Graz University of Technology
Institute of Electrical Measurement
and Measurement Signal Processing

Dissertation

Deep Learning for Video Recognition

Christoph Feichtenhofer

Monday 28th August, 2017

Advisors:

Prof. Dr. Axel Pinz
Graz University of Technology, Austria
Prof. Dr. Richard P. Wildes
York University, Toronto, Canada

Abstract

Deep learning is an emerging technique in the field of artificial intelligence. It allows us to represent signals with models of hierarchical abstraction by learning from experience. Deep representations are extremely powerful for intelligent sensing, rivaling human perception in several applications. This dissertation is concerned with learning deep spatiotemporal representations for recognition in video – an area where human level performance is still far ahead, possibly due to the way biology is processing spatiotemporal information.

The first part of the thesis considers the problem of fusing appearance and motion signals by using two-stream Convolutional Networks (ConvNets) for the task of human action recognition. We present spatiotemporal Residual Networks (ResNets) by introducing residual connections between the appearance and motion pathways to allow hierarchical learning of complex spatiotemporal features. To capture long-term dependencies, we transform pretrained image ConvNets into spatiotemporal networks by equipping these with learnable convolutional filters that are initialized as temporal residuals. Building on our spatiotemporal ResNet, we theoretically motivate multiplicative gating functions for residual networks and present a general ConvNet architecture based on multiplicative interactions of spacetime features. An irritating property of deep networks is that due to their compositional structure it is difficult to reason explicitly about what these powerful representations have learned. We shed light on deep spatiotemporal networks by visualizing what excites the learned models at the input. Our visual explanations, showing the hierarchical features of a deep spatiotemporal network, provide clear qualitative evidence for separation into appearance and motion pathways for video recognition.

A related problem is the task of dynamic scene recognition for which this thesis establishes a new state-of-the-art by presenting a novel ConvNet architecture that is based on temporal residual units and fully convolutional in spacetime. We also introduce a new video database of dynamic scenes that contains videos with and without camera motion to allow for systematic study of how this variable interacts with the defining dynamics of the scene per se. Our evaluations verify the particular strengths and weaknesses of seven previously top performing approaches as well as our novel ConvNet architecture with respect to various scene classes and camera motion parameters.

The final chapter of this thesis is concerned with detection of objects from video. We propose a ConvNet architecture for simultaneous detection and tracking, using a multi-task objective for frame-based object detection and across-frame track regression with correlation features that represent object co-occurrences across time. The frame level detections are linked based on our across-frame tracklets to produce high accuracy detections at the video level. Our approach provides better single model performance than the winning method of the last ImageNet challenge while being conceptually much simpler. Finally, we show that by increasing the temporal stride we can dramatically increase the tracker speed.

In summary, this dissertation makes several contributions for the computational sensing and analysis of spatiotemporal signals by advancing the state of the art for recognition of actions, scenes and objects, and also improving our limited understanding of deep representations for video.

Zusammenfassung

Deep Learning ist eine aufstrebende Technik auf dem Gebiet der künstlichen Intelligenz, die es uns durch das Lernen aus Erfahrung ermöglicht Signale mit hierarchischer Abstraktion darzustellen. Die gelernten Repräsentationen sind extrem leistungsfähig und erreichen in mehreren Anwendungen ähnlich gute Resultate wie Menschen. Diese Dissertation beschäftigt sich mit dem Erlernen räumlich-zeitlicher Modelle zur Erkennung von dynamischen visuellen Signalen und behandelt damit einen Bereich in dem die menschliche Leistungsfähigkeit noch weit voraus ist.

Der erste Teil der Arbeit befasst sich mit dem Problem der Fusion von Erscheinungs- und Bewegungssignalen durch die Verwendung von Convolutional Networks (ConvNets) für das Erkennen von Aktivitäten in Videos. Wir präsentieren räumlich-zeitliche Residual Networks (ResNets), indem wir Residuenverbindungen zwischen den Erscheinungs- und Bewegungspfaden einführen, um hierarchisches Lernen komplexer räumlich-zeitlicher Merkmale zu ermöglichen. Zur Erfassung von Langzeit-Abhängigkeiten werden vortrainierte Bilderkennungs-ConvNets in räumlich-zeitliche Netzwerke transformiert. Dies erfolgt durch die Ausstattung mit lernbaren Faltungsfiltern, die als zeitliche Residuen initialisiert werden. Aufbauend auf unserem räumlich-zeitlichen ResNet diskutieren wir multiplikative Gating-Funktionen für ResNets und stellen eine allgemeine ConvNet-Architektur vor, die auf multiplikativen Interaktionen von Raum-Zeit Features basiert.

Eine ungünstige Eigenschaft von tiefen Netzwerken ist, dass es aufgrund ihrer kompositorischen Struktur schwierig ist, explizite Schlussfolgerungen über die gelernten Repräsentationen zu treffen. Wir untersuchen unsere Netzwerke indem wir visualisieren was die gelernten Modelle anregt. Unsere visuellen Erkenntnisse, welche die hierarchischen Merkmale eines tiefen räumlich-zeitlichen Netzwerks zeigen, liefern klare, qualitative Beweise für die Trennung in Erscheinungsbild und Bewegungspfade in der Videoerkennung.

Ein verwandtes Problem ist die Erkennung dynamischer Szenen, wofür diese Arbeit einen neuen Stand der Technik etabliert. Dies erfolgt durch eine neuartige ConvNet-Architektur, die auf zeitlichen Residueneinheiten basiert und äquivariant in Raum und Zeit ist. Wir stellen zusätzlich eine neue Videodatenbank mit dynamischen Szenen vor, die Videos mit und ohne Kamerabewegung enthält, um die Interaktion mit der definierenden Dynamik der Szene systematisch untersuchen zu können.

Das letzte Kapitel der Arbeit beschäftigt sich mit der Erkennung von Objekten aus dem raumzeitlichen Videosignal. Wir präsentieren eine ConvNet-Architektur für das gleichzeitige Erkennen und Tracking von Objekten, welche eine Multi-Task-Zielfunktion für die Objekterkennung und Track-Regression mit Korrelationsmerkmalen verwendet. Die Frame-Level-Detektionen sind auf Basis unserer übergreifenden Tracklets miteinander verknüpft, um auf der Videoebene hochgenaue Detektionen zu erzeugen.

Zusammenfassend präsentiert diese Dissertation bedeutende Beiträge für das computerbasierte Erfassen von räumlich-zeitlichen Signalen, indem sie den Stand der Technik zur Erkennung von Aktionen, Szenen und Objekten vorantreibt und damit unser begrenztes Verständnis für tiefe, räumlich-zeitliche Repräsentationen verbessert.

Deutsche Fassung:
Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008
Genehmigung des Senates am 1.12.2008

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am ……………………………… ………………………………………………..
(Unterschrift)

Englische Fassung:

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

……………………………… ………………………………………………..
date (signature)

Contents

Acknowledgements

Advisers play a key role in the development of doctoral candidates. I would like to express deep gratitude to my advisor Axel for his invaluable guidance and support. In our numerous discussions he always contributed new insights while also providing me with the freedom to pursue my own ideas. With his deep and broad knowledge of science he taught me a lot about research, teaching and communicating. I am also grateful for his fundamental support that made my research stays in Toronto and Oxford possible. Thanks to Rick for advising me during my stays in the Vision Lab at York University, Toronto and during our regular online meetings. I am very fortunate to have such a passionate researcher as my co-advisor. Rick's ability to always keep a clear overview, even for very complex problems, were a key factor for my progress during my studies. Thanks to Andrew for hosting me in the VGG Lab at Oxford University and the numerous discussions we had in our video calls. Andrew's boundless enthusiasm for research and his abilities to always know where the field currently stands and foresee how it might evolve are just remarkable. Special thanks to all my friends and colleagues for making my time as a student very enjoyable and memorable. Finally, and most importantly, I want to profoundly thank my parents for their unconditional support that made all this possible.

Financial support: The author gratefully acknowledges support from the Austrian Academy of Sciences for receiving a DOC Fellowship at the Institute of Electrical Measurement and Measurement Signal Processing, Graz University of Technology. This work was also supported by the Austrian Science Fund (FWF P27076) and hardware donations by Nvidia.

1 Introduction

1.1 Motivation

Visual sensing and perception provide valuable information to intelligent systems. Humans are particularly proficient at recognizing and interpreting the rich visual world around them. Most of our capability for performing a variety of tasks comes from how we encode sensory measurements to extract rich representations for recognition, decision and action. How to build artificial intelligent systems that could rival or even surpass human performance is a fundamental research problem and of highest interest.

Deep learning (LeCun et al., 2015) has changed computer vision fundamentally. The core idea behind deep learning is to jointly optimize multiple layers of features for a given task of consideration. Optimization is typically performed iteratively via backpropagation of gradients through artificial neural networks with special connectivity structure (*e.g.* convolutional, recurrent) of the parameters. Since 2012 (Krizhevsky et al., 2012a), deep neural networks have made in-roads to all areas of audiovisual sensing rapidly with stunning results in all fields. Interestingly, however, deep learning did not have such a dramatic effect on video analysis where, contrary to other computer vision domains that rely on single image processing, shallow and hand-crafted representations are still powerful.

Recent results underline the lack of progress in video understanding compared to single image understanding, where even in situations where temporal information is available, it is not integrated into the basic processing. These results are unsatisfactory, as the potential richness of the temporal dimension should be exploited whenever possible, even though it comes with new challenges such as duration variation, temporal clutter and within-class variation of temporal relations.

A major drawback of Convolutional Neural Networks (CNNs) is that they require millions of parameters to be optimized during training. Thus, training problems ensue: These networks require a large amount of data to prevent overfitting and large amounts of time for learning owing to computational demand. In single image classification these issues have been ameliorated by using the huge ImageNet (Deng et al., 2009) database, as well as clever GPU implementations (Krizhevsky et al., 2012b). For video training the challenges are exacerbated, as now the network also must be optimized over the temporal domain. The lack of large datasets for the video domain could be one reason that holds back success of deep spatiotemporal image representations. To the current date, however, several attempts have been made to release large video datasets of scales that are at ImageNet level or even beyond; *e.g.* Sports-1M[1], YouTube-8M[2], YouTube Bounding Boxes[3] are orders of magnitudes larger than ImageNet. Unfortunately, due to the massive size of such datasets, they are not always well curated (e.g., labels and metadata can be inaccurate, availability can be unreliable).

Another reason than lack of data for limited success of deep learning in video could be that *video is different*, which deduces that the current still image architectures of the deep models are not ideal for spatiotemporal data. Interestingly, also nature seems to separate processing of appearance and motion information. In neuroscience, numerous studies suggest a corresponding separation into ventral and dorsal pathways of the brain. The ventral pathway is mostly tuned for appearance based perception of objects, whereas the dorsal stream is involved in localization and movement recognition and also for sensorimotor control of the observer's interactions with these objects (Goodale and Milner, 1992). Although these two pathways are not completely segregated from each other with significant crosstalk across the streams (Felleman and Van Essen, 1991, Goodale and Milner, 1992, Kourtzi and Kanwisher, 2000, Saleem et al., 2000) each pathway shows highly selective, hierarchical features for either appearance or motion detection (from local orientation selective to complex patterns in higher areas of the visual cortex). We frame the question: Why does nature devote specific neurons for both tasks and what is the benefit of decoupling the representation of appearance and motion for deep architectures?

In response to the outlined state of affairs, this thesis advances the state-of-the-art in representations for video understanding. It tries to rectify the deficit of current approaches by presenting general ConvNet architectures and design guidelines for recognition in video. Our advances come at the level of refinement of deep spatiotemporal representations that inherently capture both spatial and temporal structure of video data to abstract information most relevant for particular (e.g., recognition) tasks in an adaptive, data-driven fashion for combining local measurements across space and time to best support higher level inferences. We ground our findings both theoretically and empirically by showing visualizations of what our representations have learned and what sen-

[1]https://github.com/gtoderici/sports-1m-dataset
[2]https://research.google.com/youtube8m/
[3]https://research.google.com/youtube-bb/

sory information mostly excites them in the input space. To ensure broad practical applicability, our approaches are developed within the contexts of three major video understandings tasks: dynamic scene recognition, action recognition and video object detection. Throughout the dissertation a tight coupling of theoretical developments with rigorous empirical evaluation on applications of dynamic scene, action and object recognition fosters both fundamental and practical advances.

1.2 Applications

After decades of research dedicated to solve computer vision problems in the spatial image domain, scientific work addressing spatiotemporal representations has been relatively neglected until recent years. As motivated in the previous section, research in spacetime image understanding considerably lags behind its spatial counterpart. However, with the amount of video material on the internet growing exponentially, for example more than 400 hours of video are uploaded to YouTube every minute[4], there has never been a greater need for basic research in the field of spacetime image representation.

Recording, storing and viewing videos has become an ordinary part of our daily lives. With the rising amount of video material readily available (e.g., on the web) it is now more important than ever to develop methods for automatic video processing and content interpretation. Furthermore, the classification of image sequences is a fundamental computer vision problem on its own, as principled attacks on the challenge of revealing basic relationships between a dynamic world and images thereof.

Here, a full understanding would go beyond the ability to assign a simple label to a video to moreover encompass an integrated interpretation in terms of the captured scene, actions and objects within the scene as well as the unfolding of actions across time and space, i.e., activities. A single label is not enough for a complete understanding of video, as its sequences may contain several interesting objects, actions or events. Videos may contain stories that are told by the actions performed in a video and the causal relationship between them.

Even given these strong motivations, research in video-based computer vision has received far less attention than its single image-based counterpart. One reason for this lack of progress is that the additional temporal dimension leads to a huge amount of data to process. However, memory issues could be handled, not only by descriptor compression, but, especially for video, by online incremental processing of the data. Perhaps more fundamental, while single image interpretation can build directly on the long standing history of 2D image representation and feature extraction, full consideration of video entails the development of new approaches that inherently encompass the temporal dimension.

Vehicle autonomy is a related field that has received a surge of attention recently. The rapid recent progress seen in this field lately can be mainly devoted to the breakthrough of ConvNets in computer vision (Krizhevsky et al., 2012b). The visual sensing and perception part of an intelligent

[4]`https://www.youtube.com/yt/about/press/`

agent arguably is the most critical component, especially for avoiding fatalities in a dynamic environment. The challenges for near-zero fatality requirement are vast and only achievable with reliable, understandable models. The traditional way to approach the autonomous driving problem is by "mediated perception" where hand-designed sub-systems are responsible for performing the sensing tasks in isolation; *e.g.* estimating road structure, detecting lane markings and obstacles. A purely hand-designed approach would allow for more interpretability, but only if the overall system does not get too complex. The process of autonomous driving, however, is highly complex. There are many aspects for autonomous cars in pattern recognition, localization and mapping. It is not an easy task to drive through different types of roadways, scenery, driving and weather conditions in a dynamic environment where humans are acting unpredictably while driving or walking. We think that it is notoriously impossible to write software that can handle all these factors without artificial intelligence. Having the ability to let systems learn could tackle such a problem in an integrated manner, however, understandability of these systems has to be top priority. The contributions presented in this dissertation could be directly applied to the autonomous driving scenario where reliable recognition of objects and actions in a dynamic scene play an essential role. Next, we will briefly describe the details of the tasks on which we will apply our approaches, followed by a itemization of the contributions and outline of this thesis.

1.2.1 Scenes

The task of scene categorization is to find the categories (*e.g.* beach, city, river) to which the input sequence belongs. Humans are able to perform this task with speed and accuracy (Potter and Levy, 1969, Rousselet et al., 2004) and with little attention to the objects present in the scene (Li et al., 2002). Such a holistic understanding of the scene is also pursued by popular representations and algorithms for scene categorization (Fei-Fei and Perona, 2005, Lazebnik et al., 2006, Oliva and Torralba, 2001), where local features are used to describe a complex scene straightforwardly, without intermediately extracting semantics of the objects in the scene. Beyond being of basic scientific interested in and of itself, dynamic scenes are of interest because they can supply context for other tasks, e.g., action recognition (Marszalek et al., 2009).

Dynamic scenes are characterized by a collection of dynamic patterns and their spatial layout, as captured in short video clips. For instance, a beach scene might be characterized by drifting overhead clouds, mid-scene water waves and a foreground of static sandy texture. Other examples include forest fires, avalanches and traffic scenes. These scenes may be captured by either stationary or moving cameras; thus, while scene motion is characteristic, it can be compounded with camera induced motion. Indeed, dynamic scene classification in the presence of camera motion has proven to be more challenging than when this confounding attribute is absent. In comparison, dynamic textures (*e.g.*, (Derpanis and Wildes, 2012, Doretto et al., 2003, Szummer and Picard, 1998)) also are concerned with complicated dynamic patterns, but in simpler settings, typically with stationary cameras and the field of view completely occupied by the particular complex dynamic pattern.

In our previous research, we have studied the task of dynamic scene recognition extensively

(Feichtenhofer et al., 2013, 2014, 2016a). In our penultimate approach to the task we build on the dominant methodology to image classification and object recognition consisting of three steps: feature extraction, coding and pooling and termed our method “Dynamically Pooled Complementary Features” (DPCF) (Feichtenhofer et al., 2016a). In DPCF, an input video is analyzed in slices of the sequence, which are defined as short temporal intervals. For each slice, at each spatiotemporal location complementary spatial, temporal and color features are extracted. Next, these features are encoded into a mid-level representation that has been tuned to the task of dynamic scene recognition via a training procedure. Finally, the encoded features are pooled by a novel, dynamic spacetime pyramid that adapts to temporal scene dynamics, resulting in a feature vector, that is subject to online classification. Complementarity in terms of spatial, temporal and color channels is preserved through all three steps of processing. Note that the approach is also strongly supported by findings from neurobiology of natural visual systems (Stone, 2012), where there is a separation into parvocellular, magnocellular and konio layers that strongly suggests a similar complementarity (spatial, motion and color channels) of visual pathways.

1.2.2 Actions

Action recognition in video is an intensively researched area, with state of the art systems still being far from human performance - reflecting the difficulty of the task. Over the past years, it has been actively researched *e.g.* (Feichtenhofer et al., 2015, Jhuang et al., 2007, Karpathy et al., 2014, Laptev et al., 2008a, Simonyan and Zisserman, 2014a, Taylor et al., 2010, Wang and Schmid, 2013) and has been dominated by the Bag of Visual Word (BoW) approach, consisting of 1) feature extraction, 2) encoding, and 3) pooling and classification. Typical techniques used in these BoW steps are: *1)* SIFT (Lowe, 2004), HOG3D (Kläser et al., 2008), spacetime correlation patches (Shechtman and Irani, 2007), Histograms of Optical Flow (HOF) (Laptev et al., 2008b), Motion Boundary Histograms (MBH) (Dalal et al., 2006), trajectories (Wang et al., 2013), and Spatiotemporal Oriented Energy (SOE) (Derpanis et al., 2013); *2)* Locality-constrained Linear Coding (LLC) (Wang et al., 2010), Super Vector (SV) (Zhou et al., 2010), Vector of Locally Aggregated Descriptors (VLAD) (Jégou et al., 2012) and Fisher Vectors (FV) (Perronnin and Dance, 2007); *3)* average- and max-pooling with geometry embedded by aggregating with Spatial Pyramid Matching (SPM) (Lazebnik et al., 2006, Yang et al., 2009), or as weighted by spatiotemporal saliency (Feichtenhofer et al., 2015). Classification is mostly realized using a Support Vector Machine (SVM) (Cortes and Vapnik, 1995).

As with other areas of computer vision, recent approaches have concentrated on applying Convolutional Neural Networks (ConvNets) to this task *e.g.* (Karpathy et al., 2014, Simonyan and Zisserman, 2014a, Tran et al., 2015a), with progress over a number of strands: learning local spatiotemporal filters (Karpathy et al., 2014, Taylor et al., 2010, Tran et al., 2015a), incorporating optical flow snippets (Simonyan and Zisserman, 2014a), and modelling more extended temporal sequences (Donahue et al., 2015, Ng et al., 2015b). As actions can be understood as spatiotemporal objects, researchers have investigated carrying spatial recognition principles over to the temporal domain by learning local spatiotemporal filters (Karpathy et al., 2014, Taylor et al., 2010, Tran et al., 2015a). However, since the temporal domain arguably is fundamentally different from the spatial one, different treatment of these dimensions has been considered, *e.g.* by incorporating optical flow networks (Simonyan and Zisserman, 2014a), or modelling temporal sequences in recurrent architectures (Donahue et al., 2015, Ng et al., 2015b, Sharma et al., 2015).

However, action recognition has not yet seen the substantial gains in performance that have been achieved in other areas by ConvNets, e.g. image classification (Krizhevsky et al., 2012a, Simonyan and Zisserman, 2014b, Szegedy et al., 2015a), human face recognition (Schroff et al., 2015), and human pose estimation (Tompson et al., 2015). Indeed the current state of the art ConvNet based approaches experience performance gains by a combination with Fisher Vector encoded (Perronnin et al., 2010a) hand-crafted features (such as HOF and MBH) over dense trajectories (Wang and Schmid, 2013)). Thus, in comparison to the dramatic progress in other single image related problems, the impact of video research lags behind.

A number of datasets are available for empirical evaluation of action recognition from video. The Sports-1M (Karpathy et al., 2014) contains a large number of videos ($\approx$1M) and classes (487); however, this data was labelled automatically and therefore is not free of label noise. An alternative large scale human action dataset is the THUMOS dataset (Gorban et al., 2015) that has over 45M frames; however, only a small fraction of these actually contain the labelled action and thus are useful for supervised feature learning. Due to these circumstances, learning spatiotemporal ConvNets is still largely relying on smaller, but temporally consistent datasets such as UCF101 (Khurram Soomro and Shah, 2012) or HMDB51 (Kuehne et al., 2011), which contain short videos of actions. This circumstance facilitates learning, but comes with the risk of severe overfitting to the training data.

Part of the reason for lack of success of deep networks for action recognition is probably that current datasets used for training are either too small or too noisy (we return to this point below in related work). Compared to image classification, action classification in video has the additional challenge of variations in motion and viewpoint, and so might be expected to require *more* training examples than that of ImageNet (1000 per class) – yet UCF-101 has only 100 examples per class. Another important reason is that current ConvNet architectures are not able to take full advantage of temporal information and their performance is consequently often dominated by spatial (appearance) recognition.

1.2.3 Objects

Object detection in images has received great attention over the last years with tremendous progress mostly due to the emergence of deep Convolutional Networks (He et al., 2016a, Krizhevsky et al., 2012a, LeCun et al., 1989, Simonyan and Zisserman, 2014b, Szegedy et al., 2015a) and its region based descendants (Girshick, 2015, Girshick et al., 2014, Li et al., 2016a, Ren et al., 2016). In the case of object detection and tracking in videos, recent approaches have mostly used detection as a first step, followed by post-processing methods such as applying a tracker to propagate detection scores over time. Such variations on the 'tracking by detection' paradigm have seen impressive progress but are dominated by frame-level detection methods.

1.3 Contributions and thesis outline

The objective of this thesis is the visual analysis and representation of spatiotemporal information for recognition. The thesis is organized as follows. The next chapter describes relevant background and a literature review that serves as a basis for the for forthcoming chapters.

In Chapter 3 we study how to fuse appearance and motion features for the task of action recognition in video. Recent applications of Convolutional Neural Networks (ConvNets) for human action recognition in videos have proposed different solutions for incorporating the appearance and motion information. We study a number of ways of fusing ConvNet towers both spatially and temporally in order to best take advantage of this spatiotemporal information. We make the following findings: (i) that rather than fusing at the softmax layer, a spatial and temporal network can be fused at a convolution layer without loss of performance, but with a substantial saving in parameters; (ii) that it is better to fuse such networks spatially at the last convolutional layer than earlier, and that additionally fusing at the class prediction layer can boost accuracy; finally (iii) that pooling of abstract convolutional features over spatiotemporal neighbourhoods further boosts performance. Based on these studies we propose a new ConvNet architecture for spatiotemporal fusion of video snippets, and evaluate its performance on standard benchmarks where this architecture achieves state-of-the-art results. Our code and models are available at `http://www.robots.ox.ac.uk/~vgg/software/two_stream_action/`

Next, Chapter 4 introduces Spatiotemporal Residual Networks (ST-ResNet) as a combination of Two-stream ConvNets (Simonyan and Zisserman, 2014a) and Residual Networks (ResNets) (He et al., 2016a). Two-stream ConvNets have shown strong performance for human action recognition in videos and recently ResNets have arisen as a new technique to train extremely deep architectures. Our novel ST-ResNet architecture generalizes ResNets for the spatiotemporal domain by introducing residual connections in two ways. First, we inject residual connections between the appearance and motion pathways of a two-stream architecture to allow spatiotemporal interaction between the two streams. Second, we transform pretrained image ConvNets into spatiotemporal networks by equipping them with learnable convolutional filters that are initialized as temporal residual connections and operate on adjacent feature maps in time. This approach slowly increases the spatiotemporal receptive field as the depth of the model increases and naturally integrates image ConvNet design principles. The whole model is trained end-to-end to allow hierarchical learning of complex spatiotemporal features. As a further contribution we show that using a smaller batch size for the noisy bias and variance estimation in batch normalization fosters generalization when training very deep two-stream ResNets for action recognition. In our ablation experiments we compare additive and multiplicative interactions when connecting the two streams and also show how to best temporally pool the features for classification. We evaluate our novel spatiotemporal ResNet using two widely used action recognition benchmarks where it exceeds the previous state-of-the-art.

Chapter 5 builds on our findings from the preceding chapter. While our ST-ResNet led to state-of-the-art performance, we did not provide systematic justification for our design choices. Here, we reconsider the combination of the two-stream and ResNet approaches in a more thorough fashion to increase the understanding of how these techniques interact. We provide a thorough analysis of additive and multiplicative interactions with two ResNet streams, as well as unidirectional *vs.* bidirectional connections between the two streams. We also study ways of increasing the temporal footprint of the network by injecting temporal kernels on strided/dilated inputs to enable hierarchical learning of long-term correspondences, which is found to be especially significant experimentally. We ground our architectural design on a solid theory for multiplicative motion gating and initial identity mapping kernels that perform temporal filtering. In summary, the resulting novel architecture is a ConvNet based on multiplicative interactions of spacetime features. Our model combines the appearance and motion pathways of a two-stream architecture by motion gating and is trained end-to-end. We theoretically motivate multiplicative gating functions for residual networks and empirically study their effect on classification accuracy. To capture long-term dependencies we inject identity mapping kernels for learning temporal relationships. In empirical investigation we find that our model produces a new state-of-the-art in action recognition.

Our models for both Chapters 4 and 5 are fully convolutional in spacetime and able to evaluate a video in a single forward pass. We share them together with the source code at `https://github.com/feichtenhofer/st-resnet`.

In Chapter 6, we are concerned with understanding of what a deep spatiotemporal network has learned. We expand on the well-known visualization technique of activation maximization which has previously been used to visualize neurons in deep image-classification ConvNets. For the first time, we apply activation maximization to both, appearance and motion pathways of a patiotemporal architecture. Our results provide highly intuitive explanations for what excites the filters throughout the hierarchy of the networks. We show visual stimuli with vastly different motion speeds at the input which all excite the same neuron. Overall, the chapter provides clear qualitative evidence for separation into two streams for processing appearance and motion – a principle that has also been found in nature where numerous studies suggest a corresponding separation into ventral and dorsal pathways of the brain.

The subsequent Chapter 7 switches the task from action to dynamic scene recognition. The scenes are recognized on the basis of their image spacetime appearance, *e.g.*, as forest fire vs. beach vs. city. Here, we combine three contributions to establish a new state-of-the-art in dynamic scene recognition. First, we present a ConvNet architecture based on temporal residual units that is fully convolutional in spacetime and does not rely on optical flow inputs. Our model augments spatial ResNets with convolutions across time to hierarchically add temporal residuals as the depth of the network increases. This simple technique also allows to generalize (extremely deep) pre-trained appearance models to the spatiotemporal domain. Second, existing approaches to video-based recognition are categorized and a baseline of seven previously top performing algorithms are selected for comparative evaluation on dynamic scenes. Third, we introduce a new and challenging video database of dynamic scenes that more than doubles the size of those previously available. We analyze the two existing benchmark datasets for dynamic scene recognition and identify need and room for significant improvement: We have carefully designed a novel, extended dataset of 20 different dynamic scene categories, including an efficient evaluation protocol. The dataset encompasses a wide range of natural variations (seasonal and diurnal changes as well as those of viewing parameters) and adds six additional scene classes to those previously available. Each scene class is captured with and without camera motion to allow for systematic study of how this variable interacts with dynamics of the scene per se. Neither of the major extant dynamic scenes datasets allows for systematic control of this dimension. Camera motion is especially relevant in dynamic scene recognition, where defining scene dynamics can be obscured by camera motion. Our evaluations verify the particular strengths and weaknesses of the baseline algorithms with respect to various scene classes and camera motion parameters. While performance on previous datasets was already saturated, the new dataset is sufficiently challenging to foster further research in dynamic scene recognition. Our temporal ResNet boosts recognition performance and establishes a new state-of-the-art on dynamic scene recognition. Our code and models are available at `https://github.com/feichtenhofer/temporal-resnet` and our dynamic scene recognition dataset is available at `http://vision.eecs.yorku.ca/research/dynamic-scenes/`.

Chapter 8 considers the recently emerging problem of large-scale video object detection. Recent approaches for high accuracy detection and tracking of object categories in video consist of complex multistage solutions that become more cumbersome each year. In this work we propose a ConvNet architecture that jointly performs detection and tracking, solving the task in a simple and effective way. Our contributions are threefold: First, we set up a ConvNet architecture for simultaneous detection and tracking, using a multi-task objective for frame-based object detection and across-frame track regression; second, we introduce novel correlation features that represent object co-occurrences across time to aid the ConvNet during tracking; third, we link the frame level detections based on our across-frame tracklets to produce high accuracy detections at the video level. Our ConvNet architecture for spatiotemporal object detection is evaluated on the large-scale ImageNet VID dataset where it achieves state-of-the-art results. Our approach provides better single model performance than the winning method of the 2016 ImageNet challenge while being conceptually much simpler. Finally, we show that by increasing the temporal stride we can dramatically increase the tracker speed.

Finally, the thesis is concluded in Chapter 9 where further ideas for future directions are given.

The chapters are related to the following publications

- Chapter 3: *Convolutional Two-Stream Network Fusion for Video Action Recognition*
  Christoph Feichtenhofer, Axel Pinz, Andrew Zisserman
  IEEE Conference on Computer Vision and Pattern Recognition (**CVPR**) 2016
- Chapter 4: *Spatiotemporal Residual Networks for Video Action Recognition*
  Christoph Feichtenhofer, Axel Pinz, Richard P. Wildes
  Advances in Neural Information Processing Systems (**NIPS**) 2016
- Chapter 5: *Spatiotemporal Multiplier Networks for Video Action Recognition*
  Christoph Feichtenhofer, Axel Pinz, Richard P. Wildes
  IEEE Conference on Computer Vision and Pattern Recognition (**CVPR**) 2017
- Chapter 7: *Temporal Residual Networks for Dynamic Scene Recognition*
  Christoph Feichtenhofer, Axel Pinz, Richard P. Wildes
  IEEE Conference on Computer Vision and Pattern Recognition (**CVPR**) 2017
- Chapter 8: *Detect to Track and Track to Detect*
  Christoph Feichtenhofer, Axel Pinz, Andrew Zisserman
  IEEE International Conference on Computer Vision (**ICCV**) 2017

The publications below describe work that has been done earlier and is loosely related to, but not described in, this thesis:

- *Bags of Spacetime Energies for Dynamic Scene Recognition*
  Christoph Feichtenhofer, Axel Pinz, Richard P. Wildes
  IEEE Conference on Computer Vision and Pattern Recognition (**CVPR**) 2014
- *Dynamically Encoded Actions based on Spacetime Saliency*
  Christoph Feichtenhofer, Axel Pinz, Richard P. Wildes
  IEEE Conference on Computer Vision and Pattern Recognition (**CVPR**) 2015
- *Dynamic Scene Recognition with Complementary Spatiotemporal Features*
  Christoph Feichtenhofer, Axel Pinz, Richard P. Wildes
  IEEE Transactions on Pattern Analysis and Machine Intelligence (**PAMI**) 2016

2 Deep Learning Background

In this chapter we review methods that are closely related to the contributions of this thesis. Since the forthcoming chapters will build on deep networks for image and video recognition tasks, this chapter provides a brief introduction to notation and technical terms. Particularly, we start by discussing our employed practices to apply deep networks to large scale machine learning problems, then review ConvNets for the image classification task in Section 2.2, describe the two-stream ConvNet variant for video recognition tasks in Section 2.3, and provide an overview of the region-based detection networks serving as baseline for the task of object detection from video in Section 2.4. For a more comprehensive overview please see (Goodfellow et al., 2016) and for further implementation details please see (Chatfield et al., 2014a, Li et al., 2016a, Ren et al., 2016, Sermanet et al., 2014, Simonyan and Zisserman, 2014a, 2015, Szegedy et al., 2015a, Vedaldi and Lenc, 2015).

2.1 Feed-forward neural networks

Generally, feed-forward neural networks consist of layers of neurons that are modelled as weights and non-linearities transforming the activations of such neurons. Formally, the i^{th} neuron at the l^{th} layer takes as input the output of all the neurons from the previous layer $\mathbf{x}_l$ and applies its parameters, *i.e.* weights, $\mathbf{w}_l^{(i)}$ and biases $\mathbf{b}_l^{(i)}$, to produce an output value

$$\mathbf{a}_l^{(i)} = \mathbf{x}_l^\top \mathbf{w}_l^{(i)} \qquad \mathbf{x}_{l+1}^{(i)} = f(\mathbf{a}_l^{(i)} + \mathbf{b}_l^{(i)}), \tag{2.1}$$

where f is an activation function taking in activations, $\mathbf{a}$, and biases $\mathbf{b}$. All linear weight layers are followed by a non-linear activation function f (*e.g.* a sigmoid or tanh) to allow the learning of non-linear input-output mappings. The networks presented in this thesis will all use a Rectified Linear Unit (ReLU) as activation function which, for a given input $\mathbf{x}$, computes as output $\mathbf{y} = \max\{0, \mathbf{x}\}$. Rectifiers ease (gradient-based) optimization because they do not have curvature or saturation regions as *e.g.* a sigmoid function would have. A *deep* network is a hierarchical representation with multiple (hidden) layers of neurons (weights and non-linearities) between input and output of the network. This is very loosely motivated by biology where different rows in the weight matrices would represent a simple abstraction of the synaptic connections of one neuron to its input, with positive factors exciting and negative ones prohibiting a neuron.

The network architectures proposed in this thesis are trained by using supervised learning. The parameters of a network, θ, are learned with gradient-based optimization algorithms by backpropagation (Rumelhart et al., 1986) of an error signal (loss) that measures the discrepancy of the network predictions $\hat{\mathbf{x}}$ and the labels $\mathbf{c}$ provided as supervision.

Gradient based optimization searches in the parameter space to find $\theta^\star$ that minimizes the loss over the N training examples

$$\theta^\star = \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} \ell(\mathbf{x}_0^{(i)}, \mathbf{c}^{(i)}; \theta), \tag{2.2}$$

where the loss function ℓ is evaluated for the i^{th} training example $\mathbf{x}_0^{(i)}$ and its label $\mathbf{c}^{(i)}$ to produce a single scalar value. A scalar loss allows for efficient evaluation of the gradient on the error function $\nabla_\theta \ell(\mathbf{x}, \mathbf{c}; \theta)$ w.r.t. the network parameters θ by recursively applying the chain rule (*i.e.* the backpropagation algorithm). After the gradient is computed via backpropagation, optimization proceeds by gradient descent on the loss function and adjusting parameters by a step in this direction

$$\theta \leftarrow \theta - \eta \nabla_\theta \ell(\mathbf{x}, \mathbf{c}; \theta), \tag{2.3}$$

with η being the learning rate (or step size) for updates. In all our experiments we resort to minibatch Stochastic Gradient Descent (SGD) with momentum as optimization algorithm (Bottou, 2010). Minibatch SGD iteratively computes parameter updates for M training examples $\{\mathbf{x}_0^{(1)}, \ldots, \mathbf{x}_0^{(M)}\}$ by averaging the gradient over that batch $\frac{1}{M} \nabla_\theta \sum_{i=1}^{M} \ell(\mathbf{x}_0^{(i)}, \mathbf{c}^{(i)})$ and applying updates (2.3) based on these noisy estimates of the gradient over the whole training set. In all our experiments we set the learning rate according to a fixed schedule, *i.e.* after a fixed number of iterations the rate is decreased by some factor. Momentum (Sutskever et al., 2013) is used to speed up and smooth the gradient update by incorporating the velocity vector, $\mathbf{v}$, of the direction that consistently reduces the loss ℓ

$$\mathbf{v} \leftarrow \mu \mathbf{v} - \eta \nabla_\theta \ell(\mathbf{x}, \mathbf{c}; \theta), \tag{2.4}$$

$$\theta \leftarrow \theta + \mathbf{v}, \tag{2.5}$$

with $\mu \in [0, 1)$ being the momentum factor, which exponentially decays the estimated gradients from previous iterations. There exist techniques that use second-order information or more adaptive gradient schedules, *e.g.* Adam (Kingma and Ba, 2015) or RMSProp (Hinton et al., 2012b), but these are not used for optimizing the proposed networks of this thesis. All following chapters build on classical SGD, as it produces stable and accurate models for the tasks of image and video classification (Krizhevsky et al., 2012a, Simonyan and Zisserman, 2014a, Szegedy et al., 2015a). An exception is Chapter 6 where Adagrad (Duchi et al., 2011) is employed due to the difficulties in scaling learning rates for the underlying problem of activation maximization; an optimization algorithm description is given in the corresponding Chapter.

A standard pre-processing technique for neural networks is centering the data around zero by subtracting the mean over the training set; further pre-processing techniques include normalization by the standard deviation, PCA decorrelation or whitening transformation. Note that for images that have already a bounded input (*e.g.* between 0 and 255), and approximately equal scaling, these input normalizations are not used in practice (subtracting the mean, however, is standard practice). Normalizing the input to the layers of neural networks has been proven as important for several reasons, especially when the networks are very deep. The problem that arises in deep networks is described as internal covariate shift in (Ioffe and Szegedy, 2015), which points to the change of the layer activation distributions during training. In the gradient update step, every layer is optimized with the assumption that the other layers do not change. Each layer, however, gets as input the output from another layer and since the networks operate with (zero-centered) activation functions, a shift in the input distribution can have a dramatic effect on optimization. This effect can be addressed by careful initialization (*e.g.* (He et al., 2015)) of the network weights, but only if the network is not too deep and the learning rate is kept low. Batch Normalization (BN) (Ioffe and Szegedy, 2015) is a technique that re-parametrizes the optimization of the network and can be seen as a breakthrough for efficient training of very deep models. BN normalizes the outputs of a layer to have zero mean and unit variance across a batch. At the l^{th} layer, it first computes the mean μ_l and standard deviation σ_l of the activations $\mathbf{x}_l$ and then transforms these as

$$\mathbf{x}_{l+1}^{\text{BN}} = \gamma_l \frac{\mathbf{x}_l - \mu_l}{\sqrt{\sigma_l^2 + \epsilon}} + \beta_l, \tag{2.6}$$

with ϵ being a small constant added for stability, and γ_l and β_l being scale and shift parameters that are learned to allow the output of batch-normalization having a distribution that has non-zero mean and non-unit variance, so that the network can represent the same input-output mappings as without batch normalization. Note that μ_l and σ_l are computed over the batch and also the spatial dimensions of the input $\mathbf{x}_l$ and backpropagation operates backwards through the moment estimation. Also notably, batch-normalization injects noise into the training process since, similar as minibatch SGD, it approximates statistics over the entire training data. This process leads to different network predictions for different batch constellations. Since during testing a deterministic behaviour, irrespective of the batch constellation, is desired, μ_l and σ_l are replaced with a running average of these moments over training.

Especially if the batch-size is small, or when the batches are highly correlated (*e.g.* if features of multiple frames from a single video are in the batch), this leads to a moment estimation that is highly biased on the current batch and the dependencies of the samples. Empirically, we find in Chapter 4 that this situation increases training error and provides a regularizing effect. Regularization can be helpful for increasing generalization of trained models to previously unseen (test) data. In this thesis several additional practices for regularization are employed (*e.g.* training data augmentation) and will be described in the relevant chapters. Interestingly, the regularizing effect coming from the minibatch dependence in BN is not well understood in the literature, although, there have been very recent works which ameliorate this minibatch dependence (Ba et al., 2016, Ioffe, 2017, Salimans and Kingma, 2016). This improvement can be helpful for applying BN to tasks where underfitting is an issue *e.g.* Generative Adversarial Networks (GANs) in (Salimans and Kingma, 2016). Next, we introduce deep learning architectures that serve as a starting point for the presented approaches in this thesis.

2.2 Convolutional networks for image recognition

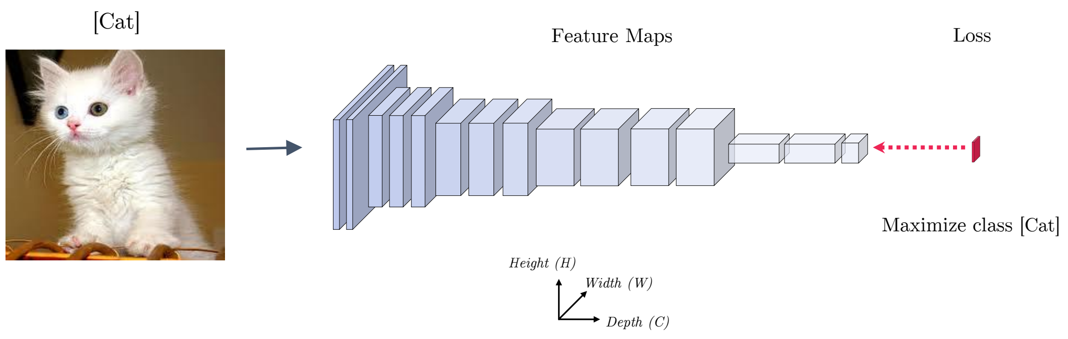


Figure 2.1: Output feature maps of an image classification network. Given an input image, forward propagation computes the output network layers which produce feature maps of decreasing size (H, W) and increasing dimensionality (C), up to the loss layer which maximizes a given class label (c) (*e.g.* "Cat").

ConvNets are artificial neural networks with special (spatial) connectivity structure based on an idea of local shift invariance that has been observed in the visual cortex of the brain. Hubel and Wiesel (Hubel and Wiesel, 1962) postulate simple cells that detect local features and complex cells that pool the outputs of simple cells within a retinotopic neighbourhood. This model inspired Fukushima (Fukushima, 1980) to develop the Neocognitron and LeCun *et al.* to develop ConvNets and apply them to digit recognition (LeCun et al., 1989) and optical character recognition (LeCun et al., 1998). In 2012, a paradigm shift enabled by large annotated datasets (*i.e.* ImageNet (Russakovsky et al., 2015a) with ≈1M training images in 1K classes) and hardware for fast parallel computing (*i.e.* GPUs that allow processing of >200 images/second) led to unprecedented performance in large scale image classification (Krizhevsky et al., 2012a), and since then, the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) (Russakovsky et al., 2015a) has become the driving force in developing new ConvNet architectures for image recognition. Similarly, deep networks have also become a standard tool for speech recognition (Hinton et al., 2012a) and natural language processing (Wu et al., 2016a). Moreover, it has been found that the features learned by ConvNets are well transferable to other, related tasks. This result enabled success in numerous machine vision applications such as object detection (Girshick et al., 2014) and semantic segmentation (Long et al., 2015).

ConvNets are functions that map an input tensor $\mathbf{x}_0 \in \mathbb{R}^{H \times W \times T \times C}$ to an output vector $\mathbf{y}$ by a sequence of operations defined by the layers of the network; here, H,W, T and C denote the width, height, duration and depth of the tensor. Note that the spatiotemporal dimensions for the tensors can be of an arbitrary number, but are usually set to two ($T = 1$) when working with images or three for working with videos. A ConvNet consists of a chain of L layers that are executed in a sequential manner. Now, considering each layer f_l in a network, it computes an output vector $\mathbf{x}_{l+1}$ that depends only on its input $\mathbf{x}_l$ and parameters $\mathbf{w}_l$: To make this more concretely, each layer computes a function

$$\mathbf{x}_{l+1} = f_l(\mathbf{x}_l; \mathbf{w}_l). \tag{2.7}$$

to produce output map $\mathbf{x}_{l+1} \in \mathbb{R}^{H_{l+1} \times W_{l+1} \times T_{l+1} \times C_{l+1}}$, where W, H, T and C are the width, height, duration and number of channels (*i.e.* depth) of the respective feature maps. The most popular layers, namely convolutional, fully-connected, pooling and nonlinearity layers, used in popular feedforward ConvNet architectures (He et al., 2016a, Simonyan and Zisserman, 2015, Szegedy et al., 2015a) are described in the remainder of this chapter.

Convolutional layers. The convolutional layers are equipped with filter weights $\mathbf{w}_l \in \mathbb{R}^{H' \times W' \times T' \times C' \times C''}$ and biases $b \in \mathbb{R}^{C'}$ which are convolved with the input $\mathbf{x}_l$

$$\mathbf{x}_{l+1}^{\text{conv}} = \mathbf{x}_l * \mathbf{w}_l + b_l, \tag{2.8}$$

where the input depth of $\mathbf{x}_l$ is C' and the output depth of $\mathbf{x}_{l+1}^{\text{conv}}$ is C''. Specifically, the layer slides these c filter kernels over all spatiotemporal locations i, j, t of the input and performs pointwise multiplication and summation

$$\mathbf{x}_{l+1}^{\text{conv}}(i,j,t,c) = b_c + \sum_{i'=1}^{H'} \sum_{j'=1}^{W'} \sum_{t'=1}^{T'} \sum_{c'=1}^{C'} \mathbf{x}_l(i',j',t',c')\mathbf{w}_l(i',j',t',c',c). \tag{2.9}$$

Besides the filter dimensions, $H' \times W' \times T' \times C' \times C''$, which correspond to the number of parameters (excluding the length of the bias vector, C', the convolutional layer (and any other locally operating layer that behaves like a filter) has two more variables, stride s and padding

p that modify the output size of the convolution. The stride defines how the filter is translated over the input dimensions (width, height, duration) and how many operations are performed; *i.e.* the increments of (i', j', t') between each multiplication in (2.9). A stride of one corresponds to a dense convolution, where the filter is slid over all input positions and a larger stride leads to a subsampling of the input which results in smaller output dimension. The padding p defines the size of the border added to the input before the filtering operation is performed. Typically the input is padded with zeros at the border to produce outputs of the same size as the input. Specifically, for a single dimension (*i.e.* width W), the size of the output is defined as

$$W'' = \left\lfloor \frac{W - W' + p_w^{\text{left}} + p_w^{\text{right}}}{s_w} \right\rfloor + 1, \tag{2.10}$$

where W is the input width, W' is the filter width, p_w^{left} is the padding added to the left of the input, p_w^{right} is the padding added to the right and s_w is the stride of the filtering operation in the horizontal direction.

Fully connected layers. Having defined a convolutional layer, a fully connected layer is a special case of a convolutional layer which has a filter size that is equal to the input size; therefore, the filters of such a layer are connected to all locations of the input and produce a single scalar output.

Pooling. The number of filters (*i.e.* the depth of the feature maps) is increased when going from input (*e.g.*, three colour channels) to the output of the network. Subsampling of feature maps has three direct motivations. First, the filters deeper in the network hierarchy should operate on larger receptive fields at the input to capture high-level semantics. Second, for classification tasks, the output of the network should be invariant to the exact location of the objects present in the input. Third, a limited memory budget in hardware can put a constraint on the overall feature map dimensions that can be held in a network. Pooling can be thought of as a filter that computes a pre-defined function in a local region $H' \times W' \times T'$ individually for each feature channel c. Two popular pooling functions are employed in current ConvNets: max and average pooling.

Dilation. Using dilated filters is a way of increasing the receptive field, irrespective of the subsampling used. Dilation d increases the offset on which the filter taps operate on the input ($d = 1$ corresponds to no dilation). Pictorially, dilating the filter-taps can be thought as padding the filter kernel by zeros. For example, a one dimensional filter $[w_1, w_2, w_3]$ with a dilation factor of $d = 2$ corresponds to $[w_1, 0, w_2, 0, w_3]$. Note that in practice this composition is implemented efficiently by striding the filter taps on the input by the dilation factor. Dilated filter processing is a long standing practice in the signal processing community where it is referred to as *spread-tap filtering*.

Example network. This thesis will build on several different network architectures. One highly effective (but computationally demanding) architecture is the VGG-16 network. Here, we illustrate the VGG-16 network (Simonyan and Zisserman, 2015) for its simplicity as it is designed by stacking

13 convolutional layers of size 3×3 each with ReLU activation functions, followed by two fully connected layers. In Fig. 2.2 we illustrate the architecture and how it is used for image classification.

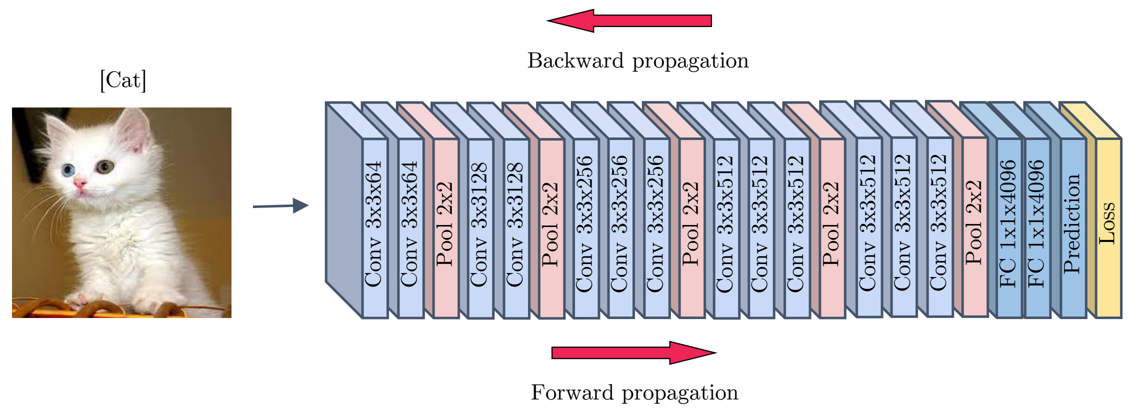


Figure 2.2: Architecture of a VGG-16 network (ReLU non-linearities not shown). Given an input image, forward propagation computes the output of the network and compares it with the correct answer (cat) to obtain an error signal (loss). The error signal is back-propagated to get derivatives for learning parameters (*i.e.* weights) of the individual layers. Supervised learning minimizes the loss (and some regularization term) w.r.t. the parameters over the training data. The layers are shown with type and the filter dimensions ($H' \times W' \times C''$), namely filter width, height and output channels. Note that the temporal extent (T') and input dimensionality (C') is omitted for brevity.

Geometric transformations. Starting from AlexNet (Krizhevsky et al., 2012a), ConvNet architectures have evolved over the last few years. In Table Table 2.1 we show three example architectures, *i.e.* AlexNet (winner ILSVRC'12), VGG-M (Chatfield et al., 2014b) (which is very similar to the ZF-network (Zeiler and Fergus, 2013), winner ILSVRC'13), and VGG-16 (Simonyan and Zisserman, 2015) (2nd place ILSVRC'14), that were popular in computer vision literature from 2012-2015 (we omit the GoogleNet (Szegedy et al., 2014), the winning architecture of ILSVRC'14, for simplicity). The geometric properties of the layers can be read from Table 2.1, where we see a slowly growing receptive field coupled with an increase of feature dimensionality. Notably, since padding is used during filtering, the overall theoretical receptive field can grow larger than the input resolution. In Fig. 2.1 we show the output feature maps for the VGG-16 architecture.

AlexNet			VGG-M			VGG-16		
type	size	stride	type	size	stride	type	size	stride
conv1	11	4	conv1	7	2	conv1_1	3	1
relu1	11	4	relu1	7	2	relu1_1	3	1
						conv1_2	5	1
						relu1_2	5	1
norm1	11	4	norm1	7	2			
pool1	19	8	pool1	11	4	pool1	6	2
conv2	51	8	conv2	27	8	conv2_1	10	2
relu2	51	8	relu2	27	8	relu2_1	10	2
						conv2_2	14	2
						relu2_2	14	2
norm2	51	8	norm2	27	8			
pool2	67	16	pool2	43	16	pool2	16	4
conv3	99	16	conv3	75	16	conv3_1	24	4
relu3	99	16	relu3	75	16	relu3_1	24	4
						conv3_2	32	4
						relu3_2	32	4
						conv3_3	40	4
						relu3_3	40	4
						pool3	44	8
conv4	131	16	conv4	107	16	conv4_1	60	8
relu4	131	16	relu4	107	16	relu4_1	60	8
						conv4_2	76	8
						relu4_2	76	8
						conv4_3	92	8
						relu4_3	92	8
						pool4	100	16
conv5	163	16	conv5	139	16	conv5_1	132	16
relu5	163	16	relu5	139	16	relu5_1	132	16
						conv5_2	164	16
						relu5_2	164	16
						conv5_3	196	16
						relu5_3	196	16
pool5	195	32	pool5	171	32	pool5	212	32
fc6	355	32	fc6	331	32	fc6	404	32
relu6	355	32	relu6	331	32	relu6	404	32
fc7	355	32	fc7	331	32	fc7	404	32
relu7	355	32	relu7	331	32	relu7	404	32
fc8	355	32	fc8	331	32	fc8	404	32
pred	355	32	pred	331	32	pred	404	32

Table 2.1: Geometric properties of the AlexNet, VGG-M and VGG-16 ConvNet architectures. The columns list the type and number of the layers, *i.e.* convolution (conv), rectification (relu), pooling (pool), fully-connected (fc), or prediction (pred); the corresponding receptive field sizes (size), and the pixelwise stride in horizontal and vertical direction that is used in the local filtering operations (stride).

2.3 Two-Stream networks for video recognition

Appearance and motion cues are vital for visual recognition. For example, try to guess which type of swimming action (*e.g.* breast stroke, crawling) a person is performing just from a single image. By just looking at the pose of the person this task can be ambiguous and is hard to determine, even for humans. For humans it is much easier to recognise the action in Fig. 2.3 if one knows what (spatial cues) is moving how (temporal cues) and the manner in which these cues evolve over time. In the above case, knowing how the limbs of the person are moving over short temporal

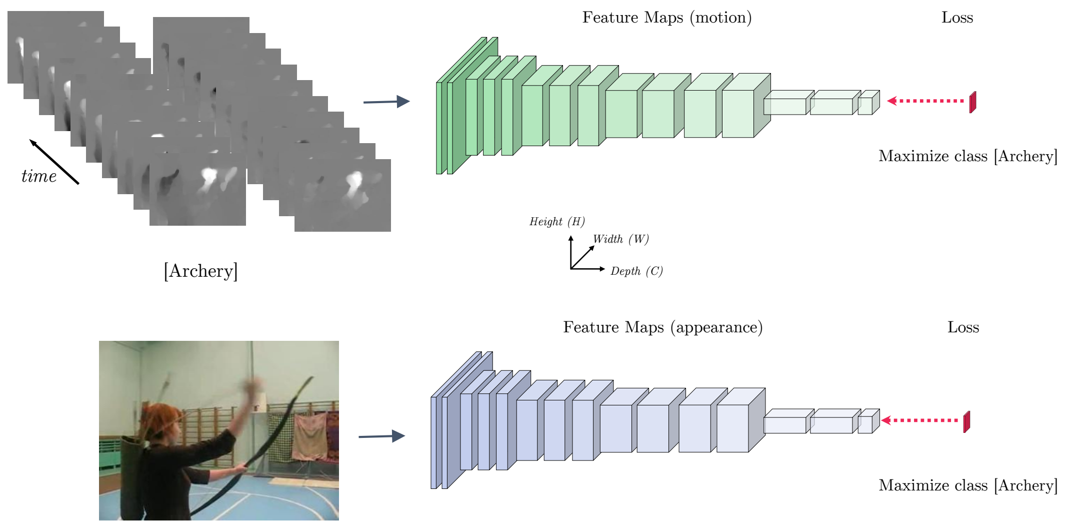


Figure 2.3: Output feature maps of a VGG-16 two-stream action classification network. Given input samples of appearance (image frame) and motion (optical flow frames), forward propagation computes the output network layers which produce feature maps of decreasing size (H, W) and increasing dimensionality C, up to the loss layers which maximize a given class label c (*e.g.* "Archery"). Note that there is no connection across the streams and training and testing is implemented in separation.

duration would ease the problem. The two-stream ConvNet architecture proposed by Simonyan and Zisserman (Simonyan and Zisserman, 2014a) incorporates that information by training separate ConvNets on still images as well as on stacks of (frame-level) optical flow.

This two-stream approach (Simonyan and Zisserman, 2014a) separately trains two ConvNet streams, operating on motion and appearance information. Each stream performs video recognition on its own and softmax class posteriors are combined by late fusion for final classification. During testing, a fixed number of frames is sampled for the videos and the prediction scores of the two networks are averaged for all frames. Notably, here, we found that training a ConvNet jointly on both RGB and optical flow input is non-trivial, as such an architecture can severely overfit to appearance information.

Appearance and motion streams. The appearance stream operates on individual RGB image frames from the video and performs classification of these. In this way, appearance information can be most effectively exploited for recognizing actions by transfer learning from powerful ImageNet models. The motion stream operates on a stack of optical flow frames. In this thesis we will show that also this stream can effectively exploit deep models trained for image classification, and moreover in Chapter 6 we will provide intuitive examples of why this is the case. The motion

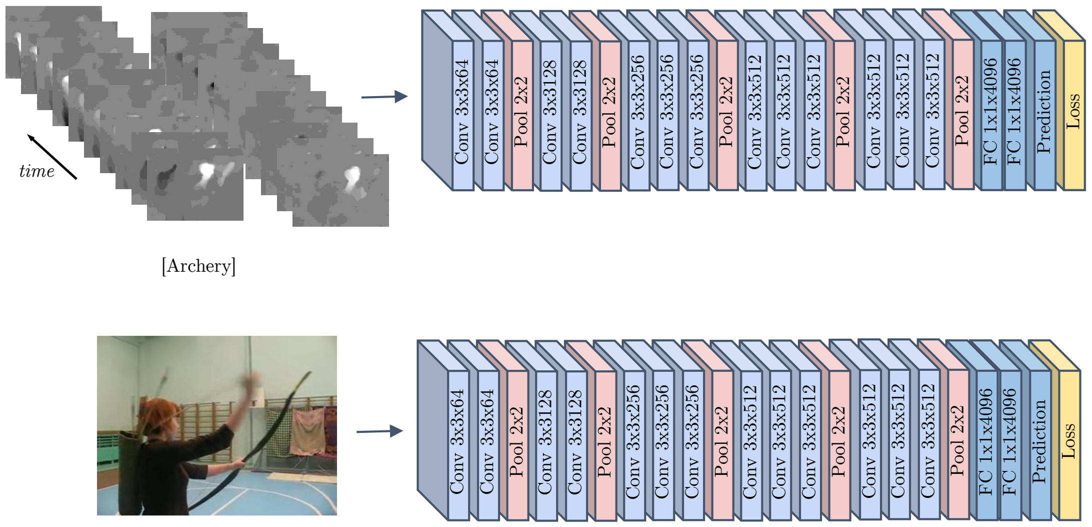


Figure 2.4: Architectural details of a VGG-16 Two-Stream network that trains two separate network architectures for the appearance and motion stream.

stream receives information based on an input stack of $L = 10$ horizontal and vertical optical flow frames. For details on the implementation and ablation studies on the parameter L as well as the optical flow input (*e.g.* mean subtraction and trajectory accumulation in the stacking) the reader is referred to the original publication (Simonyan and Zisserman, 2014a). Notably, here, the mean-flow subtraction as a form of simple camera motion compensation showed no significant gain, nor did the way of stacking the optical flow fields (accumulating trajectories *vs.* inter-frame motion). These observations may be explained by the hypothesis that such transformations are implicitly learned by the network during training. The parameter L, which represents the number of frames for each stack of optical flow input, however is crucial and increases performance for increasing number of frames L. Besides their comparison of several techniques to align the optical flow frames where they concluded that simple stacking of $L = 10$ horizontal and vertical flow fields performs well, they also employed multitask learning on UCF101 and HMDB51 to increase the amount of training data and to improve the performance on both. In the forthcoming chapters, we will build on the two-stream baseline networks by training separate models of varying complexity (*e.g.* VGG-M, VGG-16) for the appearance and motion streams. A visual example for training two such separate streams with a VGG-16 model is given in Fig. 2.4.

Biological perception. The two-stream architecture is inspired by the two-stream hypothesis (Goodale and Milner, 1992, Mishkin and Ungerleider, 1982, Ungerleider and Haxby, 1994) that postulates two pathways in the human visual cortex: the ventral pathway that responds to spatial features such as shape or colour of objects and the dorsal pathway that is sensitive to object transformations and their spatial relationship, as e.g. caused by motion.

Research in neuroscience suggests that the ventral stream is a stack of cortical areas (RGC, LGN, V1, V2, V4, IT) where each area *locally* applies a set of operations to produce a retinotopic output map that is used as an input for a higher map. Such an "object recognition" stream could be simplified to a ConvNet that applies filtering, including pooling to grow the receptive field and provide invariances to viewpoint, scale, pose, etc. The dorsal stream, on the other hand, is responsible for recognizing motion and the spatial location of different entities in the visual field of view. An interesting question that arises is why does nature separate visual information processing into a 'what' and a 'where' pathway? One answer to that question could be that it has an architectural advantage, since encoding exact spatial coordinates (where) is counter-productive for recognizing what an object is – a case where invariance to spatial transformations (*e.g.* shape, pose, scale) is beneficial. On the other hand, if we would like to know the spatial position of an object and how the location evolves over time (*i.e.* motion), object measurements such as shape, pose and scale are important cues and we don't want to be invariant against these.

Nevertheless, a full separation of 'what' and 'where' streams is not expedient when we would like to reason about spatiotemporal objects with discriminative motion. Here a fusion of the 'what' and 'where' information could provide clear benefits. Research in neuroscience also suggests that there are numerous connections between the cortical areas of the ventral and the dorsal pathways; *e.g.*, that motion information is distributed into separate visual areas (Born and Tootell, 1992, Van Essen and Gallant, 1994). In this work, our goal is to give the two-stream architecture the possibility of jointly using these cues for action recognition in video.

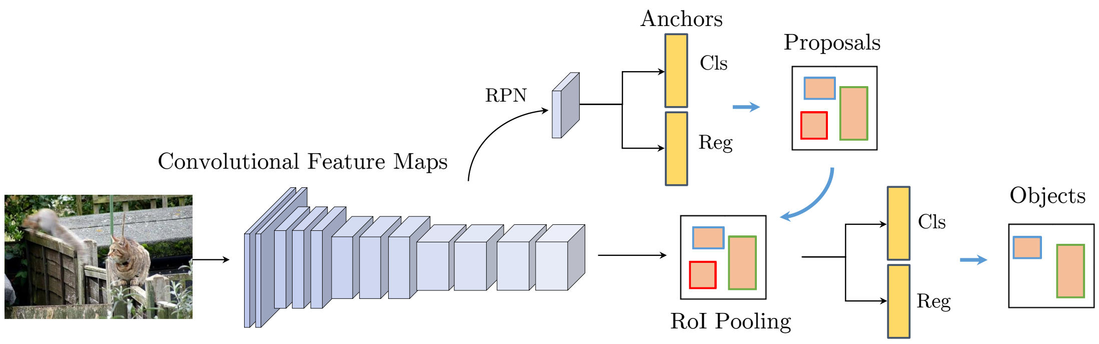


Figure 2.5: An object detection architecture working on convolutional feature maps.

2.4 Convolutional networks for object detection

Two families of detectors are currently popular: First region proposal based detectors R-CNN (Girshick et al., 2014), Fast R-CNN (Girshick, 2015), Faster R-CNN (Ren et al., 2016) and R-FCN (Li et al., 2016a) and second, detectors that directly predict boxes for an image in one step such as YOLO (Redmon et al., 2016) and SSD (Liu et al., 2016).

Our approach in Chapter 8 builds on R-FCN (Li et al., 2016a), which is a simple and efficient framework for object detection on region proposals with a fully convolutional nature see Fig. 2.5. In terms of accuracy it is competitive with Faster R-CNN (Ren et al., 2016), which uses a multi-layer network that is evaluated per-region (and thus has a cost growing linearly with the number of candidate RoIs). R-FCN reduces the cost for region classification by pushing the region-wise operations to the end of the network with the introduction of a position-sensitive RoI pooling layer, which works on convolutional features that encode the spatially subsampled class scores of input RoIs.

Here, we describe the object detection pipeline used in Chapter in 8 that is based on R-FCN (Li et al., 2016a). Given a video, R-FCN takes frames $\mathbf{I}^t \in \mathbb{R}^{H_0 \times W_0 \times 3}$ at time t and pushes them through a backbone ConvNet (*e.g.* a ResNet-101 trunk (He et al., 2016a)) to obtain feature maps $\mathbf{x}_l^t \in \mathbb{R}^{H_l \times W_l \times D_l}$ where W_l, H_l and D_l are the width, height and depth (*i.e.* number of feature channels) of the respective feature map output by layer l. Notably, since high-accuracy input resolution is of clear importance for localizing objects, R-FCN reduces the effective stride at the last convolutional layer from 32 pixels to 16 pixels by modifying the conv5 block to a spatial filter stride of one, to compensate the reduced receptive field induced by that transformation, the authors also increase its receptive field by using dilation for convolution (see Section 2.2) with a dilation factor of 2 for these filters.

Region proposal. Similar to Fast R-CNN (Girshick, 2015), the R-FCN approach (Li et al., 2016a) operates in two stages: First, it extracts candidate regions of interest (RoIs) using a Region Proposal Network (RPN) (Ren et al., 2016); second, it performs region classification into different object categories and background by using a position-sensitive RoI pooling layer (Li et al., 2016a). An example is visualized in Fig. 2.5, where the RPN branches out at an intermediate layer to produce regions of high "objectness" and subsequently RoI-pooling takes in these regions for classification. Notably, the proposal stage (RPN) performs classification (foreground/background) and bounding box regression on "anchor" boxes, which are pre-defined boxes of different aspect ratios that are slid over the feature map in a fully convolutional manner. Only the highest scoring (typically ≈ 300) "anchors" are kept and used as region proposals for the next stage.

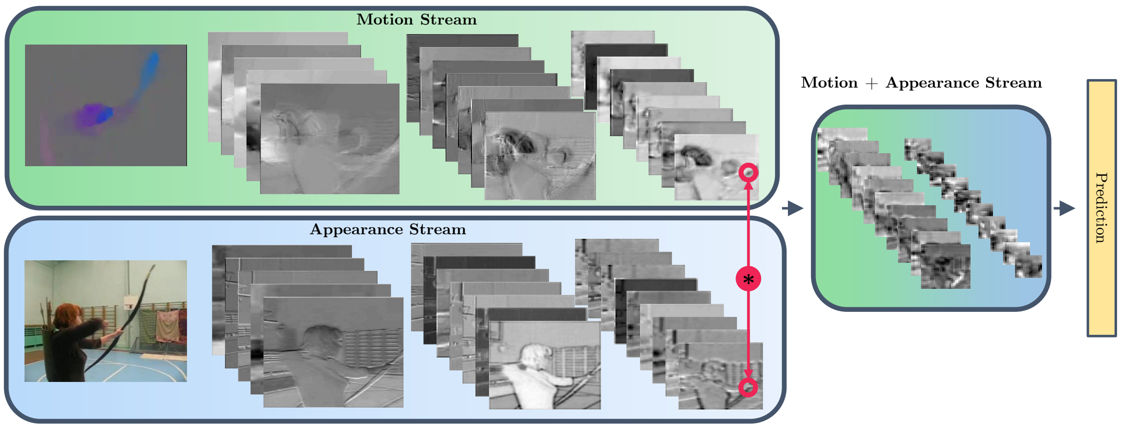
Detection. For the detection stage, the candidate regions are used as input to the RoI pooling operation which operates on an additional convolutional layer with output $\mathbf{x}^t_{cls}$ that is injected (randomly initialized) after the last convolutional layer of the trunk ConvNet (*e.g.* a ResNet (He et al., 2016a) or VGG-16 (Simonyan and Zisserman, 2015)). The position-sensitive RoI pooling operation (Li et al., 2016a) produces a bank of $D_{cls} = k^2(C+1)$ location-sensitive score maps which correspond to a $k \times k$ spatial grid describing relative positions to be used in the RoI pooling operation for each category (C) and background. Applying the softmax function to the outputs leads to a probability distribution p over $C+1$ classes for each RoI coming from the RPN sub-network.

For bounding box regression, a second branch is used. Similar to Faster R-CNN (Ren et al., 2016), a sibling convolutional layer with output $\mathbf{x}^t_{reg}$ after the last convolutional layer performs bounding box regression. However, different than Faster R-CNN, this task is also solved in a fully convolutional manner (no region-wise operations) where again a position-sensitive RoI pooling operation is performed on this bank of $D_{cls} = 4k^2$ maps for class-agnostic bounding box prediction of candidate boxes.

Multi-task loss. For learning the detectors a multi-task loss formulation (Girshick, 2015), consisting of of a combined classification L_{cls} and regression loss L_{reg} is used. For a single iteration and a batch of N RoIs the network predicts softmax probabilities $\{p_i\}_{i=1}^N$ and regression offsets $\{b_i\}_{i=1}^N$. The overall objective function is written as:

$$L(\{p_i\},\{b_i\},\{\Delta_i\}) = \frac{1}{N}\sum_{i=1}^{N} L_{cls}(p_{i,c^*}) + \lambda \frac{1}{N_{fg}} \sum_{i=1}^{N} [c_i^* > 0] L_{reg}(b_i, b_i^*) \tag{2.11}$$

Where the ground truth class label of an RoI is defined by c_i^* and its predicted softmax score is p_{i,c^*} and b_i^* is the ground truth regression target. The indicator function $[c_i^* > 0]$ is 1 for foreground RoIs and 0 for background RoIs (with $c_i^* = 0$). $L_{cls}(p_{i,c^*}) = -\log(p_{i,c^*})$ is the cross-entropy loss for box classification, and L_{reg} is the bounding box regression loss defined as the smooth L1 function in (Girshick, 2015). The tradeoff parameter $\lambda = 1$ is used to balance the different loss terms. The assignment of RoIs to ground truth is as follows: a class label c^* and regression targets b^* are assigned if the RoI overlaps with a ground-truth box at least by 0.5 in intersection-over-union (IoU). Thus, the first term of (2.11) is active for all N boxes in a training batch and the second term is only active for N_{fg} foreground RoIs.



Example outputs of the first three convolutional layers from a two-stream ConvNet model. The two networks separately capture appearance and motion information at a fine temporal scale. In this chapter, we investigate several approaches to fuse the two networks across space and time, to finally come up with a greatly improved Convolutional Two-Stream Fusion architecture.

3 Convolutional Two-Stream Network Fusion

3.1 Motivation

Some actions can be identified from a still image from their appearance alone (*e.g.* archery in the above case). For others, though, individual frames can be ambiguous, and motion cues are necessary. Consider, for example, discriminating walking from running, yawning from laughing, or in swimming, crawl from breast-stroke. The two-stream architecture (Simonyan and Zisserman, 2014a) incorporates motion information by training separate ConvNets for both appearance in still images and stacks of optical flow. Indeed, this work showed that optical flow information alone was sufficient to discriminate most of the actions in UCF101.

Nevertheless, the two-stream architecture (or any previous method) is not able to exploit two very important cues for action recognition in video: (i) recognizing what is moving where, i.e. registering appearance recognition (spatial cue) with optical flow recognition (temporal cue); and (ii) how these cues evolve over time.

Our objective in this chapter is to rectify these limitations by developing an architecture that is able to fuse spatial and temporal cues at several levels of granularity in feature abstraction, and with spatial as well as temporal integration. In particular, Sec. 3.3 investigates three aspects of fusion: (i) in Sec. 3.3.1 *how* to fuse the two networks (spatial and temporal) taking account of *spatial* registration? (ii) in Sec. 3.3.2 *where* to fuse the two networks? And, finally in Sec. 3.3.3 (iii) how to fuse the networks *temporally*? In each of these investigations we select the optimum outcome (Sec. 3.4) and then, putting these results together, propose a novel architecture (Sec. 3.3.4) for spatiotemporal fusion of two stream networks that achieves state of the art performance in Sec. 3.4.5.

We implemented our approach using the MatConvNet toolbox (Vedaldi and Lenc, 2015) and made our code publicly available at `https://github.com/feichtenhofer/twostreamfusion`

3.2 Related work

Several pieces of recent work on using ConvNets for action recognition in temporal sequences have investigated the question of how to go beyond simply using the framewise appearance information, and exploit the temporal information. A natural extension is to stack consecutive video frames and extend 2D ConvNets into time (Ji et al., 2013) so that the first layer learns spatiotemporal features. In (Karpathy et al., 2014) the authors study several approaches for temporal sampling, including early fusion (letting the first layer filters operate over frames as in (Ji et al., 2013)), slow fusion (consecutively increasing the temporal receptive field as the layers increase) and late fusion (merging fully connected layers of two separate networks that operate on temporally distant frames). Their architecture is not particularly sensitive to the temporal modelling, and they achieve similar levels of performance by a purely spatial network, indicating that their model is not gaining much from the motion information.

The recently proposed C3D method (Tran et al., 2015a) learns 3D ConvNets on a limited temporal support of 16 consecutive frames with all filter kernels of size 3×3×3. They report better performance than (Karpathy et al., 2014) by letting all filters operate over space and time. However, their network is considerably deeper than (Ji et al., 2013, Karpathy et al., 2014) with a structure similar to the very deep networks in (Simonyan and Zisserman, 2014b), which should be a basis of better performance on its own. Another way of learning spatiotemporal relationships is proposed in (Sun et al., 2015), where the authors factorize 3D convolution into a 2D spatial and a 1D temporal convolution. Specifically, their temporal convolution is a 2D convolution over time as well as the feature channels and is only performed at higher layers of the network.

The work in (Ng et al., 2015b) compares several temporal feature pooling architectures to combine information across longer time periods. They conclude that temporal pooling of convolutional layers performs better than slow, local, or late pooling, as well as temporal convolution. They also investigate ordered sequence modelling by feeding the ConvNet features into a recurrent network with Long Short-Term Memory (LSTM) cells. Using LSTMs, however, did not give an improvement over temporal pooling of convolutional features.

The most closely related work to ours, and the one we extend here, is the two-stream ConvNet architecture proposed in (Simonyan and Zisserman, 2014a). The method first decomposes video into spatial and temporal components by using RGB and optical flow frames. These components are fed into separate deep ConvNet architectures, to learn spatial as well as temporal information about the appearance and movement of the objects in a scene. Each stream is performing video recognition on its own and for final classification, softmax scores are combined by late fusion. The authors compared several techniques to align the optical flow frames and concluded that simple stacking of $L = 10$ horizontal and vertical flow fields performs best. They also employed multitask learning on UCF101 and HMDB51 to increase the amount of training data and improve the performance on both.

Also related to our work is the bilinear method (Lin et al., 2015), which correlates the output of two ConvNet layers by performing an outer product at each location of the image. The resulting bilinear feature is pooled across all locations into an orderless descriptor. Note that this approach is closely related to second-order pooling (Carreira et al., 2012) of hand-crafted SIFT features.

3.3 Approach

We build upon the the two-stream architecture (Simonyan and Zisserman, 2014a), which has been described in Section 2.3. This architecture has two main drawbacks: (i) it is not able to learn the pixel-wise correspondences between spatial and temporal features (since fusion is only on the classification scores), and (ii) it is limited in temporal scale as the spatial ConvNet operates only on single frames and the temporal ConvNet only on a stack of L temporally adjacent optical flow frames (*i.e.* $L = 10$). The implementation of (Simonyan and Zisserman, 2014a) addressed the latter problem to an extent by temporal pooling across regularly spaced samples in the video, but this does not allow the modelling of temporal evolution of actions.

3.3.1 Spatial fusion

In this section we consider different architectures for fusing the two stream networks. However, the same issues arise when spatially fusing any two networks, so they are not tied to this particular application.

To be clear, our intention in this section is to fuse the two networks (at a particular convolutional layer) such that channel responses at the same pixel position are put in correspondence. To motivate this idea, consider for example discriminating between the actions of brushing teeth and brushing hair. If a hand moves periodically at some spatial location then the temporal network can recognize that motion, and the spatial network can recognize the location (teeth or hair) and their combination then discriminates the action (*e.g.*, as brush teeth vs. brush hair).

This spatial correspondence is easily achieved when the two networks have the same spatial resolution at the layers to be fused, simply by overlaying (stacking) layers from one network on the other (we make this precise below). However, there is also the issue of which *channel* (or channels) in one network *corresponds* to the *channel* (or channels) of the other network.

Suppose for the moment that different channels in the spatial network are responsible for different facial areas (mouth, hair, etc), and one channel in the temporal network is responsible for periodic motion fields of this type. Then, after the channels are stacked, the filters in the subsequent layers must learn the correspondence between these appropriate channels (*e.g.* as weights in a convolution filter) in order to best discriminate between these actions.

To make this more concrete, we now discuss a number of ways of fusing layers between two networks, and for each describe the consequences in terms of correspondence.

A fusion function $f : \mathbf{x}_t^a, \mathbf{x}_t^b, \to \mathbf{y}_t$ fuses two feature maps $\mathbf{x}_t^a \in \mathbb{R}^{H \times W \times C}$ and $\mathbf{x}_t^b \in \mathbb{R}^{H' \times W' \times C'}$, at time t, to produce an output map $\mathbf{y}_t \in \mathbb{R}^{H'' \times W'' \times C''}$, where W, H and C are the width, height and number of channels of the respective feature maps. When applied to feedforward ConvNet architectures, consisting of convolutional, fully-connected, pooling and nonlinearity layers, f can be applied at different points in the network to implement *e.g.* early-fusion, late-fusion or multiple layer fusion. Various fusion functions f can be used. We investigate the following ones in this chapter, and, for simplicity, assume that $H = H' = H''$, $W = W' = W''$, $C = C'$, and also drop the t subscript.

Sum fusion. $\mathbf{y}^{\text{sum}} = f^{\text{sum}}(\mathbf{x}^a, \mathbf{x}^b)$ computes the sum of the two feature maps at the same spatial locations i, j and feature channels d:

$$y_{i,j,d}^{\text{sum}} = x_{i,j,d}^a + x_{i,j,d}^b, \tag{3.1}$$

where $1 \leq i \leq H, 1 \leq j \leq W, 1 \leq d \leq C$ and $\mathbf{x}^a, \mathbf{x}^b, \mathbf{y} \in \mathbb{R}^{H \times W \times C}$

Since the channel numbering is arbitrary, sum fusion simply defines an arbitrary correspondence between the networks. Of course, subsequent learning can employ this arbitrary correspondence to its best effect, optimizing over the filters of each network to make this correspondence useful. Notably, in the backward pass gradients are distributed equally to both inputs.

Max fusion. $\mathbf{y}^{\max} = f^{\max}(\mathbf{x}^a, \mathbf{x}^b)$ similarly takes the maximum of the two feature map:

$$y^{\max}_{i,j,d} = \max\{x^a_{i,j,d}, x^b_{i,j,d}\}, \tag{3.2}$$

where all other variables are defined as above (3.1).

Similarly to sum fusion, the correspondence between network channels is again arbitrary. Notably, here the backward pass acts as a gradient switch that only sends gradients backwards to the highest activating channel of an input.

Concatenation fusion. $\mathbf{y}^{\text{cat}} = f^{\text{cat}}(\mathbf{x}^a, \mathbf{x}^b)$ stacks the two feature maps at the same spatial locations i, j across the feature channels d:

$$y^{\text{cat}}_{i,j,2d} = x^a_{i,j,d} \qquad y^{\text{cat}}_{i,j,2d-1} = x^b_{i,j,d}, \tag{3.3}$$

where $\mathbf{y} \in \mathbb{R}^{H \times W \times 2C}$.

Concatenation does not define a correspondence, but leaves this to subsequent layers to define (by learning suitable filters that weight the layers), as we illustrate next.

Conv fusion. $\mathbf{y}^{\text{conv}} = f^{\text{conv}}(\mathbf{x}^a, \mathbf{x}^b)$ first stacks the two feature maps at the same spatial locations i, j across the feature channels d as above (3.3) and subsequently convolves the stacked data with a bank of filters $\mathbf{w} \in \mathbb{R}^{1 \times 1 \times 2C \times C}$ and biases $b \in \mathbb{R}^C$

$$\mathbf{y}^{\text{conv}} = \mathbf{y}^{\text{cat}} * \mathbf{w} + b, \tag{3.4}$$

where the number of output channels is C, and the filter has dimensions $1 \times 1 \times 2C$. Here, the filter $\mathbf{w}$ is used to reduce the dimensionality by a factor of two and is able to model weighted combinations of the two feature maps $\mathbf{x}^a, \mathbf{x}^b$ at the same spatial (pixel) location. When used as a trainable filter kernel in the network, $\mathbf{w}$ is able to *learn* correspondences of the two feature maps that minimize a joint loss function. For example, if $\mathbf{w}$ is learnt to be the concatenation of two permuted identity matrices $\mathbf{1}' \in \mathbb{R}^{1 \times 1 \times C \times C}$, then the ith channel of the one network is only combined with the ith channel of the other (via summation).

Note that if there is no dimensionality reducing conv-layer injected after concatenation, the number of input channels of the upcoming layer is $2C$.

Bilinear fusion. $\mathbf{y}^{\text{bil}} = f^{\text{bil}}(\mathbf{x}^a, \mathbf{x}^b)$ computes a matrix outer product of the two features at each pixel location, followed by a summation over the locations:

$$\mathbf{y}^{\text{bil}} = \sum_{i=1}^{H} \sum_{j=1}^{W} \mathbf{x}^{a\top}_{i,j} \mathbf{x}^b_{i,j}. \tag{3.5}$$

The resulting feature $\mathbf{y}^{\text{bil}} \in \mathbb{R}^{C^2}$ captures multiplicative interactions at corresponding spatial locations. The main drawback of this feature is its high dimensionality. To make bilinear features usable in practice, they are usually applied at ReLU5, the fully-connected layers are removed (Lin et al., 2015) and power- and $L2$-normalisation is applied for effective classification with linear

SVMs.

The advantage of bilinear fusion is that every channel of one network is combined (as a product) with every channel of the other network. However, the disadvantage is that all spatial information is marginalized out after this point.

Discussion: These operations illustrate a range of possible fusion methods. Others could be considered, for example: taking the pixel wise product of channels (instead of their sum or max), or the (factorized) outer product without sum pooling across locations (Oh et al., 2015). In Chapter 5 we will introduce a method to multiplicatively fuse into the bottleneck unit of a two-stream ResNet for improved fusion of such an architecture.

Injecting fusion layers can have significant impact on the number of parameters and layers in a two-stream network, especially if only the network which is fused into is kept and the other network tower is truncated, as illustrated in Fig. 3.1 (left). Table 3.1 shows how the number of layers and parameters are affected by different fusion methods for the case of two VGG-M-2048 models (used in (Simonyan and Zisserman, 2014a)) containing five convolution layers followed by three fully-connected layers each. Max-, Sum and Conv-fusion at ReLU5 (after the last convolutional layer) removes nearly *half* of the parameters in the architecture as only one tower of fully-connected layers is used after fusion. Conv fusion has slightly more parameters (97.58M) compared to sum and max fusion (97.31M) due to the additional filter that is used for channel-wise fusion and dimensionality reduction. Many more parameters are involved in concatenation fusion, which does not include dimensionality reduction after fusion and therefore doubles the number of parameters in the first fully connected layer. In comparison, sum-fusion at the softmax layer requires all layers (16) and parameters (181.4M) of the two towers.

In the experimental section of this chapter, (Sec. 3.4.1), we evaluate and compare the performance of each of these possible fusion methods in terms of their classification accuracy.

3.3.2 Where to fuse the networks

As noted above, fusion can be applied at any point in the two networks, with the only constraint being that the two input maps $\mathbf{x}_t^a \in \mathbb{R}^{H\times W\times C}$ and $\mathbf{x}_t^b \in \mathbb{R}^{H'\times W'\times C}$, at time t, have the same spatial dimensions; *i.e.* $H = H'$, $W = W'$. This constraint can be achieved by using an "upconvolutional" layer (Zeiler and Fergus, 2013), or if the dimensions are similar, upsampling can be achieved by padding the smaller map with zeros.

Table 3.2 compares the number of parameters for fusion at different layers in the two networks for the case of a VGG-M model. Fusing after different conv-layers has roughly the same impact on the number of parameters, as most of these are stored in the fully-connected layers. Two networks can also be fused at two layers, as illustrated in Fig. 3.1 (right). This arrangement achieves the original objective of pixel-wise registration of the channels from each network (at conv5), but does not lead to a reduction in the number of parameters (by half if fused only at conv5, for example). In the experimental section (Sec. 3.4.2) we evaluate and compare both the performance of fusing at different levels, and fusing at multiple layers simultaneously.

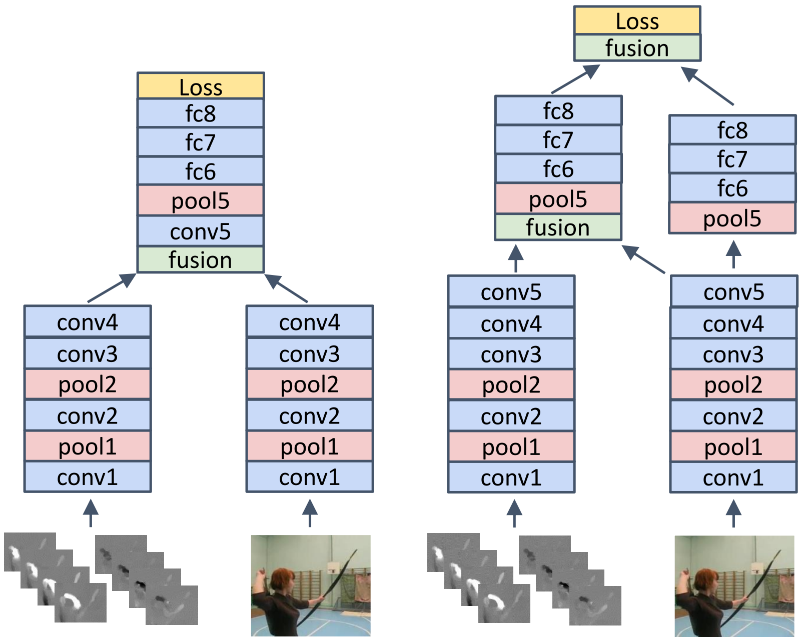


Figure 3.1: Two examples of where a fusion layer can be placed. The left example shows fusion after the fourth conv-layer. Only a single network tower is used from the point of fusion. The right figure shows fusion at two layers (after conv5 and after fc8) where both network towers are kept, one as a hybrid spatiotemporal net and one as a purely spatial network.

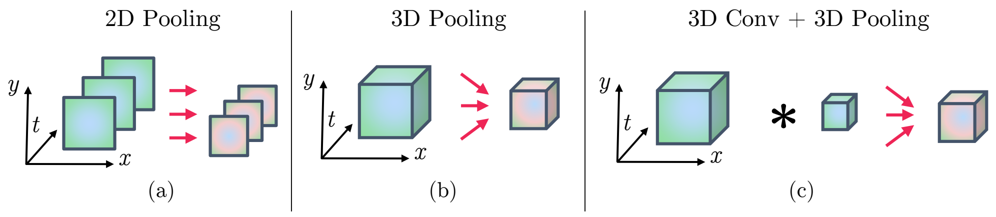


Figure 3.2: Different ways of fusing temporal information. (a) 2D pooling ignores time and simply pools over spatial neighbourhoods to individually shrink the size of the feature maps for each temporal sample. (b) 3D pooling pools from local spatiotemporal neighbourhoods by first stacking the feature maps across time and then shrinking this spatiotemporal cube. (c) 3D conv + 3D pooling additionally performs a convolution with a fusion kernel that spans the feature channels, space and time before 3D pooling.

3.3.3 Temporal fusion

We now consider techniques to combine feature maps $\mathbf{x}_t$ over time t, to produce an output map $\mathbf{y}_t$. One way of processing temporal inputs is by averaging the network predictions over time (as used in (Simonyan and Zisserman, 2014a)). In that case, the architecture only pools in 2D (xy); see Fig. 3.2(a). Specifically, the input of a temporal pooling layer receives feature maps $\mathbf{x} \in \mathbb{R}^{H \times W \times T \times C}$ that are generated by stacking spatial maps across time $t = 1 \ldots T$. Different temporal inputs can simply be fused by averaging the networks' predictions – this is the method adopted by (Simonyan and Zisserman, 2014a).

This method is compared to simple 2D pooling in figure 3.2. Note that both 2D and 3D pooling treat feature channels individually; *i.e.* there is no pooling over feature maps involved. Now consider the input of a temporal pooling layer as feature maps $\mathbf{x} \in \mathbb{R}^{H \times W \times T \times C}$ that are generated by stacking spatial maps across time $t = 1 \dots T$. The following alternatives will be discussed here.

3D Pooling: applies max-pooling to the stacked data within a 3D pooling cube of size $W' \times H' \times T'$. This is a straightforward extension of 2D pooling to the temporal domain, as illustrated in Fig. 3.2(b). For example, if three temporal samples are pooled, then a $3 \times 3 \times 3$ max pooling could be used across the three stacked corresponding channels. Note, there is no pooling *across* different channels.

3D Conv + Pooling: first convolves the four dimensional input $\mathbf{x}$ with a bank of C' filters $\mathbf{w} \in \mathbb{R}^{W'' \times H'' \times T'' \times C \times C'}$ and biases $b \in \mathbb{R}^{C}$

$$\mathbf{y} = \mathbf{x}_t * \mathbf{w} + b, \tag{3.6}$$

as *e.g.* in (Tran et al., 2015a), followed by 3D pooling as described above. This method is illustrated in Fig. 3.2(c). The filters $\mathbf{w}$ are able to model weighted combinations of the features in a local spatio-temporal neighborhood using kernels of size $H'' \times W'' \times T'' \times C$. Typically, the neighborhood is $3 \times 3 \times 3$ (spatial $\times$ spatial $\times$ temporal).

Discussion. The authors of (Ng et al., 2015b) evaluate several additional methods to combine two-stream ConvNets over time. They find temporal max-pooling of convolutional layers among the top performers. We generalize max-pooling here to 3D pooling that provides invariance to small changes of the features' position over time. Further, 3D conv allows spatio-temporal filters to be learnt (Taylor et al., 2010, Tran et al., 2015a). For example, the filter could learn to center weight the central temporal sample, or to differentiate in time or space.

To give an illustrative example that combines convolutional spatial fusion (Section 3.3.1), at the last convolutional layer (Section 3.3.2) with 3D temporal fusion (Section 3.3.3), we show a corresponding architecture for a training sample in Fig. 3.3. The sequence in this case is baseball pitch with a duration of several seconds. Theoretically, the exemplified architecture is able to capture the temporal relations of this action in spacetime both locally and globally and has an conceptual advantage over the original two-stream architecture.

3.3.4 Proposed architecture

We now bring together the ideas from the previous sections to propose a new spatio-temporal fusion architecture and motivate our choices based on our empirical evaluation in Sec. 3.4. The choice of the spatial fusion method, layer and temporal fusion is based on the experiments in sections 3.4.1, 3.4.2 and 3.4.4, respectively.

Our proposed architecture (shown in Fig. 3.4) can be viewed as an extension of the architecture in Fig. 3.1 (right) over time. We fuse the two networks, at the last convolutional layer (after ReLU)

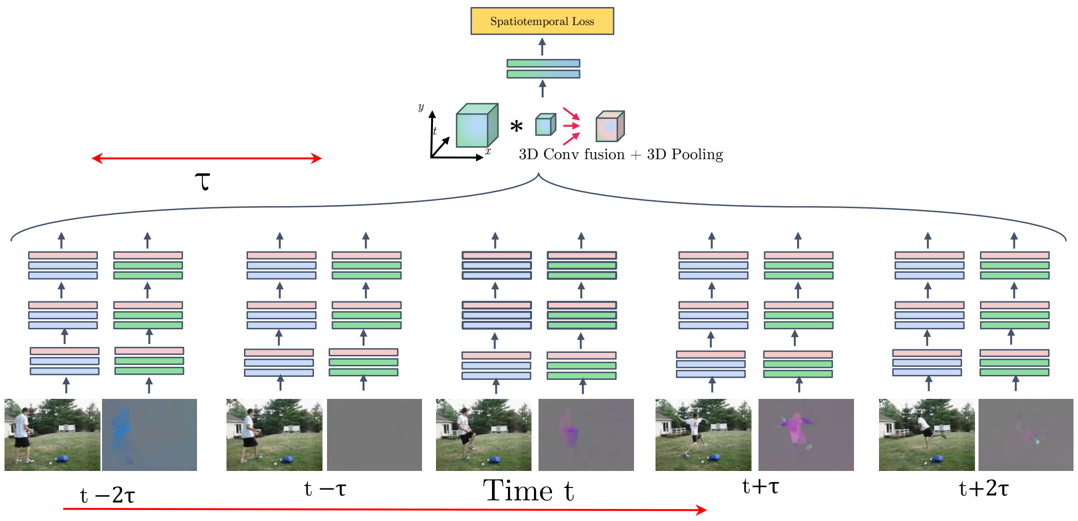


Figure 3.3: Training procedure for a 3D fusion architecture. A training sequence (*e.g.* shown for the action baseball pitch) is divided into T temporal chunks with temporal stride τ between them. The temporal chunks capture short-term information at a fine temporal scale and the fusion layer puts it into context with temporally adjacent chunks, thus operating at a coarse temporal scale $(t + T\tau)$. The fusion filter is also able to learn correspondences between highly abstract features of the spatial stream (blue) and temporal stream (green). After fusion, 3D pooling gathers the resulting features and the spatiotemporal loss is evoked for supervision.

into the spatial stream to convert it into a spatiotemporal stream by using 3D Conv fusion followed by 3D pooling (see Fig. 3.4, left). Moreover, we *do not truncate* the temporal stream and also perform 3D Pooling in the temporal network (see Fig. 3.4, right). The losses of both streams are used for training and during testing we average the predictions of the two streams. In our empirical evaluation (Sec. 3.4.5) we show that keeping both streams performs slightly better than truncating the temporal stream after fusion.

Having discussed how to fuse networks over time, we discuss here the issue of how often to sample the temporal sequence. The temporal fusion layer receives T temporal chunks that are τ frames apart; *i.e.* the two stream towers are applied to the input video at time $t, t + \tau, \dots t + T\tau$. As shown in Fig. 3.4 this enables us to capture short scale $(t \pm \frac{L}{2})$ temporal features at the input of the temporal network (*e.g.* the drawing of an arrow) and put them into context over a longer temporal scale $(t + T\tau)$ at a higher layer of the network (*e.g.* drawing an arrow, bending a bow, and shooting an arrow).

Since the optical flow stream has a temporal receptive field of $L = 10$ frames, the architecture operates on a total temporal receptive field of $T \times L$. Note that $\tau < L$ results in overlapping inputs for the temporal stream, whereas $\tau \geq L$ produces temporally non-overlapping features.

After fusion, we let the 3D pooling operate on T spatial feature maps that are τ frames apart. As features may change their spatial position over time, combining spatial and temporal pooling to 3D

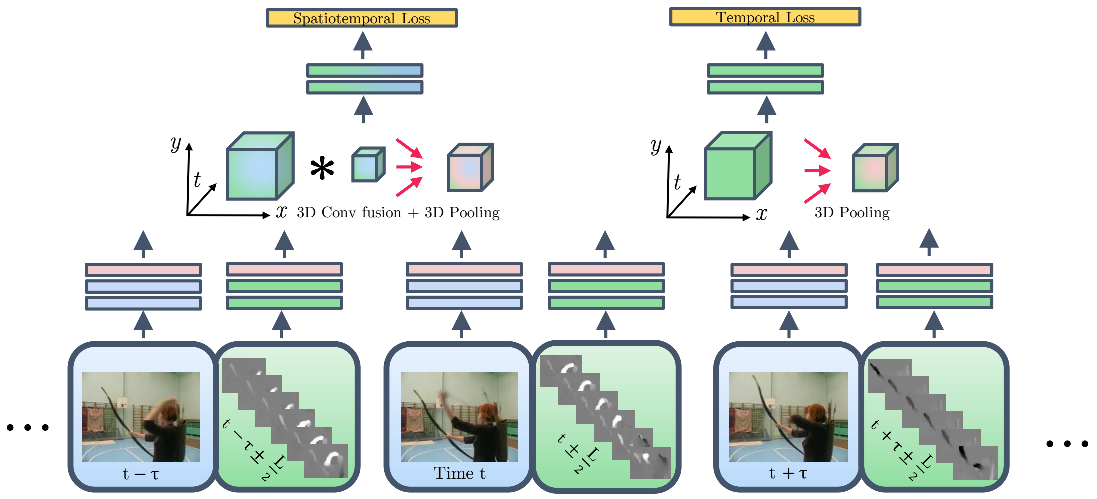


Figure 3.4: Our spatiotemporal fusion ConvNet applies two-stream ConvNets, that capture short-term information at a fine temporal scale ($t\pm\frac{L}{2}$), to temporally adjacent inputs at a coarse temporal scale ($t+T\tau$). The two streams are fused by a 3D filter that is able to learn correspondences between highly abstract features of the spatial stream (blue) and temporal stream (green), as well as local weighted combinations in x, y, t. The resulting features from the fusion stream and the temporal stream are 3D-pooled in space and time to learn spatiotemporal (top left) and purely temporal (top right) features for recognising the input video.

pooling makes sense. For example, the output of a VGG-M network at conv5 has an input stride of 16 pixels and captures high level features from a receptive field of 139×139 pixels. Spatiotemporal pooling of conv5 maps that are τ frames distant in time can therefore capture features of the same object, even if they slightly move.

3.3.5 Implementation details

Two-Stream architecture. We employ two pre-trained ImageNet models. First, for sake of comparison to the original two-stream approach (Simonyan and Zisserman, 2014a), the VGG-M-2048 model (Chatfield et al., 2014b) with 5 convolutional and 3 fully-connected layers. Second, the very deep VGG-16 model (Simonyan and Zisserman, 2014b) that has 13 convolutional and 3 fully-connected layers. We first separately train the two streams as described in (Simonyan and Zisserman, 2014a), but with some subtle differences: We do not use RGB colour jittering; Instead of decreasing the learning rate according to a fixed schedule, we lower it after the validation error saturates; For training the spatial network we use lower dropout ratios of 0.85 for the first two fully-connected layers. Even lower dropout ratios (up to 0.5) did not decrease performance significantly.

For the temporal net, we use optical flow, (Brox et al., 2004) for VGG-M architectures and (Zach et al., 2007) for VGG-16 models, by stacking with $L = 10$ frames (Simonyan and Zisserman, 2014a). We also initialised the temporal net with a model pre-trained on ImageNet, since this generally facilitates training speed without a decrease in performance compared to our model trained from scratch. The network input is rescaled beforehand, so that the smallest side of the frame equals 256. We also pre-compute the optical flow before training and store the flow fields as JPEG images (with clipping of displacement vectors larger than 20 pixels). We do not use batch normalization (Ioffe and Szegedy, 2015).

Two-Stream ConvNet fusion. For fusion, these networks are finetuned, with a batch size of 96 and a learning rate starting from 10^{-3}, which is reduced by a factor of 10 as soon as the validation accuracy saturates. We only propagate back to the injected fusion layer, since full backpropagation did not result in an improvement. In our experiments we only fuse between layers with the same output resolution; except for fusing a VGG-16 model at ReLU5_3 with a VGG-M model at ReLU5, where we pad the slightly smaller output of VGG-M (13×13, compared to 14×14) with a row and a column of zeros. For Conv fusion, we found that careful initialisation of the injected fusion layer (as in (3.4)) is very important. We compared several methods and found that initialisation by identity matrices (to sum the two networks) performs as well as random initialisation.

Spatiotemporal architecture. For our final architecture described in Sec. 3.3.4, the 3D Conv fusion kernel $\mathbf{w}$ has dimension $3 \times 3 \times 3 \times 1024 \times 512$ and $T = 5$, i.e. the spatio-temporal filter has dimension $H'' \times W'' \times T'' = 3 \times 3 \times 3$, the $C = 1024$ results from concatenating the ReLU5 from the spatial and temporal streams, and the $C' = 512$ matches the number of input channels of the following FC6 layer.

The 3D Conv filters are also initialised by stacking two identity matrices for mapping the 1024 feature channels to 512. Since the activations of the temporal ConvNet at the last convolutional layer are roughly 3 times lower than its appearance counterpart, we initialise the temporal identity matrix of $\mathbf{w}$ by a factor of 3 higher. The spatiotemporal part of $\mathbf{w}$ is initialised using a Gaussian of size $3 \times 3 \times 3$ and $\sigma = \mathbf{1}$. Further, we do not fuse at the prediction layer during training, as this would bias the loss towards the temporal architecture, because the spatiotemporal architecture requires longer to adapt to the fused features.

Training 3D ConvNets is even more prone to overfitting than the two-stream ConvNet fusion, and requires additional augmentation, as follows. During finetuning, at each training iteration we sample the $T = 5$ frames from each of the 96 videos in a batch by randomly sampling the starting frame, and then randomly sampling the temporal stride $(\tau) \in [1, 10]$ (so operating over a total of between 15 and 50 frames). Instead of cropping a fixed sized 224×224 input patch, we randomly jitter its width and height by $\pm 25\%$ and rescale it to 224×224. The rescaling is chosen randomly and may change the aspect-ratio. Patches are only cropped at a maximum of 25% distance from the image borders (relative to the width and height). Note, the position (and size, scale, horizontal flipping) of the crop is randomly selected in the first frame (of a multiple-frame-stack) and then the same spatial crop is applied to all frames in the stack.

Testing. Unless otherwise specified, only the $T = 5$ frames (and their horizontal flips) are sampled, compared to the 25 frames in (Simonyan and Zisserman, 2014a), to foster fast empirical evaluation. In addition we employ fully convolutional testing where the entire frame is used (rather than spatial crops).

3.4 Evaluation

We evaluate our presented approaches on two challenging and popular used action recognition datasets. First, we consider UCF101 (Khurram Soomro and Shah, 2012), which consists of 13320 videos showing 101 action classes. It provides large diversity in terms of actions, variations in background, illumination, camera motion and viewpoint, as well as object appearance, scale and pose. Second, we consider HMDB51 (Kuehne et al., 2011), which has 6766 videos that show 51 different actions and generally is considered more challenging than UCF0101 due to the even wider variations in which actions occur (higher intra-class variations lead to larger errors during training and testing). In our experiments we only use the authors' original (non-stabilized) versions of the videos. For both datasets, we use the provided evaluation protocol and report mean average accuracy over three splits into training and test sets when comparing against state of the art methods from the literature, while for ablation studies we report the performance on the first split of each dataset.

3.4.1 How to fuse the two streams spatially?

For these experiments we use the same network architecture as in (Simonyan and Zisserman, 2014a); i.e. two VGG-M-2048 nets (Chatfield et al., 2014b). The fusion layer is injected at the last convolutional layer, after rectification, *i.e.* its input is the output of ReLU5 from the two streams. This choice is made because, in preliminary experiments, it provided better results than alternatives such as the non-rectified output of conv5. At that point the features are already highly informative while still providing coarse location information. After the fusion layer a single processing stream is used.

We compare different fusion strategies in Table 3.1 where we report the average accuracy on the first split of UCF101. We first observe that our performance for softmax averaging (85.94%) compares favourably to the one reported in (Simonyan and Zisserman, 2014a). Second we see that Max and Concatenation perform considerably lower than Sum and Conv fusion. Conv fusion performs best and is slightly better than Bilinear fusion and simple fusion via summation. For the reported Conv-fusion result, the convolution kernel $\mathbf{w}$ is initialised by identity matrices that perform summation of the two feature maps. Initialisation via random Gaussian noise ends up at a similar performance 85.59% compared to identity matrices (85.96%), however, at a much longer training time (by factor of 8). This is interesting, since this, as well as the high result of Sum-fusion, suggest that *simply summing the feature maps* is already a good fusion technique and learning a randomly initialised combination does not lead to significantly different/better results.

For all the fusion methods shown in Table 3.1, fusion at FC layers results in lower performance compared to ReLU5, with the ordering of the methods being the same as in Table 3.1, except for bilinear fusion which is not possible at FC layers. Among all FC layers, FC8 performs better than FC7 and FC6, with Conv fusion at 85.9%, followed by Sum fusion at 85.1%. We think the reason for ReLU5 performing slightly better is that at this layer spatial correspondences between appearance

Fusion Method	Fusion Layer	Acc.	#layers	#parameters
Sum (Simonyan and Zisserman, 2014a)	Softmax	85.6%	16	181.42M
Sum (ours)	Softmax	85.94%	16	181.42M
Max	ReLU5	82.70%	13	97.31M
Concatenation	ReLU5	83.53%	13	172.81M
Bilinear (Lin et al., 2015)	ReLU5	85.05%	10	6.61M+SVM
Sum	ReLU5	85.20%	13	97.31M
Conv	ReLU5	85.96%	14	97.58M

Table 3.1: Performance comparison of different spatial fusion strategies (Sec. 3.3.1) on UCF101 (split 1). Sum fusion at the softmax layer corresponds to averaging the two networks predictions and therefore includes the parameters of both 8-layer VGG-M models. Performing fusion at ReLU5 using Conv or Sum fusion does not significantly lower classification accuracy. Moreover, this requires only half of the parameters in the softmax fusion network. Concatenation has lower performance and requires twice as many parameters in the FC6 layer (as Conv or Sum fusion). Only the bilinear combination enjoys much fewer parameters as there are no FC layers involved; however, it has to employ an SVM to perform comparably.

Fusion Layers	Accuracy	#layers	#parameters
ReLU2	82.25%	11	91.90M
ReLU3	83.43%	12	93.08M
ReLU4	82.55%	13	95.48M
ReLU5	85.96%	14	97.57M
ReLU5 + FC8	86.04%	17	181,68M
ReLU3 + ReLU5 + FC6	81.55%	17	190,06M

Table 3.2: Performance comparison for Conv fusion (3.4) at different fusion layers. An earlier fusion (than after conv5) results in weaker performance. Multiple fusions also lower performance if early layers are incorporated (last row). Best performance is achieved for fusing at ReLU5 or at ReLU5+FC8 (but with nearly double the parameters involved).

and motion are fused, which would have already been collapsed at the FC layers (Mahendran and Vedaldi, 2015).

3.4.2 Where to fuse the two streams spatially?

Fusion from different layers is compared in Table 3.2. Conv fusion is used and the fusion layers are initialised by an identity matrix that sums the activations from previous layers. Interestingly, fusing and truncating one net at ReLU5 achieves around the same classification accuracy on the first split of UCF101 (85.96% vs 86.04%) as an additional fusion at the prediction layer (FC8), but at a much lower number of total parameters (97.57M vs 181.68M). Fig. 3.1 shows how these two examples are implemented.

	UCF101 (split 1)		HMDB51 (split 1)	
Model	VGG-M-2048	VGG-16	VGG-M-2048	VGG-16
Spatial	74.22%	82.61%	36.77%	47.06%
Temporal	82.34%	86.25%	51.50%	55.23%
Late Fusion	85.94%	90.62%	54.90%	58.17%

Table 3.3: Performance comparison of deep (VGG-M-2048) vs. very deep (VGG-16) Two-Stream ConvNets on the UCF101 (split1) and HMDB51 (split1). Late fusion is implemented by averaging the prediction layer outputs. Using deeper networks boosts performance at the cost of computation time.

Fusion Method	Pooling	Fusion Layers	UCF101	HMDB51
2D Conv	2D	ReLU5 +	89.35%	56.93%
2D Conv	3D	ReLU5 +	89.64%	57.58%
3D Conv	3D	ReLU5 +	90.40%	58.63%

Table 3.4: Spatiotemporal two-stream fusion on UCF101 (split1) and HMDB51 (split1). The models used are VGG-16 (spatial net) and VGG-M (temporal net). The "+" after a fusion layer indicates that both networks and their loss are kept after fusing, as this performs better than truncating one network. Specifically, at ReLU5 we fuse from the temporal net into the spatial network, then perform either 2D or 3D pooling at Pool5 and compute a loss for each tower. During testing, we average the FC8 predictions for both towers.

3.4.3 Going from deep to very deep models

For computational complexity reasons, all previous experiments were performed with two VGG-M-2048 networks (as in (Simonyan and Zisserman, 2014a)). Using deeper models, such as the very deep networks in (Simonyan and Zisserman, 2014b) can, however, lead to even better performance in image recognition tasks (Cimpoi et al., 2015, Lin et al., 2015, Szegedy et al., 2015a). Following that line, we train a 16 layer network, VGG-16, (Simonyan and Zisserman, 2014b) on UCF101 and HMDB51. All models are pretrained on ImageNet and separately trained for the target dataset, except for the temporal HMDB51 networks, which are initialised from the temporal UCF101 models. For VGG-16, we use TV-L1 optical flow (Zach et al., 2007) and apply a similar augmentation technique as for 3D ConvNet training (described in Sec. 3.3.5) that samples from the image corners and its centre. The learning rate is set to 50^{-4} and decreased by a factor of 10 as soon as the validation objective saturates.

The comparison between deep and very deep models is shown in Table 3.3. On both datasets, one observes that going to a deeper spatial model boosts performance significantly (8.11% and 10.29%), whereas a deeper temporal network yields a lower accuracy gain (3.91% and 3.73%).

3.4.4 How to fuse the two streams temporally?

Different temporal fusion strategies are shown in Table 3.4. Conv fusion is again used and the fusion layers are initialized by an identity matrix that sums the activations from previous layers. In the first row of Table 3.4 we observe that conv fusion performs better than averaging the softmax output (*cf.* Table 3.3). Next, we find that applying 3D pooling instead of using 2D pooling after the

Method	UCF101	HMDB51
Spatiotemporal ConvNet (Karpathy et al., 2014)	65.4%	-
LRCN (Donahue et al., 2015)	82.9%	-
Composite LSTM Model (Srivastava et al., 2015)	84.3%	44.0
C3D (Tran et al., 2015a)	85.2%	-
Two-Stream ConvNet (VGG-M) (Simonyan and Zisserman, 2014a)	88.0%	59.4%
Factorized ConvNet (Sun et al., 2015)	88.1%	59.1%
Two-Stream Conv Pooling (Ng et al., 2015b)	88.2%	-
Two-Stream ConvNet (VGG-16, ours)	91.7%	58.7%
Ours (S:VGG-16, T:VGG-M)	90.8%	62.1%
Ours (S:VGG-16, T:VGG-16, single tower after fusion)	91.8%	64.6%
Ours (S:VGG-16, T:VGG-16)	92.5%	65.4%

Table 3.5: Mean classification accuracy of best performing ConvNet approaches over three train/test splits on HMDB51 and UCF101. For our method we list the models used for the spatial (S) and temporal (T) stream.

IDT+higher dimensional FV (Peng et al., 2014)	87.9%	61.1%
C3D+IDT (Tran et al., 2015a)	90.4%	-
TDD+IDT (Wang et al., 2015b)	91.5%	65.9%
Ours+IDT (S:VGG-16, T:VGG-M)	92.5%	67.3%
Ours+IDT (S:VGG-16, T:VGG-16)	93.5%	69.2%

Table 3.6: Mean classification accuracy on HMDB51 and UCF101 for approaches that use IDT features (Wang and Schmid, 2013).

fusion layer increases performance on both datasets, with larger gains on HMDB51. Finally, the last row of Table 3.4 lists results for applying a 3D filter for fusion which further boosts recognition rates.

3.4.5 Comparison with the previous state-of-the-art

Finally, we compare against the previous state-of-the-art over all three splits of UCF101 and HMDB51 in Table 3.5. We use the same method as shown above, *i.e.* fusion by 3D Conv and 3D Pooling (illustrated in Fig. 3.4). For testing we average 20 temporal predictions from each network by densely sampling the input-frame-stacks and their horizontal flips.

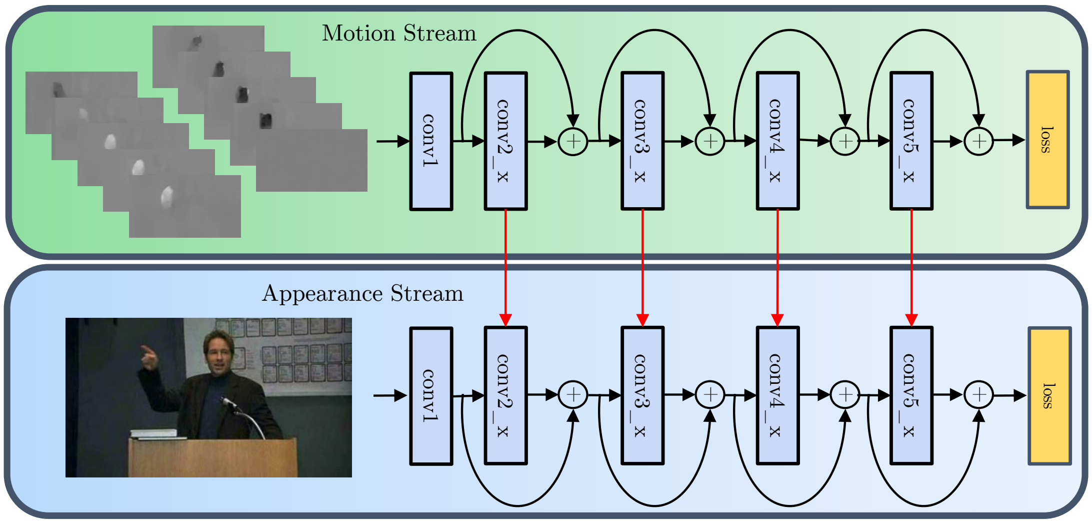
One interesting comparison is to the original two-stream approach (Simonyan and Zisserman, 2014a), we improve by 3% on UCF101 and HMDB51 by using a VGG-16 spatial (S) network and a VGG-M temporal (T) model, as well as by 4.5% (UCF) and 6% (HMDB) when using VGG-16 for both streams.

Another interesting comparison is against the two-stream network in (Ng et al., 2015b), which employs temporal conv-pooling after the last dimensionality reduction layer of a GoogLeNet (Szegedy et al., 2015a) architecture. They report 88.2% on UCF101 when pooling over 120 frames and 88.6% when using an LSTM for pooling. Here, our result of 92.5% clearly underlines the importance of our proposed approach. Note also that using a single stream after temporal fusion achieves 91.8%, compared to maintaining two streams and achieving 92.5%, but with far fewer parameters and a simpler architecture.

As a final experiment, we explore what benefit results from a late fusion of hand-crafted IDT features (Wang and Schmid, 2013) with our representation. We simply average the SVM scores of the FV-encoded IDT descriptors (*i.e.* HOG, HOF, MBH) with the predictions (taken before softmax) of our ConvNet representations. The resulting performance is shown in Table 3.6. We achieve 93.5% on UCF101 and 69.2% HMDB51. This state-of-the-art result illustrates that there is still a degree of complementary between hand-crafted representations and our end-to-end learned ConvNet approach.

3.5 Summary

In this chapter have discussed several variants for fusing two-stream networks both spatially and over time. We have proposed a new spatiotemporal architecture for two stream networks with a novel convolutional fusion layer between the networks, and a novel temporal fusion layer (incorporating 3D convolutions and pooling). The new architecture does not increase the number of parameters significantly over previous methods, yet exceeds the state of the art on two standard benchmark datasets. Our results suggest the importance of learning correspondences between highly abstract ConvNet features both spatially and temporally. One intriguing finding is that there is still such an improvement by combining ConvNet predictions with FV-encoded IDT features. We suspect that this difference may vanish in time given far more training data, but otherwise it certainly indicates where future research should attend. In the next chapter we will build on our insights for connecting motion and appearance features spatially and also on how to aggregate long-term information temporally.



Residual connections between the appearance and motion pathways of a two-stream architecture allow spatiotemporal feature learning. Temporal residual connections can be established by transforming pretrained image ConvNets into spatiotemporal networks by equipping them with learnable residual connections operating on adjacent feature maps in time. In this chapter, we introduce Spatiotemporal Residual Networks as a combination of two-stream ConvNets and ResNets.

4 Spatiotemporal Residual Networks

4.1 Motivation

Having the insights on how to fuse the two-stream ConvNets both spatially and temporally in mind, we now switch to a more general architecture for deep video recognition. Since the introduction of the “AlexNet” architecture (Krizhevsky et al., 2012a) in the 2012 ImageNet competition, ConvNets have dominated state-of-the-art performance across a variety of computer vision tasks, including object-detection (Girshick et al., 2014), image segmentation (Long et al., 2015), image classification (Simonyan and Zisserman, 2014b, Szegedy et al., 2015a), face recognition (Schroff et al., 2015) and human pose estimation (Tompson et al., 2015). In conjunction with these advances as well as the evolution of network architectures, *e.g.* from AlexNet (Krizhevsky et al., 2012a) through ZF-net (Zeiler and Fergus, 2013), Inception (Szegedy et al., 2015a, 2016), VGG (Simonyan and Zisserman, 2014b) and Residual nets (He et al., 2016a,b), several design best practices have emerged (He et al., 2016a, Simonyan and Zisserman, 2014b, Szegedy et al., 2015b, 2016). First, information bottlenecks should be avoided and the representation size should gently decrease from the input to the output as the number of feature channels increases with the depth of the network. Second, the receptive field at the end of the network should be large enough that the processing units can base operations on larger regions of the input. This functionality can be achieved by stacking many small filters or using large filters in the network; notably, the first choice can be implemented with fewer operations (faster, fewer parameters) and also allows inclusion of more nonlinearities. As a consequence deep networks with small filters outperform shallower nets with larger filters (Simonyan and Zisserman, 2014b). Third, dimensionality reduction (1×1 convolutions) before spatially aggregating filters (*e.g.* 3×3) is supported by the fact that outputs of neighbouring filters are highly correlated and therefore these activations can be reduced before aggregation (Szegedy et al., 2015b). Fourth, spatial factorization into asymmetric filters can even further reduce computational cost and ease the learning problem. Fifth, it is important to normalize the responses of each feature channel within a batch to reduce internal covariate shift; *e.g.* with batch-normalization (Ioffe and Szegedy, 2015). The last architectural guideline is to use residual connections to facilitate training of very deep models that are essential for good performance (He et al., 2016a). We carry over these good practices for designing ConvNets in the image domain to the video domain by converting the 1×1 convolutional dimensionality mapping filters in ResNets to temporal filters. By stacking several of these transformed temporal filters throughout the network we provide a large receptive field for the discriminative units at the end of the network. Further, this design allows us to convert spatial ConvNets into spatiotemporal models and thereby exploits the large amount of training data from image datasets such as ImageNet.

Also in this chapter, we will build on the two-stream approach (Simonyan and Zisserman, 2014a) that employs two separate ConvNet streams, a spatial *appearance* stream, which achieves state-of-the-art action recognition from RGB images and a temporal *motion* stream, which operates on optical flow information. The two-stream architecture is inspired by the two-stream hypothesis from neuroscience (Goodale and Milner, 1992) that postulates two pathways in the visual cortex: The ventral pathway, which responds to spatial features such as shape or colour of objects, and the dorsal pathway, which is sensitive to object transformations and their spatial relationship, as *e.g.* caused by motion. Notably, as also motivated in Section Section 2.3 further research in neuroscience suggests that motion information is distributed into separate visual areas (Born and Tootell, 1992, Van Essen and Gallant, 1994). As shown in the previous chapter, such spatial fusion of appearance and motion cues can be beneficial. In this chapter, we reconsider our findings from the previous one and build on the strengths of the additive fusion (see 3.1 and accompanying discussions) as well as on increasing the temporal receptive field by strided input sampling and convolutions across time.

In particular, this chapter extends the combination of two-stream ConvNets in the following ways. First, motivated by the recent success of residual networks (ResNets) (He et al., 2016a) for numerous challenging recognition tasks on datasets such as ImageNet and MS COCO, we apply ResNets to the task of human action recognition in videos. Here, we initialize our model with pre-trained ResNets for image categorization (He et al., 2016a) to leverage a large amount of image-based training data for the action recognition task in video. Second, we demonstrate that injecting residual connections between the two streams and jointly fine-tuning the resulting model achieves improved performance over the two-stream architecture. Third, we overcome limited temporal receptive field size in the original two-stream approach by extending the model over time. We convert convolutional dimensionality mapping filters to temporal filters that provides the network with *learnable* residual connections over time. By stacking several of these temporal filters and sampling the input sequence at large temporal strides (*i.e.* skipping frames), we enable the network to operate over large temporal extents of the input. To demonstrate the benefits of our proposed spatiotemporal ResNet architecture, it has been evaluated on two standard action recognition benchmarks where it greatly boosts the previous state-of-the-art.

4.2 Related work

Approaches for action recognition in video can largely be divided into two categories: Those that use hand-crafted features with decoupled classifiers and those that jointly learn features and classifier. Our work is related to the latter, which is outlined in the following.

Several approaches have been presented for spatiotemporal feature learning. Unsupervised learning techniques have been applied by stacking ISA or convolutional gated RBMs to learn spatiotemporal features for action recognition (Le et al., 2011, Taylor et al., 2010). In other work, spatiotemporal features are learned by extending 2D ConvNets into time by stacking consecutive video frames (Ji et al., 2013). Yet another study compared several approaches to extending ConvNets into the temporal domain, but with rather disappointing results (Karpathy et al., 2014): The architectures were not particularly sensitive to temporal modelling, with a slow fusion model performing slightly better than early and late fusion alternatives; moreover, similar levels of performance were achieved by a purely spatial network. The recently proposed C3D approach learns 3D ConvNets on a limited temporal support of 16 frames and all filter kernels having size 3×3×3 (Tran et al., 2015a). The network structure is similar to earlier deep spatial networks (Simonyan and Zisserman, 2014b).

Another research branch has investigated combining image information in network architectures across longer time periods. A comparison of temporal pooling architectures suggested that temporal pooling of convolutional layers performs better than slow, local, or late pooling, as well as temporal convolution (Ng et al., 2015b). That work also considered ordered sequence modelling, which feeds ConvNet features into a recurrent network with Long Short-Term Memory (LSTM) cells. Using LSTMs, however, did not yield an improvement over temporal pooling of convolutional features. Other work trained an LSTM on human skeleton sequences to regularize another LSTM that uses an Inception network for frame-level descriptor input (Mahasseni and Todorovic, 2016). Yet other work uses a multilayer LSTM to let the model attend to relevant spatial parts in the input frames (Sharma et al., 2015). Further, the inner product of a recurrent model has been replaced with a 2D convolution and thereby converts the fully connected hidden layers in a GRU-RNN to 2D convolutional operations (Ballas et al., 2016). That approach takes advantage of the local spatial similarity in images; however, it only yields a minor increase over their baseline, which is a two-stream VGG-16 ConvNet (Simonyan and Zisserman, 2014b) used as the input to their convolutional RNN. Finally, two recent approaches for action recognition apply ConvNets to dynamic images created by weighted averaging of video frames over long temporal extends (Bilen et al., 2016), or capture the transformation of ConvNet features from the beginning to the end of the video with a Siamese architecture (Wang et al., 2016a).

4.3 Technical approach

4.3.1 Two-Stream residual networks

As our base representation we use deep ResNets (He et al., 2016a,b). These networks are designed similarly to the VGG networks (Simonyan and Zisserman, 2014b), with small 3×3 spatial filters (except at the first layer), and similar to the Inception networks (Szegedy et al., 2015b), with 1×1 filters for learned dimensionality reduction and expansion. The network sees an input of size 224×224 that is reduced five times in the network by stride 2 convolutions followed by a global average pooling layer of the final 7×7 feature map and a fully-connected classification layer with softmax. Each time the spatial size of the feature map changes, the number of features is doubled to avoid tight bottlenecks. Batch normalization (Ioffe and Szegedy, 2015) and ReLU (Krizhevsky et al., 2012a) are applied after each convolution; the network does not use hidden fully connected, dropout, or max-pooling (except immediately after the first layer). The residual units are defined as (He et al., 2016a,b):

$$\mathbf{x}_{l+1} = f\left(\mathbf{x}_l + \mathcal{F}(\mathbf{x}_l; \mathcal{W}_l)\right), \tag{4.1}$$

where $\mathbf{x}_l$ and $\mathbf{x}_{l+1}$ are input and output of the l-th layer, $\mathcal{F}$ is a nonlinear residual mapping represented by convolutional filter weights $\mathcal{W}_l = \{\mathrm{W}_{l,k}|_{1\leq k\leq K}\}$ with $K \in \{2, 3\}$ and $f \equiv$ ReLU (He et al., 2016b). A key advantage of residual units is that their skip connections allow direct signal propagation from the first to the last layer of the network. Especially during backpropagation this arrangement is advantageous: Gradients are propagated directly from the loss layer to any previous layer while skipping intermediate weight layers that have potential to trigger vanishing or deterioration of the gradient signal.

We also leverage the two-stream architecture (Simonyan and Zisserman, 2014a). For both streams, we use the ResNet-50 model (He et al., 2016a) pretrained on the ImageNet dataset and replace the last (classifiation) layer according to the number of classes in the target dataset. The filters in the first layer of the motion stream are further modified by replicating the three RGB filter channels to a size of $2L = 20$ for operating over the horizontal and vertical optical flow stacks, each of which has a stack of $L = 10$ frames. This tack allows us to exploit the availability of a large amount of annotated training data for both streams.

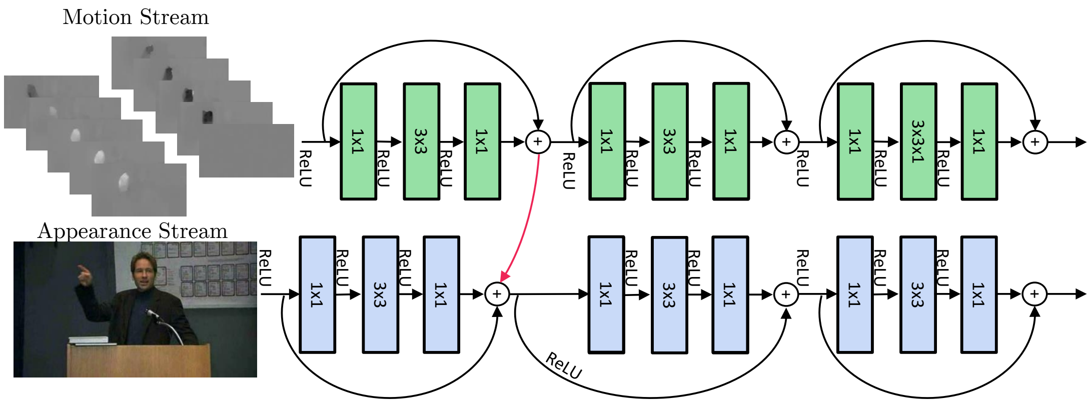


Figure 4.1: Three residual units operating on appearance and motion information. A direct way of connecting the output of the first motion unit with a residual connection into the appearance stream leads to high errors in testing. We conjecture that such a fusion hinders the representational power of the appearance network.

A drawback of the two-stream architecture is that it is unable to spatiotemporally register appearance and motion information. Thus, it is not able to represent what (captured by the spatial stream) moves in which way (captured by the temporal stream). Here, we remedy this deficiency by letting the network learn such *spatiotemporal* cues at several spatiotemporal scales. We enable this interaction by introducing residual connections between the two streams. Just as there can be various types of shortcut connections in a ResNet, there are several ways the two streams can be connected. In preliminary experiments we found that direct connections between identical layers of the two streams led to an increase in validation error. This is illustrated in Fig. 4.1, where the motion stream is connected with an additive operation to the motion stream. Since all residual units (usually consisting of two 1×1 and a 3×3 spatial convolution layer) are augmented with skip connections that distribute the signal to all such units in the network hierarchy, the newly added fusion signal will influence all layers of the appearance network; however, ResNets build on these identity connections to build up their representational capacity. We conjecture that our unsatisfactory results for this direct way of fusion are due to the large change that the signal of one network stream undergoes after injecting a fusion signal from the other stream. Notably, we were able to ameliorate this effect by inserting channel-wise learnable affine scale layers; however, in this case neither a negative nor a positive effect of the connection between the streams was observable (*i.e.* the network did not perform significantly better than the two-stream baseline). A more comprehensive ablation study on how the different cross-stream connections, including bidirectional forms, interact will be given in the next chapter and the next section continues to develop a solution based on additive interactions which are in accord with the design principles of ResNets.

4.3.2 Additive motion interaction

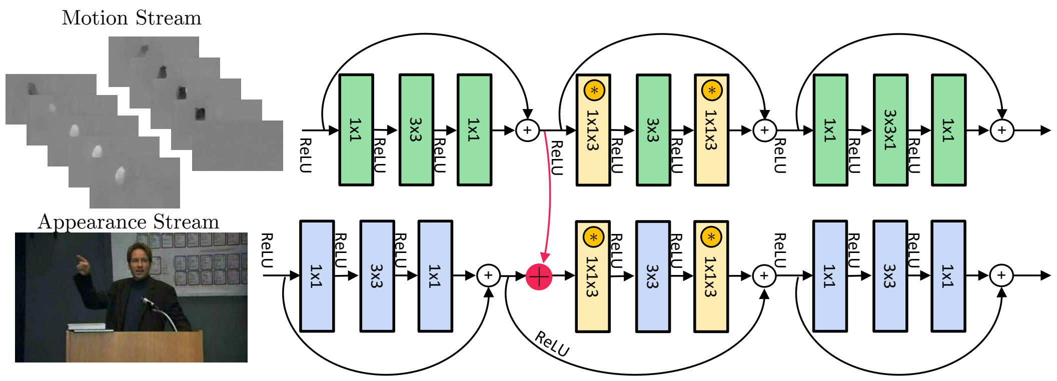


Figure 4.2: The proposed residual units in our ST-ResNet architecture. At the output of the first unit, a residual connection (highlighted in red) between the two streams enables additive motion interactions. The second residual unit also includes temporal convolutions by transforming the 1×1 dimensionality mapping filters (highlighted in yellow).

We inject a skip connection from the motion stream to the appearance stream's residual unit. To enable learning of spatiotemporal features at all possible scales, this modification is applied before the second residual unit at each spatial resolution of the network, as exemplified by the connection shown in Fig. 4.2. Formally, the corresponding appearance stream's residual units (4.1) are modified according to

$$\hat{\mathbf{x}}^{\mathrm{a}}_{l+1} = f(\mathbf{x}^{\mathrm{a}}_l) + \mathcal{F}\Big(\mathbf{x}^{\mathrm{a}}_l + f(\mathbf{x}^{\mathrm{m}}_l), \mathcal{W}^{\mathrm{a}}_l\Big), \tag{4.2}$$

where $\mathbf{x}^{\mathrm{a}}_l$ is the input of the l-th layer appearance stream, $\mathbf{x}^{\mathrm{m}}_l$ the input of the l-th layer motion stream and $\mathcal{W}^{\mathrm{a}}_l$ are the weights of the l-th layer residual unit in the appearance stream. For the gradient on the loss function $\mathcal{L}$ in the backward pass the chain rule yields

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}^{\mathrm{a}}_l} = \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{x}}^{\mathrm{a}}_{l+1}} \frac{\partial \hat{\mathbf{x}}^{\mathrm{a}}_{l+1}}{\partial \mathbf{x}^{\mathrm{a}}_l} = \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{x}}^{\mathrm{a}}_{l+1}} \left(\frac{\partial f(\mathbf{x}^{\mathrm{a}}_l)}{\partial \mathbf{x}^{\mathrm{a}}_l} + \frac{\partial}{\partial \mathbf{x}^{\mathrm{a}}_l} \mathcal{F}\Big(\mathbf{x}^{\mathrm{a}}_l + f(\mathbf{x}^{\mathrm{m}}_l), \mathcal{W}^{\mathrm{a}}_l\Big)\right) \tag{4.3}$$

for the appearance stream and similarly for the motion stream

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}^{\mathrm{m}}_l} = \frac{\partial \mathcal{L}}{\partial \mathbf{x}^{\mathrm{m}}_{l+1}} \frac{\partial \mathbf{x}^{\mathrm{m}}_{l+1}}{\partial \mathbf{x}^{\mathrm{m}}_l} + \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{x}}^{\mathrm{a}}_{l+1}} \frac{\partial}{\partial \mathbf{x}^{\mathrm{a}}_l} \mathcal{F}\Big(\mathbf{x}^{\mathrm{a}}_l + f(\mathbf{x}^{\mathrm{m}}_l), \mathcal{W}^{\mathrm{a}}_l\Big), \tag{4.4}$$

where the first additive term of (4.4) is the gradient at the l-th layer in the motion stream and the second term accumulates gradients from the appearance stream. Thus, the residual connection between the streams backpropagates gradients from the appearance stream into the motion stream.

4.3.3 Convolutional residual connections across time

The cascading of small convolutional filters is a key design choice in state-of-the-art image ConvNets. As noted above, the benefits are a large receptive field at lower cost compared to the use of large filters, coupled with having additional non-linearities in the network that follow each convolutional layer (Simonyan and Zisserman, 2014b). Another design practice is to split 2D convolutions into separate 1D convolutions; importantly, however, this choice must be realized with care: Such factorization degrades performance when applied in early layers, but when applied on intermediate to deep layers with smaller grid-sizes it yields good results (Szegedy et al., 2015b). Here we transfer both of these design practices to the spatiotemporal domain by injecting temporal convolutions into the residual units of a two-stream architecture.

Spatiotemporal coherence is an important cue when working with time varying visual data and can be exploited to learn general representations from video in an unsupervised manner (Goroshin et al., 2015). In that case, temporal smoothness is an important property and is enforced by requiring features to vary slowly with respect to time, usually implemented by penalizing with the distance of temporally neighbouring features. Further, one can expect that in many cases a ConvNet is capturing similar features across time. For example, an action with repetitive motion patterns such as "Hammering" would trigger similar features for the appearance and motion stream over time. For such cases the use of temporal residual connections would make perfect sense. However, for cases where the appearance or the instantaneous motion pattern varies over time, a residual connection would be suboptimal for discriminative learning, since the sum operation corresponds to a low-pass filtering over time and would smooth out potentially important high-frequency temporal variation of the features. Moreover, backpropagation is unable to compensate for that deficit since at a sum layer all gradients are distributed equally from output to input connections. Indeed, in preliminary experiments we observed that temporal residuals (*i.e.* sum pooling across time) in both the appearance and motion networks decreased validation accuracy while increasing training accuracy. Therefore, we conjecture that an averaging residual connection (which is not learnable) is suboptimal for capturing characteristic temporal patterns. Further support for suboptimality of this approach comes from the minor performance gain observed in attaching LSTMs to frame-level ConvNet features, as LSTMs provide a form of gated feature averaging (Ballas et al., 2016, Donahue et al., 2015, Mahasseni and Todorovic, 2016, Ng et al., 2015b, Sharma et al., 2015).

Based on the above observations, we developed a novel approach to temporal residual connections that builds on the ConvNet design guidelines of chaining small (Simonyan and Zisserman, 2014b) asymmetric (Ioannou et al., 2016, Szegedy et al., 2015b) filters. We extend the ResNet architecture with temporal convolutions by *transforming* spatial dimensionality mapping filters in the residual paths to temporal filters (see Fig. 4.2). This allows the straightforward use of standard two-stream ConvNets that have been pre-trained on large-scale datasets *e.g.* to leverage the massive amounts of training data from the ImageNet challenge. We initialize the temporal weights as an averaging filter (*i.e.* residual connections across time) and let the network *learn* to best discriminate image dynamics via backpropagation. We achieve this by replicating the learned spatial 1×1

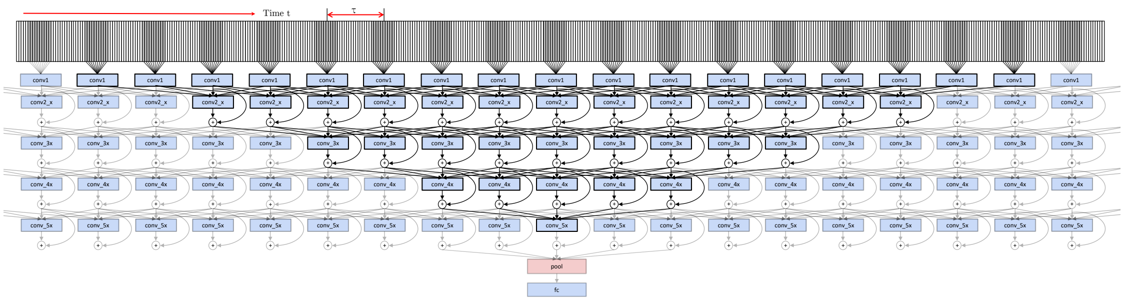


Figure 4.3: The temporal receptive field of a single neuron at the fifth residual layer of our motion network stream is highlighted. τ indicates the temporal stride between inputs. The outputs of conv5_3 are max-pooled in time and fed to the fully connected layer of our ST-ResNet*.

dimensionality mapping kernels in pretrained ResNets across time. Given the pretrained spatial weights, $\mathbf{w}_l \in \mathbb{R}^{1\times1\times C}$, temporal filters, $\hat{\mathbf{w}}_l \in \mathbb{R}^{1\times1\times T'\times C}$, are initialized according to

$$\hat{\mathbf{w}}_l(i,j,t,c) = \frac{\mathbf{w}_l(i,j,c)}{T'}, \forall t \in [1,T'], \tag{4.5}$$

and subsequently refined via backpropagation. We transform filters from both the motion and the appearance ResNets accordingly. Hence, the temporal filters are able to learn the temporal evolution of the appearance and motion features and, moreover, by stacking such filters as the depth of the network increases highly complex spatiotemporal features can be learned.

4.3.4 Proposed architecture

Our overall architecture (used for each stream) is summarized in Table 4.1. The underlying network used is a 50 layer ResNet (He et al., 2016a). Each filtering operation is followed by batch normalization (Ioffe and Szegedy, 2015) and halfway rectification (ReLU). In the columns we show "metalayers" which share the same output size. From left to right, top to bottom, the first row shows the convolutional and pooling building blocks, with the filter and pooling size shown as $(W \times H \times T, C)$, denoting width, height, temporal extent and number of feature channels, resp. Brackets outline residual units equipped with skip connections. In the last two rows we show the output size of these metalayers as well as the receptive field on which they operate. One observes that the temporal receptive field is modulated by the temporal stride τ between the input chunks. For example, if the stride is set to $\tau = 15$ frames, a unit at conv5_3 sees a window of $17 * 15 = 255$ frames on the input video; see Fig. 4.3. The pool5 layer receives multiple spatiotemporal features, where the spatial 7×7 features are averaged as in (He et al., 2016a) and the temporal features are max-pooled within a window of 5, with each of these seeing a window of 705 frames at the input. The pool5 output is classified by a fully connected layer of size $1 \times 1 \times 1 \times 2048$; note that this passes several temporally max-pooled chunks to the softmax log-loss layer afterwards. A detailed view at the conv5_x block can be seen in Fig. 4.4.

Layers	conv1	pool1	conv2_x	conv3_x	conv4_x	conv5_x	pool5
Blocks	7×7×1, 64	3×3×1 max stride 2	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}$ skip-stream $\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}$ $\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}$ skip-stream $\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}$ $\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}$ skip-stream $\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}$ $\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}$ skip-stream $\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}$ $\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}$	7×7×1 avg 1×1×5 max stride 2
Output size	112×112×11	56×56×11	56×56×11	28×28×11	14×14×11	7×7×11	1×1× 4
Recept. Field	7×7×1	11×11×1	35×35×5τ	99×99×9τ	291×291×13τ	483×483×17τ	675 × 675× 47τ

Table 4.1: Spatiotemporal ResNet architecture used in both ConvNet streams. The metalayers are shown in the columns with their building blocks showing the convolutional filter dimensions ($W \times H \times T, C$) in brackets. Each building block shown in brackets also has a skip connection to the block below and skip-stream denotes a residual connection from the motion to the appearance stream, *e.g.*, see Fig. 4.4 for the conv5_2 building block. Stride 2 downsampling is performed by conv1, pool1, conv3_1, conv4_1 and conv5_1. The output and receptive field size of these layers is shown below. For both streams, the pool5 layer is followed by a $1 \times 1 \times 1 \times 2048$ fully connected layer, a softmax and a loss.

Sub-batch normalization. Batch normalization (Ioffe and Szegedy, 2015) subtracts from all activations the batchwise mean and divides by their variance. These moments are estimated by averaging over spatial locations and multiple images in the batch. After batch normalization a learned, channel-specific affine transformation (scaling and bias) is applied. The noisy bias/variance estimation replaces the need for dropout regularization (He et al., 2016a, Szegedy et al., 2016). We found that lowering the number of samples used for batch normalization can further improve the generalization performance of the model. For example, for the appearance stream we use a low batch size of 4 for moment estimation during training. This practice strongly supports generalization of the model and nontrivially increases validation accuracy (≈4% on UCF101). Interestingly, in comparison to this approach, using dropout after the classification layer (*e.g.* as in (Szegedy et al., 2016)) decreased validation accuracy of the appearance stream. Note that only the batchsize for normalizing the activations is reduced; the batch size in stochastic gradient descent is unchanged.

4.3.5 Model training and evaluation

Our method has been implemented in our own modified version of the MatConvNet (Vedaldi and Lenc, 2015) library, which supports spatiotemporal processing. We train our model in three optimization steps with the parameters listed in Table 4.2.

Motion and appearance streams. First, each stream is trained similar to (Simonyan and Zisserman, 2014a) using Stochastic Gradient Descent (SGD) with momentum of 0.9. We rescale all videos by keeping the aspect ratio and resizing the smallest side of a frame to 256. The motion network uses optical flow stacking with $L = 10$ frames and is trained for $30K$ iterations with a learning rate of 10^{-2} followed by $10K$ iterations at a learning rate of 10^{-3}. At each iteration,

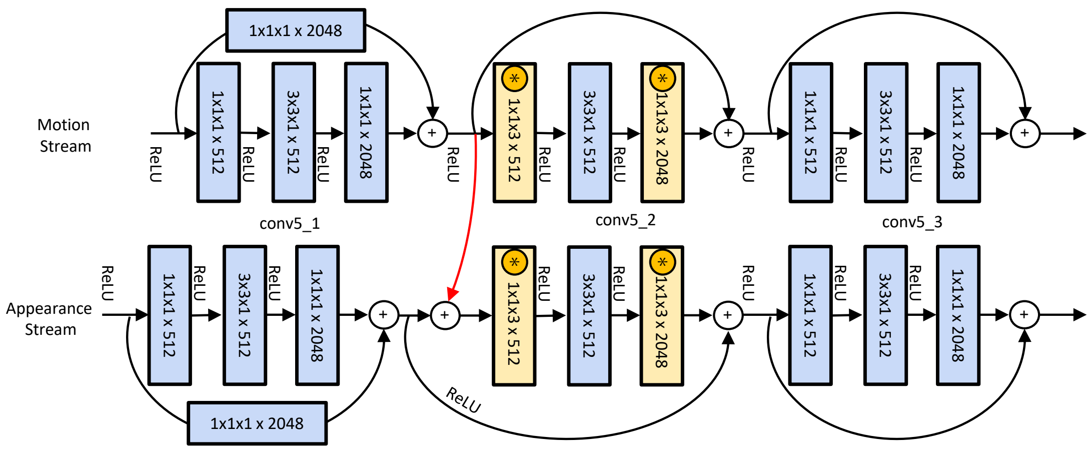


Figure 4.4: Detailed view of the conv5_x residual units of our architecture. A residual connection (highlighted in red) between the two streams enables additive motion interactions. The second residual unit, conv5_2 also includes temporal convolutions (highlighted in yellow) for learning high-level spacetime features.

Training phase	SGD batch size	Bnorm batch size	Learning Rate (#Iterations)	Temporal chunks / stride τ
Motion stream	256	86	$10^{-2}(30K)$, $10^{-3}(10K)$	1 / $\tau = 1$
Appearance stream	256	4	$10^{-2}(10K)$, $10^{-3}(10K)$	1 / $\tau = 1$
ST-ResNet	128	4	$10^{-3}(30K)$, $10^{-4}(30K)$, $10^{-5}(20K)$	5 / $\tau \in [5, 15]$
ST-ResNet*	128	4	$10^{-4}(2K)$, $10^{-5}(2K)$	11 / $\tau \in [1, 15]$

Table 4.2: Parameters for the three training phases of our model

a batch of 256 samples is constructed by randomly sampling a single optical flow stack from a video; however, for batch normalization (Ioffe and Szegedy, 2015), we only use 86 samples to facilitate generalization. We precompute optical flow (Zach et al., 2007) before training and store the flow fields as JPEGs (with displacement vectors > 20 pixels clipped). During training, we use the same augmentations as in (Ballas et al., 2016, Wang et al., 2016a); *i.e.* randomly cropping from the borders and centre of the flow stack and sampling the width and height of each crop randomly within $256, 224, 192, 168$, following by resizing to 224×224. The appearance stream is trained identically with a batch of 256 RGB frames and learning rate of 10^{-2} for $10K$ iterations, followed by 10^{-3} for another 10K iterations. Notably here we choose a very small batch size of 4 for normalization. We also apply random cropping and scale augmentations: We randomly jitter the width and height of the 224×224 input frame by $\pm 25\%$ and also randomly crop it from a maximum of 25% distance from the image borders. The cropped patch is rescaled to 224×224 and passed as input to the network. The same rescaling and cropping technique is chosen to train the next two steps of our architecture described below. In all our training steps we use random horizontal flipping and do not apply RGB colour jittering (Krizhevsky et al., 2012a).

ST-ResNet. Second, to train our spatiotemporal ResNet we sample 5 inputs from a video with random temporal stride between 5 and 15 frames. This technique can be thought of as frame-rate jittering for the temporal convolutional layers and is important to reduce overfitting of the final model. SGD is used with a batch size of 128 videos where 5 temporal chunks are extracted from each. Batch-normalization uses a smaller batch size of $128/32 = 4$. The learning rate is set to 10^{-3} and is reduced by a factor of 10 after $30K$ iterations. Notably, there is no pooling over time, which leads to temporal fully convolutional training with a single loss for each of the 5 inputs. We found that this strategy significantly reduces the training duration with the drawback that each loss does not capture all available information. We overcome this downside by the last step of our training procedure outlined next.

ST-ResNet*. For our final model, we equip the spatiotemporal ResNet with a temporal max-pooling layer after pool5 (see Table 4.1, temporal average pooling led to inferior results) and continue training as above with the learning rate starting from 10^{-4} for $2K$ iterations followed by 10^{-5}. As indicated in Table 4.2, we now use 11 temporal chunks as input with the stride τ between these being randomly chosen from $[1, 15]$.

Fully convolutional inference. For fair comparison, we follow the evaluation procedure of the original two-stream work (Simonyan and Zisserman, 2014a) by sampling 25 frames (and their horizontal flips). However, rather than using 10 spatial 224×224 crops from each of the frames, we apply fully convolutional testing both spatially (smallest side rescaled to 256) and temporally (the 25 frame-chunks) by classifying the video in a single forward pass, which takes ≈250ms on a Nvidia Titan X GPU. For inference, we average the predictions of the fully connected layers (without softmax) over all spatiotemporal locations.

4.4 Evaluation

As in the previous chapter, we evaluate our approach on the UCF101 (Khurram Soomro and Shah, 2012) and HMDB51 (Kuehne et al., 2011) datasets, where we use the provided evaluation protocol and report mean average accuracy over three splits into training and test sets.

4.4.1 Two-Stream ResNet with additive interactions

Table 4.3 shows the results of our two-stream architecture across the three training stages outlined in Sec. 4.3.5. For stream fusion, we always average the (non-softmaxed) prediction scores of the classification layer as this approach produces better results than averaging the softmax scores. Initially, let us consider the performance of the two streams, both initialized with ResNet50 models trained on the ImageNet ILSVRC12 data (He et al., 2016a), but without cross-stream residual connections (4.2) and temporal convolutional layers (4.5). The accuracies for UCF101 and HMDB51 are 89.47% and 60.59%, resp. Comparatively, a VGG16 two-stream architecture produces 91.4% and 58.5% (Ballas et al., 2016, Wang et al., 2016a). In comparing these results it is notable that the VGG16 architecture is more computationally demanding (19.6 *vs.* 3.8 billion multiply-add FLOPs) and also holds more model parameters (135M *vs.* 34M) than a ResNet50, which could explain the better performance on the larger UCF101 dataset.

Dataset	Appearance stream	Motion stream	Two-Streams	ST-ResNet	ST-ResNet*
UCF101	82.29%	79.05%	89.47%	92.76%	93.46%
HMDB51	43.42%	55.47%	60.59%	65.57%	66.41%

Table 4.3: Classification accuracy on UCF101 and HMDB51 in the three training stages of our model.

We now consider our proposed spatiotemporal ResNet (ST-ResNet), which is initialized by our two-stream ResNet50 model of above and subsequently equipped with 4 residual connections between the streams and 16 transformed temporal convolution layers (initialized as averaging filters). The model is trained end-to-end with the loss layers unchanged (we found that using a single, joint softmax classifier overfits severely to appearance information) and learning parameters chosen as in Table 4.2. The results are shown in the penultimate column of Table 4.3. Our architecture significantly improves over the two-stream baseline indicating the importance of residual connections between the streams as well as temporal convolutional connections over time. Finally, in the last column of Table 4.3 we show results for our ST-ResNet* architecture that is further equipped with a temporal max-pooling layer to consider larger temporal windows in training and testing. For training ST-ResNet* we use 11 temporal chunks at the input and the max-pooling layer pools over 5 chunks to expand the temporal receptive field at the loss layer to a maximum of 705 frames at the input. For testing, where the network sees 25 temporal chunks, we observe that this long-term pooling further improves accuracy over our ST-ResNet by around 1% on both datasets.

4.4.2 Comparison with the previous state-of-the-art

We compare to the state-of-the-art in action recognition over all three splits of UCF101 and HMDB51 in Table 4.4 (left). We use ST-ResNet*, as above, and predict the videos in a single forward pass using fully convolutional testing. When comparing to the original two-stream method (Simonyan and Zisserman, 2014a), we improve by 5.4% on UCF101 and by 7% on HMDB51. Apparently, even though the original two-stream approach has the advantage of multitask learning (HMDB51) and SVM fusion, the benefits of our deeper architecture with its cross-stream residual connections are greater. Another interesting comparison is against the two-stream network in (Ng et al., 2015b), which attaches an LSTM to a two-stream Inception (Szegedy et al., 2015b) architecture. Their accuracy of 88.6% is to date the best performing approach using LSTMs for action recognition. Here, our gain of 4.8% further underlines the importance of our architectural choices.

At time of publication of our results, the previously best performing action recognition approach, Transformations (Wang et al., 2016a), captures the transformation from start to finish of a video by using two VGG16 Siamese streams (that do not share model parameters, *i.e.* 4 VGG16 models) to discriminatively learn a transformation matrix. This method uses considerably more parameters than our approach, yet is readily outperformed by ours. Similarly, the computational demand of a single VGG16 model is much higher than the ResNet50 we employ (19.6 *vs.* 3.8 billion

Method	UCF101	HMDB51
Two-Stream ConvNet (Simonyan and Zisserman, 2014a)	88.0%	59.4%
Two-Stream+LSTM(Ng et al., 2015b)	88.6%	-
Two-Stream (VGG16) (Ballas et al., 2016, Wang et al., 2016a)	91.4%	58.5%
Transformations(Wang et al., 2016a)	92.4%	62.0%
Two-Stream Fusion (Chapter 3)	92.5%	65.4%
ST-ResNet* (50 layer)	**93.4%**	**66.4%**

Method	UCF101	HMDB51
IDT (Wang and Schmid, 2013)	86.4%	61.7%
C3D + IDT (Tran et al., 2015a)	90.4%	-
TDD + IDT (Wang et al., 2015b)	91.5%	65.9%
Dynamic Image Networks + IDT (Bilen et al., 2016)	89.1%	65.2%
Two-Stream Fusion (Chapter 3)	93.5%	69.2%
ST-ResNet* (50 layer) + IDT	**94.6%**	**70.3%**

Table 4.4: Mean classification accuracy of the state-of-the-art on HMDB51 and UCF101 for the best ConvNet approaches (top) and methods that additionally use IDT features (bottom). Our ST-ResNet obtains best performance on both datasets.

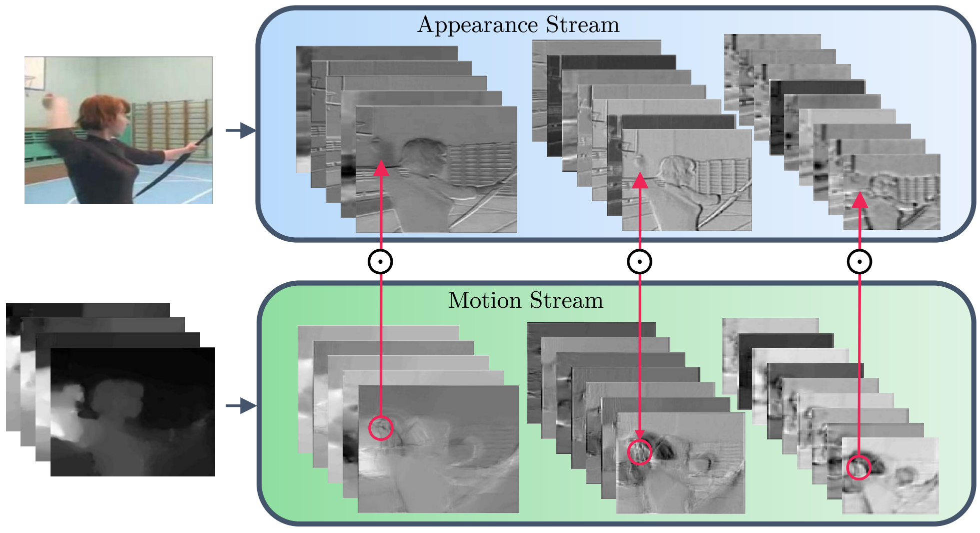
multiply-add FLOPs).

The combination of ConvNet methods with trajectory-based hand-crafted IDT features (Wang and Schmid, 2013) typically boosts performance nontrivially (Bilen et al., 2016, Tran et al., 2015a). Therefore, we further explore the benefits of adding trajectory features to our approach. We achieve this goal by simply averaging the L2-normalized SVM scores of the FV-encoded IDT descriptors (*i.e.* HOG, HOF, MBH) (Wang and Schmid, 2013) with the L2-normalized video predictions of our ST-ResNet*, again without softmax normalization. The results are shown in Table 4.4 (right) where we observe a notable boost in accuracy of our approach on HMDB51, albeit less on UCF101. Note that unlike our approach, the other approaches in Table 4.4 (right) suffer considerably larger performance drops when used without IDT, *e.g.* C3D (Tran et al., 2015a) reduces to 85.2% on UCF101, while Dynamic Image Networks (Bilen et al., 2016) reduces to 76.9% on UCF101 and 42.8% on HMDB51. These relatively larger performance decrements again underline that our approach is better able to capture the available dynamic information, as there is less to be gained by augmenting it with IDT. Still, there is a benefit from the hand-crafted IDT features even with our approach, which could be attributed to its explicit compensation of camera motion. Overall, our 94.6% on UCF101 and 70.3% HMDB51 has exceeded the previous state-of-the-art on these widely used action recognition datasets.

Finally, it is interesting to note that across all of the presented results, the peformance of our ST-ResNet* is particularly outstanding on HMDB51. While performance on UCF101 is becoming saturated and thereby provides less opportunity to demonstrate advances, HMDB51 provides greater challenge and our approach is able to respond with a decidedly increased performance margin over the alternatives.

4.5 Summary

In this chapter, we have presented a novel spatiotemporal ResNet architecture for video-based action recognition. In particular, our approach is the first to combine two-stream with residual networks and to show the great advantage that results. Our ST-ResNet allows the hierarchical learning of spatiotemporal features by connecting the appearance and motion channels of a two-stream architecture. Furthermore, we transfer both streams from the spatial to the spatiotemporal domain by transforming the dimensionality mapping filters of a pre-trained model into temporal convolutions, initialized as residual filters over time. The whole system is trained end-to-end and achieves substantial accuracy gains over the standard two-stream architecture, boosting state-of-the-art performance on two popular action recognition datasets.



Motion gating at the feature level requires feature correspondence in the forward pass that is enforced through the gradient update of multiplicative interactions in the backward pass. In this chapter, we introduce Spatiotemporal Multiplier Networks for learning multiscale spacetime representations to advance discriminative video recognition.

5 Spatiotemporal Multiplier Networks

5.1 Motivation

Building on our findings from the preceding chapter, we now discuss in greater deal our design choices and have a more in-depth look into alternatives. The spatiotemporal ResNet (ST-ResNet) presented in the previous chapter non-trivially extended the performance of the original two-stream approach in application to action recognition on standard datasets. Although, the ST-ResNet (Chapter 4) yielded state-of-the-art performance, it did not provide systematic justification for its design choices. Our work in this chapter reconsiders the combination of the two-stream and ResNet approaches in a more thorough fashion to increase the understanding of how these techniques interact, with a resulting novel architecture that exceeds our performance from Chapter 4.

More specifically, three main contributions are provided in this chapter. First, we show that a multiplicative motion gating of the appearance stream provides nontrivial performance boost over an additive formulation. We discuss the advantages of multiplicative interactions by the effect on the gradient in a residual network. We also verify their effectiveness in a series of ablation experiments where we systematically explore various alternatives for connecting the two streams, including bidirectional connections.

Second, we discuss several approaches to generalizing the ST-ResNet architecture over long-term input. Note that in Chapter 3, we have found that 3D convolutional fusion kernels that are initialized by identity matrices that sums the activations from previous layers performed especially well. Here, we propose to inject temporal filters that are initialized as identity mapping kernels at the feature level. These temporal filters inject new layers in an existing model, while preserving the feature identity property of residual networks. This approach allows injection of new temporal aggregation filters even into the skip path of the network. We provide ablation studies for where to inject these mappings and how to initialize the temporal aggregation kernel.

Third, based on what is learned from our investigation of fusing two-streams with residual connections and extending temporal support, we propose a *general* ConvNet architecture for action recognition in video. We provide details for how to learn a deep two-stream architecture that is fully convolutional in spacetime in an end-to-end fashion. We empirically show how multiplicative motion gating between the streams and injected temporal aggregation filters can enhance performance substantially, leading to state-of-the-art performance on two popular action recognition datasets.

Our code and models are available at `https://github.com/feichtenhofer/st-resnet`

5.2 Related work

Historically, research on video-based action recognition has mostly focused on crafting spatiotemporal features from optical flow-based motion information, *e.g.* Histograms Of Flow (HOF) (Laptev et al., 2008b), Motion Boundary Histograms (MBH) (Dalal et al., 2006) and trajectories (Wang and Schmid, 2013), or spatiotemporal oriented filtering *e.g.* HOG3D (Kläser et al., 2008), Cuboids (Dollár et al., 2005) and SOEs (Derpanis et al., 2012, Feichtenhofer et al., 2015).

More recently, researchers have focused on learning spatiotemporal features in an end-to-end fashion. Some work along these lines has concentrated on use of unsupervised learning (Le et al., 2011, Taylor et al., 2010). Other work makes use of a combination of hand-crafted and learned features (Ji et al., 2013). In contrast, an alternative 3D spatiotemporal ConvNet, directly learned all of its filter kernels (Tran et al., 2015a). Interestingly, a work that compared a variety of approaches to extending 2D spatial ConvNets into time found little benefit of the temporal data (Karpathy et al., 2014).

Another relevant research direction for our concerns has addressed aggregation of temporal information over extended time periods. Here, a comparison of pooling approaches suggested good performance for temporal pooling of convolutional layers (Ng et al., 2015b), as well as longer convolutions across time (Varol et al., 2016). Perhaps the most straightforward approach comes from simple weighted averaging of video frames across time (Bilen et al., 2016). Complexity can be found in the various efforts that have incorporated LSTMs into their architectures to extend their temporal support (*e.g.* (Mahasseni and Todorovic, 2016, Ng et al., 2015b, Sharma et al., 2015, Wang et al., 2016b)). Alternatively, RNNs have been applied for similar purposes (Ballas et al., 2016, Li et al., 2016b). Other recent approaches rely on a Siamese architecture to abstract the temporal transformation of features across a video (Wang et al., 2016a) or identify key volumes in the sequences (Zhu et al., 2016).

The most closely related work to our contributions in this chapter is two-stream ConvNet architecture (Simonyan and Zisserman, 2014a), which initially processes colour and optical flow information in segregation for subsequent late fusion of their separate classification scores, and our own extensions to that work that investigated spatiotemporal fusion (Chapter 3) and residual connections (Chapter 4) are of particular relevance for the work presented in this chapter, as they serve as points of departure. In contrast to those previous efforts, the current work provides a more systematic investigation of the design space that leads to a novel architecture with improved performance.

5.3 Two-stream multiplier networks

5.3.1 Baseline architecture

We build our architecture on the two-stream approach (Simonyan and Zisserman, 2014a), which separately trains two ConvNet streams: One stream exploits spatial appearance based on input of RGB image frames; the second exploits motion based on an input stack of $L = 10$ horizontal and vertical optical flow frames. As in the previous chapter, for each stream we use ResNets (He et al., 2016a,b) as the base network architecture. ResNets are fully convolutional architectures that, after an initial 7×7 filter, chain small spatial 3×3 convolutions with 1×1 dimensionality mapping filters (Szegedy et al., 2015b) followed by batch normalization (Ioffe and Szegedy, 2015) and ReLU (Krizhevsky et al., 2012a) non-linearities. The input is of size 224×224 and reduced five times in the network by stride 2 convolutions followed by global average pooling of the final 7×7 feature map.

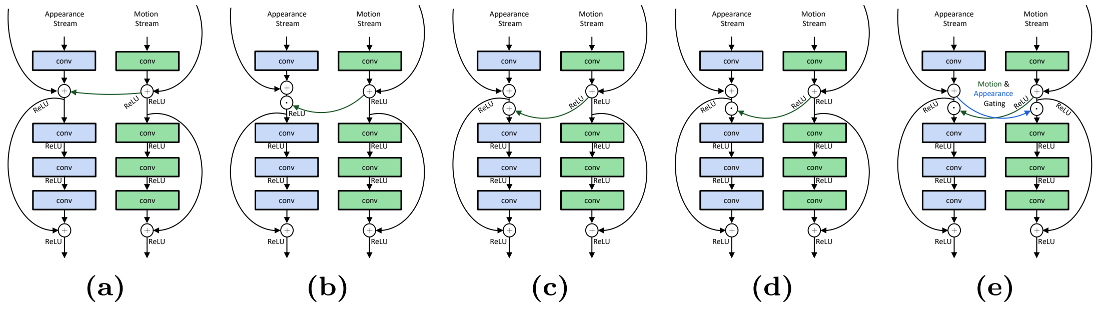


Figure 5.1: Different types of motion interactions between the two streams enables the learning of local spatiotemporal features. (a)-(d) show unidirectional connections from the motion into the appearance path and (e) illustrates bidirectional gating across streams.

ResNets are equipped with additive skip connections to directly propagate signals to all layers of the network. The building blocks of the network are residual units defined as (He et al., 2016a,b):

$$\mathbf{x}_{l+1} = f\left(\mathbf{x}_l + \mathcal{F}(\mathbf{x}_l; \mathcal{W}_l)\right), \tag{5.1}$$

where $\mathbf{x}_l$ and $\mathbf{x}_{l+1}$ are input and output of the l-th layer, $\mathcal{F}$ is a nonlinear residual mapping represented by convolutional filter weights $\mathcal{W}_l = \{\mathrm{W}_{l,k}|_{1\leq k\leq K}\}$ with $K \in \{2, 3\}$ and $f \equiv \mathrm{ReLU}$ (He et al., 2016b).

For both streams, we use the ResNet model (He et al., 2016a) pretrained on the ImageNet CLSLOC dataset and replace the last layer according to the number of classes in the target dataset. Since the motion stream receives a stack of $2L = 20$ horizontal and vertical flow fields at the input, we replicate the first layer filters to fit that dimensionality.

5.3.2 Connecting the two streams

The original two-stream architecture only allowed the two processing paths to interact via late fusion of their respective softmax predictions (Simonyan and Zisserman, 2014a). That design did not support the learning of truly spatiotemporal features, which require the appearance and motion paths to interact earlier on during processing. This interaction, however, can be important for the discrimination of actions that have similar motion or appearance patterns and can only be disentangled by the combination of the two *e.g.* brushing teeth, applying a lipstick or shaving a beard. To address this limitation, we inject cross-stream residual connections. There are numerous ways in which such connections can be embodied. In our ablation studies we compare several variants (Fig. 5.1). We show that simple cross-residual connections between identical layers of the two streams leads to inferior classification performance compared to the (non-connected) two-stream baseline. We conjecture that the decrease in performance is due to the large change of the input distribution that the layers in one network stream undergo after injecting a fusion signal from the other stream.

5.3.2.1 Additive interaction

Our approach presented in Chapter 4 provided a natural extension of ResNets for the spatiotemporal domain by adding motion residuals to the appearance stream (Chapter 4), as illustrated in Fig. 5.1c and formalized as

$$\hat{\mathbf{x}}^{\mathrm{a}}_{l+1} = f(\mathbf{x}^{\mathrm{a}}_l) + \mathcal{F}\Big(\mathbf{x}^{\mathrm{a}}_l + f(\mathbf{x}^{\mathrm{m}}_l), \mathcal{W}^{\mathrm{a}}_l\Big), \tag{5.2}$$

where $\mathbf{x}^{\mathrm{a}}_l$ and $\mathbf{x}^{\mathrm{m}}_l$ are the inputs of the l-th layers of the appearance and motion streams (resp.), while $\mathcal{W}^{\mathrm{a}}_l$ holds the weights of the l-th layer residual unit in the appearance stream. Correspondingly, the gradient on the loss function, $\mathcal{L}$, in the backward pass is given via the chain rule as

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \mathbf{x}^{\mathrm{a}}_l} &= \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{x}}^{\mathrm{a}}_{l+1}} \frac{\partial \hat{\mathbf{x}}^{\mathrm{a}}_{l+1}}{\partial \mathbf{x}^{\mathrm{a}}_l} \\ &= \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{x}}^{\mathrm{a}}_{l+1}} \left(\frac{\partial f(\mathbf{x}^{\mathrm{a}}_l)}{\partial \mathbf{x}^{\mathrm{a}}_l} + \frac{\partial}{\partial \mathbf{x}^{\mathrm{a}}_l} \mathcal{F}\Big(\mathbf{x}^{\mathrm{a}}_l + f(\mathbf{x}^{\mathrm{m}}_l), \mathcal{W}^{\mathrm{a}}_l\Big) \right) \end{aligned} \tag{5.3}$$

for the appearance stream and similarly for the motion stream as

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}^{\mathrm{m}}_l} = \frac{\partial \mathcal{L}}{\partial \mathbf{x}^{\mathrm{m}}_{l+1}} \frac{\partial \mathbf{x}^{\mathrm{m}}_{l+1}}{\partial \mathbf{x}^{\mathrm{m}}_l} + \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{x}}^{\mathrm{a}}_{l+1}} \frac{\partial}{\partial \mathbf{x}^{\mathrm{a}}_l} \mathcal{F}\Big(\mathbf{x}^{\mathrm{a}}_l + f(\mathbf{x}^{\mathrm{m}}_l), \mathcal{W}^{\mathrm{a}}_l\Big). \tag{5.4}$$

Note that these expressions were presented in the previous chapter and are repeated here for the convenience of the reader.

5.3.2.2 Multiplicative interaction

An interesting variation on the between stream interaction relates to multiplicative motion models (*e.g.* (Memisevic and Hinton, 2010, Oh et al., 2015, Taylor and Hinton, 2009)) and treats the motion signal as gated modulation of the appearance features, illustrated in Fig. 5.1d, and formalized as

$$\hat{\mathbf{x}}^{\mathrm{a}}_{l+1} = f(\mathbf{x}^{\mathrm{a}}_l) + \mathcal{F}\Big(\mathbf{x}^{\mathrm{a}}_l \odot f(\mathbf{x}^{\mathrm{m}}_l), \mathcal{W}_l\Big), \tag{5.5}$$

where $\odot$ corresponds to elementwise multiplication. A more detailed schematic is shown in Fig. 5.2. In this case, the gradient on the loss function, $\mathcal{L}$, during the backward pass can be expressed as

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \mathbf{x}^{\mathrm{a}}_l} &= \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{x}}^{\mathrm{a}}_{l+1}} \frac{\partial \hat{\mathbf{x}}^{\mathrm{a}}_{l+1}}{\partial \mathbf{x}^{\mathrm{a}}_l} \\ &= \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{x}}^{\mathrm{a}}_{l+1}} \left(\frac{\partial f(\mathbf{x}^{\mathrm{a}}_l)}{\partial \mathbf{x}^{\mathrm{a}}_l} + \frac{\partial}{\partial \mathbf{x}^{\mathrm{a}}_l} \mathcal{F}\Big(\mathbf{x}^{\mathrm{a}}_l \odot f(\mathbf{x}^{\mathrm{m}}_l), \mathcal{W}^{\mathrm{a}}_l\Big) f(\mathbf{x}^{\mathrm{m}}_l) \right) \end{aligned} \tag{5.6}$$

where the gradient flowing through the appearance stream's residual unit is modulated by the motion signal, $f(\mathbf{x}^{\mathrm{m}}_l)$. Mutually, the residual unit's gradient is modulated by the forwarded appearance signal $\mathbf{x}^{\mathrm{a}}_l$,

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}^{\mathrm{m}}_l} = \frac{\partial \mathcal{L}}{\partial \mathbf{x}^{\mathrm{m}}_{l+1}} \frac{\partial \mathbf{x}^{\mathrm{m}}_{l+1}}{\partial \mathbf{x}^{\mathrm{m}}_l} + \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{x}}^{\mathrm{a}}_{l+1}} \frac{\partial}{\partial \mathbf{x}^{\mathrm{a}}_l} \mathcal{F}\Big(\mathbf{x}^{\mathrm{a}}_l \odot f(\mathbf{x}^{\mathrm{m}}_l), \mathcal{W}^{\mathrm{a}}_l\Big) \mathbf{x}^{\mathrm{a}}_l, \tag{5.7}$$

before addition to the motion stream gradient $\frac{\partial \mathcal{L}}{\partial \mathbf{x}^{\mathrm{m}}_{l+1}} \frac{\partial \mathbf{x}^{\mathrm{m}}_{l+1}}{\partial \mathbf{x}^{\mathrm{m}}_l}$. Thus, during backpropagation the current inputs of the motion $\mathbf{x}^{\mathrm{m}}_l$ and appearance $\mathbf{x}^{\mathrm{a}}_l$ streams are explicitly involved, acting as a gating mechanism on the gradient. This formulation makes the architecture particularly capable of learning spatiotemporal feature correspondences.

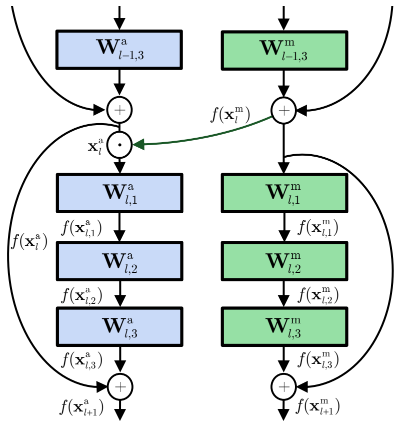


Figure 5.2: Illustration of multiplicative residual gating between the two streams (detailing Fig. 5.1d). During backpropagation the gradient is factored with the opposing stream's forward signal.

5.3.2.3 Discussion

Inclusion of the multiplicative interaction increases the order of the network fusion from first to second order (Goudreau et al., 1994). Here, this multiplicative interaction between the two streams implies a much stronger signal change based on spatiotemporal feature correspondence compared to the additive interaction (5.2): In the former case, (5.5), the motion information directly scales the appearance information through the term $\mathbf{x}^{a}_{l} \odot f(\mathbf{x}^{m}_{l})$, rather than via a more subtle bias, $\mathbf{x}^{a}_{l} + f(\mathbf{x}^{m}_{l})$, as in the additive case, (5.2). During backpropagation, instead of the fusion gradient flowing through the appearance, (5.3), and motion, (5.4), streams being distributed uniformly due to additive forward interaction (5.2), it now is multiplicatively scaled by the opposing stream's current inputs, $f(\mathbf{x}^{m}_{l})$ and $\mathbf{x}^{a}_{l}$ in equations (5.6) and (5.7), respectively. This latter type of interaction allows the streams to more effectively interact during the learning process and corresponding spatiotemporal features thereby ultimately are captured (*cf.* similar discussion in the context of recurrent networks (Wu et al., 2016b)).

Finally, rather than asymmetrically injecting the motion information into the appearance stream, bidirectional connections could be employed. Such processing could be realized for either additive or multiplicative interactions and is illustrated for the multiplicative case in Fig. 5.1e. In empirical evaluation, we show that such interactions yield inferior performance to the asymmetric case of injecting motion into appearance. We conjecture that this result comes about because the spatial stream comes to dominate the motion stream during training.

5.3.3 Temporal filtering with feature identity

Beyond very limited means for interaction between its processing paths, the original two-stream network also employed only a small temporal window (10 frames) in making its predictions, which subsequently were averaged over the video (Simonyan and Zisserman, 2014a). In contrast, many real world actions required larger intervals of time to be defined unambiguously (*e.g.* consider a “lay-up” in basketball). Thus, the second way that we improve on the two-stream architecture is to provide it with greater temporal support (*cf.* Chapters 3 & 4 for previous work with similar motivations).

We employ 1D temporal convolutions combined with feature space transformations initialized as identity mappings to achieve our goal. 1D convolutions provide a learning-efficient way to capture temporal dependencies, *e.g.* with far less overhead than LSTMs. Initialization of the feature transformations as identity mappings is appropriate when injecting into deep architectures, as any significant change in the network path would distort the (pretrained) model and thereby remove most of its representational power. Furthermore, preserving the feature identity is essential to preserve the design principles of residual networks. The corresponding kernels can be injected at any point in the network since they do not impact the information flow at initialization; however, during training they can adapt their representation under the gradient flow.

Formally, we inject temporal convolutional layers into the network that operate across C_l feature channels

$$\mathbf{x}_{l+1} = \mathbf{x}_l * \hat{W}_l + b_l, \tag{5.8}$$

where the biases b_l are initialized as $\mathbf{0}$ and $\hat{W}_l \in \mathbb{R}^{1\times1\times T\times C_l\times C_l}$ are temporal filters with weights initialized by stacking identity mappings between feature channels, $\mathbf{1} \in \mathbb{R}^{1\times1\times1\times C_l\times C_l}$, across time $t = 1 \ldots T$. Specifically,

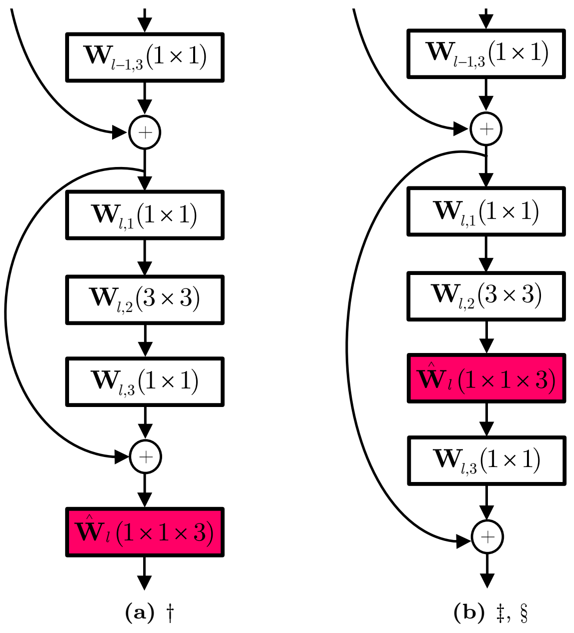
$$\hat{W}_l = \mathbf{1} \otimes \mathbf{f}, \tag{5.9}$$

where $\otimes$ denotes the tensor outer product and $\mathbf{f}$ is a 1D temporal filter of length T. Notably, eq. (5.9) initializes temporal kernels to perform identity transforms at feature level.

Since our kernels preserve the feature identity, we can place them after any layer in the network without affecting its representational ability (at initialization). During training, however, the newly added temporal conv layers affect the overall model. Here we distinguish two main variants, either inserting the layers in the shortcut path which directly affects all other layers in the network, or into the residual units which locally affects the surrounding blocks. These two variants for learning temporal relationships are illustrated in Fig. 5.3 and evaluated in our experiments (Section 5.5.2).

Recent ConvNet architectures (He et al., 2016a, Szegedy et al., 2015a) are fully convolutional and use a global average pooling after the last conv layer. Generally global pooling is reasonable, since the exponentially expanding receptive field for deeper units typically spans the whole input, *e.g.* for the ResNet-50 the last convolutional layer has a theoretical receptive field of 483×483 pixels on the 224×224 sized input.

By design, our temporal convolutional layers, (5.8), provide broad temporal support, analogous to the broad spatial support that motivates global spatial pooling. Therefore, it is equally motivated to employ global temporal pooling. Here, our intial investigations showed that global max pooling over time led to superior results compared to average pooling, presumably because it allowed the network to capitalize on the most discriminative temporal sample. Therefore, given $\mathbf{x}(i, j, t, c)$



$\mathbf{W}_{l-1,3}(1\times1)$
$\mathbf{W}_{l,1}(1\times1)$
$\mathbf{W}_{l,2}(3\times3)$
$\mathbf{W}_{l,3}(1\times1)$
$\hat{\mathbf{W}}_{l}(1\times1\times3)$
(a) †
$\mathbf{W}_{l-1,3}(1\times1)$
$\mathbf{W}_{l,1}(1\times1)$
$\mathbf{W}_{l,2}(3\times3)$
$\hat{\mathbf{W}}_{l}(1\times1\times3)$
$\mathbf{W}_{l,3}(1\times1)$
(b) ‡, §

Figure 5.3: Injection of identity mapping kernels as temporal filters, $\hat{W}_l$, into the skip path (a) or into the residual unit (b). Symbols †, ‡ and § are used in upcoming presentation to distinguish these cases.

observed over $1 \leq t \leq T$ we pool according to

$$\mathbf{x}(i,j,c) = \max_{1\leq t\leq T} \mathbf{x}(i,j,t,c). \tag{5.10}$$

During preliminary experiments, we also considered application of temporal max-pooling earlier in the network; however, it always produced inferior results to pooling after the last convolutional layer.

5.4 Architecture details

In our experiments we use 50 and 152 layer ResNets (He et al., 2016a), pretrained on ImageNet. The building blocks of our architecture are shown in Table 5.1 which reads from left to right, top to bottom. The operations at each block are convolution or pooling with dimensions $(W \times H, C)$, denoting width, height and number of feature channels, respectively. Each conv block is accompanied by batch normalization (Ioffe and Szegedy, 2015) and ReLU nonlinearities. The brackets indicate conv blocks that are grouped to residual units as outlined in equation (7.1). $\odot$ indicates the point of multiplicative motion gating into the appearance stream (Section 5.3.2.2) and the symbols ‡, § and † denote three variants of where to inject temporal convolutions (Sec. 5.3.3). In our ablation experiments we compare injecting temporal kernels into the skip path of every conv block (†), into a residual unit at every conv block (‡), or only into a residual unit at the last conv block (§); locally these variants are shown in Fig. 5.3.

Layers	conv1	pool1	conv2_x	conv3_x	conv4_x	conv5_x	pool5
Blocks	7×7, 64	3×3 max stride 2	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}$ $\odot$ $\begin{bmatrix}1\times1, 64\\3\times3, 64\ \ddagger\\1\times1, 256\end{bmatrix}$ $\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}$ †	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}$ $\odot$ $\begin{bmatrix}1\times1, 128\\3\times3, 128\ \ddagger\\1\times1, 512\end{bmatrix}$ $\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times N$ †	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}$ $\odot$ $\begin{bmatrix}1\times1, 256\\3\times3, 256\ \ddagger\\1\times1, 1024\end{bmatrix}$ $\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times M$ †	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}$ $\odot$ $\begin{bmatrix}1\times1, 512\\3\times3, 512\ \ddagger, \S\\1\times1, 2048\end{bmatrix}$ $\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}$ †	7×7 avg

Table 5.1: ResNet architecture used in our ConvNet streams. The layers are shown in the columns with brackets indicating residual units with skip connections. The filter dimensions are shown as $(W\times H, C)$ in brackets. $\odot$ denotes a multiplicative gate from the motion into the appearance stream, and the symbols ‡, § and † denote three alternatives for injecting the temporal filters. Notably for ‡ and § filters are injected only in the residual units (after a 3×3 convolution), whereas for † temporal filters also are injected into the shortcut path of the network. $N = 2$, $M = 4$ for a ResNet-50 and $N = 6, M = 32$ for a ResNet-152.

5.4.1 Training procedure

We first separately train the two streams similar to the previous chapters. We start with a learning rate of 10^{-2} and lower it two times after the validation error saturates. Our motion network uses optical flow stacks with $L = 10$ frames and is trained with dropout of 0.8 after the final classification layer. Optical flow (Zach et al., 2007) is precomputed prior to training and stored as JPEG images (with displacement vectors > 20 pixels clipped). During training, we use the same augmentations as in (Ballas et al., 2016, Wang et al., 2016a); *i.e.* randomly cropping from the borders and centre of the flow stack and sampling the width and height of each crop randomly as $W, H \in \{256, 224, 192, 168\}$, followed by resizing to 224×224. We use a batch size of 128 by randomly sampling a single optical flow stack from a video. The effective batch size during each forward and backward pass is reduced to fit GPU memory constraints and the gradient update is applied after aggregating gradients for all 128 samples. Notably, for batch normalization (Ioffe and Szegedy, 2015) (which is applied at every forward/backward computation), the batch-size is smaller. We found this fact facilitates generalization performance of our model, because smaller batches increase the regularization effect of the noisy bias/variance estimates in batch normalization.

The appearance stream is trained analogously with a batch size of 256 RGB frames. Here, we apply a less aggressive scale augmentation than for the motion network: We randomly jitter the width and height of the 224×224 input frame by ±25% and also randomly crop it from a maximum of 25% distance from the image borders. The crop is rescaled to 224×224 and passed as input to the network.

The same rescaling and cropping technique is chosen to train our proposed model. Here, we sample 5 inputs from a video with random temporal stride between 5 and 35 frames (*i.e.* temporal jittering for the injected temporal conv-layers). The batch size is set to 128 videos where 5 temporal

case	into	Fig.	UCF101	HMDB51
direct $\oplus$	$\leftarrow$	Fig. 5.1a	24.78	54.85
direct $\odot$	$\leftarrow$	Fig. 5.1b	81.98	77.89
residual $\oplus$ Section 5.3.2.1	$\leftarrow$	Fig. 5.1c	9.38	41.89
residual $\odot$ Section 5.3.2.2	$\leftarrow$	Fig. 5.1d	<u>8.72</u>	<u>37.23</u>
residual $\oplus$	$\rightarrow$	Fig. 5.1c	16.76	49.54
residual $\odot$	$\rightarrow$	Fig. 5.1d	16.68	48.43
residual $\odot$	$\leftrightarrow$	Fig. 5.1e	15.15	48.56

Table 5.2: Classification error (%) on the first split of UCF101 and HMDB51 under different cross-stream connections.

chunks are extracted from each one. Importantly, batch-normalization again uses a smaller batch size to fit GPU memory. The learning rate starts at 10^{-3} and is reduced by a factor of 10 after 20 epochs and again reduced by an order of magnitude after 10 epochs more.

5.4.2 Fully convolutional testing

We follow the evaluation procedure of the original two-stream work (Simonyan and Zisserman, 2014a) by sampling 25 frames (and their horizontal flips). However, we apply fully convolutional testing both spatially (smallest side rescaled to 256) and temporally (the 25 frame-chunks) by classifying the video in a single forward pass. For inference, we average the predictions of the classification layers over all spatiotemporal locations. Despite our slightly better results with 10-crop testing, we prefer fully convolutional testing in spacetime as this method greatly increases inference speed (a video can be tested in ≈250ms on a single Nvidia Titan X GPU).

5.5 Experimental results

As in the previous chapters, we again validate our approach on the two popular action recognition datasets: UCF101 (Khurram Soomro and Shah, 2012) and HMDB51 (Kuehne et al., 2011). Our experiments are structured into four sections. First, we present ablation experiments on how to connect the two streams (Section 5.5.1). Second, we compare different strategies to inject identity mapping kernels for learning long temporal relationships (Section 5.5.2). Third, we investigate the impact of network depth (Section 5.5.3). Finally, we provide a comparison with the state-of-the-art (Section 5.5.4).

5.5.1 Analysis of Two-Stream connections

In Section 5.3.2 we discuss two specific ways of how to connect the two streams of our architecture. More generally, there are numerous ways how one could insert cross-stream connections. The goal here is to fuse the two networks (at a particular convolutional layer) such that channel responses at the same pixel position are put in correspondence. In previous work, different fusion functions have been discussed Chapter 3 where it has been shown that additive fusion performed better than maxout or concatenation of feature channels from the two paths. Additive fusion of ResNets has been used in Chapter 4, but was not compared to alternatives. In this section, we provide a more systematic analysis on cross-stream connections between the streams.

In Fig. 5.1 we illustrate the various possible flows of information through the network and in Table 5.2 we compare their performance as error on the first splits of UCF101 and HMDB51. Note that Fig. 5.1 only illustrates the connection structure for a single layer; the cross-stream connections are inserted at every conv block (as marked in Table 5.1). Table 5.2 lists the type of connection (direct or into residual units), the fusion function (additive $\bigoplus$ or multiplicative $\bigodot$), the direction (from the motion into the appearance stream $\leftarrow$, conversely $\rightarrow$ or bidirectional $\leftrightarrow$). Note that we do not show schematic figures for fusing from the appearance into the motion stream ($\rightarrow$), as these simply are horizontal reflections of the converse ones ($\leftarrow$).

We first focus on straightforward connection of the streams with an additive shortcut connection, Fig. 5.1a, directly enabling forward and backward signal flow between the two paths. This strategy produces inferior results because it induces too large a change in the propagated signals, thereby disturbing the network's representation abilities. This overly aggressive change is induced in two ways: via the forwarded signal as it passes through the deep layers; via the backpropagated signal in all preceding layers that emit the fusion signal. Not surprisingly, the detriment is exacerbated when directly multiplying the shortcut paths of the two streams (Fig. 5.1b and second row of Table 5.2), because the change induced by multiplication is stronger than that from addition. More generally, injecting into the skip path breaks the identity shortcut of residual networks, thereby producing optimization difficulties (He et al., 2016b) and significantly increased test error, as verified by our results.

Next, we compare the additive and multiplicative motion gating into the residual units, as presented in Section 5.3.2.1 and Section 5.3.2.2, resp. Injection of residuals from the motion stream has been employed previously in Chapter 4 and produces test errors of 9.38% and 41.89% on the first splits of UCF101 and HMDB51, resp. In comparison, merely changing the interaction to multiplicative gating reduces that error to 8.72% and 37.23%. The impact of multiplicative motion interaction is twofold: First, it directly gates corresponding appearance residual units; second, it modulates the gradient in the appearance and the motion stream by each other's current input features, thereby enforcing spatiotemporal feature correspondences.

As a further experiment, we invert the direction of the connection to fuse from the appearance into the motion stream. This variation again leads to inferior results, both for additive and multiplicative residual fusion. These results can be explained by a severe overfitting of the network to appearance information. In fact, when fusing from the appearance into the motion stream ($\rightarrow$), the training loss of the motion stream decreases much faster and ends up at a much lower value. The fast training error reduction is due to the networks' focus on appearance information, which is a much stronger modality for discriminating different training frames. This effect is not only the reason for fusing from the motion stream into the appearance stream, but also supports the design of having two loss layers at the end of the network. Finally, we performed an experiment for bidirectional connections Fig. 5.1e, which also suffers under the effect of appearance dominating training.

Having verified the superior performance of multiplicative cross-stream residual connections, Fig. 5.1d, compared to the alternatives considered, we build on that design for the remainder of the experiments, unless otherwise noted.

5.5.2 Experiments on temporal aggregation

This section provides experiments for our injection of temporal filter kernels that preserve feature identity at initialization. We explore several choices for injecting such kernels within the hierarchy of the network. Table 5.3 again reports the error on the first split of UCF101 and HMDB51, as we vary operations in three ways. First, we vary where the temporal kernel is injected into overall architecture (see Table 5.1, §, †, ‡). Second, we vary the initialization of the temporal filter kernels, which is the same for all feature channels, by setting them to perform either averaging, [⅓, ⅓, ⅓], or centering, [0, 1, 0], in time. Third, we vary whether or not max-pooling in time is employed. Significantly, during training the network inputs consist up to 11 chunks that are temporally strided in the range between 5 and 35 frames; thus, these kernels are able to learn long-term temporal relationships between the features.

The results show a clear benefit of adding temporal kernels that are able to learn longer temporal relationships. Compared to the multiplicative cross-stream gating baseline, benefit is had even if just a single temporal layer is injected into each stream (§). Further, when comparing the temporal initialization of the filters, the results indicate particular benefit of a centre initialization on HMDB51. We conjecture this is due to the temporal nature of HMDB51 in comparison to UCF101: HMDB51 exhibits a higher degree of inter-video diversity *e.g.* due to camera motion, whereas videos in UCF101 typically capture temporally consistent scenes.

Finally, we investigate the impact of temporal max pooling of features before the classification layer. We notice a further decrease in error rates to 6.00% and 30.98% on UCF101 and HMDB51, respectively. Here, the relative gains again can be explained by the temporal natures of the datasets. Notably, even if max-pooling is performed after the last convolutional layer, the network will use information from all frames that are within the span of the temporal kernels. Therefore, max-pooling conceptually is a meaningful operation, because it allows the network to set the focus on a particularly discriminating instance in time, even while considering long-term information captured by stacked temporal conv-layers. This property also holds during backpropagation. Here, max-pooling only backpropagates a single temporal gradient map, even while the stacked temporal

case	temporal init.	pool time	UCF101	HMDB51
-	-	✗	8.72	37.23
§	[⅓, ⅓, ⅓]	✗	7.85	35.29
†	[⅓, ⅓, ⅓]	✗	7.72	35.96
‡	[⅓, ⅓, ⅓]	✗	<u>7.45</u>	35.94
†	[0, 1, 0]	✗	7.61	34.72
§	[0, 1, 0]	✗	7.74	34.90
‡	[0, 1, 0]	✗	7.79	<u>34.38</u>
†	[0, 1, 0]	✓	6.74	34.05
‡	[0, 1, 0]	✓	<u>6.00</u>	<u>30.98</u>

Table 5.3: Classification error (%) on the first split of UCF101 and HMDB51 under different temporal filtering layers. The symbols in the first column, §, †, ‡, denote where the new layers are placed within the overall architecture (see Table 5.1), temporal init indicates how the temporal filter taps are initialized (*i.e.* averaging or centre frame) and pool time indicates whether max-pooling in time is used during training and testing.

	UCF101		HMDB51	
Model	ResNet-50	ResNet-152	ResNet-50	ResNet-152
Appearance	82.3%	83.4%	48.9%	46.7%
Motion	87.0%	87.2%	55.8%	60.0%
Late Fusion	91.7%	91.8%	61.2%	63.8%

Table 5.4: Classification accuracy for 50 layer deep (ResNet-50), and extremely deep (ResNet-152) two-stream ConvNets on UCF101 and HMDB51. Using deeper networks boosts performance, except for the spatial network on HMDB51, which might be due to overfitting of the 152 layer model.

convolutions expand their receptive field on the gradient inversely from the output to the input. Therefore, long-range information is also used for gradient updates.

5.5.3 Going deeper

So far, all our ablation studies were conducted with a 50 layer deep ResNet model. We now switch from reporting classification error (that has been used in the ablation studies above) to accuracy, as this is the common measure in the action recognition literature. In Table 5.4 we report the accuracy for our two-stream baseline networks (no connections across streams or time) on all three splits the UCF101 and HMDB51 datasets. We train 50 and 152 layer models (He et al., 2016a). Classification is performed by averaging the prediction layer outputs from 25 uniformly sampled input video frames. Late fusion is implemented by averaging the prediction layer outputs. On HMDB51 we weight the temporal network scores by a factor of three before averaging. By comparing the performance we observe that the deeper appearance network degrades performance on HMDB51 (we were not able to ameliorate this effect with stronger regularization), while producing slightly better results on UCF101. In contrast, for the deeper motion network we see quite a sizable gain on HMDB51. These results motivate us to use a ResNet50 for the appearance and a ResNet-152 for the motion stream of our final architecture.

5.5.4 Comparison with the state-of-the-art

In comparison to the current state-of-the-art in video action recognition (Table 5.5), we consistently improve classification accuracy when comparing to the spatiotemporal ResNet presented in the previous Chapter and other competitors. Confusion matrices for our proposed spatiotemporal Multiplier Network can be found in appendix A.

Method	UCF101	HMDB51
Improved Dense Trajectories (IDT) (Wang and Schmid, 2013)	86.4%	61.7%
Spatiotemporal ConvNet (Karpathy et al., 2014)	65.4%	-
Two-Stream ConvNet (Simonyan and Zisserman, 2014a)	88.0%	59.4%
Long-term recurrent ConvNet (Donahue et al., 2015)	82.9%	-
Composite LSTM Model (Srivastava et al., 2015)	84.3%	44.0
Two-Stream+LSTM (Ng et al., 2015b)	88.6%	-
C3D (Tran et al., 2015a)	85.2%	-
C3D + IDT (Tran et al., 2015a)	90.4%	-
Dynamic Image Nets (Bilen et al., 2016)	76.9%	42.8 %
Dynamic Image Nets (Bilen et al., 2016) + IDT	89.1%	65.2%
Transformations(Wang et al., 2016a)	92.4%	62.0%
Two-Stream Fusion (Chapter 3)	92.5%	65.4%
Two-Stream Fusion (Chapter 3) + IDT	93.5%	69.2%
Long-term ConvNets (Varol et al., 2016)	91.7%	64.8%
Long-term ConvNets (Varol et al., 2016) + IDT	92.7%	67.2%
VideoLSTM + IDT (Li et al., 2016b)	92.2%	64.9%
Hierarchical Attention Nets (Wang et al., 2016b)	92.7%	64.3%
Key Volume Mining (Zhu et al., 2016)	93.1%	63.3%
RNN-FV (Lev et al., 2016) + C3D (Tran et al., 2015a) + IDT	94.1%	67.7%
Spatiotemporal ResNets (Chapter 4)	93.4%	66.4%
TSN (Wang et al., 2016) + IDT flow	94.2%	69.0%
Spatiotemporal ResNets (Chapter 4) + IDT	94.6%	70.3%
Spatiotemporal MulNets	**94.2%**	**68.9%**
Spatiotemporal MulNets + IDT	**94.9%**	**72.2%**

Table 5.5: Mean classification accuracy of the state-of-the-art on HMDB51 and UCF101.

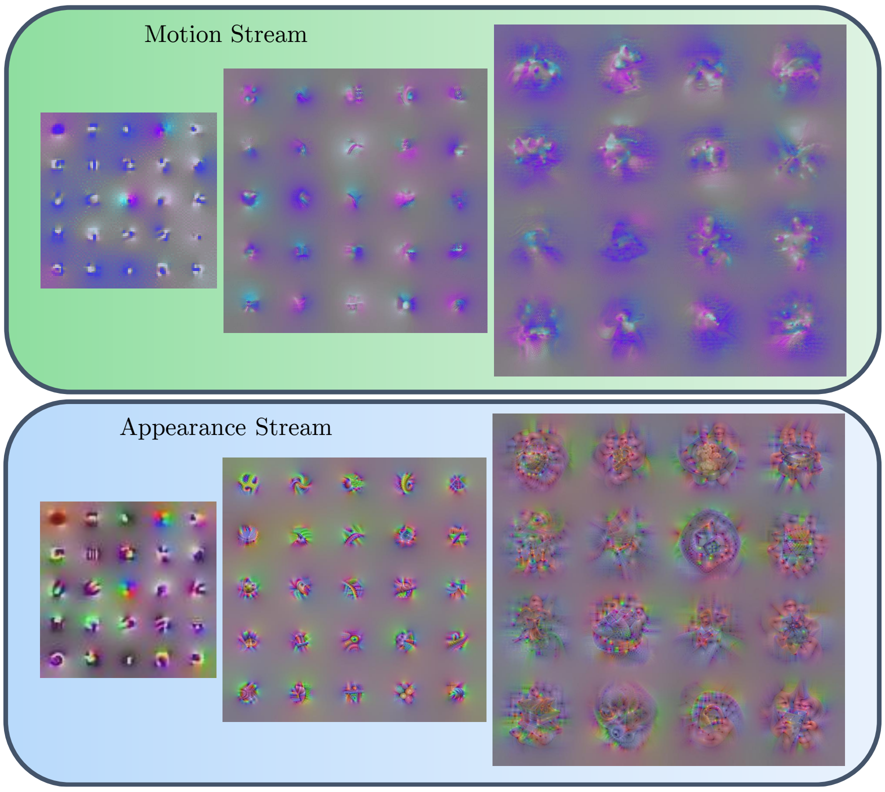
As a final experiment, we are interested if there is still something to gain from a fusion with hand-crafted IDT features (Wang and Schmid, 2013). We simply average the $L2$ normalized SVM scores of Fisher vector encoded IDT features (*i.e.* HOG, HOF, MBH) with the prediction layer output of our ConvNet model. The resulting performance is shown in the final row of Table 5.5. We achieve 94.9% on UCF101 and 72.2% on HMDB51. These results indicate that the degree of complementary between hand-crafted representations and our end-to-end learned ConvNet approach is vanishing for UCF101, given the fact that other representations see much larger gains by fusion with IDT. Nevertheless, there is still a 3.7% increase on HMDB51, which we think is mainly due to dominant camera motion in HMDB51 that is explicitly compensated by IDT's warped flow. Second, HMDB51 has a small training set with large intra-class variability which handicaps deep ConvNet representations; in contrast, both of these properties can be tackled by explicit Fisher vector encoding and SVM classifiers as in IDT.

An alternative approach that performs competitively to ours is TSN (Wang et al., 2016). Notably, however, TSN relies on 2 more input modalities for training than does our approach: It pretrains the motion stream on TVL1 optical flow (Zach et al., 2007), IDT-flow (Wang and Schmid, 2013) (termed warped flow in (Wang et al., 2016)) and difference images. This extra data yields much higher accuracy of their baseline two-stream net (*e.g.* 87.2% on UCF101's most difficult split1 in the motion stream where we have 84.9%). Thus, due to the implicit use of IDT during training, TSN (Wang et al., 2016) is not directly comparable to our results without IDT. Significantly, even given the advantage of using multiple input modalities for training a stronger baseline, the final TSN performance is merely equal to our results (without IDT).

Finally, given the large capacity of our models, we anticipate that our approach would even further improve when additional training data is used.

5.6 Summary

This chapter has addressed the challenging problem of learning multiscale spacetime representations for discriminative video recognition. We have presented a novel spatiotemporal architecture for video action recognition that builds on multiplicative interaction of appearance and motion features coupled with injected identity mapping kernels for learning long-term relationships. Our model is trained end-to-end and fully convolutional in spacetime to enable video classification in a single forward pass. In our systematic ablation studies we have underlined the importance of learning correspondences between highly abstract ConvNet features both spatially and temporally. Our results are in line with our derivations and suggest that our architecture should be applicable to related tasks such as localization and detection in video.



Example outputs of our visualization technique for filters at three convolutional layers of a two-stream ConvNet. In this chapter we propose an approach to visualize deep spatiotemporal representations to better understand what the underlying models are capturing.

6 Understanding Deep Video Representations

6.1 Motivation

Principled understanding of how deep networks operate and achieve their strong performance significantly lags behind their realizations. Since these models are being deployed to all fields from medicine to transportation, this issue becomes of even greater importance. The previous chapters have described our advances towards more effective architectures for recognizing actions in video and we have shown that significant strides towards higher accuracies could be made by deep spatiotemporal representations. We can understand our approaches from two viewpoints. First, the *architectural viewpoint* defines the network as a computational structure (*e.g.* a directed acyclic graph) of mathematical operations in feature space (*e.g.* affine scaling and shifting, local convolution and pooling, nonlinear activation functions, etc.) with defined analytic (or numeric) gradient for backpropagation. In the previous chapters, these tools have been structurally used to build architectures that are theoretically motivated. We can thus reason about their expected computation and the quantitative performance for a given task justifies their design, but overall the understanding from an architectural viewpoint is a very loose one as it does not explain how a network actually arrives at these results. The second way of understanding deep networks is the *representational viewpoint* that is concerned with the actually learned representation in the parameters of the network. Understanding these is inherently hard as networks consist of an ever higher number of parameters with a vast space of possible functions they can model. The hierarchical nature in which these parameters are arranged makes the task of understanding even harder especially for ever deeper representations. Due to their compositional structure it is difficult to explicitly reason about what these powerful models actually have learned. In this chapter we shed some light on deep spatiotemporal networks by visualizing what excites the learned models using activation maximization by backpropagating on the input. We are the first to visualize the hierarchical features learned by a deep motion network. Our visual explanations are highly intuitive and indicate qualitative evidence for the separation into two pathways for processing spatiotemporal information – a principle that has also been found in nature where numerous studies suggest a corresponding separation into ventral and dorsal pathways of the brain.

It would be interesting to investigate these deep models from a biological point of view. As already outlined in Section 2.3, research in neuroscience suggests different pathways in the hierarchical processing of visual information, namely the ventral ('what') and dorsal ('where') pathway (Felleman and Van Essen, 1991, Goodale and Milner, 1992, Mishkin et al., 1983). The ventral pathway is mostly tuned for appearance based perception, whereas the dorsal stream is involved in localization and movement recognition. Notably, there is evidence that these two pathways are not decoupled and there are numerous connections across the streams (Felleman and Van Essen, 1991, Goodale and Milner, 1992, Kourtzi and Kanwisher, 2000, Saleem et al., 2000); thus the two streams are not as segregated as has been hypothesized earlier (Maunsell and Van Essen, 1983, Mishkin et al., 1983, Ungerleider and Desimone, 1986). Information is exchanged between the multiple areas in the two paths, even a 'fusion stream' could be hypothesized that is intimately connected with both the dorsal and ventral streams and projecting into the rostral superior temporal sulcus (STS) – a region that shows stronger activations for voices *vs.* environmental sound, stories *vs.* nonsense speech, moving faces *vs.* moving objects, and biological motion (Gusnard and Raichle, 2001). Thus, the rostral part of the STS seems to be an interconnection between ventral and dorsal streams (Karnath, 2001). This study also speculates that the evolutionary development from monkey to human brain led to the lateralization of the superior temporal cortex functions that were bilateral in earlier forms (Karnath, 2001). Another rather new study proposes that the dorsal stream can be better categorized as 'how' instead of 'where' pathway, and that there are three further streams emerging from the dorsal pathway, underlying its multifaceted nature for tasks of motion perception, visually guided action and navigation (Kravitz et al., 2011).

In the light of this dissertation, an interesting field of consideration is also how motion-selective neurons are tuned and represented in the cortex. Research on macaque monkeys suggests that there are numerous cells that selectively see motion and flicker. Studies on the medial superior temporal area (MST) of the dorsal stream in the superior temporal sulcus (STS) of the extrastriate visual cortex identify neurons responding to small and large field motion (Maunsell and Van Essen, 1983, Ungerleider and Desimone, 1986). In the lateralventral (MSTl) region of the medial superial temporal area evidence for "small field object motion detectors" has been found, which are selective for visual motion caused by objects (Eifuku and Wurtz, 1998). Specifically, they report that 57% of the recoded MSTl neurons responded to center receptive field motion with a surround moving in the opposite direction and nearly 70% of the neurons responded for center motion while the surround being stationary.

There is also evidence for "large field motion detectors" (Duffy and Wurtz, 1991, Tanaka and Saito, 1989). That study found that the neurons in the dorsomedial region of the medial superior temporal area (MSTd) are directionally selective for moving visual stimuli, for three types of motion: planar (frontoparralel translations), circular (clockwise or counter-clockwise rotations) and radial motion (expanding motion patterns caused by ego-motion). MST neurons have large receptive fields and activate on pure optical flow stimuli. From all recorded neurons, 23% responded to a single motion type (planar, circular, radial expanding), 34% responded to two components planocircular or planoradial (but not circuloradial) and finally, 29% responded to all three motion components (Duffy and Wurtz, 1991). A later study suggests that the recorded MST neurons are resposible for detection of self-movement and disambiguating it from optic flow originating from the movement of large objects in the field of view of the observer (Duffy, 1998). For example, that motion responsive MT/MST areas in the brain are responding stronger to static images with implied animate dynamics into the containing objects (*e.g.* humans, animals) than on purely static images (Kourtzi and Kanwisher, 2000).

6.2 Related work

Activation maximization techniques. Activation maximization has been used by backpropagating on the input and applying gradient ascent to the input to find an image that increases the activity of some neuron of interest (Erhan et al., 2009). The method was employed to visualize units of Deep Belief Networks (DBNs) (Hinton et al., 2006) in (Erhan et al., 2009) and adopted for deep auto-encoder visualizations in (Le et al., 2012). The activation maximization idea was first applied to visualizing ConvNet representations trained on ImageNet (Simonyan et al., 2014). That work also showed that the activation maximization techniques generalize the deconvolutional network reconstruction procedure introduced earlier (Zeiler and Fergus, 2013), which can be viewed as a special case of one iteration in the gradient based activation maximization. In an unconstrained setting, these methods can exploit the full dimensionality of the input space; therefore, plain gradient based optimization on the input can generate images that do not reflect natural signals. Regularization techniques can be used to compensate for this deficit. In the literature, the following regularizers have been applied to the inputs to make them perceptually more interpretable: $L2$ norms (Simonyan et al., 2014), total-variation norms (Mahendran and Vedaldi, 2016), Gaussian blurring, and suppressing of low values and gradients (Yosinski et al., 2015), as well as spatial sifting (jittering) of the input during optimization, (Mordvintsev et al.).

Most recently, an even stronger natural image prior, generative adversarial networks (GANs) (Goodfellow et al., 2014) also have been used (Nguyen et al., 2016, 2017) to visualize class level representations. Activation maximization results produced by GANs offer visually impressive results, because these methods do not have to use extra regularization terms to prevent extremely high input signals, high frequency patterns or translated copies of similar patterns that highly activate some neuron, since the optimization is performed in a high-dimensional, abstract space (typically FC6 in AlexNet) that already induces strong regularization on the possible signals it could produce. In other words, GAN-based activation maximization does not start the optimization process from scratch, but from a high-level model that has been trained for the same or a similar task (Dosovitskiy and Brox, 2016). More specifically, (Nguyen et al., 2016) train the generator network on ImageNet and activation maximization in some target (ImageNet) network is achieved by optimizing a high-level code (*i.e.* FC6) of this generator network. Therefore, the produced result of this maximization technique is in direct correspondence to the generator, the data used to train this model, and not a random sample from the network under inspection. Since we are interested in the raw input that excites our representations, we do not employ any generative priors in this chapter.

Adversarial examples. In preliminary work to GANs, another related, irritating property of neural networks has been revealed: They can be "fooled" by applying a perturbation to the input that is hardly perceptible to humans (Szegedy et al., 2014). This input perturbation can be found by gradient ascent to maximize the network's prediction error. Most important, it has been shown that these so perturbed "adversarial examples" are not network, nor data specific, as the same perturbation can fool another network, which was trained on a different subset of the dataset (Szegedy et al., 2014) – This fact makes sense, since it is the basis for generalization of models trained by backpropagating gradients. Such examples suggest that high performance of a given model for a task it was trained for does not justify interpretability and means that we truly understand the models.

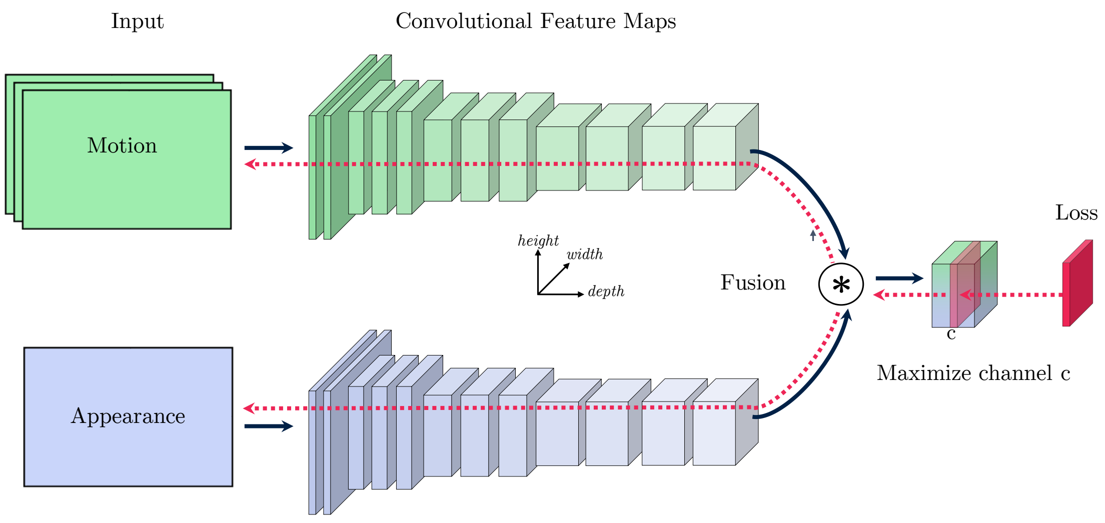


Figure 6.1: Schematic of our spatiotemporal visualization approach (see Section 6.3 for details).

6.3 Approach

There are several techniques that perform activation maximization for image classification ConvNets (Mahendran and Vedaldi, 2016, Simonyan et al., 2014, Szegedy et al., 2014, Yosinski et al., 2015). In this section, we build on the algorithm presented in (Mahendran and Vedaldi, 2016), since it provides visually most pleasing results while being conceptually simple, and extend it for the video domain. This section describes our method for finding the preferred input of a single unit in the model by maximizing its activation. We formulate the problem as a (regularized) gradient-based optimization problem that searches in the input space. An overview of our approach is shown in Fig. 6.1. First, a randomly initialized input is presented to the optical flow and the appearance towers of our model. We compute the feature maps up to a particular layer that we would like to visualize. This point could be at the last layer of the network where a neuron corresponds to the classification targets, or at any intermediate layer of the representation. A single target feature channel, c, is selected and activation maximization is performed to generate the preferred input as follows. By setting the loss for our target channel and applying backpropagation to compute gradients with respect to the input we can find the derivatives for the input to affect the target neuron, c. The gradient vector propagated to the input is scaled by the learning rate and added to the current input such that it minimizes the loss. This operation is illustrated by the dotted red line in Fig. 6.1. A gradient based optimization algorithm is used that performs these steps iteratively with an adaptively decreasing learning rate until the input converges. Note that during this optimization process the network weights are not altered, only the inputs receive changes. The detailed procedure is outlined in the remainder of this section.

6.3.1 Activation maximization

To make the above more concrete, activation maximization of unit c at layer l seeks an input $\mathbf{x}^* \in \mathbb{R}^{H\times W\times T\times C}$, with H being the height, W the width, T the duration, and C the color and optical flow channels of the input. We find $\mathbf{x}^*$ by optimizing the following objective

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmax}} \frac{1}{\rho_l^2 \hat{\mathbf{a}}_{l,c}} \langle \mathbf{a}_l(\mathbf{x}), e_c \rangle - \lambda_r \mathcal{R}_r(\mathbf{x}) \tag{6.1}$$

where $\mathbf{a}_l$ are the activations at layer l, e_c is the natural basis vector corresponding to the c^{th} feature channel, and $\mathcal{R}_r$ are regularization term(s) with weight(s) λ_r. To produce plausible inputs, the unit-specific normalization constant depends on ρ_l, which is the size of the receptive field at layer l (*i.e.* the input space), and $\hat{\mathbf{a}}_{l,c}$, which is the maximum activation of c recorded on a validation set.

6.3.2 Regularized optimization

Since the space of possible inputs that satisfy (6.1) is vast, and natural signals only occupy a small manifold of this high-dimensional space, we use regularization to constrain the input in terms of range and smoothness to better fit statistics of natural video signals. Specifically, we apply the following two regularizers, $\mathcal{R}_B$ and $\mathcal{R}_{TV}$, explicitly to the appearance and motion input of our networks.

As first regularizer, $\mathcal{R}_B$, we enforce a local norm that penalizes large input values

$$\mathcal{R}_B(\mathbf{x}) = \begin{cases} \sum_{i,j} \left(\sum_d \mathbf{x}(i,j,k,d)^2\right)^{\frac{\alpha}{2}} & \forall i,j,k,d : \sqrt{\sum_d \mathbf{x}(i,j,k,d)^2} \leq B \\ +\infty, & \text{otherwise.} \end{cases} \tag{6.2}$$

where i, j, k are spatiotemporal indices of the input volume and d indexes either color channels for appearance input, or optical flow channels for motion input, and B is the allowed range of the input. Similar norms are also used in (Mahendran and Vedaldi, 2016, Simonyan et al., 2014, Yosinski et al., 2015), with the motivation of preventing extreme input scales from dominating the visualization.

The second regularizer should penalize the high frequency content of our input, since natural signals tend to be dominated by low frequencies. We use a total variation regularizer (Mahendran and Vedaldi, 2016) that works on top of image gradients

$$\mathcal{R}_{TV2D}(\mathbf{x}) = \sum_{ijkd} \Big((\mathbf{x}(i+1,j,k,d) - \mathbf{x}(i,j,k,d)))^2 + (\mathbf{x}(i,j+1,k,d) - \mathbf{x}(i,j,k,d)))^2 \Big) \tag{6.3}$$

where i, j, k are used to index the spatiotemporal input dimensions and d indexes the color and optical flow channels. Note that a similar, more explicit regularizer has been used previously, where a 2D Gaussian blur kernel is applied after each iteration of maximization (Yosinski et al., 2015). Notably, (6.3) does not penalize variation over the temporal dimension, k.

For motion signals modeling the variation in the temporal dimension is a well studied principle in literature. For learning general representations from video in an unsupervised manner, minimizing the variation across time is seen both in biological, *e.g.* (Földiák, 1991, Wiskott and

Sejnowski, 2002), and artificial, *e.g.*, (Goroshin et al., 2015) systems. The motivation for such an approach comes about how the brain solves object recognition by building a stable, slowly varying feature space with respect to time (Wiskott and Sejnowski, 2002) in order to model temporally contiguous objects for recognition. The straightforward extension of (6.3) to the three dimensional spatiotemporal domain is given by

$$\begin{aligned}\mathcal{R}_{TV3D}(\mathbf{x},\kappa) = \kappa \sum_{ijkd} \Big(&(\mathbf{x}(i+1,j,k,d) - \mathbf{x}(i,j,k,d))^2 + (\mathbf{x}(i,j+1,k,d) - \mathbf{x}(i,j,k,d))^2 \\ &+ (\mathbf{x}(i,j,k+1,d) - \mathbf{x}(i,j,k+1,d))^2 \Big),\end{aligned} \tag{6.4}$$

where again i, j, k are used to index the input dimensions and d indexes the color and optical flow channels and κ is used for weighting the degree of spatiotemporal variation for reconstructing the input.

As so far developed, the high frequency regularizer, (6.4), is isotropic in the spatiotemporal domain, but this might not be desired. For example, what if we would want to visualize fast varying features in time that are smooth in space. For such a anisotropic case, (6.4) would bias the visualization to be smooth both in space and time, but not allow us to balance between the two dimensions. Therefore, we split up the regularization term (6.4) into separate regularizers that penalizes fast variation across space and time, explicitly

$$\begin{aligned}\mathcal{R}_{TV2D1D}(\mathbf{x},\chi) = \sum_{ijkd} \Big(&(\mathbf{x}(i+1,j,k,d) - \mathbf{x}(i,j,k,d))^2 + (\mathbf{x}(i,j+1,k,d) - \mathbf{x}(i,j,k,d))^2 \Big) \\ &+ \chi \sum_{ijkd} \Big((\mathbf{x}(i,j,k+1,d) - \mathbf{x}(i,j,k+1,d))^2 \Big),\end{aligned} \tag{6.5}$$

where χ is the slowness parameter that accounts for the regularization strength on the temporal gradients. By varying $0 < \chi < \infty$ we can selectively regularize with respect to the slowness of the features at the input. Moreover, by choosing $\chi = 0$ we can leave the temporal dimension unpenalized and produce reconstructions with unconstrained temporal gradients that only considers two-dimensional spatial variation as in (6.3) (which is also a special case of (6.4), occurring when the temporal regularization term is omitted). In the next section, such a temporally unconstrained TV regularizer is used for visualizing fast varying motion signals.

Overall, the regularization of the objective, (6.1), comes from

$$\mathcal{R}_r(\mathbf{x}) = \mathcal{R}_B(\mathbf{x}) + \mathcal{R}_{TV}(\mathbf{x}), \tag{6.6}$$

where

$$\mathcal{R}_{TV}(\mathbf{x}) = \begin{cases} \mathcal{R}_{TV2D}(\mathbf{x}), & \text{for appearance } \mathbf{x} \\ \mathcal{R}_{TVm}(\mathbf{x}), & \text{for motion } \mathbf{x}. \end{cases} \tag{6.7}$$

with $m \in \{2D, 3D, 2D1D\}$. Thus, $\mathcal{R}_r(\mathbf{x})$ serves to bias the visualizations to the space of natural images in terms of their magnitudes and spatiotemporal rates of change. Note that the three different variational regularizers for the motion input should allow us to reconstruct signals that are varying slowly in space (TV2D), uniformly in spacetime (TV3D) and non-uniformly in spacetime (TV2D1D).

For optimizing the overall objective (6.1), we directly adopt a variation of the AdaGrad algorithm (Duchi et al., 2011) from (Mahendran and Vedaldi, 2016) which adaptively scales the gradient updates on the input (as in AdaGrad), while aggregating the gradients in a sliding window over previous iterations (as in AdaDelta (Zeiler, 2012). In all the experiments shown in this chapter, we chose the regularization/loss trade-off factors λ_r to provide similar weights for the different terms (6.2) - (6.5); details are given in the next section.

6.4 Experiments

As noted above, we apply the regularizers separately to the optical flow and appearance input. The regularization terms for the appearance input are chosen to $\lambda_{B,\text{rgb}} = \frac{1}{HWB^\alpha}$ and $\lambda_{TV2D,\text{rgb}} = \frac{1}{HWV^2}$, with $V = B/6.5$, $B = 160$ and $\alpha = 3$, *i.e.* the default parameters in (Mahendran and Vedaldi, 2016). The motion input's regularization differs from the appearance one as follows. In order to visualize fast and slow motion inputs, we use different weights for the variational regularizer of the optical flow input as follows: First, for visualizing different speeds of motion signals, we use different weight terms for the variational regularizers of the motion input. In particular, to reconstruct different uniformly regularized spatiotemporal inputs (eq. (6.4)) we vary κ for penalizing the degree of spatiotemporal variation for reconstructing the motion input. Note that since optical flow is assumed to be smoother than appearance input, the total-variation regularization term of motion inputs has higher weight than the one for the appearance input. Therefore, also for using the anisotropic spatiotemporal regularizer, we set the weight to $\lambda_{TV2D1D,\text{flow}} = 10\lambda_{TV2D,\text{rgb}}$, and vary the temporal slowness parameter, χ. As for the isotropic spatiotemporal case aboce, we will chose discrete samples for χ corresponding to various variation degrees at the input (a continuum could be used). The special case corresponding to unconstrained temporal variation, $\chi = 0$ is implemented as a purely spatial regularizer (eq. (6.3)); *i.e.* $\lambda_{TV2D,\text{flow}} = 10\lambda_{TV2D,\text{rgb}}$. Note that the values in all visualizations are scaled to min-max over the whole sequence for effectively visualizing the full range of motion.

For the visualizations shown in the remainder of this chapter, we visualize units at multiple layers of the VGG-16 two-stream fusion model from Chapter 3 that is trained on UCF101.

6.4.1 Visualization of early layers

We first visualize filters of the early convolutional layers from the appearance and motion streams of the VGG-16 architecture. While we can simply visualize the appearance input as an rgb image showing the color channels, the question of what is the best way to visualize the optical flow stream arises. We do not use a variational regularizer in the temporal domain (*i.e.* the next visualizations correspond to the unconstrained temporal case, $\chi = 0$ above). In Fig. 6.3 we subsample five convolutional layers from VGG-16 (conv1_2 to conv5_3) and show different optical flow visualization encodings. We compare three alternatives for visualizing the optical flow stream: A magnitude plot that plays the optical flow inputs with the rgb channels representing horizontal, vertical and magnitudes of the flow; the classic optical flow encoding used in several optical flow benchmarks that encodes flow in the HSV color space, as explained in Fig. 6.2; and the horizontal and vertical optical flow vectors plotted as grayscale videos. Visualizations for all convolutional layers of the VGG-16 appearance and motion streams are shown in Appendix B.1.

In Fig. 6.3 we observe that, despite similar ImageNet initialization, the two network streams learn to represent different properties of the spatiotemporal signals. Nevertheless, if one looks closely, the spatial structures of the filters for appearance and motion show some similarities, despite being trained separately from different input modalities. In other words, there is no connection between appearance and motion filters during training, but still the filters share similarities in their spatial structures. This might also indicate why the sum fusion architecture (in Chapters 3 & 4) and multiplicative fusion (in Chapter 5) architectures, that are initialized from the visualized representation, provide strong performance. Finally, we also observe that temporal variation increases when going deeper in the network hierarchy. For example, in Fig. 6.3, the bottom-right filters at conv1_2 and conv2_2 show no temporal variation whereas we see increased temporal variation at the deeper network layers (*e.g.* at conv3_3).

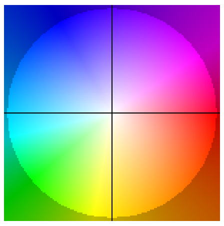


Figure 6.2: Flow field coding for the figures presented in this thesis. The displacement of a point corresponds to the vector from the centre of the figure, marked with a cross.

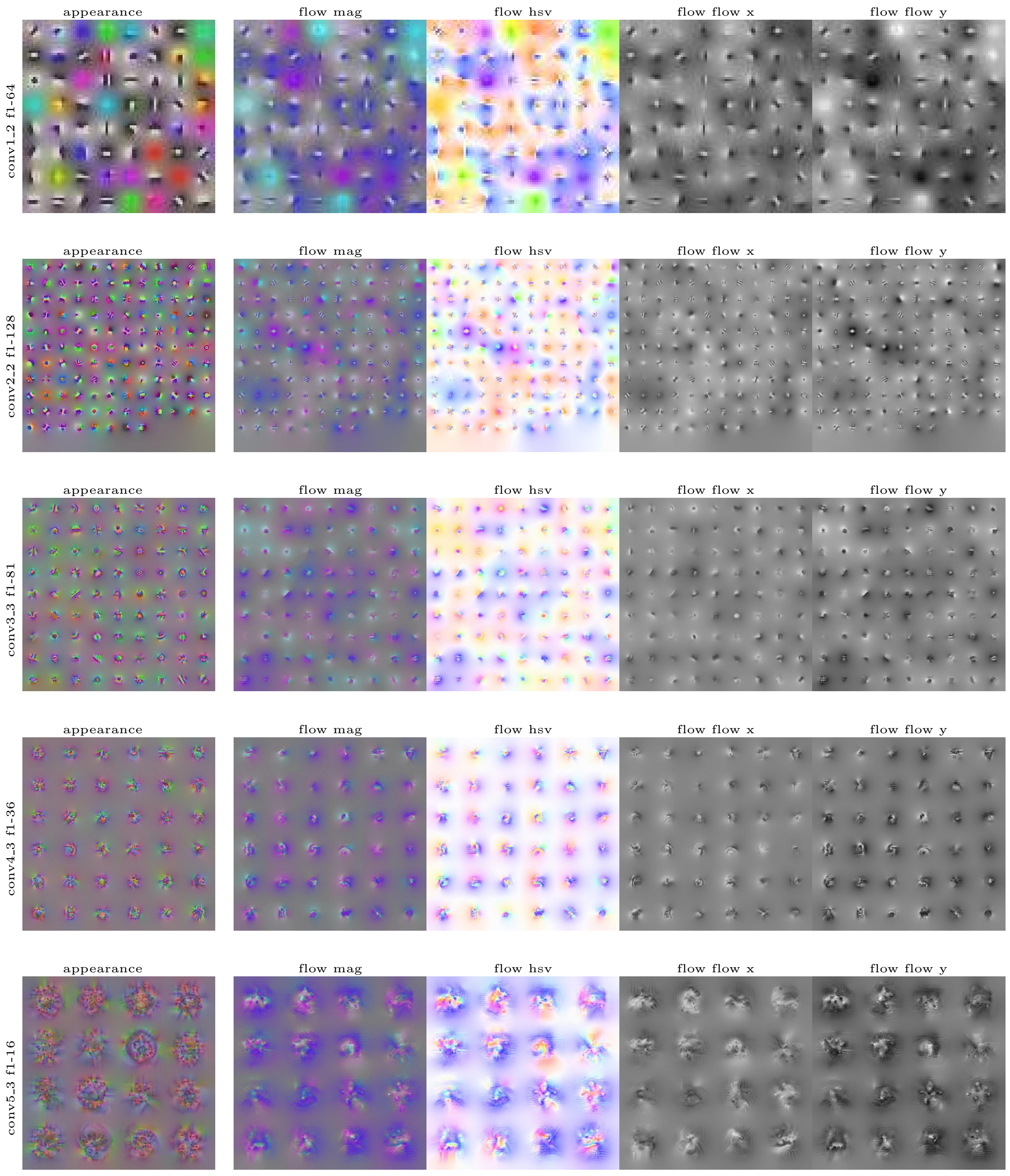


Figure 6.3: Different visualizations of the motion input for five convolutional layers of VGG-16 from conv1_2 to conv5_3. Similar spatial patterns are seen in both input modalities; further, temporal variation increases with network depth.

6.4.2 Visualization of fusion layers

Next, we move to the last convolutional layer of the VGG-16 fusion architecture, specifically to the conv5-fusion layer that fuses the appearance and flow features. We pick out a single interesting neuron (namely, filter #021) and visualize it for two spatiotemporal variation cases at the input, *i.e.* $\mathcal{R}_{TVm}(\mathbf{x}) = \mathcal{R}_{TV3D}(\mathbf{x})$ regularization for slow spatiotemporal variation and $\mathcal{R}_{TVm}(\mathbf{x}) = \mathcal{R}_{TV2D}(\mathbf{x})$ regularization for fast temporal variation (the temporal dimension is unconstrained). Fig. 6.4 shows unit f021 that we think is related to the Billiards class in UCF101. We observe that this neuron is fundamentally different in the slow and the fast motion case: It looks for linearly moving circular objects in the slow spatiotemporal variation case, while it looks for an exploding, accelerating motion pattern into various directions in the temporally unconstrained (fast) motion case. Thus, it appears that this neuron is able to detect particular spatial patterns of motion,

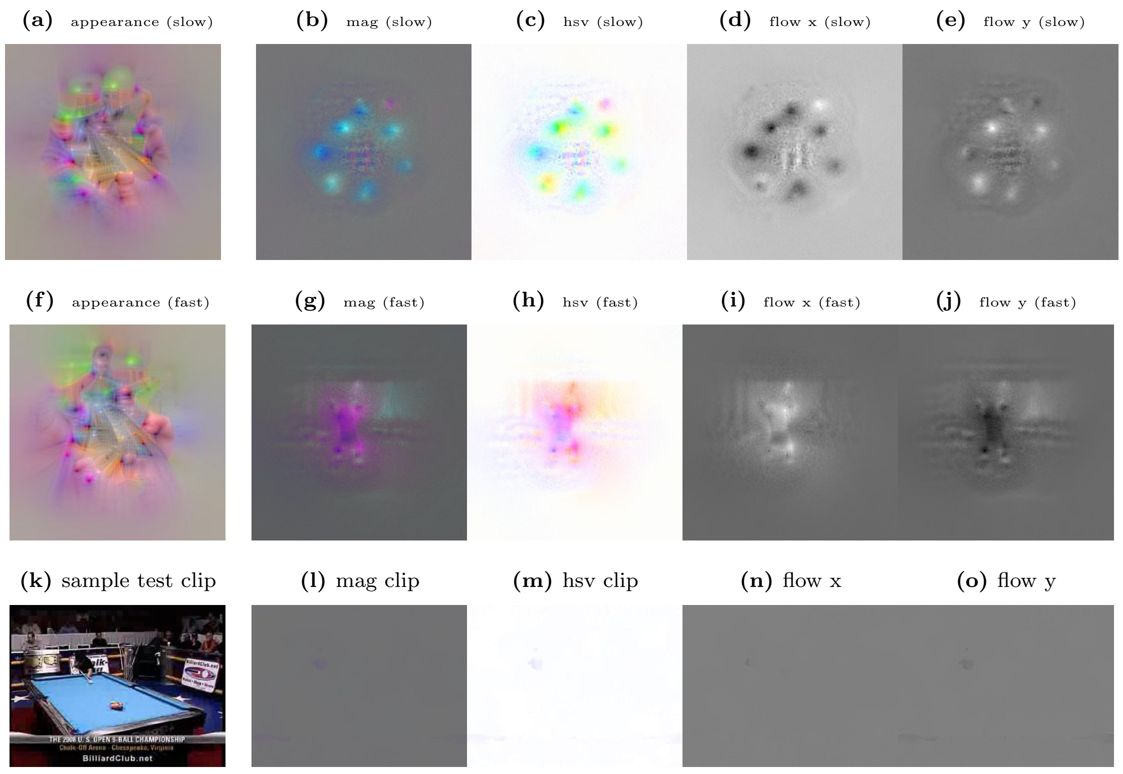


Figure 6.4: Studying a single neuron (#21) at layer conv5_fusion of a two-stream VGG-16 architecture from Chapter 3. In the first row we show what highly activates the filter in the rgb input (a) space and in the optical flow input with different forms of presentation (b) rgb image as horizontal, vertical and magnitude; (c) the directional encoding in hsv space; horizontal (d) vertical flow (e) as grayscale brightness plots. The second row shows what excites the filter when there is no restriction on the temporal variation of the input. In the last row we see a sample clip from the test set of UCF101 and the corresponding optical flow.

while allowing for a range of speeds and accelerations. Such an abstraction presumably has value in recognizing an action class with a degree of invariance to exact manner in which it unfolds across time. Another interesting fact is that switching the regularizer for the motion input also has an impact on the appearance input, even though the regularization for appearance is held constant to $\mathcal{R}_{TV2D}(\mathbf{x})$. This fact empirically verifies that the fusion unit also expects specific appearance input when confronted with differently varying motion signals in time.

6.4.3 Visualization under varying spatiotemporal regularization

In the previous section we have seen that some filters might correspond to vastly different acceleration patterns. We now explore this dimension further. For this purpose, we again fall back to a single visualization style of optical flow, the magnitude plots, which show rgb images and the channels corresponding to the horizontal, vertical and magnitudes of the optical flow vectors (*i.e.* the first flow visualization variant, mag, in the above figures) as we think that this style is perceptually easier to interpret than the other variants. Consequently, in Fig. 6.5 we show the appearance input as an rgb image, while the motion input is shown as a video that plays the optical flow inputs with the rgb channels representing horizontal, vertical and magnitudes of the flow. The motion signals are reconstructed under spatiotemporal, $\mathcal{R}_{TV3D}(\mathbf{x}, \kappa)$, regularization with different regularization strengths, κ, in the isotropic TV norm (6.4). Varying the regularization in spacetime reveals interesting properties of the underlying representation. We discuss Fig. 6.5 from two perspectives: First from the temporal perspective, we see that the first layer filters are more robust to regularization in spacetime, whereas higher layers show larger dependency on the regularization strength, κ. We think that this originates from the temporally consistent nature of the first layer filters (*e.g.* at conv2_2), which exhibit temporal low-pass characteristics. Second, from the spatial perspective, we see that with decreasing the spacetime regularization strength, κ, high-frequency inputs become dominant. Especially for low regularization factors $\kappa \leq 2.5$, we see high-frequency patterns dominating and reconstruction artifacts appearing in the background. These artifacts can be explained by the linear nature in higher dimensional spaces and also are known as adversarial examples (Goodfellow et al., 2015) when used for fooling a classifier.

Next, we switch the regularization term to $\mathcal{R}_{TV2D1D}(\mathbf{x}, \chi)$, (6.5), that anisotropically induces smoothness in space and time. Specifically, the term regularizes at a constant rate across space and varies according to the temporal regularization strength χ over time. In Fig. 6.6 we again show the appearance input as an rgb image and the motion input in the same color encoding as above. Our anisotropic regularizer reveals the robustness to temporal variation of the filters at all convolutional layers of the base VGG-16 two-stream architecture. We see that the spatial patterns are preserved throughout various temporal regularization factors χ, at all layers. From the temporal perspective, we see that, as expected, for decreasing χ the temporal variation increases; interestingly however, the direction of the motion patterns are preserved while the spatial scale of the motion increases with increasing χ. For example, consider the last shown unit f36 of layer conv4_3 (bottom right filter in the penultimate row of Fig. 6.6). This filter looks for accelerating motion blobs in an upward facing direction direction, *e.g.*, a persons' head that moves upwards in an accelerating manner when he/she is standing up. In the temporally regularized case, $\chi > 0$, this acceleration is smaller compared to the fast acceleration pattern seen in the temporally unconstrained case, $\chi = 0$. Notably, all these motion patterns strongly excite the same unit. Based on these observations, we postulate that the underlying representation in the motion network builds invariances to the spatial displacement (*i.e.* changes in magnitude of the motion vector) while preserving the coarse direction of the motion.

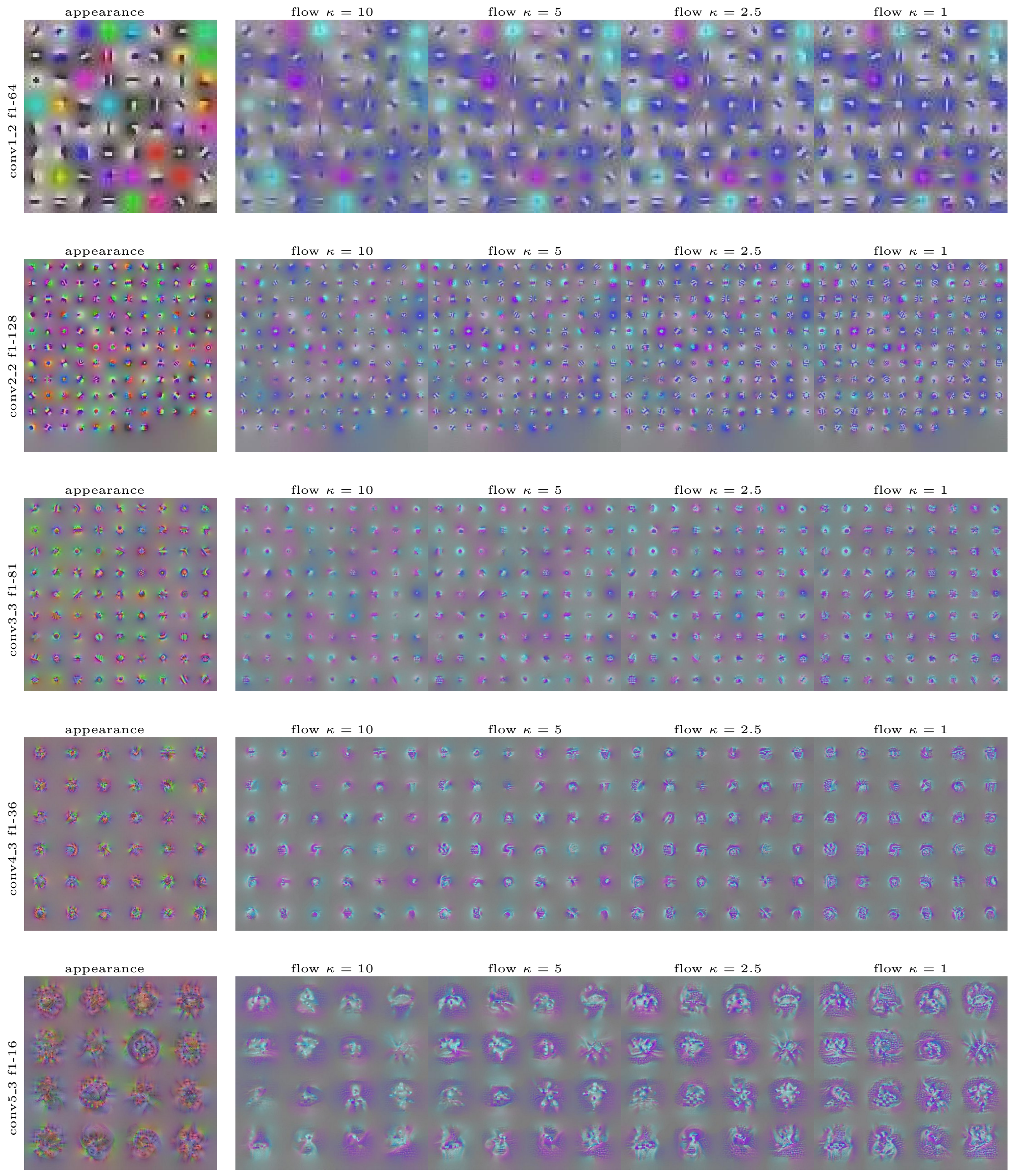


Figure 6.5: Visualization of two-stream conv filters under diverse $\mathcal{R}_{TV3D}(\mathbf{x}, \kappa)$ spacetime TV regularization. We show appearance and the optical flow inputs for slowest $\kappa = 10$, slow $\kappa = 5$, fast $\kappa = 2.5$, and faster $\kappa = 1$, spatiotemporal variation. Best viewed electronically, with zoom.

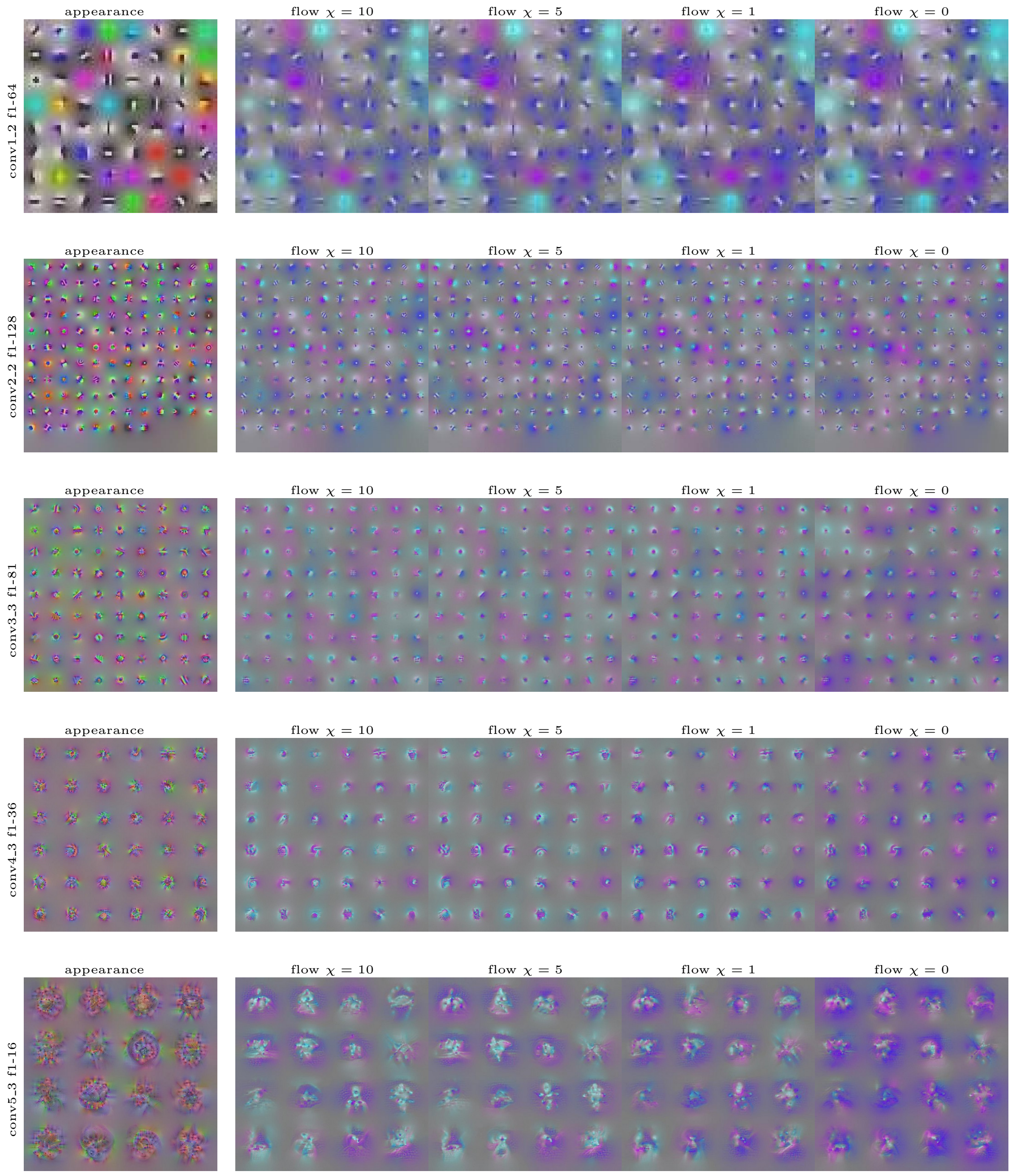


Figure 6.6: Visualization of two-stream conv filters under diverse $\mathcal{R}_{TV2D1D}(\mathbf{x}, \chi)$ temporal TV regularization. We show appearance and the optical flow inputs for slowest $\chi = 10$, slow $\chi = 5$, fast $\chi = 1$, and unconstrained (fastest) $\chi = 0$, spatiotemporal variation regularization. Spatial 2DTV regularization is held constant. Best viewed electronically, with zoom.

Now, we move on to the convolutional fusion layer (see Chapter 3), that takes in the features from the appearance and motion stream and learns a local fusion representation for the subsequent higher level layers with global receptive fields. Therefore, this layer is of particular interest as it is the first point in the network forward pass where appearance and motion information come together. Below in Fig. 6.7, we show the filters at the conv5_fusion_a layer, which fuses from the motion into the appearance stream, for multiple spatiotemporal 3D TV regularization strengths. This is again achieved by varying the parameter κ in the isotropic TV norm (6.4). The visualizations reveal that these first 6 fusion filters at this last convolutional layer show reasonable combinations of appearance and motion information, a qualitative proof that our proposed fusion model in Chapter 3 performs as expected. For example the center of the receptive field of conv5_fusion_a f002 could activate for some music instrument (*e.g.* a flute) with the motion corresponding to movement of the instrument; the upper and central surrounding region of the receptive field seems to react on skin-colored mouth-shaped inputs that are relatively static in the motion input. We also observe that the units are tuned to various degrees of spatiotemporal input variation (all the different inputs highly activate the same given unit).

Next, in Fig. 6.8 we show visualizations for varying the temporal regularization only, while keeping the spatial regularization constant. This is again achieved by varying the parameter χ in the anisotropic TV norm (6.5). Here, we see further evidence that the spatial pattern of the motion remains relatively consistent across varying values of the temporal regularization parameter, χ. Overall, it is best to view the two regularization forms in a two-page, side by side, view to see the full variability of each local filters (in its 8 different visualizations shown).

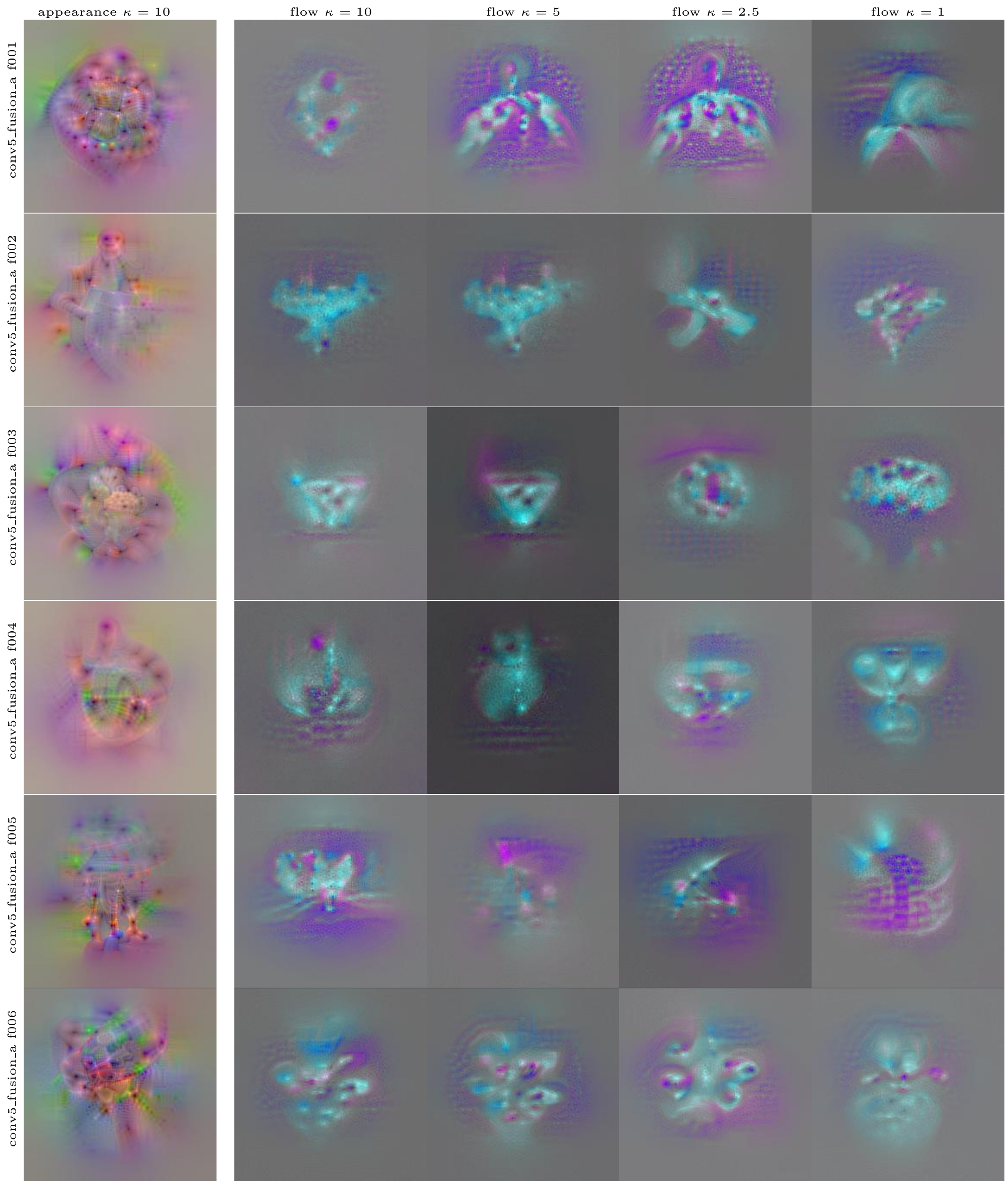


Figure 6.7: Visualization of 6 filters of the conv5_fusion (appearance stream) layer under different 3D spacetime TV regularization. We show appearance and the optical flow inputs for slowest $\kappa = 10$, slow $\kappa = 5$, fast $\kappa = 2.5$, and faster $\kappa = 1$, spatiotemporal variation.

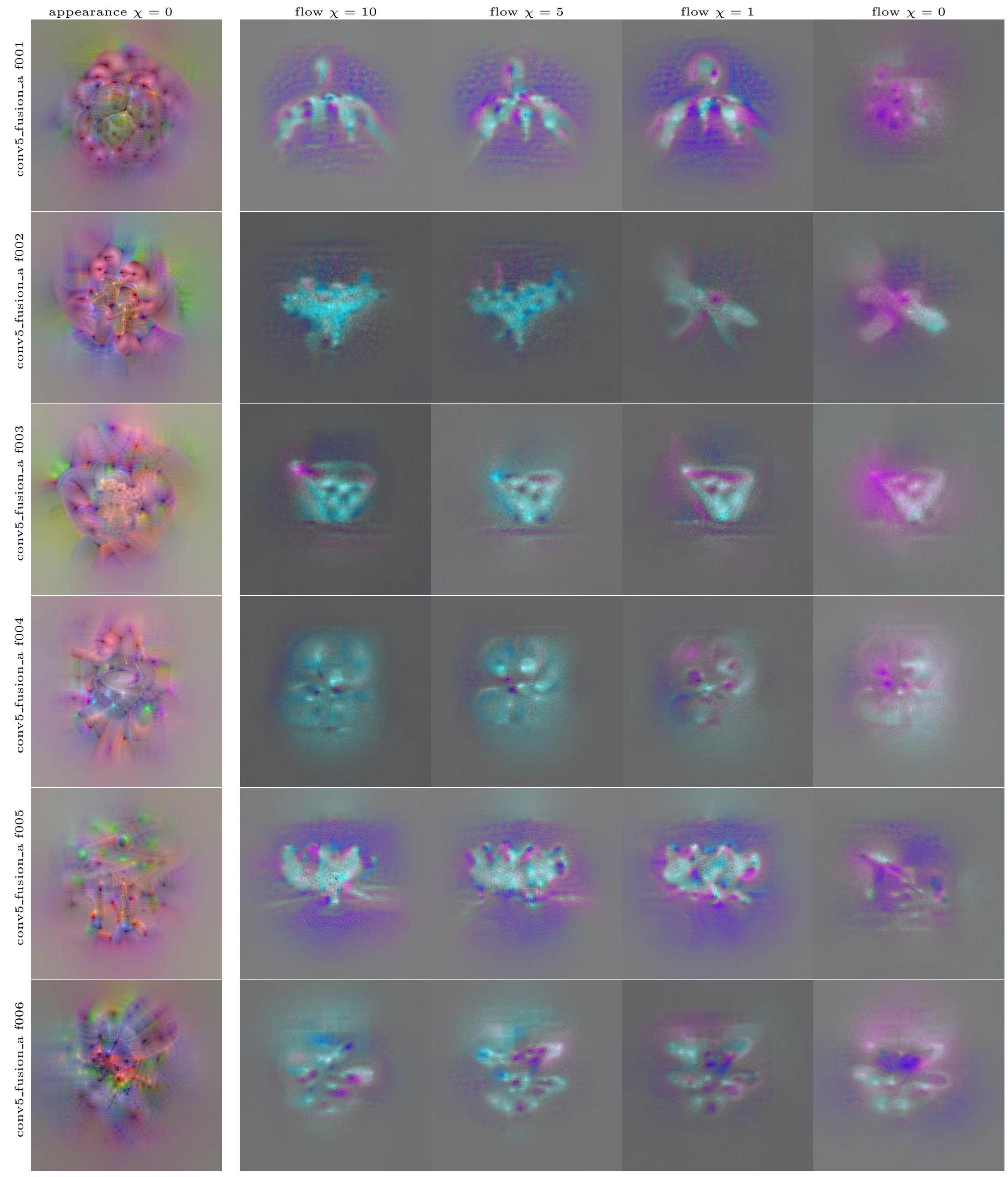


Figure 6.8: Visualization of 6 filters of the conv5_fusion (appearance stream) layer under different temporal TV regularization. We show the appearance input and the optical flow inputs for slowest $\chi = 10$, slow $\chi = 5$, fast $\chi = 1$, and unconstrained $\chi = 0$, temporal variation regularization.

We now pick a single neuron from the above Figures 6.7 & 6.8 and discuss the differences of TV regularization in spacetime. We focus on unit f004 at conv_fusion5, as it seems to capture some drum-like structure in the center of the receptive field, with skin-colored structures in the upper region. This unit could correspond to the PlayingTabla class in UCF101. In Fig. 6.9 we show the unit under the different spacetime regularizers for the motion stream (*i.e.* the fourth row in Figures 6.7 & 6.8) and also show sample frames from three PlayingTabla videos from the test set. First, we point an interesting observation about the interactions between appearance and motion input. In particular, we see that changing the motion regularization also has an effect on the appearance reconstruction per-se. Our primary hypothesis is that the neuron expects specific appearance for specific speeds. More concretely, when comparing the appearance inputs for unconstrained temporal variation in the motion input, *i.e.* $\mathcal{R}_{TV2D1D}(\mathbf{x}, \chi)$, with $\chi = 0$, to the appearance reconstruction under spatiotemporal, $\mathcal{R}_{TV3D}(\mathbf{x}, \kappa)$, motion regularization $\kappa = 10$, we see that the appearance inputs differs slightly. This means that when maximizing for a different spatiotemporal scale at the motion input, the appearance input that maximally activates the unit is changing. This is reasonable as a fast moving object (that might be captured at conv5) would have slightly different appearance than a slowly moving object (one could even interpret that the model hallucinates motion blur in the temporally unconstrained case $\chi = 0$, compared to the spatially crisper looking appearance examples for spacetime TV regularization on the motion input with $\kappa = 10$). An alternative explanation for the subtle appearance difference could be that the optimization process invests more signal energy into the motion input for the temporally unconstrained case, $\chi = 0$, and therefore the appearance reconstruction is less saturated.

When looking at the motion input for maximizing this unit, we observe it activates for the drumming motion at different spatiotemporal scales. At strong spatiotemporal input regularization ($\chi, \kappa = 10$), the unit looks for linear motion of the person's head (*e.g.* at $\kappa = 10$ the red blob in the flow field activates for horizontally moving circular structures) and body posture, whereas for increased temporal variation ($\chi, \kappa < 10$), we observe that the neuron seeks for drumming like motion of the hands. In terms of regularization comparison, we see that the isotropic spacetime TV norm, (6.4) $\mathcal{R}_{TV3D}(\mathbf{x}, \kappa)$ leads to spatially dissimilar optical flow patterns for varying the regularization strength, κ. On the other hand, the anisotropic spacetime TV norm, (6.5) $\mathcal{R}_{TV2D1D}(\mathbf{x}, \chi)$, produces more coherent spatial optical flow patterns. This is reasonable, since under isotropic spacetime regularization the network is optimizing in a constrained space in 3D, whereas for the anisotropic case, variation of $\mathbf{x}$ is separately restricted in space (2D) and time (1D); a lower dimensional subspace that limits the number of possible input functions.

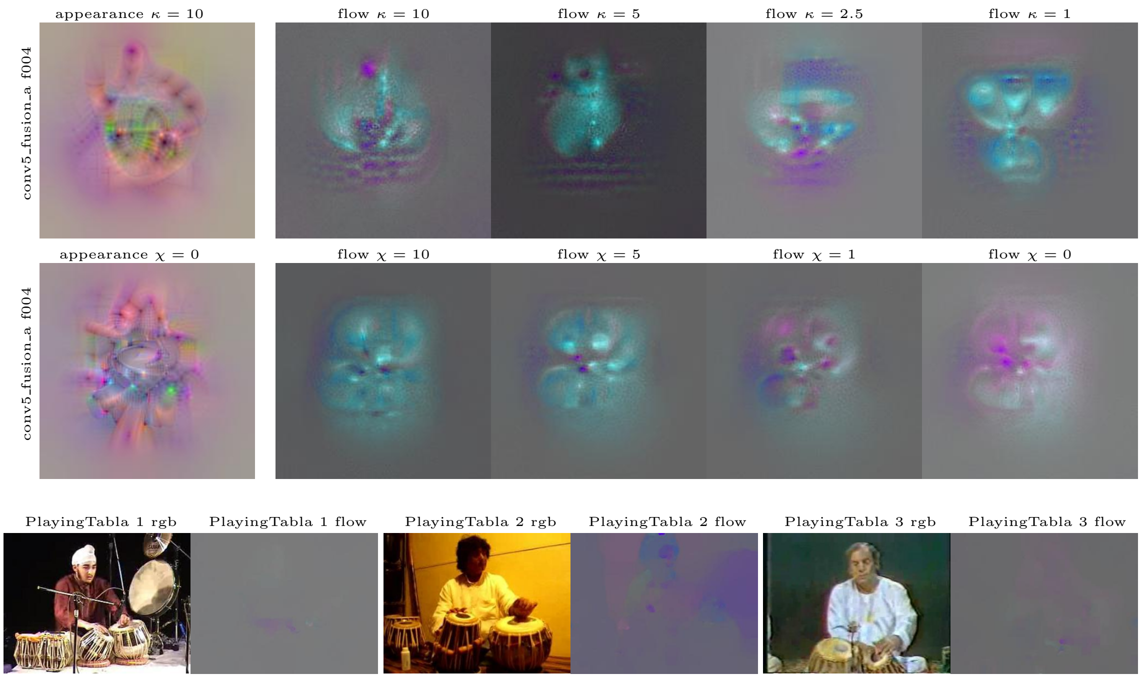


Figure 6.9: Specific unit at conv5_fusion into the appearance stream. Comparison between isotropic and anisotropic spatiotemporal regularization for a single filter at the last convolutional layer. The columns show the appearance and the motion input generated by maximizing the unit, under different degrees of isotropic spatiotemporal (κ) and anisotropic spatiotemporal TV regularization (χ). The last row shows appearance and optical flow of 40 sample frames from three videos of the PlayingTabla class in the test set.

Compared to the specific example of above, we now show two units from the conv5_fusion_a layer, that fuses from the motion into the appearance stream, to discuss generality of these units for representation across classes. Namely, the filters f006 and f009 shown in Fig. 6.10 seem to capture general spatiotemporal patterns for recognizing classes such as YoYo and Nunchucks, as seen when comparing the units visualization to the sample videos from the test set. Next, in Fig. 6.10, we similarly show general feature examples for the conv5_fusion_m layer, that fuses from the appearance into the motion stream. Namely, the filters shown in Fig. 6.11 seem to capture general spatiotemporal patterns for recognizing classes corresponding to multiple ball sport actions such as Soccer or TableTennis. These visualizations reveal that at the last convolutional layer, the network builds a representation that is on the one hand distributed over multiple classes, but on the other hand can also be quite specifically tuned to a particular class (the specific case in Fig. 6.9 above). Overall, we observe a consequent increase of class-specificity when moving from the features at close to the network input to higher layers, in the same manner as we observe invariances to classification-irrelevant factors building up in the network hierarchy. Additional visualizations showing multiple specific and general filters at the local conv5_fusion_a and conv5_fusion_m layers, can be found in Appendix B.2, where more of the multi-modal nature of these units is shown. Of particular note is the diversity of the features even for the same unit; for example filter conv5_fusion_m f020, shown in Fig. B.17 acts as a cake candle detector under irratic camera motion.

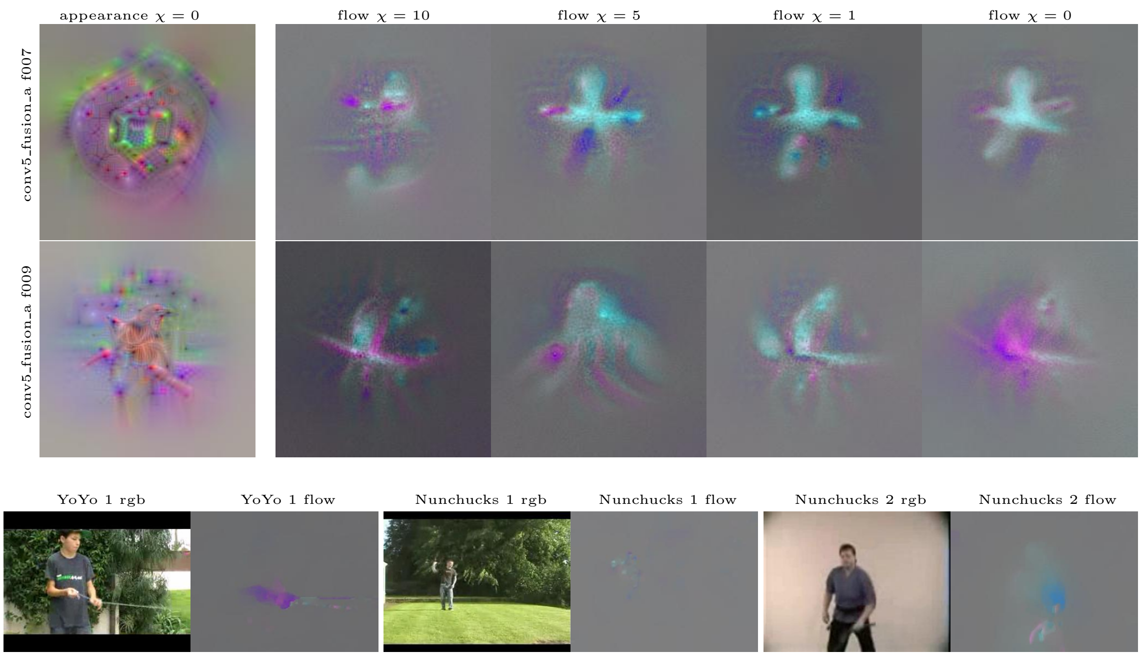


Figure 6.10: Two general units at the convolutional fusion layer. The columns show the appearance and the motion input generated by maximizing the unit, under different degrees of anisotropic spatiotemporal regularization (χ). The last row shows appearance and optical flow of 40 sample frames from three videos of the YoYo and Nunchucks classes in the test set.

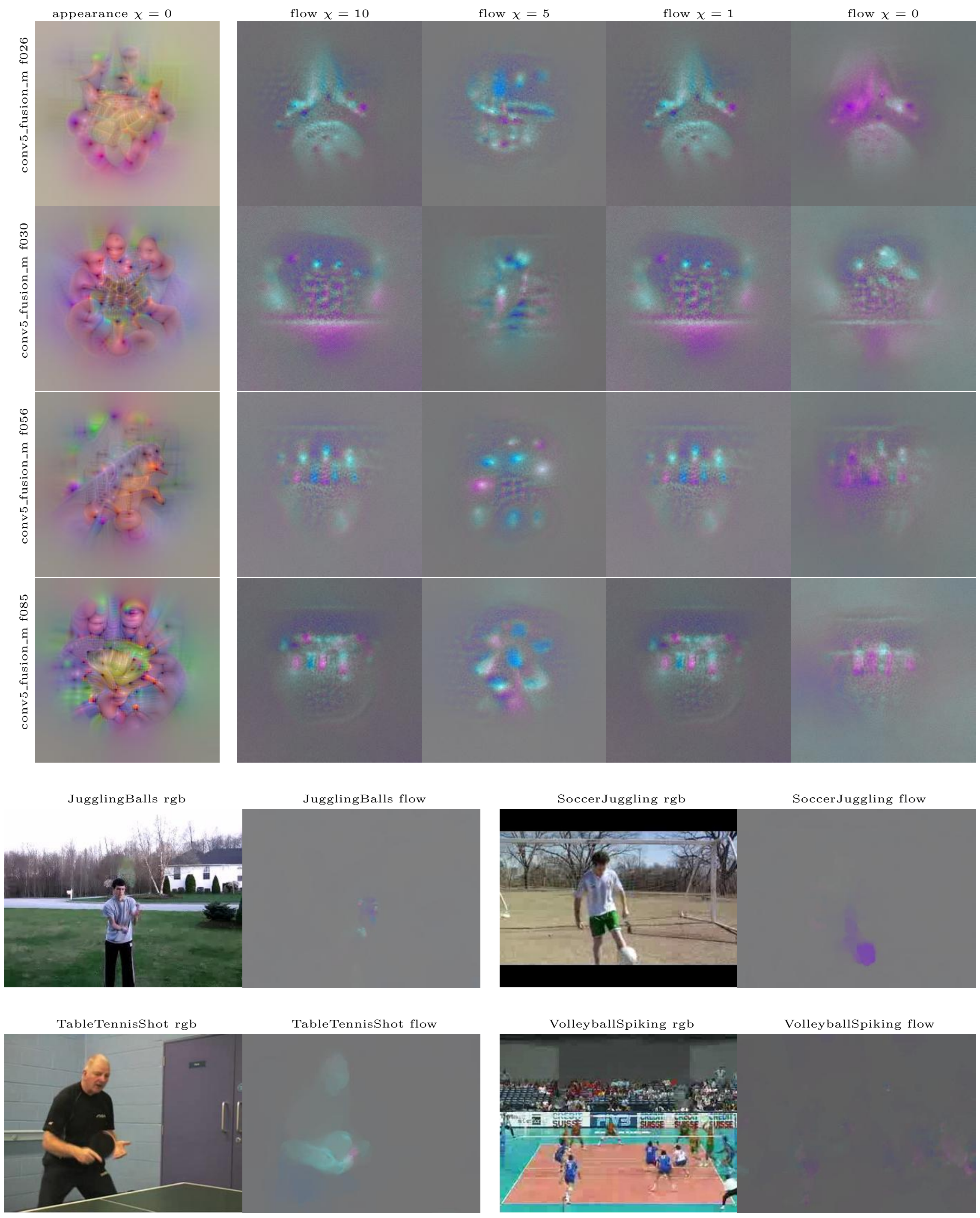


Figure 6.11: Four general units at the convolutional fusion layer that could be useful for representing ball sports. The columns show the appearance and the motion input generated by maximizing the unit, under different degrees of temporal regularization (χ). The last row shows appearance and optical flow of 40 sample frames from the test set.

6.4.4 Visualization of global layers

We now visualize the layers that have non-local filters, *e.g.* fully-connected layers that operate on top of the convolutional fusion layer illustrated above. Fig. 6.12 shows the first five filters of the fully-connected 6 (fc_6) layer in the motion stream of the VGG-16 fusion architecture. In contrast to the local features above, we observe a holistic representation that consists of a mixture of the local units seen in the previous layer.

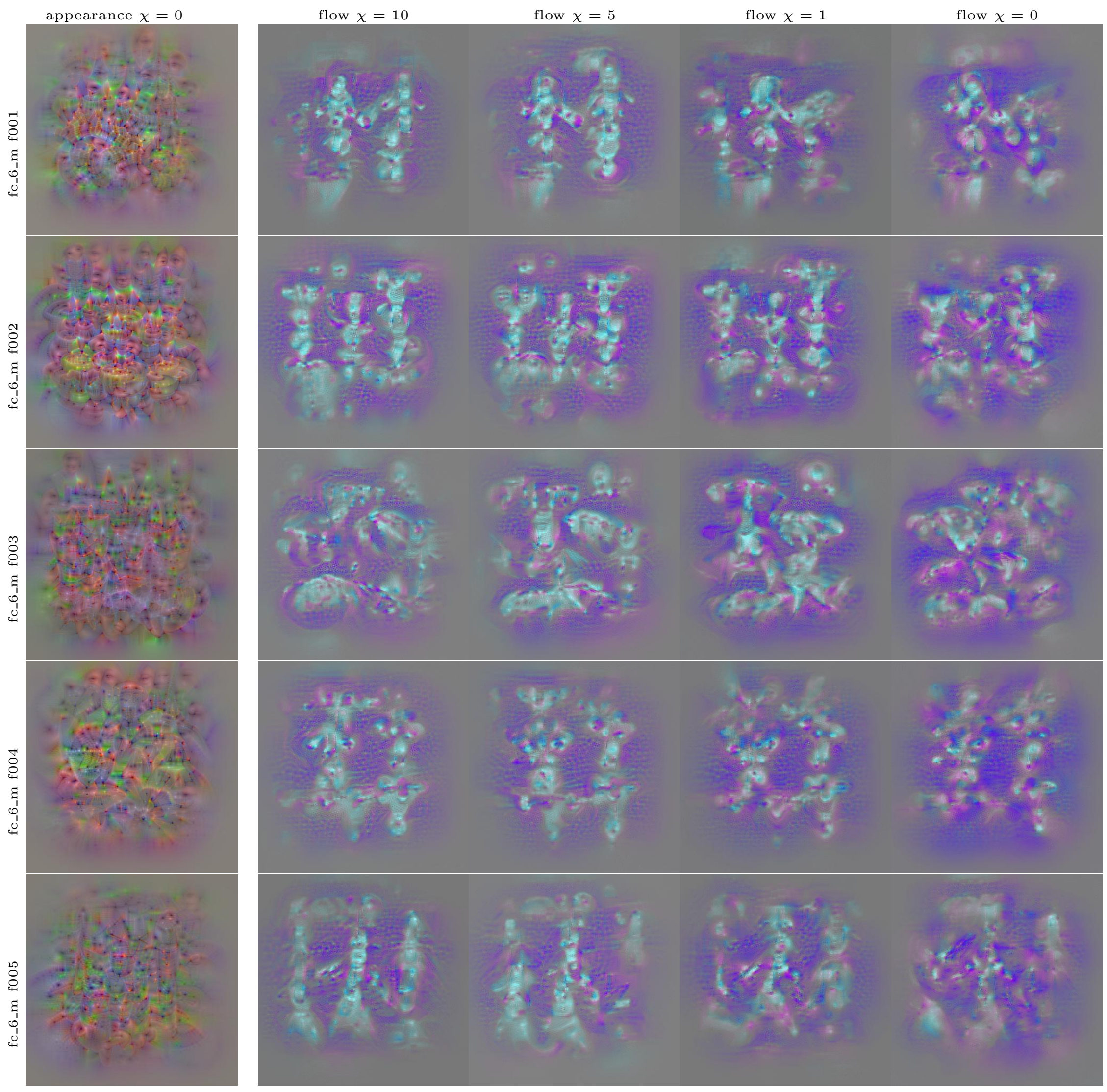


Figure 6.12: Visualization of 5 filters of the fc_6 (motion stream) layer under different temporal TV regularization. We show the appearance input and the optical flow inputs for slowest $\chi = 10$, slow $\chi = 5$, fast $\chi = 1$, and unconstrained (fastest) $\chi = 0$, temporal variation regularization.

A similar visualization showing the subsequent fully-connected_7 (fc_7) layer is shown in Fig. 6.13 where we already observe class-like patterns that serve as a linear subspace for the final prediction layer. Additional visualizations of these global layers can be found in Appendix B.3.

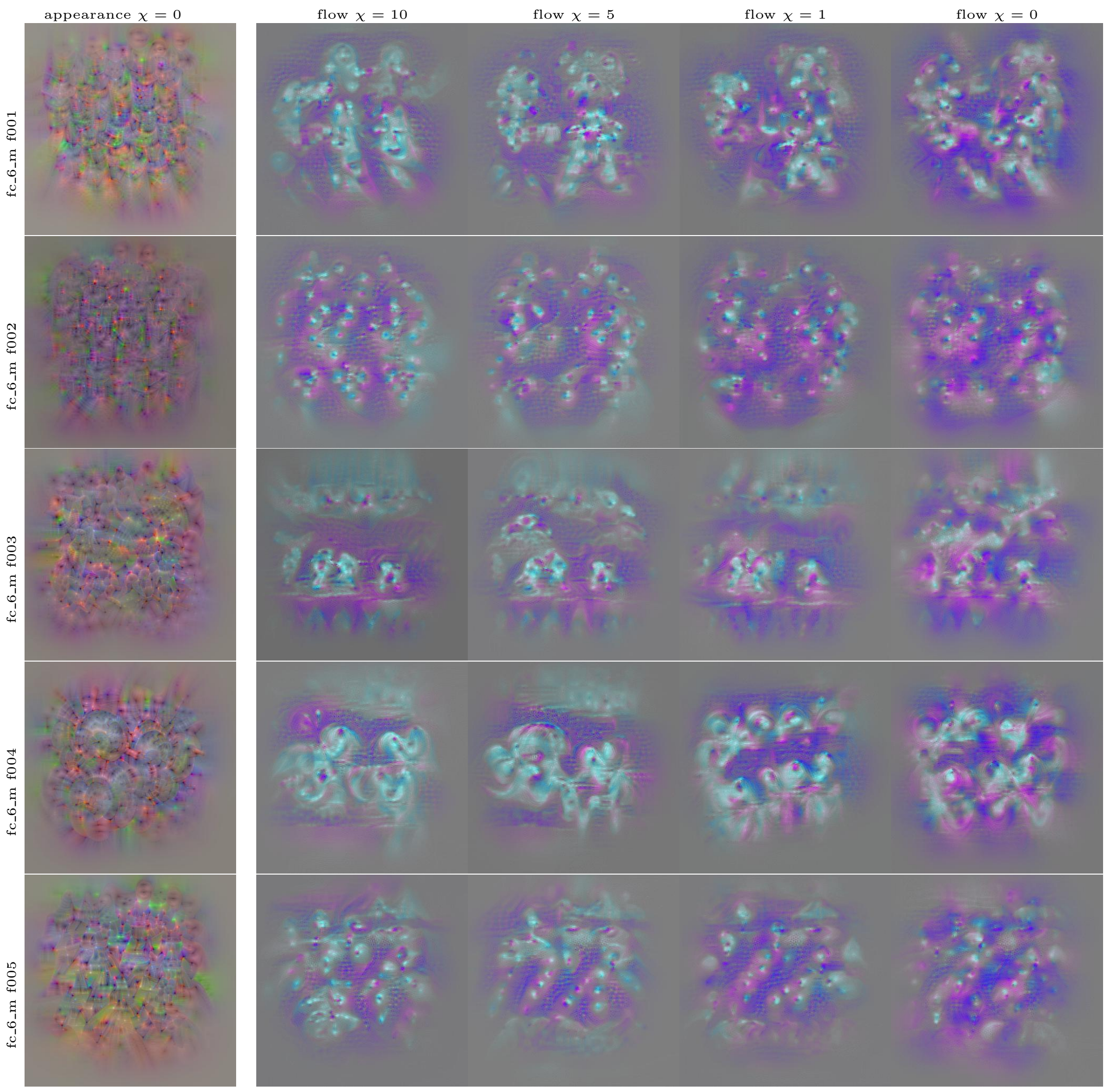


Figure 6.13: Visualization of 5 filters of the fc_7 (motion stream) layer under different temporal TV regularization. We show the appearance input and the optical flow inputs for slowest $\chi = 10$, slow $\chi = 5$, fast $\chi = 1$, and unconstrained (fastest) $\chi = 0$, temporal variation regularization.

Finally, we visualize the ultimate class prediction layers of the architecture, where the neuron outputs corresponds to different classes (thus, we know what they should be matched to). In Fig. 6.14, we show the fast motion activation of the first four classes in UCF101, ApplyEyeMakeup, ApplyLipstick, Archery and BabyCrawling (see Appendix B.4 for additional examples). The learned features for archery (*e.g.*, the elongated bow shape and positioning of the bow, but also shooting the arrow in the fast variation case) are markedly distinct from those of the make-up examples (*e.g.*, capturing facial features, such as eyes, and the motion of applicator). Interestingly, it seems that ApplyEyemakeup and ApplyLipstick are being distinguished, at least in part, by the fact that eyes tend to move in the latter case, while they are held static in the former case. Here, we see a benefit of visualizations beyond revealing what the network has learned – they also can reveal idiosyncrasies of the data on which the network has been trained. The final baby crawling example is also markedly distinct from any of the other examples (*e.g.*, capturing the facial parts of the babies' appearance while focusing on the arm and head movement in the motion representation). Thus, we find that the class prediction units have learned representations that are well matched to their classes.

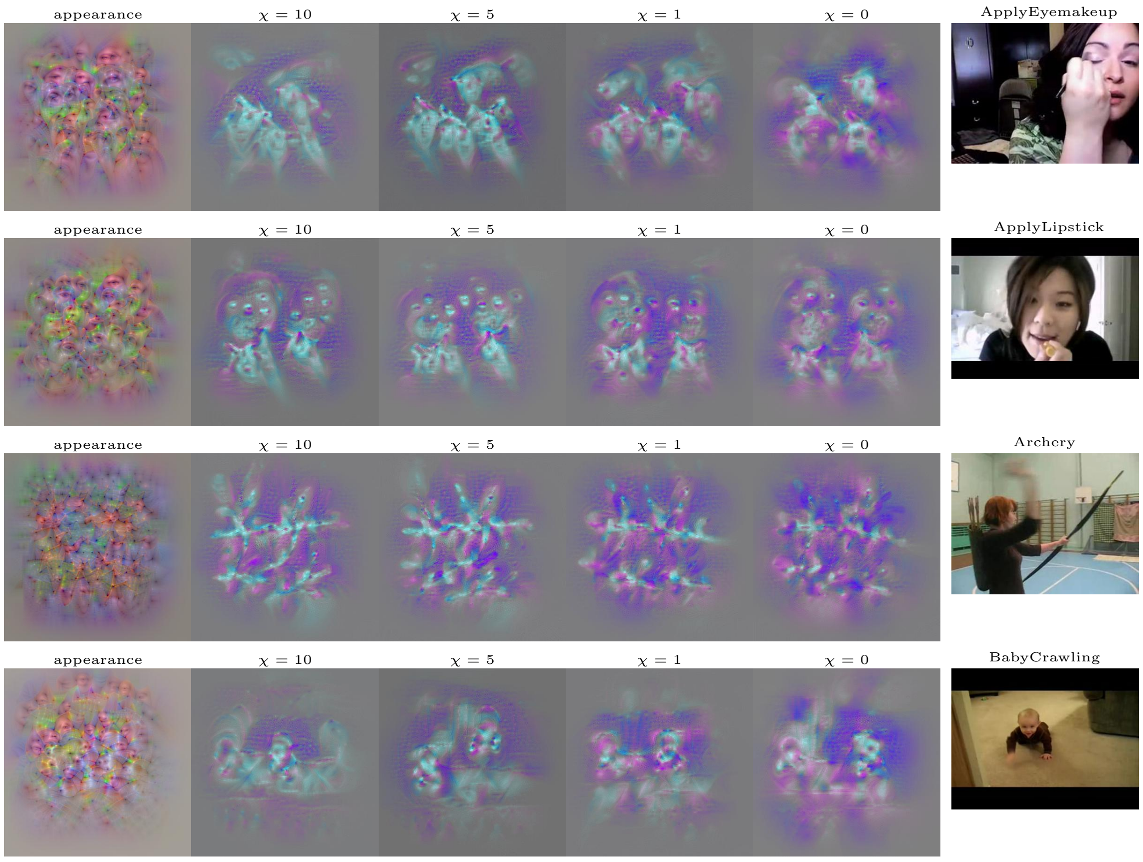
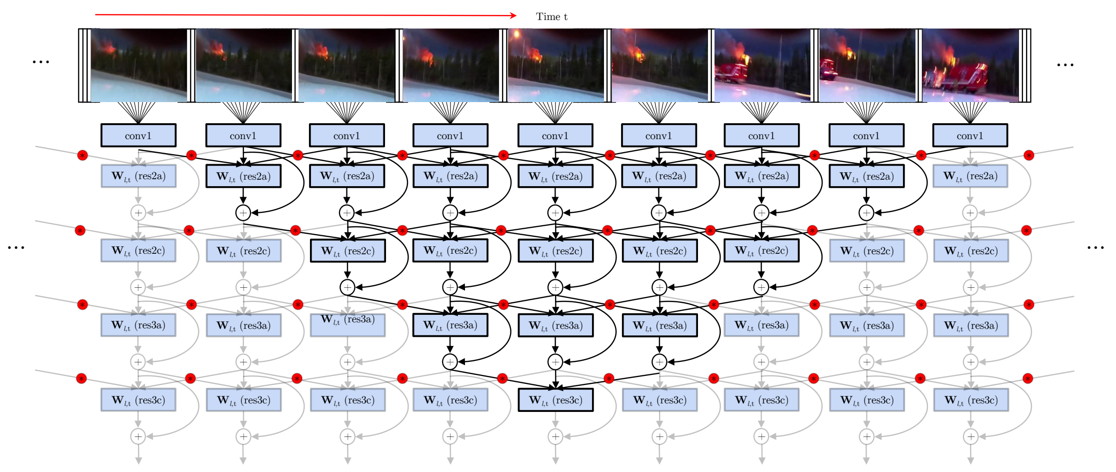


Figure 6.14: Classification units at the last layer of the network. The first column shows the appearance and the second to fifth columns the motion input generated by maximizing the prediction layer output for the respective classes, with different degrees of temporal variation regularization (χ). The last column shows 40 sample frames from the first video of that class in the test set.

6.5 Summary

An irritating property of deep networks is that due to their compositional structure it is difficult to explicitly reason about what these models actually have learned. As the success of deep architectures for processing visual signals has led to their deployment in all areas of perceptual sensing, it is of utmost importance to understand how these representations work and what they are capturing. In this chapter, we have shed light on deep spatiotemporal representations by visualizing what our models have learned for recognizing actions in video. We have shown that local detectors for appearance and motion objects arise to form distributed representations for recognizing human actions. Key observations include the following. First, learned patterns show correlated spatial structure across the appearance and motion streams even prior to cross stream connections. This fact might be an artifact of pretraining of both streams on ImageNet. Second, throughout the hierarchy of the network, features become more abstract (e.g., as illustrated in visualizations ascending through layers in Figs. 6.6-6.14) and show increasing invariance to aspects of the data that are unimportant to desired distinctions (e.g., neurons matched to billiard ball shape and motion patterns across various speeds and accelerations). Third, the network has the ability to learn representations that are highly class specific (e.g., shape and motion of bow for archery), but it also can learn more generic representations that can subserve a range of actions (e.g. various sports). Fourth, visualizations can be used not only to shed light on learned representations, but also to reveal idiosyncracies of training data (e.g., distinguishing application of lipstick vs. eye make-up based on eye motion). Finally, the results provide clear qualitative evidence for separation into two streams for learning highly abstract appearance and motion representations – a principle that has also been found in nature where numerous studies suggest a corresponding separation into ventral and dorsal pathways of the mammalian brain.



Time t
...
conv1
$\mathbf{W}_{l,t}$ (res2a)
$\mathbf{W}_{l,t}$ (res2c)
$\mathbf{W}_{l,t}$ (res3a)
$\mathbf{W}_{l,t}$ (res3c)
...

Temporal receptive field of a single neuron at the third conv block of our spatiotemporal ConvNet architecture. In this chapter, we present T-ResNet, an architecture that is fully convolutional in spacetime and performs temporal filtering at residual units to hierarchically inject temporal information as depth increases.

7 Deep Learning for Dynamic Scene Recognition

7.1 Motivation

Image-based scene recognition is a basic area of study in visual information processing. Humans are capable of recognizing scenes with great accuracy and speed (Potter, 2012). Reliable automated approaches can serve to provide priors for subsequent operations involving object and action recognition, e.g., (Marszalek et al., 2009, Torralba et al., 2010). Moreover, scene recognition could serve in browsing image databases, e.g. (Vailaya et al., 2001). While early computational research in scene recognition was concerned with operating on the basis of single images, e.g., (Fei-Fei and Perona, 2005, Lazebnik et al., 2006, Oliva and Torralba, 2001), more recently dynamic scene recognition from video has emerged as a natural progression, e.g., (Derpanis et al., 2012, Feichtenhofer et al., 2014, Shroff et al., 2010).

Beyond dynamic scene recognition, considerable research has addressed allied tasks in video-based recognition. Arguably, the most heavily researched has been action recognition (Marszalek et al., 2009, Ng et al., 2015b, Simonyan and Zisserman, 2014a); although, a variety of additional video-based recognition tasks also have been considered, *e.g.* (Over et al., 2013, Park et al., 2013, Poleg et al., 2015). In response to the challenges these tasks pose, a wide variety of approaches have been developed. Here, it is worth noting that recent extensions of Convolutional Networks (ConvNets) to video have shown particularly strong results, *e.g.* (Ng et al., 2015a, Tran et al., 2015a, Xu et al., 2015). While many of these approaches have potential to be generalized and applied to dynamic scene recognition, that avenue has been under researched to date. The current chapter addresses this situation by applying a representative sampling of state-of-the-art video recognition techniques to dynamic scenes, including a novel ConvNet. This work extends our understanding of not only the individual techniques under evaluation, but also the nature of dynamic scenes as captured in video.

7.2 Related work

Currently, there are two standard databases to support the study of scene recognition from videos (Derpanis et al., 2012, Shroff et al., 2010). Both of these databases capture a range of scene classes and natural variations within class (seasonal and diurnal changes as well as those of viewing parameters). A significant difference between the two datasets is that one includes camera motion (Shroff et al., 2010), while the other does not (Derpanis et al., 2012). Unfortunately, neither database provides balanced scene samples acquired with and without camera motion to support systematic study of how scene dynamics can be disentangled from camera motion. Moreover, at this time performance on both datasets is at saturation (Feichtenhofer et al., 2016b, Tran et al., 2015a). Correspondingly, research in dynamic scene recognition is at risk of stagnation, unless new, systematically constructed and challenging video databases relevant to this task are introduced.

Video-based dynamic scene classification has been approached based on linear dynamical systems (Doretto et al., 2003), chaotic invariants (Shroff et al., 2010), local measures of spatiotemporal orientation (Derpanis et al., 2012, Feichtenhofer et al., 2013, 2014), slowly varying spatial orientations (Theriault et al., 2013) and spatiotemporal ConvNets (Tran et al., 2015a), with spatiotemporal orientation and ConvNets showing strongest recent performance (Feichtenhofer et al., 2016b).

The most closely related work to T-ResNet is the spatiotemporal residual network, ST-ResNet from Chapter 4, that is based on two-stream (Simonyan and Zisserman, 2014a) and residual networks (He et al., 2016a). The ST-ResNet architecture injects residual connections between the appearance and motion pathways of a two-stream architecture and transforms spatial filter kernels into spatiotemporal ones to operate on adjacent feature maps in time. Our work in this chapter, instead extends the spatial residual units with a temporal kernel that is trained from scratch and hence is able to learn complex temporal features as it is initialized to receive more informative temporal gradients. Importantly, different from the previous chapter, our T-ResNet architecture does not rely on optical flow input. In many cases the brightness constancy assumption used in optical flow computation does not hold for dynamic texture patterns (*e.g.* water, fire), but recognizing these is essential to discriminate scenes based on their defining dynamics.

In summary, this chapter makes the following contributions to advance dynamic scene classification. First, a novel spatiotemporal ConvNet architecture, T-ResNet, is introduced that is based on transformation of a spatial network to a spatiotemporal ConvNet. This transformation entails a particular form of transfer learning from spatial image classification to spatiotemporal scene classification. Second, the superiority in scene recognition from video of the newly developed T-ResNet is documented by comparing it to a representative sampling of alternative approaches. Results show that our spatiotemporally trained ConvNet greatly outperforms the alternatives, including approaches hand-crafted for dynamic scenes and other networks trained directly for large scale video classification. Third, a new dynamic scenes dataset is introduced. This dataset more than doubles the size of the previous collections in common use in dynamic scene recognition (Derpanis et al., 2012, Shroff et al., 2010), while including additional challenging scenarios. Significantly, for each scene class that is represented an equal number of samples is included with and without camera motion to support systematic investigation of this variable in scene recognition. Our Code uses our own modified spatiotemporal version of the MatConvNet toolbox (Vedaldi and Lenc, 2015) and is available at
https://github.com/feichtenhofer/temporal-resnet and our novel dynamic scene recognition dataset is available at http://vision.eecs.yorku.ca/research/dynamic-scenes/.

7.3 Temporal residual networks

Various paths have been followed to extend ConvNets from the 2D spatial domain, (x, y), to the 3D spatiotemporal domain, (x, y, t), including building on optical flow fields (Poleg et al., 2015, Simonyan and Zisserman, 2014a), learning local spatiotemporal filters (Karpathy et al., 2014, Taylor et al., 2010, Tran et al., 2015a) and modeling as recurrent temporal sequences (Ballas et al., 2016, Donahue et al., 2015, Srivastava et al., 2015). To date, however, these approaches have not triggered the dramatic performance boost over hand-crafted representations (*e.g.* IDT (Wang and Schmid, 2013)) that ConvNets brought to the spatial domain *e.g.* in image classification (Krizhevsky et al., 2012b, Simonyan and Zisserman, 2014b). This relative lack of impact has persisted even when large new datasets supported training of 3D spatiotemporal filters (Karpathy et al., 2014, Tran et al., 2015a). This section documents a novel approach that proceeds by transforming a spatial ConvNet, ResNet (He et al., 2016a), to a spatiotemporal kindred, T-ResNet. In empirical evaluation (Sec. 7.5) it will be shown that this approach yields a network with state-of-the-art performance.

7.3.1 Spatiotemporal residual unit

The main building blocks of the ResNet architecture are residual units (He et al., 2016a). Let the input to a residual unit be a feature map, $\mathbf{x}_l \in \mathbb{R}^{H\times W\times T\times C}$, where W and H are spatial dimensions, C is the feature dimension and T is time. Such maps can be thought of as stacking spatial maps of C dimensional features along the temporal dimension. At layer l with input $\mathbf{x}_l$, a residual block is defined as (He et al., 2016a,b)

$$\mathbf{x}_{l+1} = f\left(\mathbf{x}_l + \mathcal{F}(\mathbf{x}_l; \mathcal{W}_l)\right), \tag{7.1}$$

with $f \equiv$ ReLU, $\mathcal{W}_l = \{\mathrm{W}_{l,k}|_{1\leq k\leq K}\}$ holding the K corresponding filters and biases in the unit, and $\mathcal{F}$ denoting the residual function representing convolutional operations. Formally, each of the K layers in the l^{th} residual unit performs the following filtering operation

$$\mathbf{x}_{l,k+1} = \mathrm{W}_{l,k}\mathbf{x}_{l,k}, \tag{7.2}$$

where $\mathrm{W}_{l,k}|_{1\leq k\leq K}$ are the convolutional filter kernels arranged as a matrix and batch normalization layers are omitted for simplicity. We use the original ResNet architecture (He et al., 2016a) where $K = 3$, consisting of 1×1 dimensionality reduction, 3×3 spatial aggregation and 1×1 dimensionality restoration filtering operations. These choices lead to the residual unit

$$\mathcal{F} = f(\mathrm{W}_{l,3}f(\mathrm{W}_{l,2}f(\mathrm{W}_{l,1}\mathbf{x}_l))), \tag{7.3}$$

as illustrated in Fig. 7.1a.

Our proposed spatiotemporal residual unit $\hat{\mathcal{F}}$ injects temporal information into residual blocks via 1D temporal filtering. Building on the inception idea (Szegedy et al., 2015a), our temporal convolution block operates on the dimensionality reduced input, $\mathbf{x}_{l,1}$, with a bank of spatiotemporal filters, $\mathrm{W}_{l,t}$, and by applying biases, $b \in \mathbb{R}^C$, according to

$$\mathbf{x}_{l,t} = \mathrm{W}_{l,t}\mathbf{x}_{l,1} + b_l, \tag{7.4}$$

$$\mathbf{x}_{l,t} = \mathrm{W}_{l,t}f(\mathrm{W}_{l,1}\mathbf{x}_l) + b_l, \tag{7.5}$$

where biases $b_l \in \mathbb{R}^C$ are initialized to zero and the weights, $\mathrm{W}_{l,t}$, come as a 3-tap temporal filter bank. These filters are initialized randomly for the same feature dimensionality as the spatial 3×3 filters, $\mathbf{W}_{l,2}$, working in parallel on input $\mathbf{x}_{l,1}$. $\mathrm{W}_{l,t}$ is able to model the temporal structure of the features from the previous layer. Moreover, by stacking several such kernels through the hierarchy of the network we are able to grow the temporal receptive field.

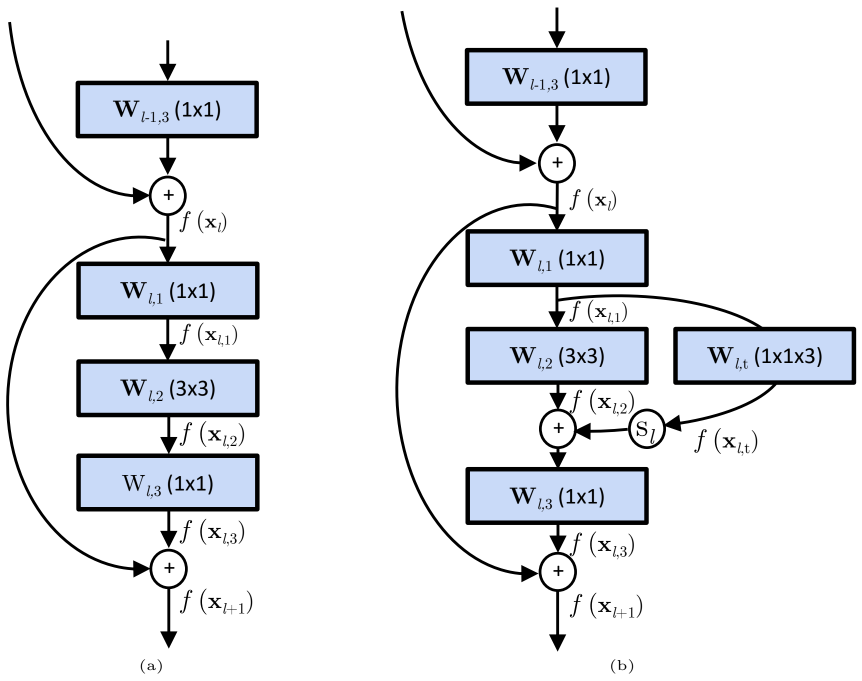


Figure 7.1: Comparison between the original residual units (a) and our proposed spatiotemporal residual units (b), which augment the "bottleneck structure" with an additional temporal conv block and an affine scaling layer S_l.

Our proposed spatiotemporal residual unit $\hat{\mathcal{F}}$ is now defined as

$$\hat{\mathcal{F}} = f\Big(\mathrm{W}_{l,3}\big(\mathrm{S}_l f(\mathbf{x}_{l,t}) + f(\mathrm{W}_{l,2} f(\mathbf{x}_{l,1}))\big)\Big), \tag{7.6}$$

where S_l is a channel-wise affine scaling weight initialized with a scaling of .01 and zero biases. We found adaptive scaling of the temporal residuals to facilitate generalization performance. The final unit is illustrated in Fig. 7.1b.

Discussion. Our design builds on two good practices for designing image ConvNets: First, it builds on the inception concept that dimensionality reduction should be performed before spatially/temporally aggregating filters since outputs of neighbouring filters are highly correlated and therefore these activations can be reduced before aggregation (Szegedy et al., 2015b). Second, it exploits spatial factorization into asymmetric filters, which reduces computational cost and also has been found to ease the learning problem (Szegedy et al., 2016).

Scaling down residuals also has been found important for stabilizing training (Szegedy et al., 2016), where residuals are scaled down by a constant factor before being added to the accumulated layer activations. Activations also have been usefully rescaled before combining them over several layers of the network (Bell et al., 2016). As an alternative to scaling has been used where the network was first pre-conditioned by training with a very low learning rate, before the training with higher learning rate proceeded (He et al., 2016a).

7.3.2 Global pooling over spacetime

The “Network In Network” (Lin et al., 2013) architecture has shown that the fully connected layers used in previous models (Krizhevsky et al., 2012b, Simonyan and Zisserman, 2014b) can be replaced by global average pooling after the last convolutional layer and this replacement has been adopted by recent ConvNet architectures (He et al., 2016a, Szegedy et al., 2015a). The general idea behind global pooling of activations is that at the last conv-layer the units see all pixels at the input due to the growth of the receptive field; *e.g.* for the ResNet-50 architecture that we use in this work, the last convolutional layer theoretically has a receptive field that covers 483×483 pixels at the input, even though the input is only of size 224×224. Practically, however, the utilized receptive field of a unit is postulated to be smaller (Zhou et al., 2014).

Correspondingly, a temporal network can be expected to capture similar spatial features across time. Therefore, it is reasonable to develop a spatiotemporal network by global pooling over the temporal support. We found in our experiments that globally max pooling over time, *i.e.*

$$\mathbf{x}(i,j,c) = \max_{1\leq k'\leq T'} \mathbf{x}(i,j,k',c), \tag{7.7}$$

works better ($\approx 2\%$ accuracy gain) than global averaging of temporal activations. We conjecture that this result is due to the derivative of the sum operation uniformly backpropagating gradients to the temporal inputs. Thus, the network is not able to focus on the most discriminating instance in time when employing temporal averaging. Even though max-pooling over time only backpropagates a single temporal gradient map to the input, it can guide the learning of long-term temporal features because the filter’s temporal receptive field on the gradient maps grows *from the output to the input.*

Discussion. We conducted several experiments using max-pooling earlier in the network and it consistently led to reduced performance with accuracy decreasing more, the earlier pooling starts ($\approx$1−6%). We also experimented with a more natural decrease of frames by valid convolutions in time. Typically, ConvNet architectures zero-pad the inputs before each spatial (*e.g.* 3×3) convolution such that the output size is unchanged. A cleaner strategy is to use valid convolutions (*i.e.* not filtering over the border pixels) together with lager sized inputs *e.g.* used in the early layers of inception-v4 (Szegedy et al., 2016). We investigated if there is any gain in performance for having a temporal architecture with valid filtering operations across time. For this experiment, we increase the number of frames at the input and use temporal residual blocks that do not pad the input in time. Since the network now hierarchically downsamples the input by two frames at each temporal residual block, the final max-pooling layer now receives less frames (when keeping GPU-memory constant compared to a padded design). In our experiments this architectural change leads to an error increase of 2.4%, in comparison to the padded architecture equivalent. In conclusion, pooling as late as possible, together with padded convolutions across time, enables best accuracy for our T-ResNet in dynamic scene classification.

7.3.3 Implementation details

We build on the ResNet-50 model pretrained on ImageNet (He et al., 2016a) and replace the last (prediction) layer. Next we transform every first and third residual unit at each conv stage (conv2_x to conv5_x) that hold residual units to our proposed temporal residual units. For our temporal residual blocks, we switch the order of batch normalization (Ioffe and Szegedy, 2015) and ReLU from post-activation to pre-activation (He et al., 2016b). The temporal filters are of dimension $W' \times H' \times T' \times C \times C = 1 \times 1 \times 3 \times C \times C$ and initialized randomly. We use 16 frame inputs and temporal max-pooling is performed immediately after the spatial global average pooling layer.

The training procedure follows standard ConvNet training (He et al., 2016a, Krizhevsky et al., 2012b, Simonyan and Zisserman, 2014b), with some subtle differences. We set the learning rate to 10^{-2} and decrease it by an order of magnitude after the validation error saturates. We use batch normalization (Ioffe and Szegedy, 2015) and no dropout. To accelerate training, we train the network in a two-stage process with a batchsize of 256: First, we train the network in a purely spatial manner where we randomly sample a single frame from different videos (ResNet); second, we transform the residual units to spacetime and re-start the training process by sampling 16-frame stacks from 256/32 videos per batch (T-ResNet).

For data augmentation we obtain multiple frame-stacks by randomly selecting the position of the first frame and apply the same random crop to all samples. Instead of cropping a fixed sized 224×224 input patch, we perform multi-scale and aspect-ratio augmentation by randomly jittering its width and height by $\pm 25\%$ and resizing it to obtain a fixed sized 224×224 network input. We randomly crop translated patches at a maximum of 25% distance from the image borders (relative to the width and height). Compared to training spatial ConvNets, training spatiotemporal ConvNets is even more prone to overfitting. In response, we use temporal frame jittering: In each training iteration we sample the 16 frames from each of the training videos in a batch by randomly sampling the starting frame, and then randomly sampling the temporal stride $\in [5, 15]$. We do not apply RGB colour jittering (Krizhevsky et al., 2012b).

During testing, we take a sample of 16 equally spaced frames from a video and propagate these through the net to yield a single prediction for each video. Instead of cropping the image corners, centre and their horizontal flips, we apply a faster fully convolutional testing strategy (Simonyan and Zisserman, 2014b) on the original image and their horizontal flips and average the predictions from all locations. Thus inference can be performed in a *single* forward pass for the whole video.

7.4 Dynamic scenes dataset

As discussed in the beginning of this chapter, the previously best performing algorithms on dynamic scene recognition have saturated performance on extant datasets (Feichtenhofer et al., 2016b, Tran et al., 2015a). In response, this section introduces a new dynamic scenes dataset to support the current and future studies in this domain.

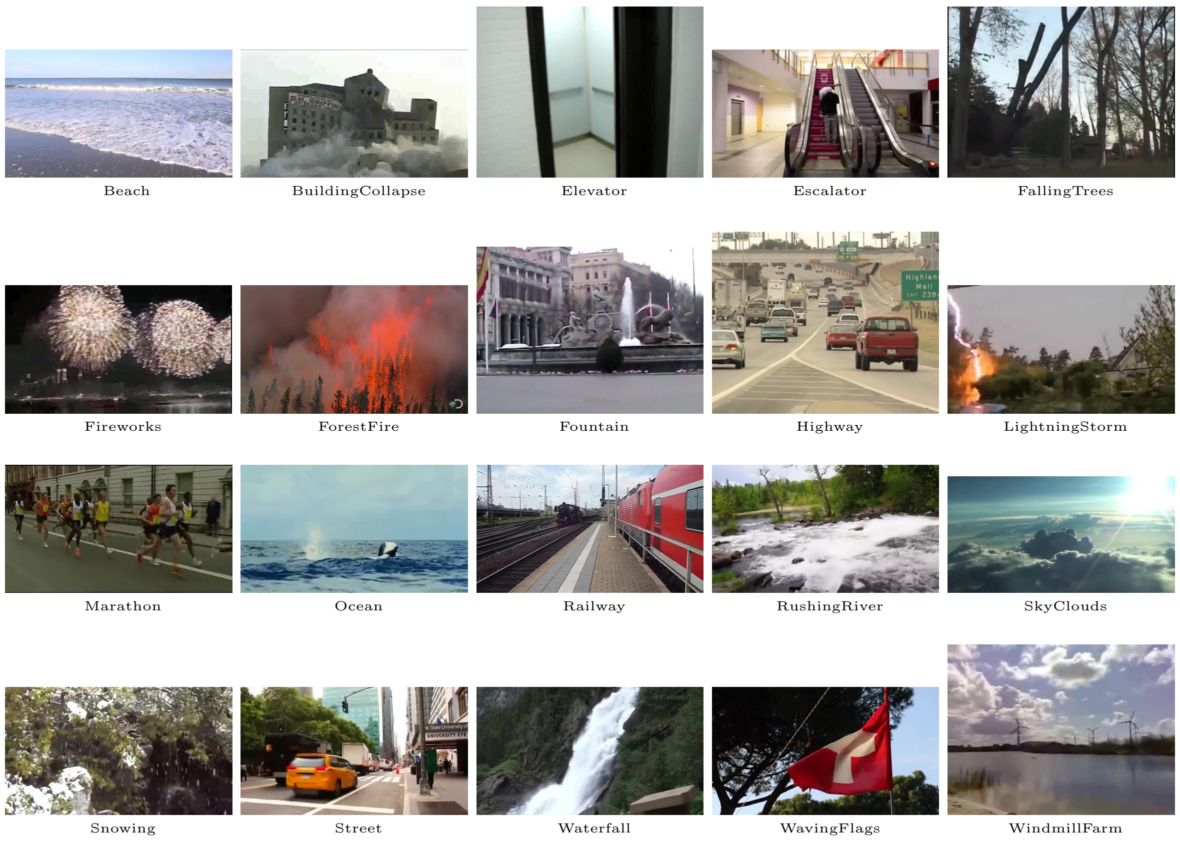


Figure 7.2: Thumbnail examples of the **YUP++** dataset.

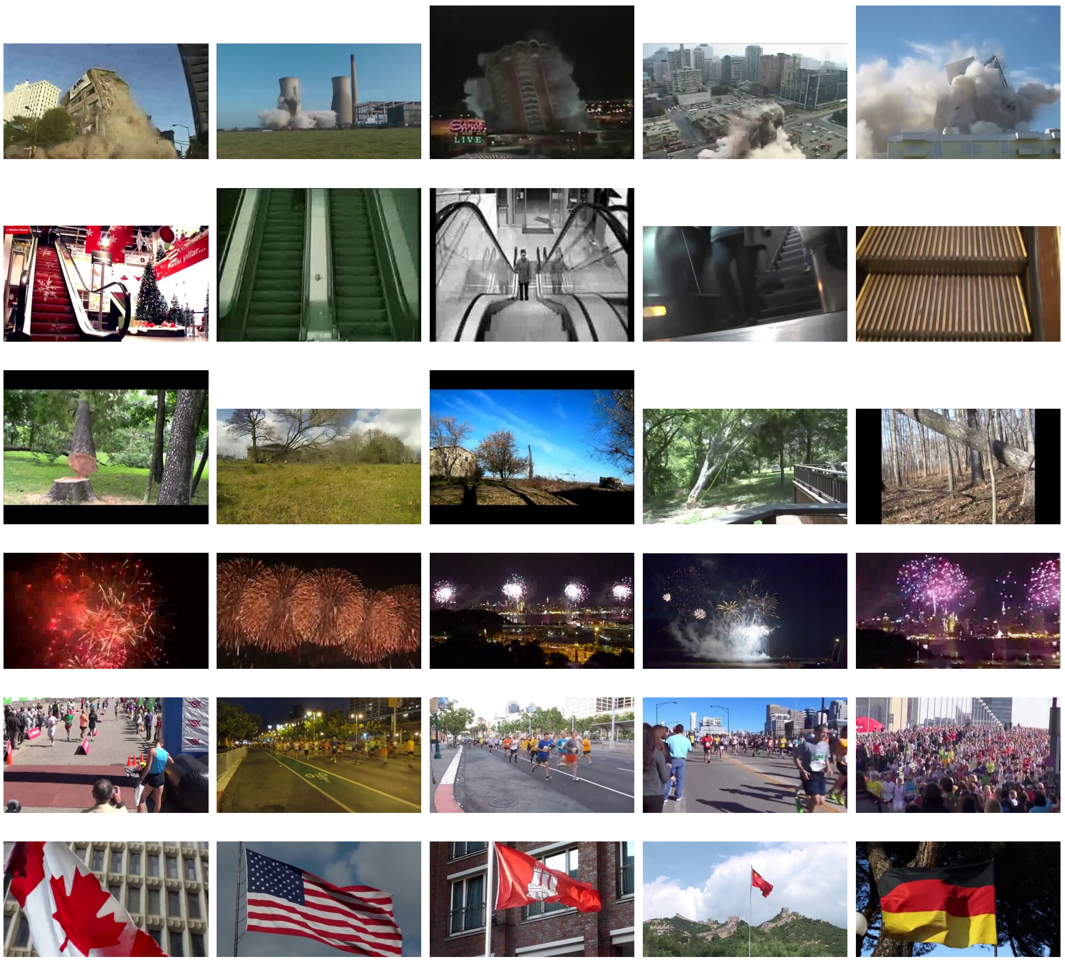


Figure 7.3: Variations within the new classes in the **YUP++**. Top-to-bottom: Building Collapse, Escalator, Falling Trees, Fireworks, Marathon and Waving Flags.

7.4.1 Specifications

The new dynamic scenes dataset samples 20 scene classes, while encompassing a wide range of conditions, including those arising from natural within scene category differences, seasonal and diurnal variations as well as viewing parameters. Thumbnail examples of each class are shown in Figs. 7.2 and 7.3. Details of the dataset are provided in the remainder of this section.

The new dataset builds on the earlier YUPenn dataset (Derpanis et al., 2012). This dataset is taken as a point of departure rather than the Maryland dataset (Shroff et al., 2010) as it includes one additional scene class and three times as many videos for each class represented. To the original dataset, six additional classes have been added for a total of twenty. The final set of classes represented in the dataset are as follows: beach, city street, elevator, forest fire, fountain, highway, lightning storm, ocean, railway, rushing river, sky clouds, snowing, waterfall, windmill farm, building collapse, escalator, falling trees, fireworks, marathon, waving flags. The last six listed classes are in addition to those available in the earlier YUPenn. Due to its extended number of classes and addition of moving camera videos this novel dataset is termed **YUP++**.

For each scene class in the dataset, there are 60 colour videos, with no two samples for a given class taken from the same physical scene. Half of the videos within each class are acquired with a static camera and half are acquired with a moving camera, with camera motions encompassing pan, tilt, zoom and jitter. Having both static and moving camera instances for each class allows for systematic consideration of the role this variable plays in scene categorization, something that was not supported in either of the previous dynamic scenes datasets. Beyond camera motion and natural variation of individual scenes within a given class, a wide range of additional acquisition parameters are varied, including illumination (*e.g.* diurnal variations), seasonal, scale and camera viewpoint.

The videos were acquired from online video repositories (YouTube, BBC Motion Gallery and Getty Images) or a handheld camcorder. All videos have been compressed with H.264 codec using the ffmpeg video library. Duration for each video is 5 seconds, with original frame rates ranging between 24 and 30 frames per second. All have been resized to a maximum width of 480 pixels, while preserving their original aspect ratio.

Overall, the new dynamic scenes dataset more than doubles the size of previous datasets for this task. All of the videos are distinct from the earlier Maryland dataset. In comparison to the YUPenn dataset, six new scene classes have been added and all moving camera videos are new.

7.4.2 Experimental protocol

For the purpose of dynamic scene recognition, the dataset has been divided into training and test sets. A random split is employed to generate the two sets, by randomly choosing for each class an equal number of static camera and moving camera videos. This random split protocol is in contrast to the previously used leave-one-out protocol on the YUPenn and Maryland datasets. As documented in Sec. 7.5, using a random split with such a train/test ratio is better suited to providing a challenging benchmark protocol in a computationally tractable fashion. Moreover, a random split protocol is common practice in other domains, e.g. action recognition on HMDB51 (Kuehne et al., 2011) and UCF101 (Khurram Soomro and Shah, 2012) as well as indoor scene classification on MIT67 (Quattoni and Torralba, 2009).

7.5 Empirical evaluation

To establish the state-of-the-art in dynamic scene recognition, 7 representative algorithms for video-based recognition are evaluated along with T-ResNet introduced in Sec. 7.3. Three of the evaluated algorithms, C3D (Tran et al., 2015a), BoSE (Feichtenhofer et al., 2014) and SFA (Theriault et al., 2013), have shown the first, second and fourth best performance in previous dynamic scenes evaluations. The third best performer, (Feichtenhofer et al., 2013), is an ancestor of BoSE and is not considered here. The remaining algorithms, while not previously evaluated on dynamic scene recognition, are selected to provide a balanced coverage of contemporary strong performers on image-based classification tasks. To see how well a strong performer on single image classification can perform on video-based scene classification, very deep convolutional networks with Fisher vector encoded features are considered (S-CNN) (Cimpoi et al., 2015). To see how well an approach that serves as the basis for a variety of strong performing action recognition algorithms can be adapted to scene recognition, (improved) dense trajectories (IDTs) are considered (Wang et al., 2013). Also, to test further temporal ConvNets (in addition to spatiotemporal C3D), a representative based on optical flow is considered (T-CNN) (Simonyan and Zisserman, 2014a). Finally, to judge the improvements that the spatiotemporal T-ResNet offers over the spatial ResNet (He et al., 2016a), we report results for the fine-tuned ResNet. Details of how these approaches are applied to dynamic scene recogntion are supplied in the Appendix C.

7.5.1 Is there a need for a new dataset?

The first question investigated is if a new dynamic scenes dataset is truly needed to challenge the available algorithms or if only the existing evaluation protocols need an improvement. To answer this question, the largest previously available dataset, **YUP** (Derpanis et al., 2012), was explored in the following fashion with three of the best performing algorithms to date (C3D, BoSE, and SFA): Instead of using the leave-one-video-out (LOO) protocol, as in previous evaluations (Derpanis et al., 2012, Feichtenhofer et al., 2014, Shroff et al., 2010, Theriault et al., 2013), fixed train/test splits are used, as it is common practice in action recognition tasks (Khurram Soomro and Shah, 2012, Kuehne et al., 2011). Splits were generated by randomly choosing training and testing clips from each class. Three splits are employed for any given ratio; final recognition accuracy is taken as the average across the three. This experiment was performed for several train/test ratios. The results for the three considered algorithms are reported in Table 7.1, left. Surprisingly, performance can remain on par with the leave-one-out results on this dataset (Feichtenhofer et al., 2014, Theriault et al., 2013, Tran et al., 2015a); it is also surprising that even very low train/test ratios can still score high. It is concluded that simply changing the relative sizes of the training and testing sets does not have a significant enough impact on recognition rates to continue using this dataset in evaluations.

In comparing results for various 10/90 splits we find little difference, even for the most difficult moving camera component of the new dataset, **YUP++ moving camera**; see Table 7.1, right. This finding suggests that the 10/90 split supports stable algorithm performance evaluation, even while being most challenging. Moreover, since there is little variation between random splits it justifies using just a single split in evaluation, as it provides for less overhead in evaluation, especially for cases involving ConvNet training. Therefore, in all subsequent experiments a single 10/90 split is employed. In particular, we employ split #1 of Table 7.1.

Train/Test:	LOO	90/10	70/30	50/50	30/70	10/90
C3D Acc:	98.1	97.6	96.8	94.8	94.2	86.0
BoSE Acc:	96.2	95.3	95.1	94.8	94.1	82.54
SFA Acc:	85.5	84.7	83.4	81.0	80.0	70.0

#split	SFA	BoSE	T-CNN	S-CNN	IDT	C3D
1	51.1	61.9	36.3	68.1	70.4	76.3
2	49.3	60.2	38.8	72.2	69.4	77.6
3	44.8	60.0	36.5	72.8	69.3	78.3

Table 7.1: Left: Performance of 3 of the previously best approaches for dynamic scene recognition on on the **YUP** (Derpanis et al., 2012) dataset. Different train/test ratios have no significant effect on the classification accuracy, except for a very aggressive ratio of 10/90 (i.e. using 3 videos for training and 27 videos for testing per class). Right: Comparison of different algorithms on the **YUP++ moving camera** dataset using a 10/90 train test ratio. Performance levels are consistent across different random splits.

7.5.2 Does adding new classes solve the problem?

The next question investigated is if adding additional classes would lead to a sufficiently more challenging dynamic scene benchmark. Table 7.2 (left) lists the results for including six additional classes BuildingCollapse, Escalator, FallingTrees, Fireworks, Marathon and WavingFlags to the previously existing YUPenn. Note that still all videos are taken from a static camera and thereby this subset is called the **YUP++ static camera**. While all algorithms decrease in performance compared to the original YUP, the best performers suffer only negligible deficits. It is desirable to further increase the challenge.

7.5.3 Does more challenging data help?

Since adding more classes has too limited an effect on performance, this section presents a way of increasing the difficulty of the data. The challenge is increased by including camera motion during acquisition of videos. The overall size of the datasets is thereby doubled, as each class contains an equal number of videos captured with and without camera motion. Details are provided in Sec. 7.4. The results for just the new videos are reported in Table 7.2 (right), with this subset referred to as **YUP++ moving camera**. Here it is seen that the challenge has increased so that even the top performing algorithm scores at 81.5% accuracy and there is spread between algorithms that allows for interesting comparisons, as discussed next.

7.5.4 Detailed algorithm comparisons

Consistent with previous results (*e.g.* (Derpanis et al., 2012)), top performers on the static (Table 7.2, left) and moving (Table 7.2, right) camera subsets as well as the entirety (Table 7.3, left) of YUP++ are dominated by algorithms that include both spatial and temporal measurements, *i.e.* our novel T-ResNet, C3D, IDT and BoSE. Interestingly, algorithms based on purely spatial features, S-CNN and ResNet, also show reasonable performance. Apparently, even for dynamic scenes defining features can be abstracted on a spatial basis. In contrast, basing feature abstraction on motion alone (T-CNN) apparently loses too much information.

Class (static)	SFA	BoSE	T-CNN	S-CNN	IDT	C3D	ResNet	T-ResNet
Beach	74.1	88.9	85.2	74.1	100.0	92.6	74.1	96.3
BuildingCollapse	74.1	92.6	74.1	96.3	100.0	92.6	100.0	100.0
Elevator	81.5	96.3	100.0	100.0	96.3	100.0	100.0	100.0
Escalator	40.7	66.7	22.2	81.5	51.9	70.4	81.5	88.9
FallingTrees	63.0	63.0	29.6	74.1	96.3	92.6	88.9	77.8
Fireworks	63.0	85.2	44.4	77.8	92.6	85.2	88.9	96.3
ForestFire	25.9	85.2	25.9	96.3	74.1	92.6	92.6	92.6
Fountain	14.8	55.6	22.2	44.4	74.1	33.3	77.8	92.6
Highway	66.7	63.0	55.6	63.0	85.2	70.4	81.5	88.9
LightningStorm	33.3	59.3	88.9	77.8	96.3	81.5	74.1	92.6
Marathon	48.1	85.2	92.6	96.3	88.9	100.0	96.3	100.0
Ocean	96.3	85.2	88.9	100.0	100.0	100.0	100.0	100.0
Railway	33.3	48.1	51.9	88.9	74.1	59.3	81.5	96.3
RushingRiver	66.7	92.6	44.4	96.3	74.1	100.0	100.0	85.2
SkyClouds	85.2	100.0	63.0	96.3	96.3	100.0	96.3	100.0
Snowing	44.4	77.8	63.0	66.7	85.2	51.9	37.0	77.8
Street	96.3	92.6	63.0	100.0	96.3	96.3	100.0	96.3
Waterfall	74.1	66.7	25.9	70.4	33.3	96.3	59.3	70.4
WavingFlags	48.1	81.5	55.6	88.9	100.0	96.3	100.0	96.3
WindmillFarm	92.6	85.2	81.5	96.3	100.0	96.3	100.0	100.0
Average	61.1	78.5	58.9	84.3	85.7	85.4	86.5	92.41

Class (moving)	SFA	BoSE	T-CNN	S-CNN	IDT	C3D	ResNet	T-ResNet
Beach	77.8	77.8	18.5	70.4	66.7	81.5	96.3	96.3
BuildingCollapse	44.4	33.3	0.0	40.7	44.4	44.4	40.7	51.9
Elevator	81.5	100.0	77.8	100.0	100.0	100.0	100.0	100.0
Escalator	51.9	74.1	29.6	88.9	59.3	85.2	92.6	96.3
FallingTrees	55.6	77.8	63.0	77.8	96.3	88.9	85.2	96.3
Fireworks	48.1	74.1	25.9	33.3	85.2	77.8	59.3	81.5
ForestFire	29.6	66.7	14.8	88.9	59.3	55.6	88.9	96.3
Fountain	29.6	11.1	18.5	18.5	37.0	25.9	55.6	74.1
Highway	14.8	22.2	29.6	37.0	44.4	48.1	25.9	55.6
LightningStorm	25.9	59.3	59.3	85.2	81.5	85.2	88.9	92.6
Marathon	74.1	77.8	92.6	96.3	100.0	100.0	100.0	100.0
Ocean	40.7	37.0	33.3	51.9	55.6	85.2	22.2	48.1
Railway	18.5	66.7	25.9	92.6	59.3	88.9	100.0	100.0
RushingRiver	55.6	59.3	66.7	81.5	77.8	96.	85.2	85.2
SkyClouds	63.0	70.4	63.0	77.8	55.6	96.3	92.6	92.6
Snowing	14.8	40.7	14.8	22.2	77.8	40.7	25.9	37.0
Street	70.4	85.2	3.7	77.8	85.2	96.3	77.8	92.6
Waterfall	77.8	66.7	18.5	77.8	77.8	88.9	66.7	63.0
WavingFlags	70.4	70.4	51.9	77.8	81.5	74.1	92.6	96.3
WindmillFarm	77.8	66.7	18.5	66.7	63.0	66.7	74.1	74.1
Average	51.1	61.9	36.3	68.1	70.4	76.3	73.5	81.5

Table 7.2: Performance of different algorithms on the **YUP++ static camera** (top) and **YUP++ moving camera** (bottom) subsets.

The top performing algorithm on the static, moving and entire YUP++ is the newly proposed T-ResNet. It is particularly interesting to compare it to ResNet, as T-ResNet is initialized with ResNet and transformed from the spatial to spatiotemporal domain; see Sec. 7.3. Surprisingly, this transformation succeeds on the basis of a very small training set, *i.e.* a mere 10% of the dynamic scenes dataset. These results show that well initialized spatial networks can be transformed very efficiently to extract discriminating spatiotemporal information. Indeed, this discrimination tops that of a rival spatiotemporal network, C3D, as well as the best hand-crafted spatiotemporal performer IDT.

Comparing performance on the static (Table 7.2, left) vs. moving (Table 7.2, right) camera subsets, it is seen that all algorithms show a decrement in performance in the presence of camera motion. Apparently, the algorithms have difficulty disentangling image dynamics that arise from scene intrinsics vs. camera motion and this is an area where future research should be focused. As it stands, the greatest performance loss is suffered by T-CNN, which suggests that building representations purely on motion information makes extraction of scene intrinsic dynamics especially difficult in the presence of camera motion. The smallest decrease in performance is seen by C3D, which, once again, shows that combined spatial and temporal information provides the strongest basis for dynamic scene characterization, even in the presence of camera motion. Here, it is worth noting that because no previous dynamic scenes dataset contained both static and moving camera examples for each class, it was more difficult to draw such conclusions.

No single algorithm is the top performer across all scene categories (Table 7.3). It is particularly interesting to compare the two approaches based on hand-crafted features (BoSE and IDT), as the nature of what they are extracting is most explicitly defined. Trajectory-based IDT excels where scenes can be characterized by motion of features across time, *e.g.* the operation of an elevator or the falling of a tree. In complement, spatiotemporal orientation-based BoSE excels where scenes can be characterized by dynamic texture, *e.g.* flickering of forest fires and turbulence of waterfalls. Along similar lines, while T-ResNet is the overall better performer than IDT, it seems that T-ResNet exhibits weaker performance for scene dynamics with rather irregular or mixed defining motion patterns as snowing or fireworks, both categories being resolved quite well by IDT. It also is interesting to note that the most challenging classes for the spatially-based approaches, S-CNN and ResNet, are those where motion is particularly important, *e.g.* with snowing being the most or second most difficult for each. More generally, classes that are most strongly defined by motion tend to be the most difficult for most algorithms considered, suggesting that capturing differences between scenes based on their dynamics remains an area for additional research.

Class	SFA	BoSE	T-CNN	S-CNN	IDT	C3D	ResNet	T-ResNet
Beach	92.6	83.3	72.2	75.9	87.0	83.3	90.7	74.1
BuildingCollapse	66.7	66.7	37.0	81.5	87.0	83.3	83.3	94.4
Elevator	85.2	98.1	79.6	100.0	100.0	98.1	100.0	100.0
Escalator	48.1	74.1	37.0	90.7	66.7	87.0	88.9	92.6
FallingTrees	42.6	79.6	53.7	88.9	98.1	88.9	92.6	88.9
Fireworks	51.9	83.3	38.9	66.7	98.1	81.5	87.0	96.3
ForestFire	29.6	77.8	9.3	92.6	72.2	79.6	96.3	100.0
Fountain	18.5	44.4	11.1	38.9	57.4	35.2	83.3	75.9
Highway	55.6	50.0	50.0	63.0	68.5	64.8	74.1	79.6
LightningStorm	42.6	79.6	77.8	81.5	94.4	87.0	90.7	90.7
Marathon	66.7	88.9	92.6	96.3	98.1	100.0	100.0	100.0
Ocean	64.8	70.4	51.9	83.3	74.1	96.3	66.7	85.2
Railway	29.6	83.3	53.7	96.3	88.9	88.9	100.0	100.0
RushingRiver	55.6	81.5	72.2	87.0	87.0	100.0	88.9	85.2
SkyClouds	83.3	94.4	74.1	90.7	88.9	98.1	96.3	96.3
Snowing	14.8	57.4	33.3	51.9	90.7	46.3	33.3	53.7
Street	79.6	90.7	44.4	92.6	96.3	98.1	100.0	98.1
Waterfall	77.8	85.2	13.0	88.9	66.7	90.7	57.4	75.9
WavingFlags	53.7	81.5	61.1	87.0	98.1	88.9	96.3	98.1
WindmillFarm	79.6	70.4	50.0	87.0	92.6	83.3	94.4	94.4
Average	56.9	77.0	50.6	82.0	85.6	84.0	85.9	89.0

Table 7.3: Performance comparison of different algorithms on the entire **YUP++ dataset (static and moving camera)**.

7.5.5 Impact of the new dataset

The new **YUP++** dataset has allowed for empirical study of the state-of-the-art in visual recognition approaches applied to dynamic scenes in ways not previously possible. First, by providing increased overall difficulty in comparison to previous datasets, it has allowed for clear performance distinctions to be drawn across a range of algorithms. Second, it has documented that even the strongest extant approaches suffer non-negligible performance decrements when operating in the presence of camera motion in comparison to a stabilized camera scenario. For example, the top overall performer, T-ResNet, has an overall decrement of over 10% in moving from static to moving camera scenarios. Third, the dataset has been shown to have adequate diversity to support ConvNet training on as little as 10% of its total, *e.g.* with T-ResNet transformed from ResNet for great performance improvements on that basis. Fourth, the dataset has provided insight into how different scene characteristics can impact algorithm performance, *e.g.* the relative impact of regular vs. irregular motion patterns.

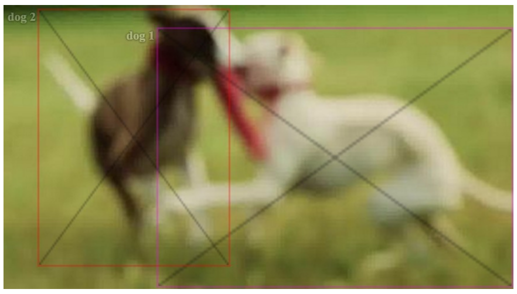
Moving forward the dataset can continue to support advances in dynamic scene research. First, algorithmic advances focused on discounting camera motion can be developed relative to a dataset that controls exactly for this variable. For example, the impact of image stabilization preprocessing can be studied. Similarly, the development of feature representations that aim for invariance with respect to camera motion can be supported. Second, from a learning perspective the impact of training on stabilized and testing on moving camera scenarios (and vice versa) can be studied. Third, and more generally, given that top performance of the evaluated algorithms exhibits less than 90% accuracy on the entire dataset and less than 82% on the moving camera subset, there is ample room for further benchmarking of improved algorithms using **YUP++**.

7.6 Summary

We have presented a general spatiotemporal ConvNet, T-ResNet, based on transforming a purely spatial network to one that can encompass spacetime via hierarchical injection of temporal residuals. In comparison to a representative set of strong performing alternative approaches to video-based recognition, our approach has produced the best overall performance on a novel dynamic scene recognition dataset.

Our new database extends previous dynamic scenes evaluation sets in both diversity and size: It adds new scene classes and provides balanced samples with and without camera motion. Significantly, all algorithms show a decrement in performance when confronted with camera motion, suggesting that a promising research direction for future studies is the development of approaches that are robust to this variable. Moving forward, the new dataset can continue to support development and comparative evaluation of algorithms for dynamic scene understanding.

(a)



(b)

(c)

(d)

Challenges for video object detection in realistic video. The images show training examples from the ImageNet video object detection challenge for the classes: (a) bicycle, bird, rabbit; (b) dog; (c) fox; and (d) red panda. In this chapter, we develop a unified approach for video object detection & tracking. Best viewed electronically and with zoom.

8
Deep Video Detection & Tracking

8.1 Motivation

Object detection in video has seen a surge in interest lately, especially since the introduction of the ImageNet (Russakovsky et al., 2015b) object detection from video challenge (VID). Different from the ImageNet object detection (DET) challenge, VID shows objects in image sequences and comes with additional challenges of (i) size: video provides a massive number of frames (VID has around 1.3M images, compared to around 400K in DET or 100K in COCO (Lin et al., 2014)), (ii) motion blur: due to rapid camera or object motion, (iii) quality: internet video clips are typically of lower quality than static photos, (iv) partial occlusion: due to change in objects/viewer positioning and (v) pose: unconventional object-to-camera poses are frequently seen in video. In the figure above, we show example images from the VID dataset; for more examples please see[1]. On the positive side, however, characteristic object motions have potential to be useful in recognition.

To solve this challenging task, recent top entries in the ImageNet (Russakovsky et al., 2015b) video detection challenge use exhaustive post-processing on top of frame-level detectors. For example, the winner (Kang et al., 2016a) of ILSVRC'15 uses two multi-stage Faster R-CNN (Ren et al., 2016) detection frameworks, context suppression, multi-scale training/testing, a ConvNet tracker (Wang et al., 2015a), optical-flow based score propagation and model ensembles.

In this chapter we propose Detect and Track (D&T), a unified approach to tackle the problem of object detection in realistic video. Our objective is to directly infer a 'tracklet' over multiple frames by simultaneously carrying out detection and tracking with a ConvNet. To achieve this we propose to extend the R-FCN (Li et al., 2016a) detector with a tracking formulation that is inspired by current correlation and regression based trackers (Bertinetto et al., 2016, Held et al., 2016, Ma et al., 2015). We train a fully convolutional architecture end-to-end using a detection and tracking based loss and term our approach D&T for joint Detection and Tracking. The input to the network consists of multiple frames which are first passed through a ConvNet trunk (*e.g.* a ResNet-101 (He et al., 2016a)) to produce convolutional features that are shared for the task of detection and tracking. We compute convolutional cross-correlation between the feature responses of adjacent frames to estimate the local displacement at different feature scales. On top of the features, we employ an RoI-pooling layer (Li et al., 2016a) to classify and regress box proposals as well as an RoI-tracking layer that regresses box transformations (translation, scale, aspect ratio) across frames. Our architecture is fully convolutional up to RoI-pooling/tracking and can be trained end-to-end for object detection and tracking. Finally, to infer long-term tubes of objects across a video we link detections based on our tracklets.

An evaluation on the large-scale ImageNet VID dataset shows that our approach is able to achieve better single-model performance than the winner of the last ILSVRC'16 challenge, despite being conceptually simple and much faster. We show that including a tracking loss may improve feature learning for better static object detection and also show a very fast version of D&T that works on temporally-strided input frames. We think that given its high accuracy and speed, our unified framework can foster further research and applications in large scale video detection.

[1]`http://vision.cs.unc.edu/ilsvrc2015/ui/vid`

Our code is publicly available at https://github.com/feichtenhofer/Detect-Track.

8.2 Related work

Object detection. Object detection has been studied for decades. Most of the recent progress can be attributed to the rise of deep ConvNets. Two families of detectors are currently popular: First region proposal based detectors R-CNN (Girshick et al., 2014), Fast R-CNN (Girshick, 2015), Faster R-CNN (Ren et al., 2016) and R-FCN (Li et al., 2016a) and second, detectors that directly predict boxes for an image in one step such as YOLO (Redmon et al., 2016) and SSD (Liu et al., 2016).

Our approach builds on R-FCN (Li et al., 2016a) which is a simple and efficient framework for object detection on region proposals with a fully convolutional nature. In terms of accuracy it is competitive with Faster R-CNN (Ren et al., 2016), which uses a multi-layer network that is evaluated per-region (and thus has a cost growing linearly with the number of candidate RoIs). R-FCN reduces the cost for region classification by pushing the region-wise operations to the end of the network with the introduction of a position-sensitive RoI pooling layer that works on convolutional features that encode the spatially subsampled class scores of input RoIs.

Tracking. Tracking is also an extensively studied problem in computer vision, with most recent progress devoted to trackers operating on deep ConvNet features. In (Nam and Han, 2016) a ConvNet is fine-tuned at test-time to track a target from the same video via detection and bounding box regression. Training on the examples of a test sequence is slow and also not applicable in the object detection setting. Other methods use pre-trained ConvNet features to track and have achieved strong performance either with correlation (Bertinetto et al., 2016, Ma et al., 2015) or regression trackers on heat maps (Wang et al., 2015a) or bounding boxes (Held et al., 2016). The regression tracker in (Held et al., 2016) is related to our method. It is based on a a siamese ConvNet that predicts the location in the second image of the object shown in the center of the previous image. Since this tracker predicts a bounding box instead of just the position, it is able to model changes in scale and aspect of the tracked template. The major drawback of this approach is that it only can process a single target template and it also has to rely on significant data augmentation to learn all possible transformations of tracked boxes. The approach in (Bertinetto et al., 2016) is an example of a correlation tracker and inspires our method. The tracker also uses a fully-convolutional Siamese network that takes as input the tracking template and the search image. The ConvNet features from the last convolutional layer are correlated to find the target position in the response map. One drawback of many correlation trackers (Bertinetto et al., 2016, Ma et al., 2015) is that they only work on single targets and do not account for changes in object scale and aspect ratio.

Video object detection. Action detection is also a related problem and has received increased attention recently mostly with methods building on two-stream ConvNets (Simonyan and Zisserman, 2014a). In (Gkioxari and Malik, 2015) a method is presented that uses a two-stream R-CNN

(Girshick et al., 2014) to classify regions and link them across frames based on the action predictions and their spatial overlap. This method has been adopted by (Saha et al., 2016) and (Peng and Schmid, 2016) where the R-CNN was replaced by Faster R-CNN with the RPN operating on two streams of appearance and motion information.

One area of interest is learning to detect and localize in each frame (e.g. in video co-localization) with only weak supervision. The YouTube Object Dataset (Prest et al., 2012), has been used for this purpose, *e.g.* (Joulin et al., 2014, Kwak et al., 2015).

Since the object detection from video task has been introduced at the ImageNet challenge, it has drawn significant attention. In (Kang et al., 2016b) tubelet proposals are generated by applying a tracker to frame-based bounding box proposals. The detector scores across the video are re-scored by a 1D CNN model. In their corresponding ILSVRC submission the group (Kang et al., 2016a) added a propagation of scores to nearby frames based on optical flows between frames and suppression of class scores that are not under the top classes in a video. A more recent work (Kang et al., 2017) introduces a tubelet proposal network that regresses static object proposals over multiple frames, extracts features by applying Faster R-CNN, which are finally processed by an encoder-decoder LSTM. In deep feature flow (Zhu et al., 2017) a recognition ConvNet is applied to key frames only and an optical flow ConvNet is used for propagating the deep feature maps via a flow field to the rest of the frames. This approach can increase detection speed by a factor of 5 at a slight accuracy cost. The approach is error-prone due largely to two aspects: First, propagation from the key frame to the current frame can be erroneous and, second, the key frames can miss features from current frames. Very recently a new large-scale dataset for video object detection has been introduced (Real et al., 2017) with single objects annotations over video sequences.

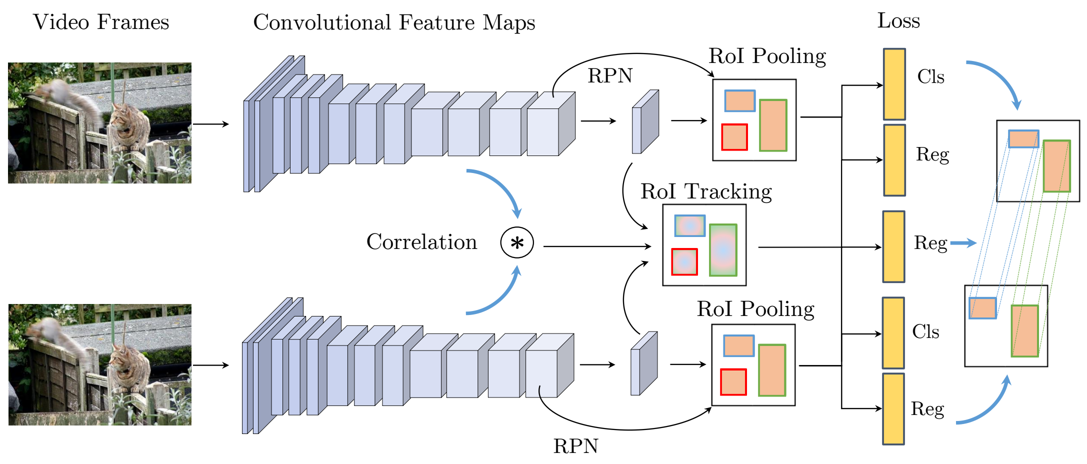


Figure 8.1: Architecture of our Detect and Track (D&T) approach (see Section 8.3 for details).

8.3 Approach

In this section we first give an overview of our Detect and Track (D&T) approach (Section 8.3.1), then summarize the baseline R-FCN detector (Li et al., 2016a) (Section 8.3.2) and formulate our tracking objective as cross-frame bounding box regression (Section 8.3.3), finally, we introduce correlation features (Section 8.3.4) to aid the network in the tracking process.

The next Section 8.4 shows how we link across-frame tracklets to tubes over the temporal extent of a video and Section 8.5 describes how we apply our approach to the ImageNet VID challenge.

8.3.1 D&T overview

We aim at jointly detecting and tracking (D&T) objects in video. Fig. 8.1 illustrates our D&T architecture. We build on the R-FCN (Li et al., 2016a) object detection framework that is fully convolutional up to region classification and regression, and extend it for multi-frame detection and tracking. Given a set of two high-resolution input frames our architecture first computes convolutional feature maps that are shared for the tasks of detection and tracking (*e.g.* the features of a ResNet-101(He et al., 2016a)). A Region Proposal Network (RPN) is used to propose candidate regions in each frame based on the objectness likelihood for pre-defined candidate boxes (*i.e.* "anchors"(Ren et al., 2016)). Based on these regions, RoI pooling aggregates position-sensitive score maps produced from an intermediate convolutional layer to classify boxes and refine their coordinates (regression).

We extend this architecture by introducing a regressor that takes the intermediate position-sensitive regression maps from both frames (together with correlation maps, see below) as input to an RoI tracking operation which outputs the box transformation from one frame to the other. We train the RoI tracking task by extending the multi-task objective of R-FCN with a tracking loss that regresses object coordinates across frames. Our tracking loss operates on ground truth objects and evaluates a soft L1 norm (Girshick, 2015) between coordinates of the predicted track and the ground truth track of an object.

Such a tracking formulation can be seen as a multi-object extension of the single target tracker in (Held et al., 2016) where a ConvNet is trained to infer an object's bounding box from features of the two frames. One drawback of such an approach is that it does not exploit translational equivariance which means that the tracker has to learn all possible translations from training data. Thus such a tracker requires exceptional data augmentation (artificially scaling and shifting boxes) during training (Held et al., 2016) .

A tracking representation that is based on correlation filters (Bolme et al., 2010, Danelljan et al., 2016, Henriques et al., 2015) can exploit the translational equivariance as correlation is equivariant to translation. Recent correlation trackers (Bertinetto et al., 2016, Ma et al., 2015) typically work on high-level ConvNet features and compute the cross correlation between a tracking template and the search image (or a local region around the tracked position from the previous frame). The resulting correlation map measures the similarity between the template and the search image for all circular shifts along the horizontal and vertical dimension. The displacement of a target object can thus be found by taking the maximum of the correlation response map.

Different from typical correlation trackers that work on single target templates, we aim to track multiple objects simultaneously. We compute correlation maps for all positions in a feature map and let RoI tracking additionally operate on these feature maps for better track regression. Our architecture is able to be trained end-to-end taking as input frames from a video and producing object detections and their tracks. The next sections describe how we structure our architecture for end-to-end learning of object detection and tracklets.

8.3.2 Object detection and tracking in R-FCN

Our architecture takes frames $\mathbf{I}^t \in \mathbb{R}^{H_0 \times W_0 \times 3}$ at time t and pushes them through a backbone ConvNet (*i.e.* ResNet-101 (He et al., 2016a)) to obtain feature maps $\mathbf{x}_l^t \in \mathbb{R}^{H_l \times W_l \times D_l}$ where W_l, H_l and D_l are the width, height and number of channels of the respective feature map output by layer l. As in R-FCN (Li et al., 2016a) we reduce the effective stride at the last convolutional layer from 32 pixels to 16 pixels by modifying the conv5 block to have unit spatial stride, and also increase its receptive field by dilated convolutions (Long et al., 2015).

Our overall system builds on the R-FCN (Li et al., 2016a) object detector that works in two stages: First it extracts candidate regions of interest (RoI) using a Region Proposal Network (RPN) (Ren et al., 2016) and second, it performs region classification into different object categories and background by using a position-sensitive RoI pooling layer (Li et al., 2016a). The input to this RoI pooling layer comes from an extra convolutional layer with output $\mathbf{x}_{cls}^t$ that operates on the last convolutional layer of a ResNet (He et al., 2016a). The layer produces a bank of $D_{cls} = k^2(C+1)$ position-sensitive score maps which correspond to a $k \times k$ spatial grid describing relative positions to be used in the RoI pooling operation for each category (C) and background. Applying the softmax function to the outputs leads to a probability distribution p over $C+1$ classes for each RoI. In a second branch R-FCN puts a sibling convolutional layer with output $\mathbf{x}_{reg}^t$ after the last convolutional layer for bounding box regression, again a position-sensitive RoI pooling operation is performed on this bank of $D_{cls} = 4k^2$ maps for class-agnostic bounding box prediction of a box $b = (b_x, b_y, b_w, b_h)$.

Let us now consider a pair of frames $\mathbf{I}^t, \mathbf{I}^{t+\tau}$, sampled at time t and $t+\tau$, given as input to the network. We introduce an inter-frame bounding box regression layer that performs position sensitive RoI pooling on the concatenation of the bounding box regression features $\{\mathbf{x}_{reg}^t, \mathbf{x}_{reg}^{t+\tau}\}$ to predict the transformation $\Delta^{t+\tau} = (\Delta_x^{t+\tau}, \Delta_y^{t+\tau}, \Delta_w^{t+\tau}, \Delta_h^{t+\tau})$ of the RoIs from t to $t+\tau$. The correlation features, that are also used by the bounding box regressors, are described in section 8.3.4. We show an illustration of this approach in Fig. 8.2.

8.3.3 Multitask detection and tracking objective

To learn this regressor, we extend the multi-task loss of Fast R-CNN (Girshick, 2015), consisting of a combined classification L_{cls} and regression loss L_{reg}, with an additional term that performs tracking across two frames L_{tra}. For a single iteration and a batch of N RoIs the network predicts softmax probabilities $\{p_i\}_{i=1}^{N}$, regression offsets $\{b_i\}_{i=1}^{N}$, and cross-frame RoI-tracks $\{\Delta_i^{t+\tau}\}_{i=1}^{N_{tra}}$. Our overall objective function is written as:

$$\begin{aligned} L(\{p_i\},\{b_i\},\{\Delta_i\}) = \frac{1}{N}\sum_{i=1}^{N} L_{cls}(p_{i,c^*}) \\ +\lambda\frac{1}{N_{fg}}\sum_{i=1}^{N}[c_i^* > 0]L_{reg}(b_i, b_i^*) \\ +\lambda\frac{1}{N_{tra}}\sum_{i=1}^{N_{tra}} L_{reg}(\Delta_i^{t+\tau}, \Delta_i^{*,t+\tau}). \end{aligned} \tag{8.1}$$

Where the ground truth class label of an RoI is defined by c_i^* and its predicted softmax score is p_{i,c^*}. b_i^* is the ground truth regression target, and $\Delta_i^{*,t+\tau}$ is the track regression target. The indicator function $[c_i^* > 0]$ is 1 for foreground RoIs and 0 for background RoIs (with $c_i^* = 0$). $L_{cls}(p_{i,c^*}) = -\log(p_{i,c^*})$ is the cross-entropy loss for box classification, and L_{reg} is the bounding box regression loss defined as the smooth L1 function in (Girshick, 2015). The tradeoff parameter is set to $\lambda = 1$ as in (Girshick, 2015, Li et al., 2016a). The assignment of RoIs to ground truth is as follows: a class label c^* and regression targets b^* are assigned if the RoI overlaps with a ground-truth box at least by 0.5 in intersection-over-union (IoU) and the tracking target $\Delta^{*,t+\tau}$ is assigned only to ground truth targets which are appearing in both frames. Thus, the first term of (8.1) is active for all N boxes in a training batch, the second term is only active for N_{fg} foreground RoIs and the last term is only active for N_{tra} ground truth RoIs that have a track correspondence across the two frames.

For track regression we use the bounding box regression parametrisation of R-CNN (Girshick, 2015, Girshick et al., 2014, Ren et al., 2016). For a single object we have ground truth box coordinates $B^t = (B_x^t, B_y^t, B_w^t, B_h^t)$ in frame t, and similarly $B^{t+\tau}$ for frame $t+\tau$, denoting the horizontal & vertical centre coordinates and its width and height. The tracking regression values for the target $\Delta^{*,t+\tau} = \{\Delta_x^{*,t+\tau}, \Delta_y^{*,t+\tau}, \Delta_w^{*,t+\tau}, \Delta_h^{*,t+\tau}\}$ are then

$$\Delta_x^{*,t+\tau} = \frac{B_x^{t+\tau} - B_x^t}{B_w^t} \qquad \Delta_y^{*,t+\tau} = \frac{B_y^{t+\tau} - B_y^t}{B_h^t} \tag{8.2}$$

$$\Delta_w^{*,t+\tau} = \log(\frac{B_w^{t+\tau}}{B_w^t}) \qquad \Delta_h^{*,t+\tau} = \log(\frac{B_h^{t+\tau}}{B_h^t})). \tag{8.3}$$

8.3.4 Correlation features for object tracking

Different from typical correlation trackers on single target templates, we aim to track multiple objects simultaneously. We compute correlation maps for all positions in a feature map and let RoI pooling operate on these feature maps for track regression. Considering all possible circular

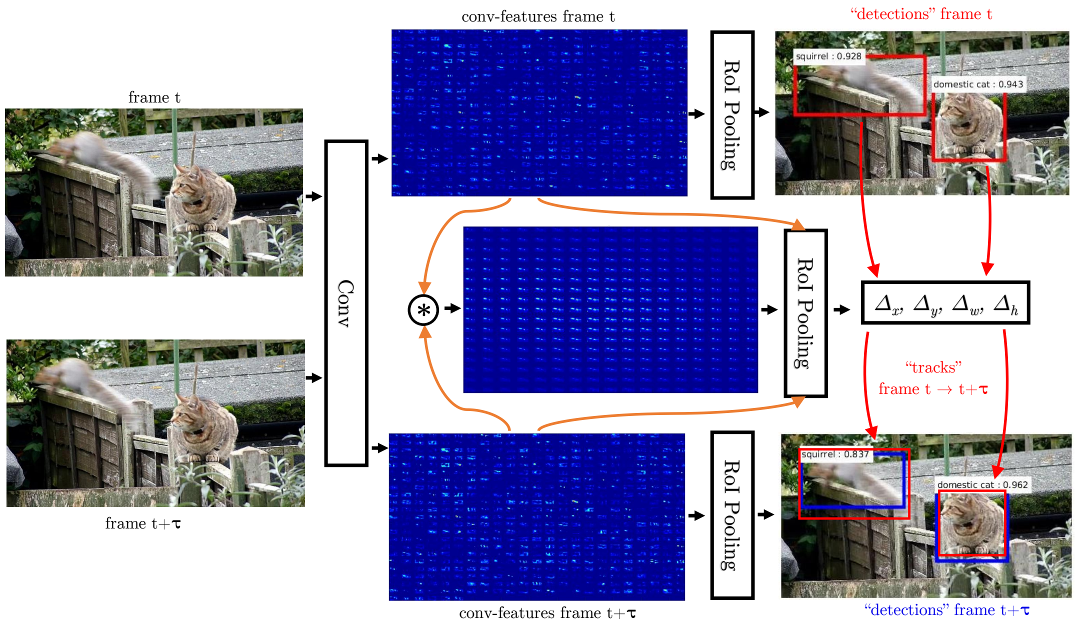


Figure 8.2: Schematic of our approach for two frames at time t and $t + \tau$. The inputs are first passed through a fully-convolutional network to produce feature maps. A correlation layer operates at multiple feature maps of different scales (only the coarsest scale is shown in the figure) and estimates local feature similarity for various offsets between the two frames. Finally, position sensitive RoI pooling (Li et al., 2016a) operates on the convolutional features of the individual frames to produce per-frame detections and also on a stack of individual frame-features as well as the between frame correlation features to output regression offsets of the detection boxes across the two frames.

shifts in a feature map would lead to large output dimensionality and also give responses for too large displacements. Therefore, we restrict correlation to a local neighbourhood. This idea was originally used for optical flow estimation in (Dosovitskiy et al., 2015), where a correlation layer is introduced to aid a ConvNet in matching feature points between frames. The correlation layer performs point-wise feature comparison of two feature maps $\mathbf{x}_l^t, \mathbf{x}_l^{t+\tau}$

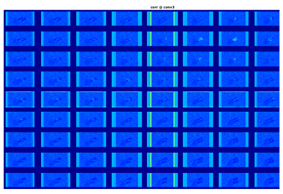
$$\mathbf{x}_{corr}^{t,t+\tau}(i,j,p,q) = \left\langle \mathbf{x}_l^t(i,j), \mathbf{x}_l^{t+\tau}(i+p, j+q) \right\rangle \tag{8.4}$$

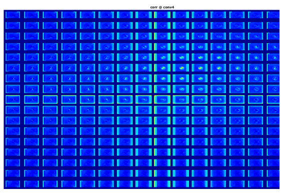
where $-d \leq p \leq d$ and $-d \leq q \leq d$ are offsets to compare features in a square neighbourhood around the locations i, j in the feature map, defined by the maximum displacement, d. Thus the output of the correlation layer is a feature map of size $\mathbf{x}_{corr} \in \mathbb{R}^{H_l \times W_l \times 2d+1 \times 2d+1}$. Equation (8.4) can be seen as a correlation of two feature maps within a local square window defined by d. We compute this local correlation for features at layers conv3, conv4 and conv5 (we use a stride of 2 in i, j to have the same size in the conv3 correlation). We show an illustration of these features for two sample sequences in Fig. 8.3.

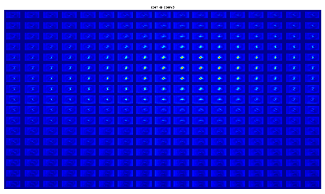
To use these features for track-regression, we let RoI pooling operate on these maps by stacking them with the bounding box features in Section 8.3.2 $\{\mathbf{x}_{corr}^{t,t+\tau}, \mathbf{x}_{reg}^t, \mathbf{x}_{reg}^{t+\tau}\}$.

(a)

(b)

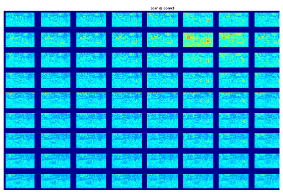

(c)

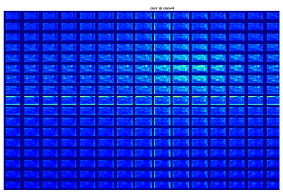

(d)


(e)

(f)

(g)


(h)


(i)

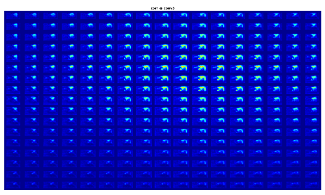

(j)

Figure 8.3: Correlation features for two frames of two validation sequences. For the frames in (a) & (b) , we show in (c),(d) and (e) the correlation maps computed by using features from conv3, conv4 and conv5, respectively. The feature maps are shown as arrays with the map centre corresponding to zero offsets p, q between the frames and the neighbouring rows and columns correspond to shifted correlation maps of increasing p, q. We observe that the airplane moves to the top-right; hence the feature maps corresponding to $p = 3, q = 3$ show strong responses. Note that the features at conv4 and conv5 have the same resolution, whereas at conv3 we use stride 2 correlation sampling to produce same sized outputs. In (h),(i) and (j) we show additional multiscale correlation maps for the frames in (f) & (g) which are affected by camera motion resulting in correlation patterns that correctly estimate this at the lower layer (conv3 correlation responds on the grass and legs of the animal (h)), and also handles the independent motion of the animals at the higher conv5 correlation (j).

8.4 Linking tracklets to object tubes

One drawback of high-accuracy object detection is that high-resolution input images have to be processed which puts a hard constraint on the number of frames a (deep) architecture can process in one iteration (due to memory limitations in GPU hardware). Therefore, a tradeoff between the number of frames and detection accuracy has to be made. Since video possesses much redundant information and objects typically move smoothly in time we can use our inter-frame tracks to link detections in time and build long-term object tubes. To this end, we adopt an established technique from action localization (Gkioxari and Malik, 2015, Peng and Schmid, 2016, Saha et al., 2016), which is used to to link frame detections in time to tubes.

Consider the class detections for a frame at time t, $D_i^{t,c} = \{x_i^t, y_i^t, w_i^t, h_i^t, p_{i,c}^t\}$, where $D_i^{t,c}$ is a box indexed by i, centred at (x_i^t, y_i^t) with width w_i^t and height h_i^t, and $p_{i,c}^t$ is the softmax probability for class c. Similarly, we also have tracks $T_i^{t,t+\tau} = \{x_i^t, y_i^t, w_i^t, h_i^t; x_i^{t+\tau}, y_i^{t+\tau}, w_i^{t+\tau}, h_i^{t+\tau}\}$ that describe the transformation of the boxes from frame t to $t+\tau$. We can now define a class-wise linking score that combines detections and tracks across time

$$s_c(D_{i,c}^t, D_{j,c}^{t+\tau}, T^{t,t+\tau}) = p_{i,c}^t + p_{j,c}^{t+\tau} + \psi(D_i^t, D_j, T^{t,t+\tau}) \tag{8.5}$$

where the pairwise score is

$$\psi(D_{i,c}^t, D_{j,c}^{t+\tau}, T^{t,t+\tau}) = \begin{cases} 1, & \text{if } D_i^t, D_j^{t+\tau} \in T^{t,t+\tau}, \\ 0, & \text{otherwise.} \end{cases} \tag{8.6}$$

Here, the pairwise term ψ evaluates to 1 if the IoU overlap a track correspondences $T^{t,t+\tau}$ with the detection boxes $D_i^t, D_j^{t+\tau}$ is larger than 0.5. This is necessary, because the output of the track regressor does not have to exactly match the output of the box regressor.

The optimal path across a video can then be found by maximizing the scores over the duration $\mathcal{T}$ of the video (Gkioxari and Malik, 2015)

$$\bar{D}_c^\star = \underset{\bar{D}}{\operatorname{argmax}} \frac{1}{\mathcal{T}} \sum_{t=1}^{\mathcal{T}-\tau} s_c(D^t, D^{t+\tau}, T^{t,t+\tau}). \tag{8.7}$$

Eq. (8.7) can be solved efficiently by applying the Viterbi algorithm (Gkioxari and Malik, 2015). Once the optimal tube $\bar{D}_c^\star$ is found, the detections corresponding to that tube are removed from the set of regions and (8.7) is applied again to the remaining regions.

After having found the class-specific tubes $\bar{D}_c$ for one video, we re-weight all detection scores in a tube by adding the mean of the $\alpha = 50\%$ highest scores in that tube. We found that overall performance is largely robust to that parameter, with less than 0.5% mAP variation when varying $10\% \leq \alpha \leq 100\%$. Our simple tube-based re-weighting aims to boost the scores for positive boxes on which the detector fails. Note that our approach enforces the tube to span the whole video and, for simplicity, we do not prune any detections in time. Removing detections with subsequent low scores along a tube could clearly improve the results, but we leave that for future work. In the following section our approach is applied to the video object detection task.

8.5 Experiments

8.5.1 Dataset sampling and evaluation

We evaluate our method on the ImageNet (Russakovsky et al., 2015b) object detection from video (VID) dataset[2] which contains 30 classes in 3862 training and 555 validation videos. The objects have ground truth annotations of their bounding box and track ID in a video. Since the ground truth for the test set is not publicly available, we measure performance as mean average precision (mAP) over the 30 classes on the validation set by following the protocols in (Kang et al., 2016a,b, 2017, Zhu et al., 2017), as is standard practice.

The 30 object categories in ImageNet VID are a subset of the 200 categories in the ImageNet DET dataset. Thus we follow previous approaches (Kang et al., 2016a,b, 2017, Zhu et al., 2017) and train our R-FCN detector on an intersection of ImageNet VID and DET set (only using the data from the 30 VID classes). Since the DET set contains large variations in the number of samples per class, we sample at most 2k images per class from DET. We also subsample the VID training set by using only 10 frames from each video. The subsampling reduces the effect of dominant classes in DET (*e.g.* there are 56K images for the dog class in the DET training set) and very long video sequences in the VID training set.

8.5.2 Training and testing

RPN. Our RPN is trained as originally proposed (Ren et al., 2016). We attach two sibling convolutional layers to the stride-reduced ResNet-101 (Section 8.3.2) to perform proposal classification and bounding box regression at 15 anchors corresponding to 5 scales and 3 aspect ratios. As in (Ren et al., 2016) we also extract proposals from 5 scales and apply non-maximum suppression (NMS) with an IoU threshold of 0.7 to select the top 300 proposals in each frame for training/testing our R-FCN detector. We found that pre-training on the full ImageNet DET set helps to increase the recall; thus, our RPN is first pre-trained on the 200 classes of ImageNet DET before fine-tuning on only the 30 classes which intersect ImageNet DET and VID. Our 300 proposals per image achieve a mean recall of 96.5% on the ImageNet VID validation set.

R-FCN. Our R-FCN detector is trained similar to (Li et al., 2016a, Zhu et al., 2017). We use the stride-reduced ResNet-101 with dilated convolution in conv5 (see Section 8.3.2) and online hard example mining (Shrivastava et al., 2016). A randomly initialized 3×3, dilation 6 convolutional layer is attached to conv5 for reducing the feature dimension to 512 (Zhu et al., 2017) (in the original R-FCN this is a 1×1 convolutional layer without dilation and an output dimension of 1024). For object detection and box regression, two sibling 1×1 convolutional layers provide the $D_{cls} = k^2(C + 1)$ and $D_{reg} = 4k^2$ inputs to position-sensitive RoI pooling layer. We use a $k \times k = 7 \times 7$ spatial grid for encoding relative positions as in (Li et al., 2016a).

[2]`http://www.image-net.org/challenges/LSVRC/`

In both training and testing, we use single scale images with shorter dimension of 600 pixels. We use a batch size of 4 in SGD training and a learning rate of 10^{-3} for 60K iterations followed by a learning rate of 10^{-4} for 20K iterations. For testing we apply NMS with IoU threshold of 0.3.

D & T. For training our D&T architecture we start with the R-FCN model from above and further fine-tune it on the full ImageNet VID training set with randomly sampling a set of two adjacent frames from a different video in each iteration. In each other iteration we also sample from the ImageNet DET training set to not bias our model to the VID training set. When sampling from the DET set we send the same two frames through the network as there are no sequences available. Besides not forgetting the images from the DET training set, this has an additional beneficial effect of letting our model prefer small motions over large ones (*e.g.* the tracker in (Held et al., 2016) samples motion augmentation from a Laplacian distribution with zero mean to bias a regression tracker on small displacements). Our correlation features (8.4) are computed at layers conv3, conv4 and conv5 with a maximum displacement of $d = 8$ and a stride of 2 in i, j for the the conv3 correlation. For training, we use a learning rate of 10^{-4} for 40K iterations and 10^{-5} for 20K iterations at a batch size of 4. During testing our architecture is applied to a sequence with temporal stride τ, predicting detections D and tracklets T between them. We perform non-maximum suppression with bounding-box voting (Gidaris and Komodakis, 2015) before the tracklet linking step to reduce the number of detections per image and class to 25. These detections are then used in eq. (8.7) to extract tubes and the corresponding detection boxes are re-weighted as outlined in Section 8.4 for evaluation.

8.5.3 Results

We show experimental results for our models and the current state-of-the-art in Table 8.1. Qualitative results for difficult validation videos can be seen in Appendix D.

Single frame methods. First we compare methods working on single frames without any temporal processing. Our R-FCN baseline achieves 74.2% mAP which compares favourably to the best performance of 73.9% mAP in (Zhu et al., 2017). We think our slightly better accuracy comes from the use of 15 anchors for RPN instead of the 9 anchors in (Zhu et al., 2017). The Faster R-CNN models working as single frame baselines in (Kang et al., 2016b), (Kang et al., 2017) and (Kang et al., 2016a) score with 45.3%, 63.0% and 63.9%, respectively. We think their lower performance is mostly due to the difference in training procedure and data sampling, and not originating from a weaker base ConvNet, since our frame baseline with a weaker ResNet-50 produces 72.1% mAP (*vs.* the 74.2% for ResNet-101). Next, we are interested on how our model performs after fine-tuning with the tracking loss, operating via RoI tracking on the correlation and track regression features (*i.e.* D (& T loss) in Table 8.1). The resulting performance for single-frame testing is 75.8% mAP. This 1.6% gain in accuracy shows that merely adding the tracking loss can aid the per-frame detection. A possible reason for that result is because the correlation features propagate gradients back into the base ConvNet and therefore make the features more sensitive to important objects

Methods	airplane	antelope	bear	bicycle	bird	bus	car	cattle	dog	domestic cat	elephant	fox	giant panda	hamster	horse	lion
TCN (Kang et al., 2016b)	72.7	75.5	42.2	39.5	25.0	64.1	36.3	51.1	24.4	48.6	65.6	73.9	61.7	82.4	30.8	34.4
TPN+LSTM (Kang et al., 2017)	84.6	78.1	72.0	67.2	68.0	80.1	54.7	61.2	61.6	78.9	71.6	83.2	78.1	91.5	66.8	21.6
Winner ILSVRC'15 (Kang et al., 2016a)	83.7	85.7	84.4	74.5	73.8	75.7	57.1	58.7	72.3	69.2	80.2	83.4	80.5	93.1	84.2	67.8
D (R-FCN)	87.4	79.4	84.5	67.0	72.1	84.6	54.6	72.9	70.9	77.3	76.7	89.7	77.6	88.5	74.8	57.9
D (& T loss)	89.4	80.4	83.8	70.0	71.8	82.6	56.8	71.0	71.8	76.6	79.3	89.9	83.3	91.9	76.8	57.3
D&T ($\tau = 1$)	90.2	82.3	87.9	70.1	73.2	87.7	57.0	80.6	77.3	82.6	83.0	97.8	85.8	96.6	82.1	66.7
D&T ($\tau = 10$)	89.1	79.8	87.5	68.8	72.9	86.1	55.7	78.6	76.4	83.4	82.9	97.0	85.0	96.0	82.2	66.0

Methods	lizard	monkey	motorcycle	rabbit	red panda	sheep	snake	squirrel	tiger	train	turtle	watercraft	whale	zebra	mAP (%)
TCN (Kang et al., 2016b)	54.2	1.6	61.0	36.6	19.7	55.0	38.9	2.6	42.8	54.6	66.1	69.2	26.5	68.6	47.5
TPN+LSTM (Kang et al., 2017)	74.4	36.6	76.3	51.4	70.6	64.2	61.2	42.3	84.8	78.1	77.2	61.5	66.9	88.5	68.4
Winner ILSVRC'15 (Kang et al., 2016a)	80.3	54.8	80.6	63.7	85.7	60.5	72.9	52.7	89.7	81.3	73.7	69.5	33.5	90.2	73.8
Winner ILSVRC'16 (Yang et al., 2016)	(single model performance)														76.2
D (R-FCN)	76.8	50.1	80.2	61.3	79.5	51.9	69.0	57.4	90.2	83.3	81.4	68.7	68.4	90.9	74.2
D (& T loss)	79.0	54.1	80.3	65.3	85.3	56.9	74.1	59.9	91.3	84.9	81.9	68.3	68.9	90.9	75.8
D&T ($\tau = 1$)	83.4	57.6	86.7	74.2	91.6	59.7	76.4	68.4	92.6	86.1	84.3	69.7	66.3	95.2	**79.8**
D&T ($\tau = 10$)	83.1	57.9	79.8	72.7	90.0	59.4	75.6	65.4	90.5	85.6	83.3	68.3	66.5	93.2	78.6

Table 8.1: Performance comparison on the ImageNet VID validation set. The average precision (in %) for each class and the mean average precision over all classes is shown. τ corresponds to the temporal sampling stride.

in the training data. We see significant gains for classes like panda, monkey, rabbit or snake which likely exhibit significant motion.

Multi-frame methods. Next we investigate the effect of multi-frame input during testing. In Table 8.1 we see that linking our detections to tubes based on our tracklets, D&T ($\tau = 1$), raises performance substantially to 79.8% mAP. Some class-AP scores can be boosted significantly (*e.g.* cattle by 9.6, dog by 5.5, cat by 6, fox by 7.9, horse by 5.3, lion by 9.4, motorcycle by 6.4 rabbit by 8.9, red panda by 6.3 and squirrel by 8.5 points AP). This gain is mostly due to the following reason: If an object is captured in an unconventional pose, is distorted by motion blur, or appears at a small scale, the detector might fail; however, if its tube is linked to other potentially highly scoring detections of the same object, these failed detections can be recovered (even though we use a very simple re-weighting of detections across a tube). The only class that loses AP is whale

(-2.6 points) and this has an obvious reason. In most validation snippets the whales successively emerge and submerge from the water and our detection rescoring based on tubes would assign false positives when they submerge for a couple of frames.

When comparing our 79.8% mAP against the current state of the art, we make the following observations. The method in (Kang et al., 2016b) achieves 47.5% by using a temporal convolutional network on top of the still image detector. An extended work (Kang et al., 2017) uses an encoder-decoder LSTM on top of a Faster R-CNN object detector that work on proposals from a tubelet proposal network, produces 68.4% mAP. An even more complex approach is used by the winner of the ILSVRC 2015 challenge (Kang et al., 2016a) which combines two Faster R-CNN detectors, multi-scale training/testing, context suppression, high confidence tracking (Wang et al., 2015a) and optical-flow-guided propagation to achieve 73.8%. Finally, we compare to the challenge winner from ILSVRC2016 (Yang et al., 2016), which uses a cascaded R-FCN detector, context inference, cascade regression and a correlation tracker (Ma et al., 2015) to achieve 76.19% mAP validation performance with a single model (multi-scale testing and model ensembles boost their accuracy to 81.1%). Here it is noteworthy that our method does not have to rely on these complicated elaborations.

Online capabilities and runtime. The only component limiting online application is the tube rescoring. We have evaluated an online version which performs only causal rescoring across the tracks. The performance for this method is 78.7%mAP, compared to the noncausal method (79.8%mAP). Since the correlation layer and track regressors are operating fully convolutional (no additional per-ROI computation is added except at the ROI-tracking layer), the extra runtime cost for testing a 1000x600 pixel image is 14ms (i.e. 141ms vs 127ms without correlation and ROI-tracking layers) on a Titan X GPU. The (unoptimized) tube linking takes on average 46ms per frame on a single CPU core).

Increasing temporal stride. Lastly, we reconsider a finding from our architecture design in the previous chapters, and look at larger temporal strides τ for detecting and tracking in video, which we found useful for the related task of video action recognition Chapters 3, 4) and 5. Our D & T architecture is evaluated only at every τ^{th} frame of an input sequence and tracklets have to link detections over larger temporal strides. The performance for a temporal stride of $\tau = 10$ is 78.6% mAP which is 1.2% below the full-frame evaluation. We think that such a minor drop is remarkable as the duration for processing a video is now roughly reduced by a factor of 10.

A potential point of improvement is to extend the detector to operate over multiple frames of the sequence. We found that such an extension did not have a clear beneficial effect on accuracy for short temporal windows (*i.e.* augmenting the detection scores at time t with the detector output at the tracked proposals in the adjacent frame at time $t+1$ only raises the accuracy from 79.8 to **80.0%** mAP). Increasing this window to frames at $t \pm 1$ by bidirectional detection and tracking from the t^{th} frame did not lead to any gain. Interestingly, when testing with a temporal stride of $\tau = 10$ and augmenting the detections from the current frame at time t with the detector output at the tracked proposals at $t + 10$ raises the accuracy from 78.6 to 79.2% mAP.

We conjecture that the insensitivity of the accuracy for short temporal windows originates from the high redundancy of the detection scores from the centre frames with the scores at tracked locations. The accuracy gain for larger temporal strides, however, suggests that more complementary information is integrated from the tracked objects; thus, a potentially promising direction for improvement is to detect and track over multiple temporally strided inputs.

8.6 Summary

In this chapter we have presented a unified framework for simultaneous object detection and tracking in video. Our fully convolutional D&T architecture allows end-to-end training for detection and tracking in a joint formulation. In evaluation our method achieves accuracy competitive with the winner of the last ImageNet challenge while being simple and efficient. We demonstrate clear mutual benefits of jointly performing the task of detection and tracking, a concept that can foster further research in video analysis.

9 Conclusion

Deep networks have taken by storm the state-of-the-art in nearly all computer vision domains. Their impact on academia and industry has been spectacular. In some cases they even outperform humans *e.g.* for classifying 1000 ImageNet classes [1] or for reading lips[2]. As a consequence, deep networks are deployed in all kinds of applications, from medicine to transportation.

The first part of this dissertation was devoted to a challenging problem in which machine perception is still far from rivaling human performance – action recognition in video, an area that is intensively researched, but still far from the level of performance at which humans perform this task. Chapters 3-5 represent the core contributions of this thesis towards advancing this difficult task. We build on two-stream ConvNets which have shown strong performance for action recognition in videos. Nevertheless, the two-stream architecture (or similar previous methods) is not able to explicitly model two very important cues for recognition in video: First, recognizing what is moving where, *i.e.* registering appearance recognition with motion recognition and second, modeling how these instantaneous spatiotemporal measurements evolve over time. Our main objective towards advancing video action recognition, was to rectify these limitations by developing architectures that are able to fuse appearance and motion information at several levels of granularity in feature abstraction, and with hierarchical integration over space and time. In particular, we investigated several aspects of spatiotemporal feature learning which can be grouped into the two concepts of learning of hierarchical spatiotemporal features and temporal integration over long-term input.

[1]This might not be a fair comparison since the classes in ImageNet are dominated by fine sub-categories such as different kinds of mushrooms or dogs.

[2]`https://www.theverge.com/2016/11/24/13740798/google-deepmind-ai-lip-reading-tv`

For the concept of learning spatiotemporal features, we first studied spatial fusion of two deep appearance and motion recognition networks while taking into account spatial feature registration. This approach is motivated by the importance of such local cues for discriminating between difficult cases such as actions of brushing teeth and shaving beard which can be discriminated by subtle differences in the appearance cues representing the tools used (*e.g.* tooth brush or razor blade) and the periodic motion patterns of the hand movement for performing these actions. An architecture that explicitly models these local spatiotemporal cues should have a theoretical advantage over an architecture that does not form explicit representations to model such spacetime correspondences.

In Chapter 3 we attempt spatial fusion at a particular convolutional layer such that features at the same spatial position are put in correspondence to form intermediate, local spacetime representations. Since the two networks have the same spatial resolution at the layers to be fused one can overlay (concatenate) features from one network on the other to perform fusion. However, there is also the issue of which feature channel (or channels) in one network correspond to a particular channel (or channels) of the other network. We studied several fusion functions that fuse two feature maps at the same instance in time, *i.e.* sum, max, concatenation, bilinear and convolutional fusion. Our experimental results revealed that a simple sum operation performed near optimal among the alternatives. This result is surprising, because it either assumes some direct correspondence of the channels in the two networks at the fusion layer, or the feature averaging is fully compensated by learning in the later layers of the network that work on top of the sum-fused features. A more advanced fusion, which provided overall best results in Chapter 3, was an extra convolutional layer that performed the fusion by learned, weighted summation on the stacked features of the two networks. Another surprising finding was that the filters of this convolutional fusion layer can be initialized to perform the sum operation of the corresponding channels from the two networks, which led to the same performance as if the filters in that layer are initialized randomly, but convergence is an order of magnitude faster if the filters are initialized to perform the sum operation across corresponding channels. A noteworthy result from Chapter 3 was also that two networks can be fused at the last convolution layer while removing the fully connected layers of one stream completely, to allow a substantial saving in parameters, without a significant sacrifice of performance.

An alternative viewpoint is to consider the sum-fusion as a residual connection across two networks of different modalities. In Chapter 4 we build on our insights to present spatiotemporal Residual Networks by using residual connections for spatial fusion between the appearance and motion pathways of a two-stream ResNet architecture. These connections are naturally integrated in ConvNet design principles to allow learning of multiscale spatiotemporal features by designing such interactions across multiple layers of the network hierarchy. Importantly, we have shown that a direct fusion between identical layers of the two networks leads to poor results, but a spatial fusion into the mirror residual unit of the opposing stream allows local learning of spacetime features. As this connection of the appearance and motion channels is conducted at multiple levels in the network hierarchy, our ST-ResNet allows the hierarchical learning of spacetime features. As a further contribution of Chapter 4, we showed that using a smaller batch size for the noisy bias and variance estimation in batch normalization fosters generalization when training very deep two-stream ResNets for action recognition.

In Chapter 5 we thoroughly discussed the combination of the two-stream and ResNet approaches by providing ablation studies that increased our understanding of how these techniques interact. We compared asymmetrical injection of motion information into the appearance stream as well as symmetric (bidirectional) connections across the two streams. In empirical evaluation, we showed that bidirectional interactions yield inferior results to the asymmetric case of injecting motion into appearance and conjectured that this originates from the dominance of the appearance stream over the motion stream during training, since appearance is a much stronger cue for reducing the training objective. Another core contribution of Chapter 5 is multiplicative motion gating of the appearance stream. Our proposed spatiotemporal Multiplier Network provides a nontrivial performance boost over an additive (sum) formulation. We theoretically discussed the advantages of multiplicative interactions by the effect on the backward signal in a residual network, which enforces strong feature correspondence by multiplying the gradient of one stream with the current forward signal of the opposing stream. In comparison to the additive fusion, multiplicative gating increases the order of the network fusion from first to second order. In the forward pass, multiplication implies a much stronger signal change based on spatiotemporal feature correspondence compared to the additive interaction by the direct scaling of the appearance information through the motion signal, rather than via a more subtle bias as in the additive case. During backpropagation, instead of the fusion gradient flowing through the appearance and motion streams being distributed uniformly due to additive forward interaction, it is now multiplicatively factored by the opposing stream's current inputs. This latter type of interaction allows the streams to more effectively interact during the learning process and makes the system more capable of learning corresponding spatiotemporal features.

In summary, our multiplicative formulation for spatial fusion exhibits several theoretical benefits for recognition in video. Suppose for the moment that different channels in the appearance network are responsible for facial areas (mouth, hair, etc), and particular channels in the motion network are responsible for periodic optical flow fields which align with these concepts in provided input data. Multiplicative feature fusion combines the channels of both streams by enforcing strong spatiotemporal correspondence of the appearance and motion stream via backpropagation through the respective filters. The weights in the subsequent layers of the residual unit are able to effectively learn the relationships across these channels in order to best discriminate between related actions using spacetime features capturing information at similar spatiotemporal scales. As a final note, our findings for deep network fusion are generally applicable to any problem that attempts to fuse information from deep architectures, which are capturing different input modalities (*e.g.* speech, language, depth), and are not tied to the particular modalities of motion and appearance input.

For the concept of learning long-term correspondences, we first studied in Chapter 3 how to best fuse two networks temporally. To capture the long-term nature of input sequences and allow the architecture to best discriminate training sequences, we introduced temporally strided input sampling. A training sequence is divided into temporal chunks with temporal stride between them.

This approach is a memory efficient form of dilated filtering in the first convolutional layer, with the dilation factor being the temporal stride. The temporal chunks are fed into the ConvNets to capture short-term information at a fine temporal scale and spatiotemporal convolution layers put that into context with temporally adjacent chunks, thus operating at a coarse temporal scale. Our explored ways of fusing over time include 3D pooling from local spatiotemporal neighbourhoods by first stacking the feature maps across time and then shrinking this spatiotemporal cube by applying local average or maximum pooling, or additionally performing a convolution with a 3D fusion kernel that spans the feature channels, space and time before 3D pooling. Notably, for using 3D fusion kernels, we found it essential to initialize as sum operation across the feature channels, since otherwise learning such a 3D filters would require significant amounts of training data. The fusion filter is able to learn correspondences between highly abstract features of the spatial stream and temporal stream over multi-frame input. After fusion, 3D pooling gathers the resulting features and the spatiotemporal loss is evoked for supervision. We also discussed the issue of how often to sample the temporal sequence, however, our final architecture in Chapter 3 was limited in its capacity for temporal modelling. Our temporal fusion layer received temporal chunks that are multiple frames apart by applying the two stream towers to the input video at temporally strided input locations. This enabled us to capture short scale temporal features at the input of the temporal network and putting them into context over a longer temporal scale at the higher fusion layer of the network. Our technique has already had an impact in the field, with popular methods adopting similar techniques, *e.g.* (Wang et al., 2016) where the two streams are applied to temporal segments of the input. Although, that design provided the fusion layer with explicit capabilities to model long-term structure in the input, it did not allow a hierarchical integration of temporal information into the architecture.

Building upon our findings for long-term temporal aggregation over strided input samples, we developed a spatiotemporal residual architecture in Chapter 4. Our ST-ResNet not only allows the hierarchical learning of spacetime features by fusion of the appearance and motion channels at multiple scales, it further employs a transformation of pre-trained spatial kernels to temporal kernels. In particular, we transfer both streams from the spatial to the spatiotemporal domain by transforming the dimensionality mapping filters of a pre-trained model into temporal convolutions, initialized as residual filters over time. This technique builds on the ConvNet design principle of stacking small convolutional filters throughout the hierarchy of the network. The resulting benefits are a large receptive field at higher layers that comes with lower cost compared to the use of large filters. Moreover, we factorize spatiotemporal modelling into separate 2D spatial convolutions and 1D temporal convolutions which is a general ConvNet design principle to ease the learning problem and keep computational demand low. These techniques allow the straightforward use of standard ConvNets that have been pre-trained on large datasets and therefore to leverage the massive amounts of training data in the image domain. We initialize our temporal kernels as an averaging filter that can be thought as residual connections across time and let the network learn to best discriminate image dynamics via backpropagation. This is achieved by simply replicating the learned spatial kernels in pretrained ResNets across time for initialization as a temporal smoothing kernel. Hence, the temporal filters are able to learn the temporal evolution of the appearance and motion features and, moreover, by stacking such filters as the depth of the network increases, highly complex spatiotemporal features can be learned. Also our technique of transforming spatial filters has already had an impact in the field, with popular methods building on it, *e.g.* (Carreira and Zisserman, 2017) where the filter transformation is termed "filter inflation".

In summary, unlike some other approaches that explicitly learn spatiotemporal features using 3D convolutions, our approach has several benefits: Due to the asymmetric kernels the learning problem is factorized and thus eased. We can learn features over large temporal scales by modelling the temporal footprint of our architecture with the input stride in time (*e.g.* typical 3D ConvNets have limited temporal support of around 16 frame snippets (Tran et al., 2015b)). Finally our method has a conceptual benefit for temporal feature learning over a sequence modelling approach (*e.g.* as in recurrent networks), because our learned filter weights can model spatiotemporal information from local to global spatiotemporal scales.

In Chapter 5, we take these ideas of long-term feature learning even further by employing temporal convolutions combined with feature space transformations initialized as identity mappings. Using identity initialization preserves the design principles of residual networks as any significant change in the network path would distort the (pretrained) model and thereby remove most of its representational power. Therefore, the corresponding kernels can be injected at any point in the network since they do not impact the information flow at initialization and during training the filters can adapt their representation under the gradient flow. Our results show clear benefits of using temporal kernels that are able to learn longer temporal relationships. We also compared different temporal initializations and found that low-pass filters are more applicable for videos that typically capture temporally consistent scenes and high-pass temporal initialization is more appropriate for videos that exhibit a higher degree of inter-video diversity, for example caused by camera motion. In summary, our ultimately presented architecture for video action recognition, the spatiotemporal Multiplier Network, enables learning of multiscale spacetime representations by multiplicative interaction of appearance and motion features coupled with injected identity mapping kernels for learning long-term relationships. Our model is trained end-to-end and fully convolutional in spacetime to process a video in a single forward pass and produces overall best results on widely used action recognition datasets.

In Chapter 6, we focused on obtaining a better understanding of what deep spatiotemporal representations are capturing. We visualized deep two-stream representations trained for action recognition at multiple levels of granularity. Our results provided intuitive explanations for what most excites filters across the different network layers. We found that lower layer filters learn similar spatial patterns, even if trained on different input modalities of appearance and flow. For intermediate layers, we found visual stimuli of vastly different acceleration patterns that strongly activate a single neuron; thereby, suggesting invariances to wide spatiotemporal scale changes. When looking at the ultimate classification layers of the visualized networks, we found that these neurons are very broadly tuned to similar concepts at the input. More generally, the chapter provided clear qualitative evidence for separation into two streams for processing appearance and motion

– a principle that has also been found in nature where numerous studies suggest a corresponding separation into ventral and dorsal pathways of the brain.

A weakness of the temporal filters from Chapters 4 and 5 is that they are initialized as temporal low-pass or high-pass kernels across time. In some cases, temporal coherence is a useful property if one can expect that a ConvNet is capturing similar features across time (*e.g.* for repetitive motion patterns seen in actions such as playing instruments). In other cases where the appearance and the instantaneous motion pattern strongly vary over time, a high-pass filtering can be advantageous for finding discriminative information over time, as a low-pass kernel would smooth out potentially important high-frequency temporal variation. If the patterns, however, are broadly tuned in frequency none of the two initializations are optimal. Of course, learning can alter temporal filter kernels, but we argue that such initializations are suboptimal for modelling instantaneous temporal patterns in a video, arising for example as characteristic temporal textures in dynamic scenes. In Chapter 7, we face the task of dynamic scene recognition, which is a relatively new and growing area of investigation in computer vision. Going beyond recognition of scenes from single still images, it addresses how scene dynamics (e.g., waves of water, fluttering of leaves, flow of traffic) can assist in scene classification. Notably, these dynamic patterns typically subsist at short temporal scales and do not satisfy brightness constancy assumptions – a reason for why optical flow based methods usually perform poorly on that task. Different from our methods in Chapters 4 and 5, we propose a temporal ResNet architecture that does not rely on optical flow input. T-ResNet holds spatiotemporal residual units that inject temporal information into residual blocks via temporal filters that are initialized randomly and operate over short temporal scales for capturing defining dynamics of the scenes. The training of temporal filters from scratch is able to learn complex temporal features, as it is initialized to receive more informative temporal gradients (than low- or high-pass initialization); furthermore, our spatiotemporal residual unit can effectively ignore or enhance temporal information by an extra non-linearity packed into the temporal residual path. As noted, T-ResNet directly operates on the video input (without explicitly using optical flow) and can also be viewed as a temporal inception architecture, since the temporal convolution block operates on the dimensionality reduced input for injecting temporal signals.

Research in the area of dynamic scene classification received a major impetus via the introduction of two video datasets designed to support study of this task. Since that time, aggressive algorithm development has led to the need for new datasets of dynamic scenes to drive further efforts. Toward that end, Chapter 7 introduced a new and challenging video database of dynamic scenes that more than doubles the size of those previously available. The dataset encompasses a wide range of natural variations (seasonal and diurnal changes as well as those of viewing parameters) and adds six additional scene classes to those previously available. Moreover, each scene class is captured with and without camera motion to allow for systematic study of how this variable interacts with dynamics of the scene per se. Neither of the major previous dynamic scenes datasets allows for systematic control of this dimension (Derpanis et al., 2012, Shroff et al., 2010). As a further contribution, a representative sampling of the currently best performing methods for visual recognition are benchmarked on the database to provide up to date documentation of the state-of-the-art in dynamic scene recognition. Our evaluation of seven state-of-the-art algorithms on the new dataset shows that algorithms combining spatial and temporal measurements tend to perform best; although, purely spatial measurements also can perform well on its own. In contrast, an algorithm based more purely on optical flow information performs poorly. Significantly, all algorithms show a decrement in performance when confronted with camera motion, suggesting that a promising research direction for future studies is the development of approaches that are robust to this variable. Moving forward, the new dataset can continue to support development and comparative evaluation of algorithms for dynamic scene understanding.

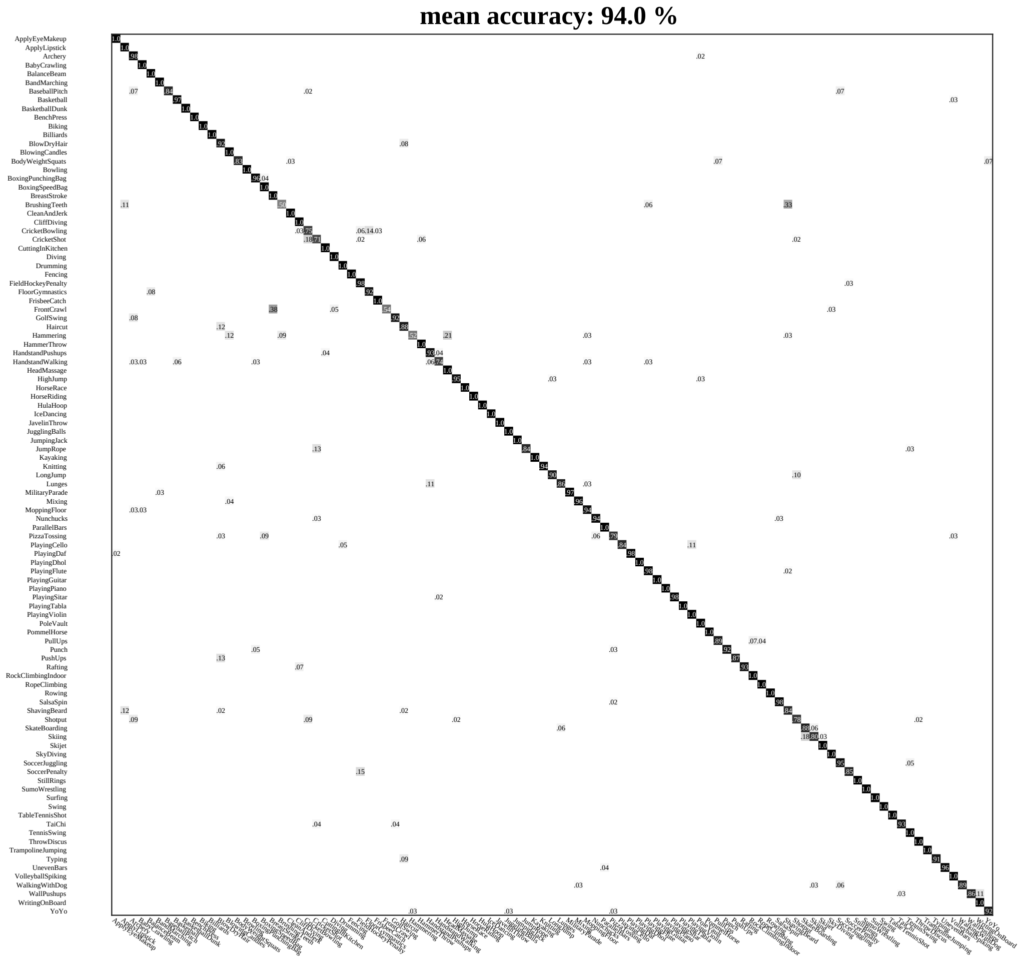
In the final Chapter 8, we have presented a straightforward and unified approach for simultaneous object detection and tracking in video. Our D&T method achieves state of the art on the ImageNet VID challenge over all previous single model entries while being highly efficient. We also carried over our ideas of temporally strided sampling from the previous chapters to the task of video object detection. By looking at larger temporal windows during testing, our D & T architecture is evaluated only at sparse frames of an input sequence and tracklets have to link detections over larger temporal strides. The performance for such sampling is only slightly below the full-frame evaluation, which we think is remarkable as the duration for processing a video can thereby be reduced by orders of magnitudes. A potential point of improvement for D&T is to extend the detector to operate over multiple frames of the sequence. It is straightforward to extend the D&T approach in such a way in order to lengthen the temporal footprint. For example, correlation maps could be computed for a set of neighbouring frames (with stride) symmetrically in time. Notably, we found in preliminary experiments that such an extension did not have a clear beneficial effect on accuracy for short temporal windows. Interestingly, however, when testing with a larger temporal stride and augmenting the detections from the current frame with the detector output from temporally strided frames leads to clear gains in detection accuracy. We conjecture that the insensitivity of the accuracy for short temporal windows originates from the high redundancy of the detection scores from the centre frames with the scores at tracked locations. The accuracy gain for larger temporal strides, however, suggests that more complementary information is integrated from the tracked objects; thus, a potentially promising direction for improvement is to detect and track over multiple temporally strided inputs.

Even though progress of deep learning for single image related tasks has been impressive, deep video recognition has lagged behind. This thesis provided several significant contributions for advancing video recognition with deep learning. The core concept of the dissertation is that video is different from the image domain. Instead of developing approaches that operate directly on the spacetime volume, we design spatiotemporal architectures that explicitly model the interplay of appearance and motion information. This allows us to best take advantage of the regularities, as well as inherent differences, of the spatial and temporal domain instead of directly approaching the problem by a purely data-driven concept. Besides empirical justification, our approach is also loosely supported by the way nature attends this problem where numerous studies suggest a corresponding separation into ventral and dorsal pathways of the human brain.

A Confusion matrices for Spatiotemporal Multiplier Networks

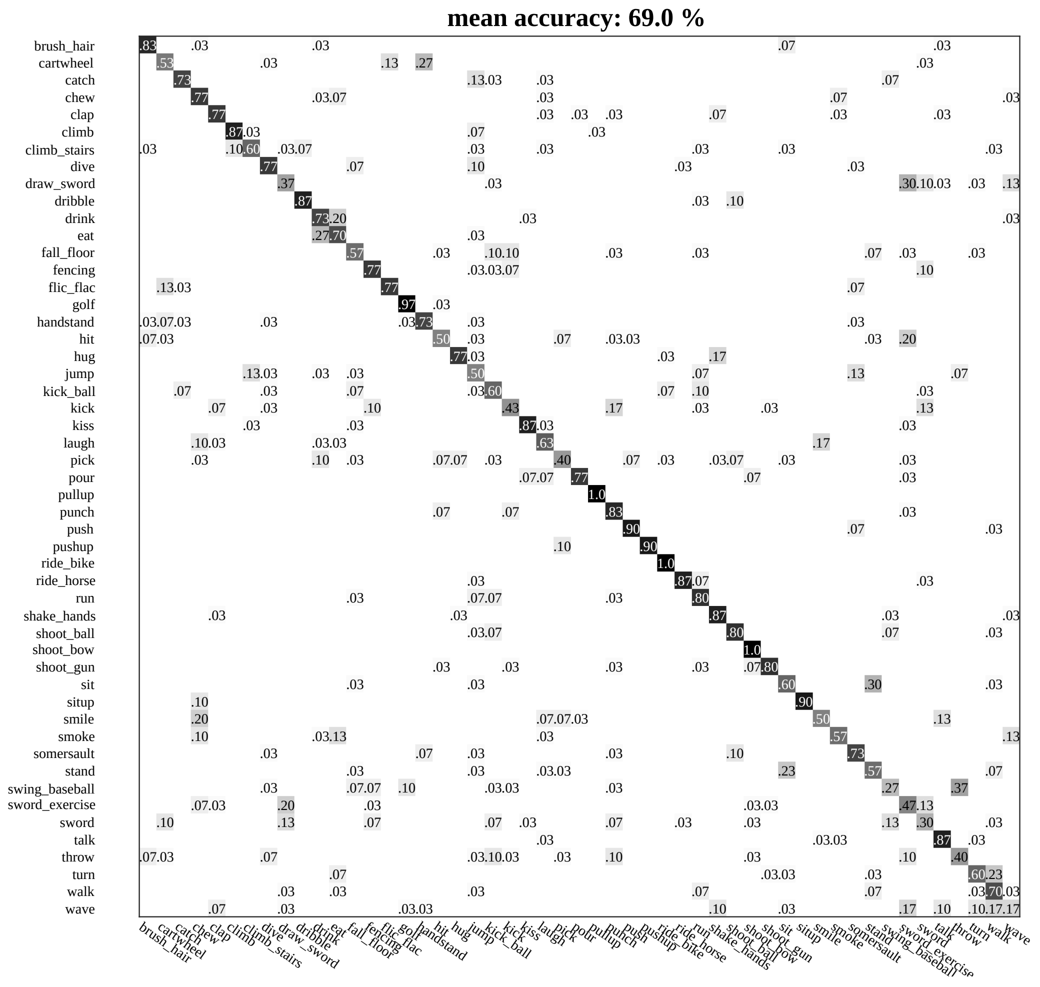
Confusion matrices for our proposed architecture using 50-layer ResNets for both appearance and motion streams are shown in Fig. A.1 for UCF101 and in Fig. A.2 for HMDB51. We observe that on UCF101 lowest accuracy is achieved by the classes BrushingTeeth, 52% (33% confused with ShavingBeard), Hammering, 52% (21% confused with HeadMassage), FrontCrawl, 54% (38% confused with BreastStroke).

On HMDB51 we report lowest performance for the classes wave 17% (17% confused with sword_exercise), swing_baseball 27% (37% confused with throw) and sword 30% (13% confused with draw_sword). In Fig. A.3 and Fig. A.4 we show confusion matrices for our final model that uses a deeper architecture (ResNet-152) for the motion stream (corresponding to the penultimate row of Table 5.5 where the average performance over all three splits is listed, while here for confusion matrices we report results for split 1). Confusions are similar to the case above. Moreover, we see that there is just a slight gain for using a deeper model in the motion stream.



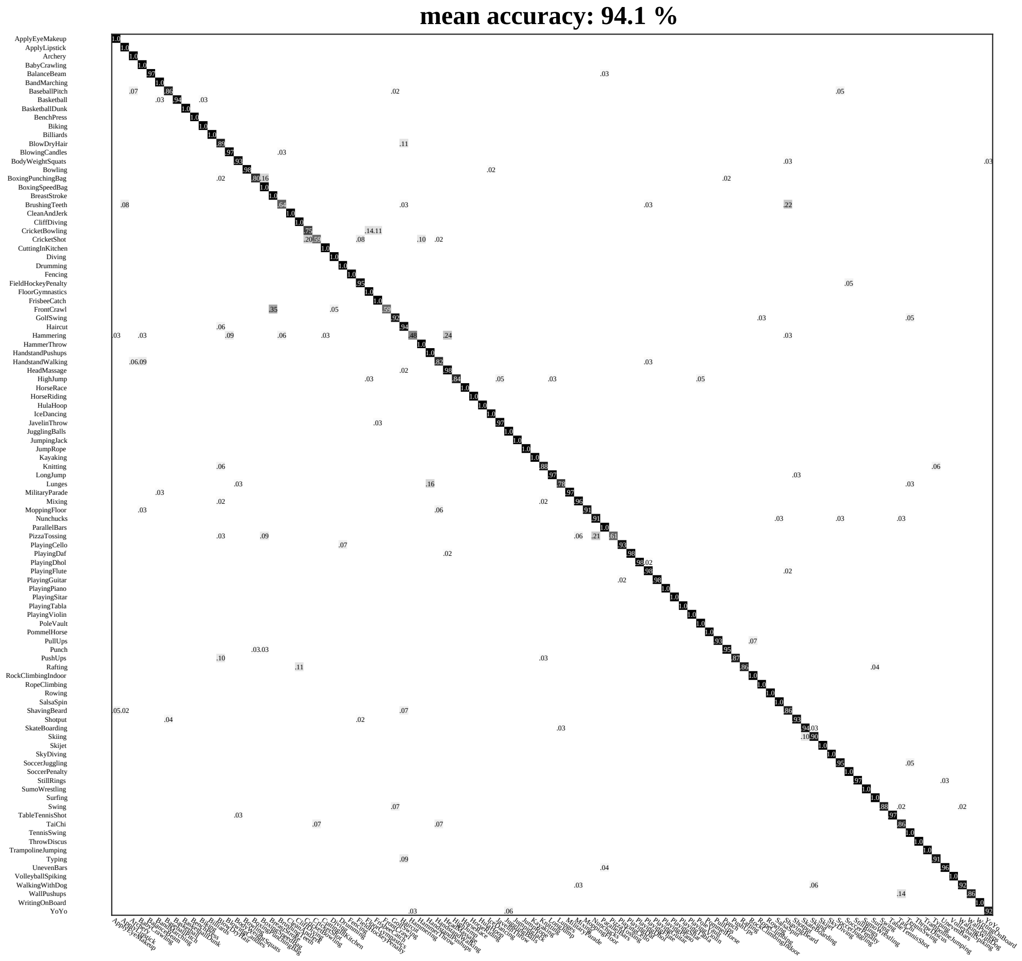
(a) ResNet-50 & ResNet-50; UCF101 (split1)

Figure A.1: Confusion matrix for our model on the first split of UCF101 when using ResNet-50 in both streams. Best viewed electronically, with zoom.



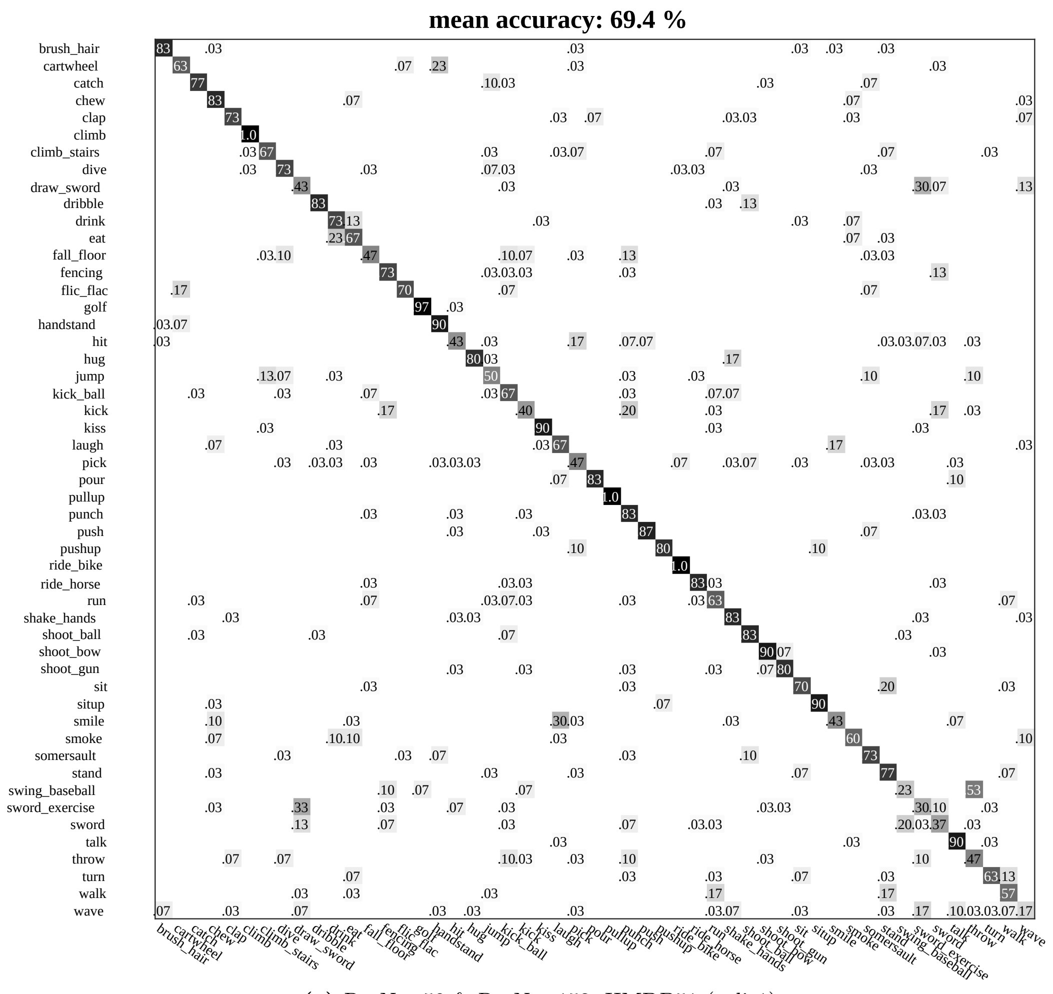
(a) ResNet-50 & ResNet-50; HMDB51 (split1)

Figure A.2: Confusion matrix for our model on the first split of HMDB51 when using ResNet-50 in both streams. Best viewed electronically, with zoom.



(a) ResNet-50 & ResNet-152; UCF101 (split1)

Figure A.3: Confusion matrix for our model on the first split of UCF101 when using ResNet-50 in the appearance and ResNet-152 in the motion streams. Best viewed electronically, with zoom.



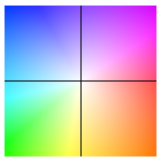
(a) ResNet-50 & ResNet-152; HMDB51 (split1)

Figure A.4: Confusion matrix for our model on the first split of HMDB51 when using ResNet-50 in the appearance and ResNet-152 in the motion streams. Best viewed electronically, with zoom.

B Visualizations for Understanding Deep Video Representations

B.1 Early layers under different optical flow encodings

In Figures B.1, B.2 and B.3 we show further examples for visualization of the early layers in the VGG-16 two-stream representation. For the optical flow components, we use three different encodings as outlined in Section 6.4.1. For the hsv encoding of optical flow, we show the vector mapping in the top-right corner of the page. We observe that when going deeper in the network hierarchy, filters get more specific capturing complex structures at the last convolutional layers. Also interesting is the correlation of spatial patterns between the appearance and the optical flow inputs.



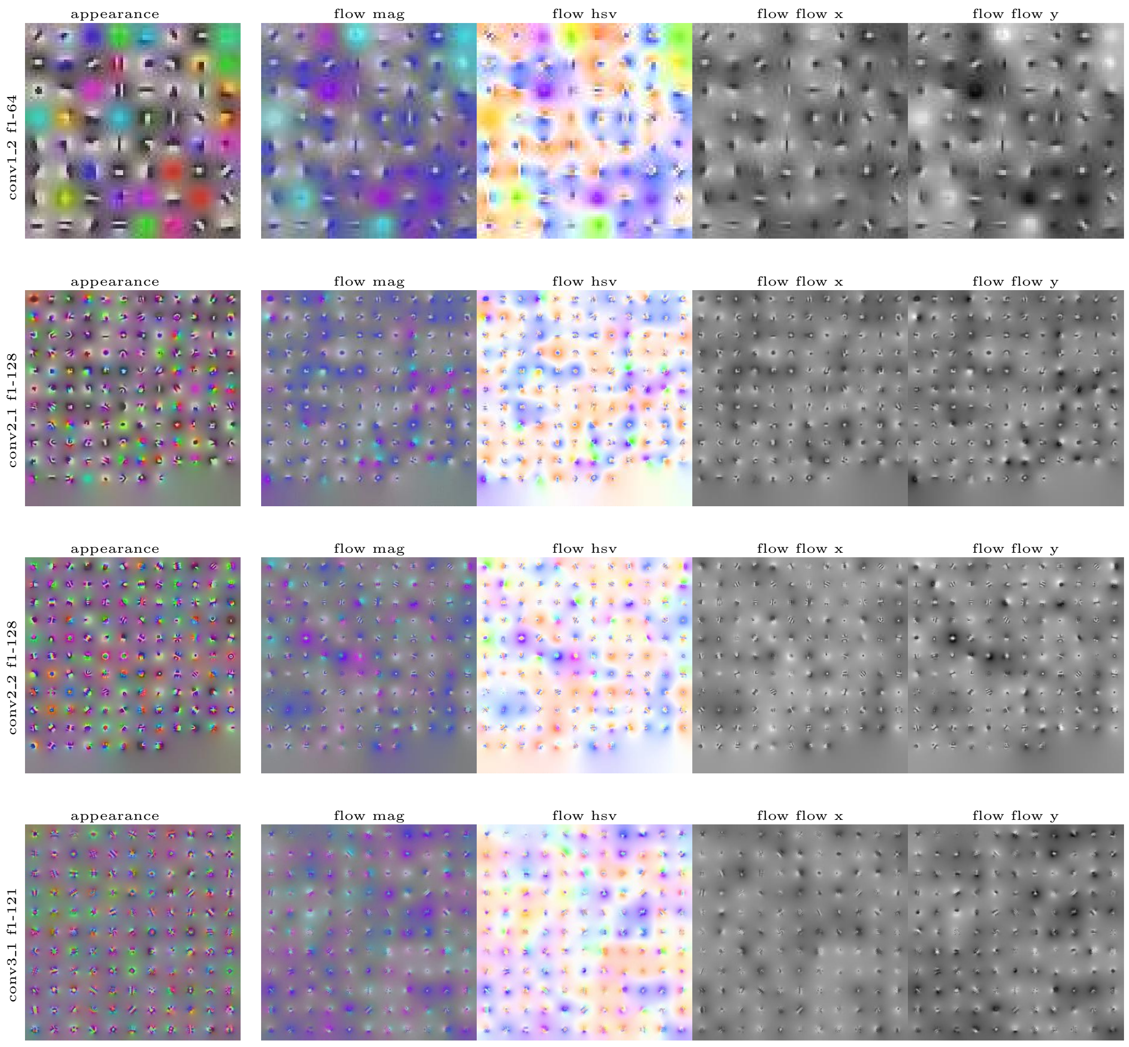
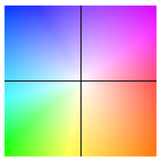


Figure B.1: Different visualizations of the motion input for VGG-16 layers conv1_1 to conv3_1.



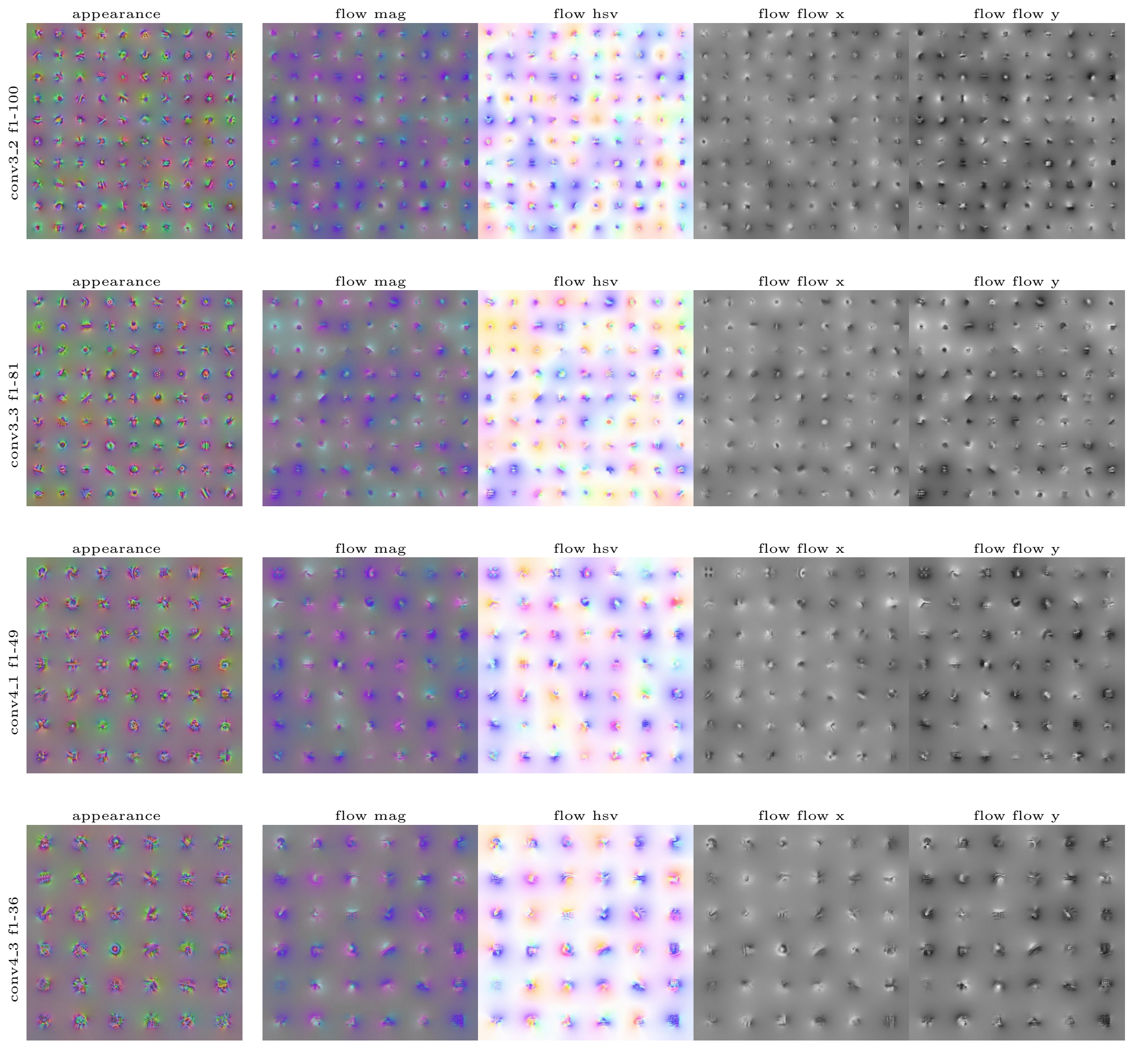
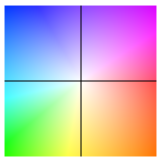


Figure B.2: Different visualizations of the motion input for VGG-16 layers conv3_2 to conv4_2.



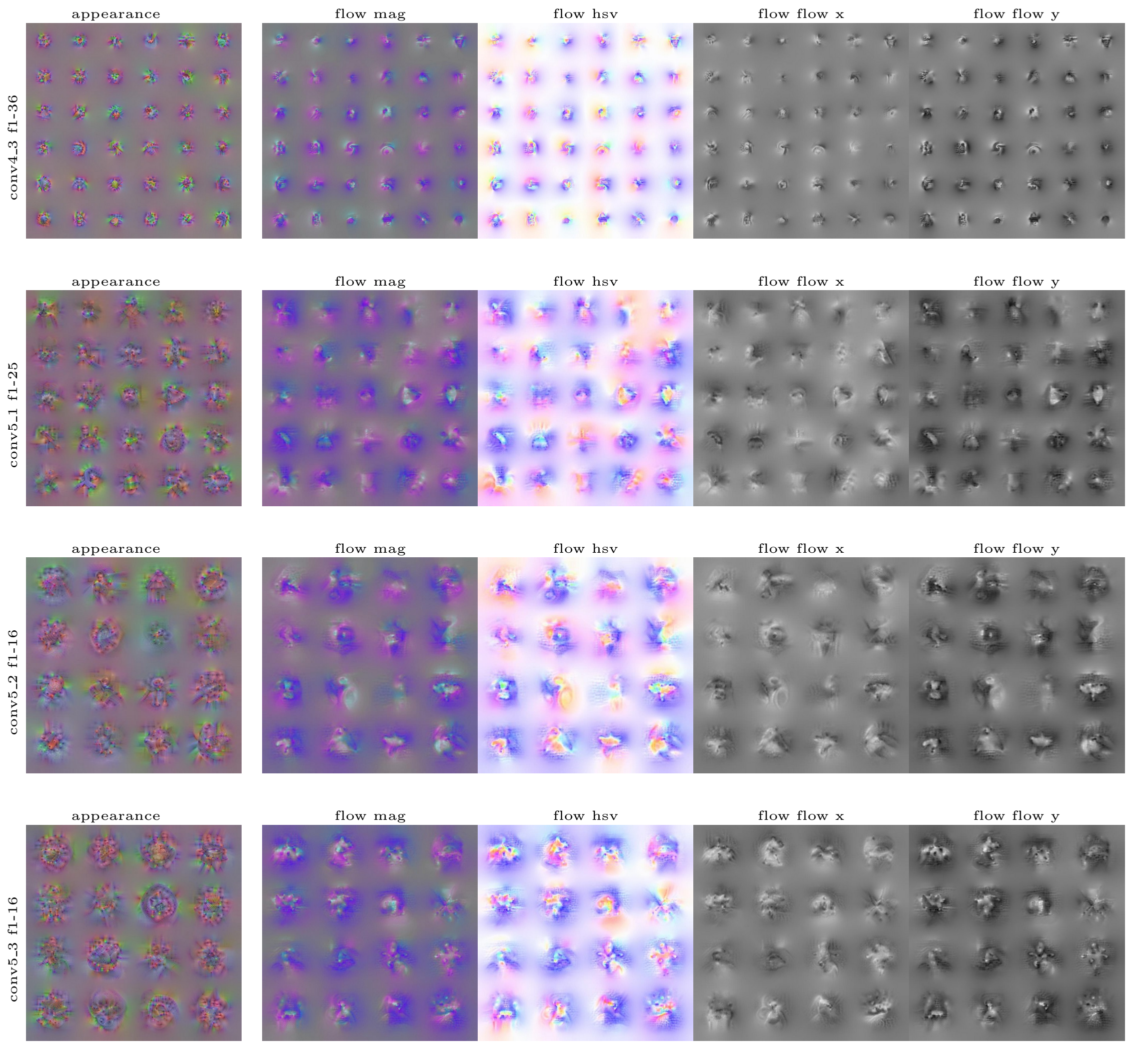


Figure B.3: Different visualizations of the motion input for VGG-16 layers conv4_3 to conv5_3.

B.2 Convolutional fusion layer visualizations

The visualizations shown in Figures B.4 - B.19 alternately show samples from the conv5_fusion layers under isotropic $\mathcal{R}_{TV3D}(\mathbf{x}, \kappa)$ and non-isotropic, (6.5) $\mathcal{R}_{TV2D1D}(\mathbf{x}, \chi)$, spacetime TV norms as defined in equations (6.4) and (6.5). It is recommended to view the two regularization forms in a two-page, side by side, view to see the full variability of these local filters (each unit is shown in 8 different visualizations). Interesting patterns appear that act as a local, distributed representation for multiple abstract features at the higher layers, shown in the next section.

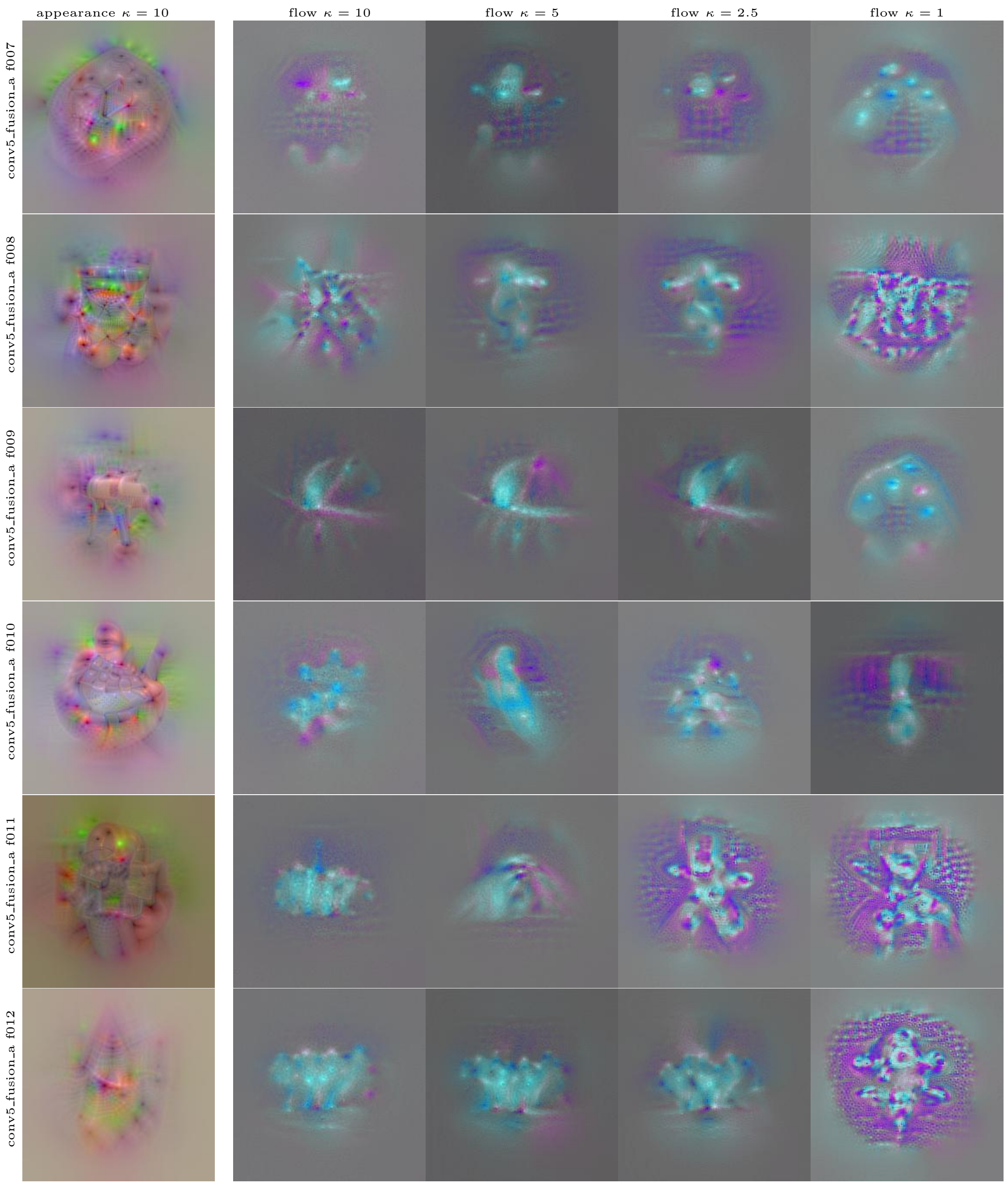


Figure B.4: Visualization of 6 filters of the conv5_fusion (appearance stream) layer under different 3D spacetime TV regularization. We show appearance and the optical flow inputs for slowest $\kappa = 10$, slow $\kappa = 5$, fast $\kappa = 2.5$, and faster $\kappa = 1$, spatiotemporal variation.

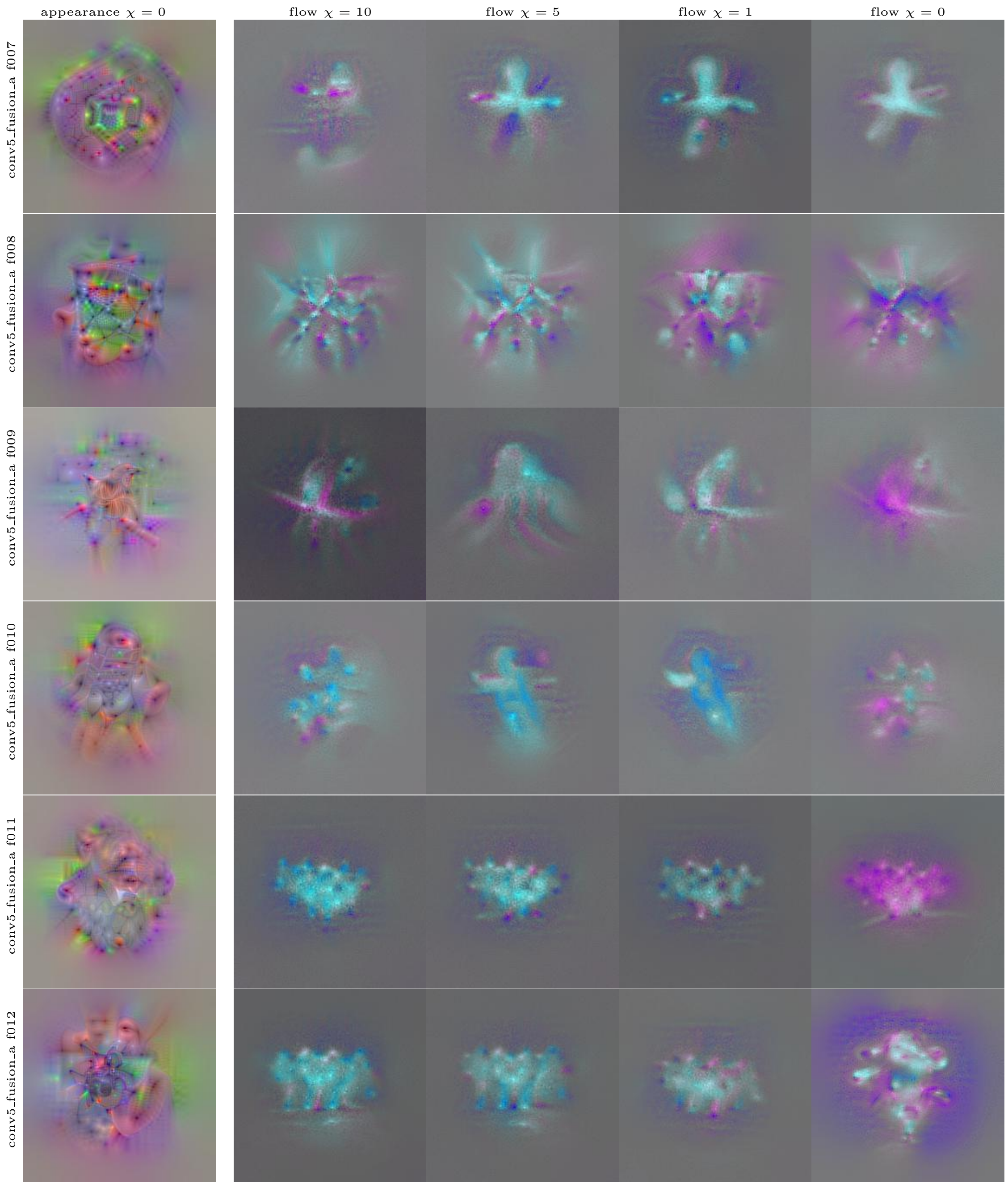


Figure B.5: Visualization of 6 filters of the conv5_fusion (appearance stream) layer under different temporal TV regularization. We show the appearance input and the optical flow inputs for slowest $\chi = 10$, slow $\chi = 5$, fast $\chi = 1$, and unconstrained $\chi = 0$, temporal variation regularization.

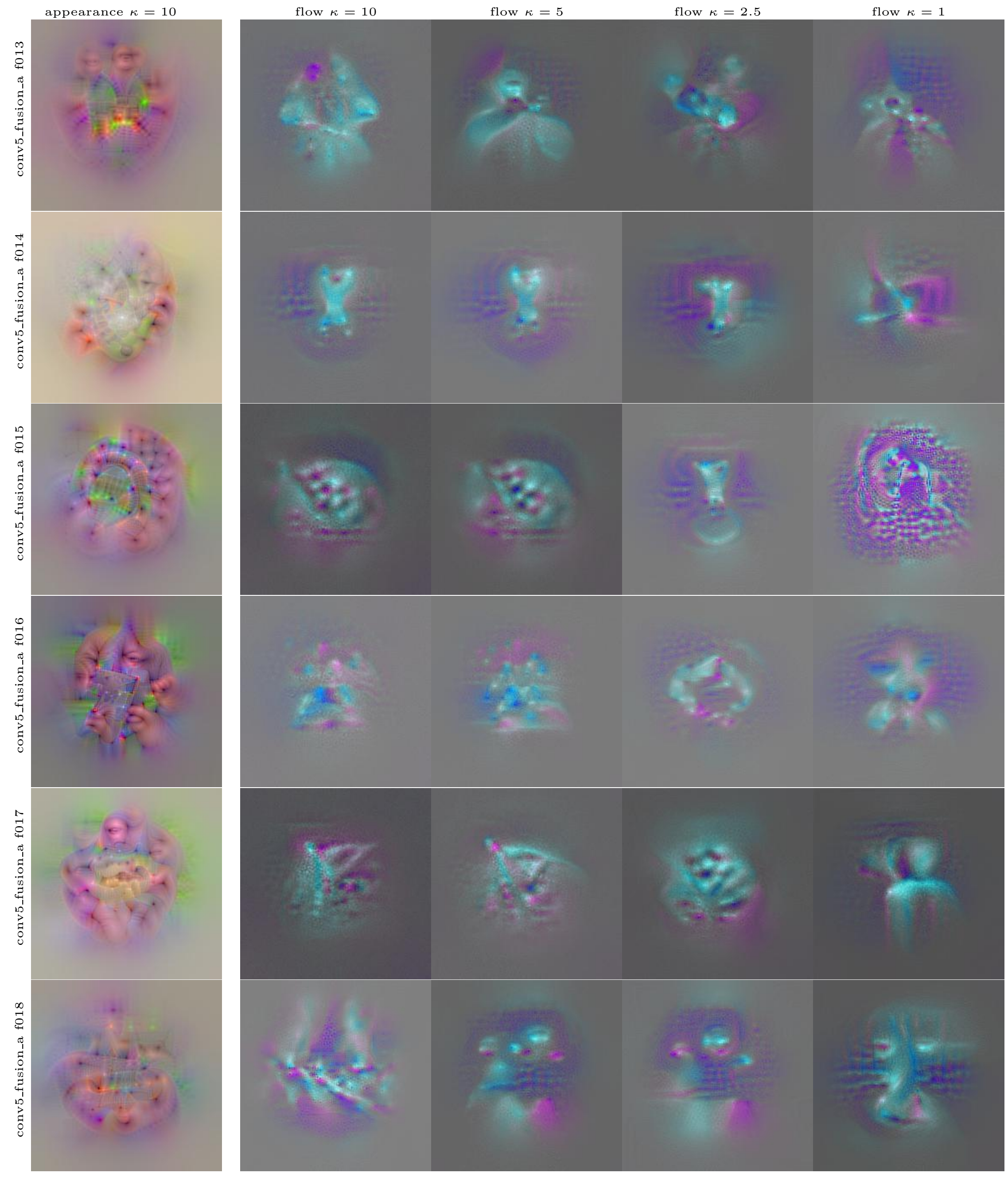


Figure B.6: Visualization of 6 filters of the conv5_fusion (appearance stream) layer under different 3D spacetime TV regularization. We show appearance and the optical flow inputs for slowest $\kappa = 10$, slow $\kappa = 5$, fast $\kappa = 2.5$, and faster $\kappa = 1$, spatiotemporal variation.

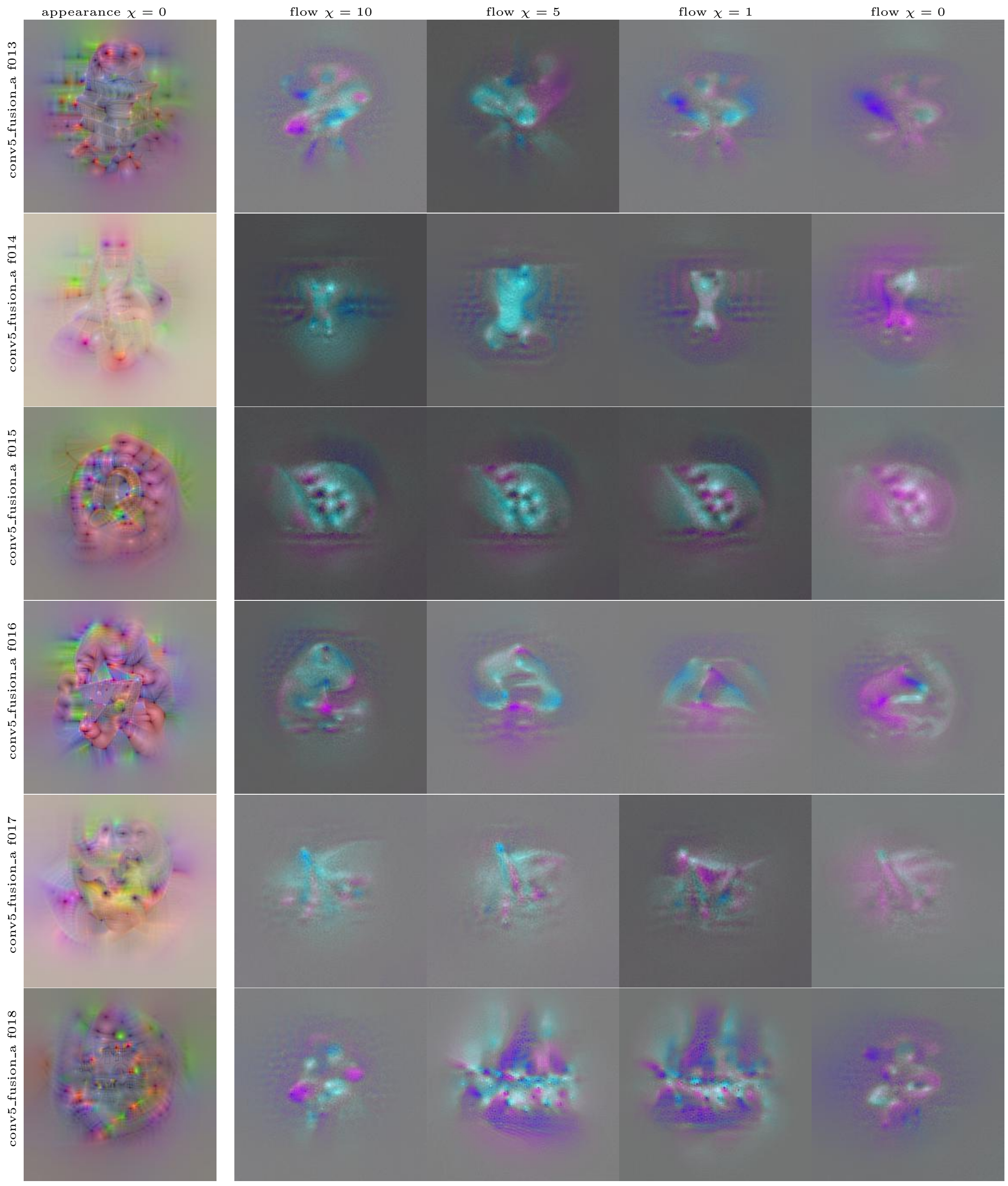


Figure B.7: Visualization of 6 filters of the conv5_fusion (appearance stream) layer under different temporal TV regularization. We show the appearance input and the optical flow inputs for slowest $\chi = 10$, slow $\chi = 5$, fast $\chi = 1$, and unconstrained $\chi = 0$, temporal variation regularization.

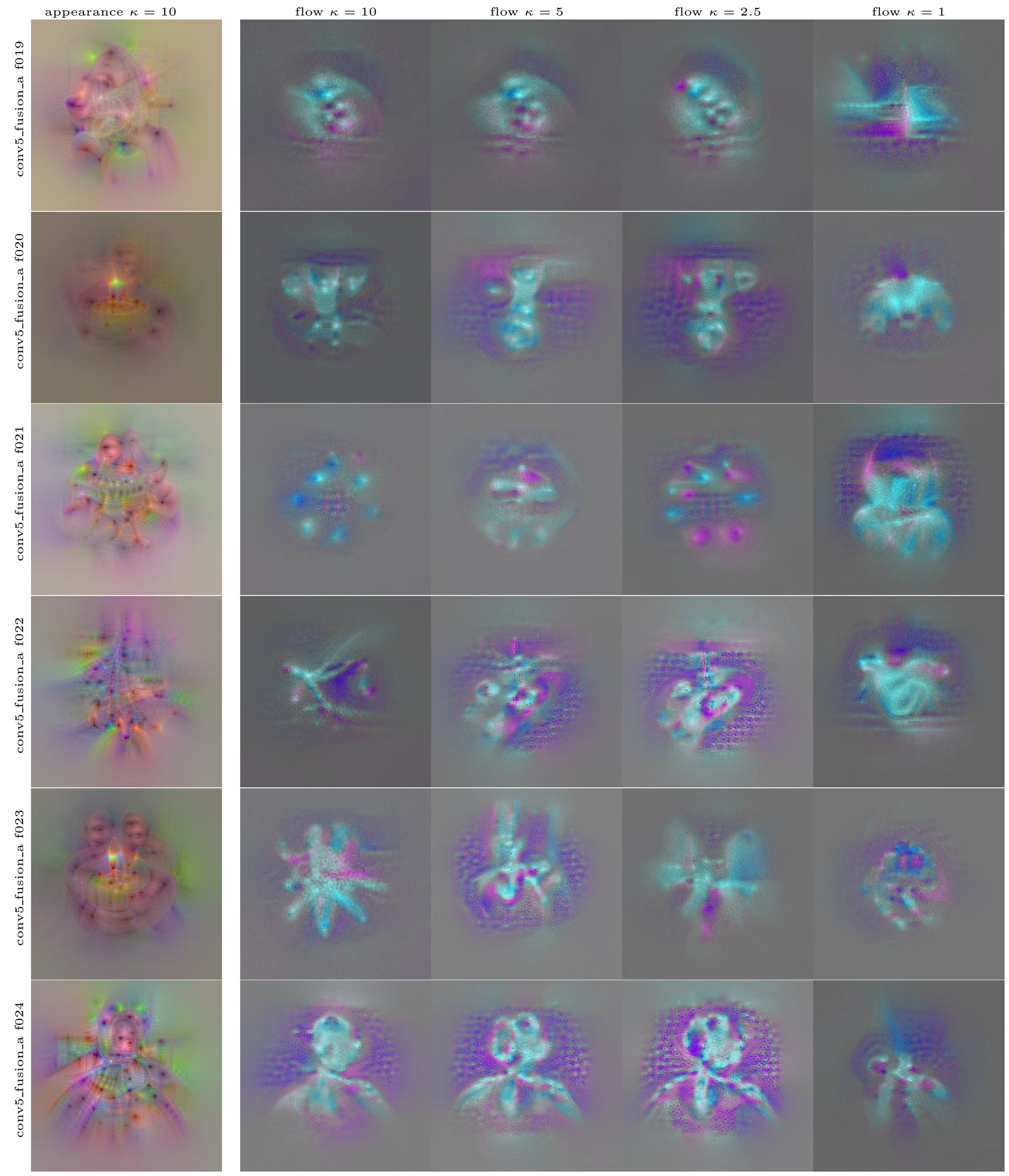


Figure B.8: Visualization of 6 filters of the conv5_fusion (appearance stream) layer under different 3D spacetime TV regularization. We show appearance and the optical flow inputs for slowest $\kappa = 10$, slow $\kappa = 5$, fast $\kappa = 2.5$, and faster $\kappa = 1$, spatiotemporal variation.

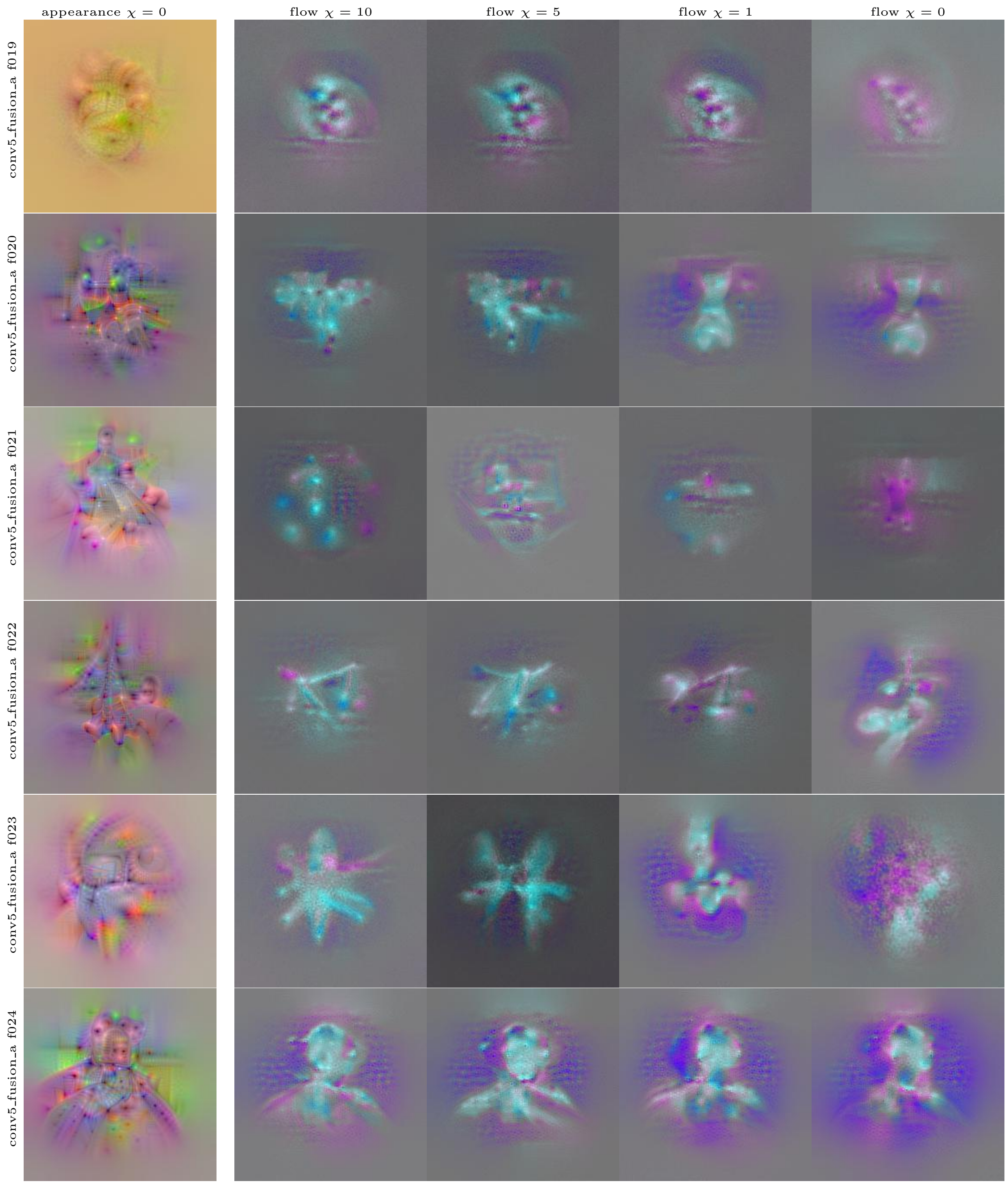


Figure B.9: Visualization of 6 filters of the conv5_fusion (appearance stream) layer under different temporal TV regularization. We show the appearance input and the optical flow inputs for slowest $\chi = 10$, slow $\chi = 5$, fast $\chi = 1$, and unconstrained $\chi = 0$, temporal variation regularization.

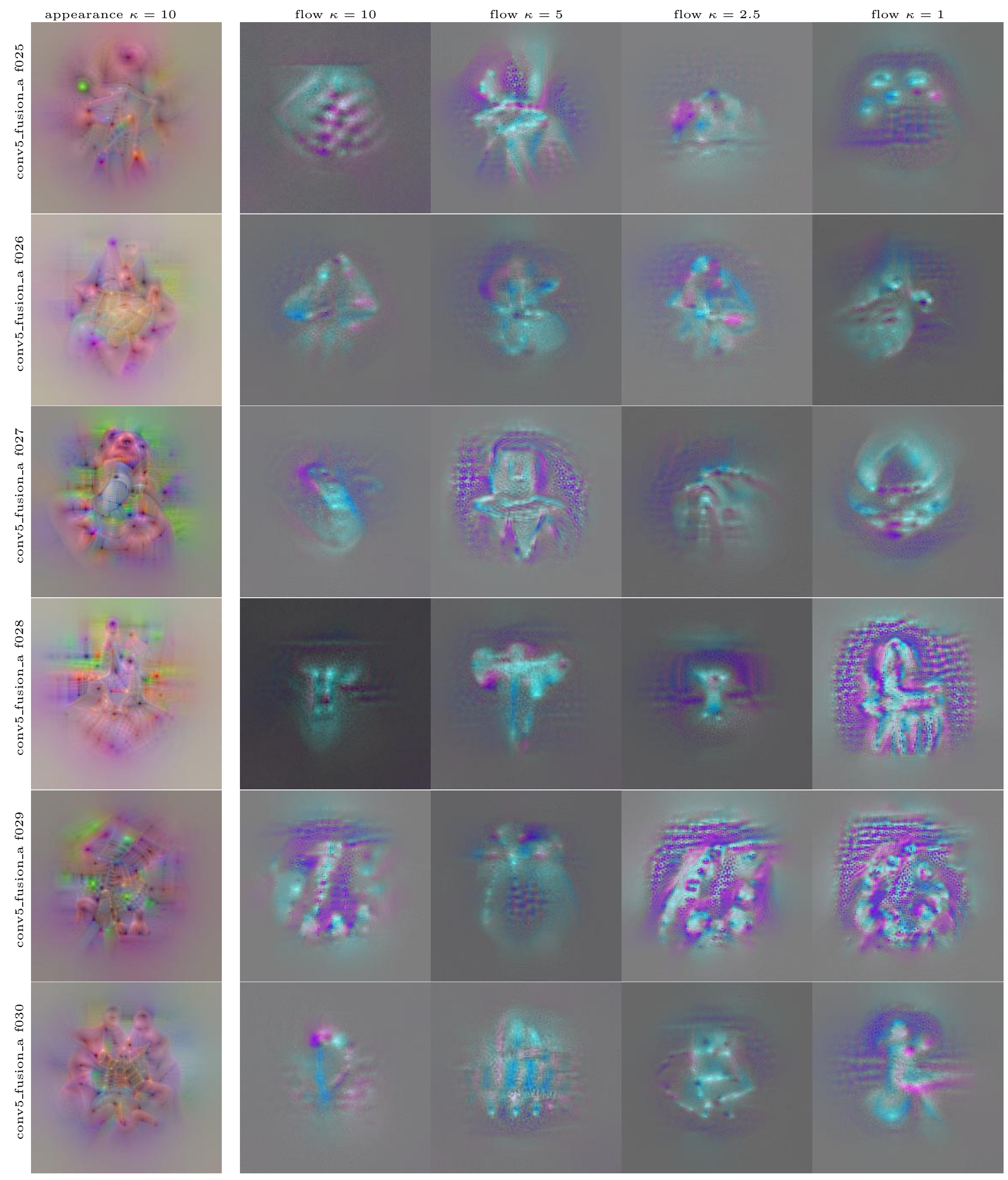


Figure B.10: Visualization of 6 filters of the conv5_fusion (appearance stream) layer under different 3D spacetime TV regularization. We show appearance and the optical flow inputs for slowest $\kappa = 10$, slow $\kappa = 5$, fast $\kappa = 2.5$, and faster $\kappa = 1$, spatiotemporal variation.

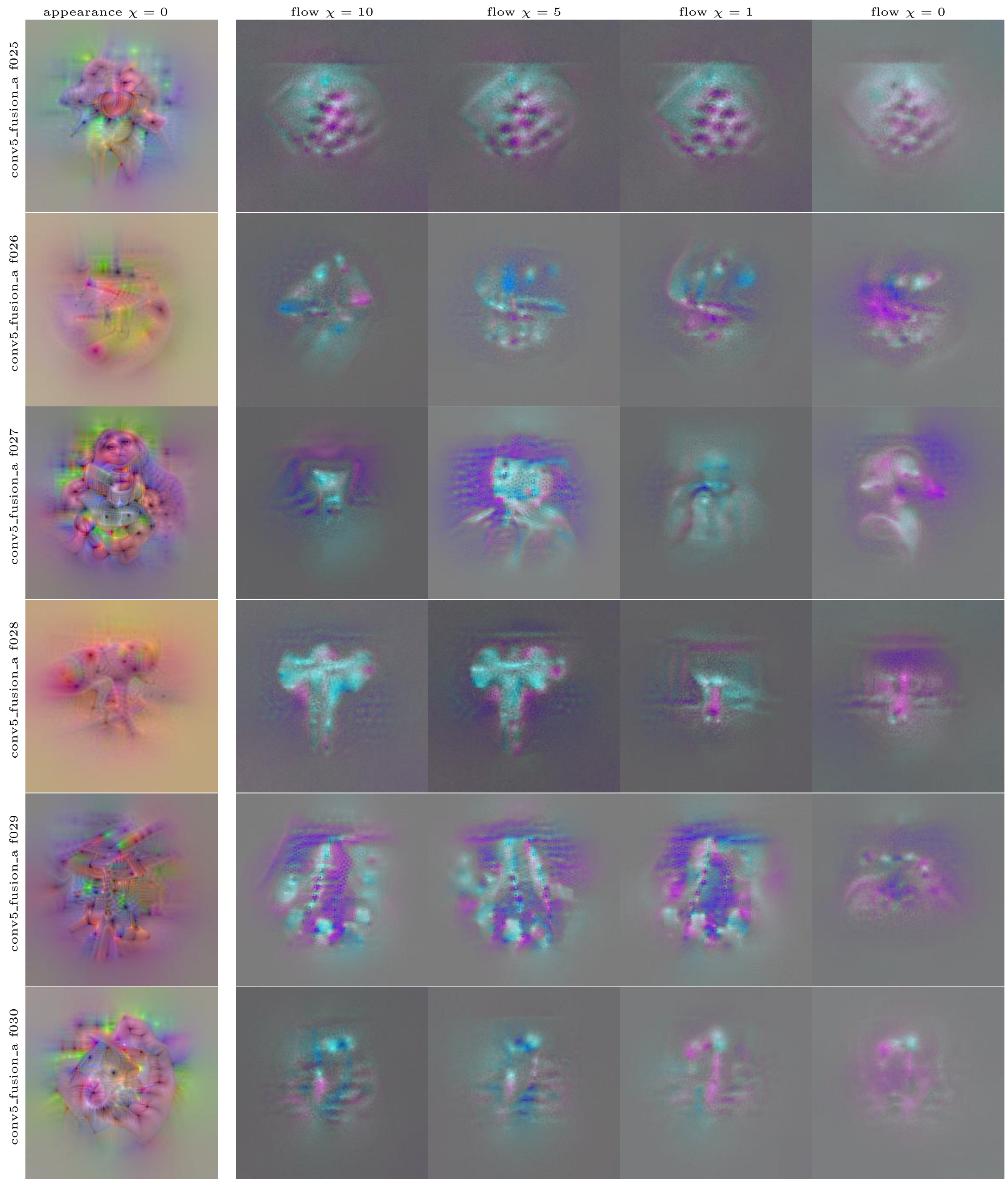


Figure B.11: Visualization of 6 filters of the conv5_fusion (appearance stream) layer under different temporal TV regularization. We show the appearance input and the optical flow inputs for slowest $\chi = 10$, slow $\chi = 5$, fast $\chi = 1$, and unconstrained $\chi = 0$, temporal variation regularization.

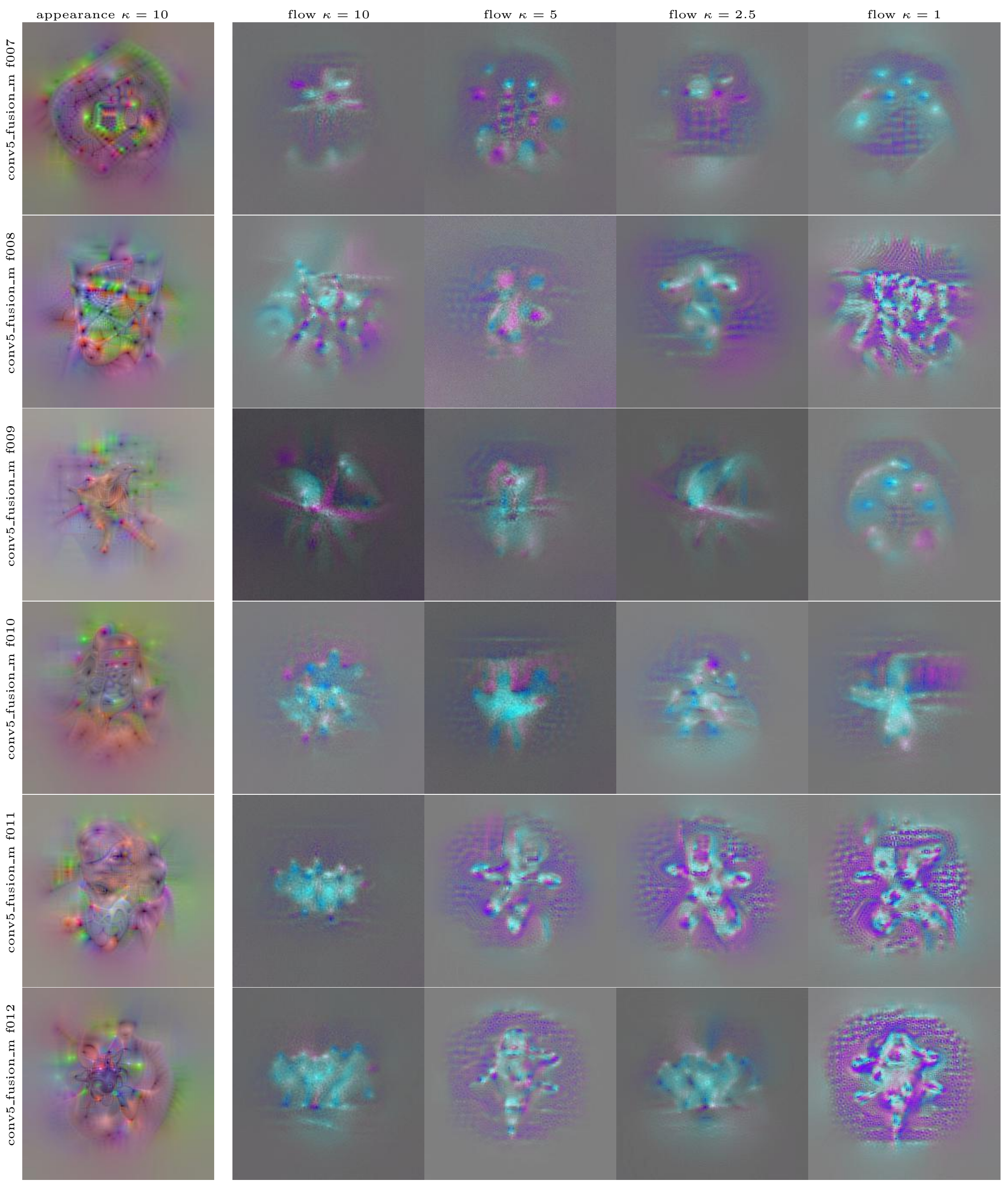


Figure B.12: Visualization of 6 filters of the conv5_fusion (motion stream) layer under different 3D spacetime TV regularization. We show appearance and the optical flow inputs for slowest $\kappa = 10$, slow $\kappa = 5$, fast $\kappa = 2.5$, and faster $\kappa = 1$, spatiotemporal variation.

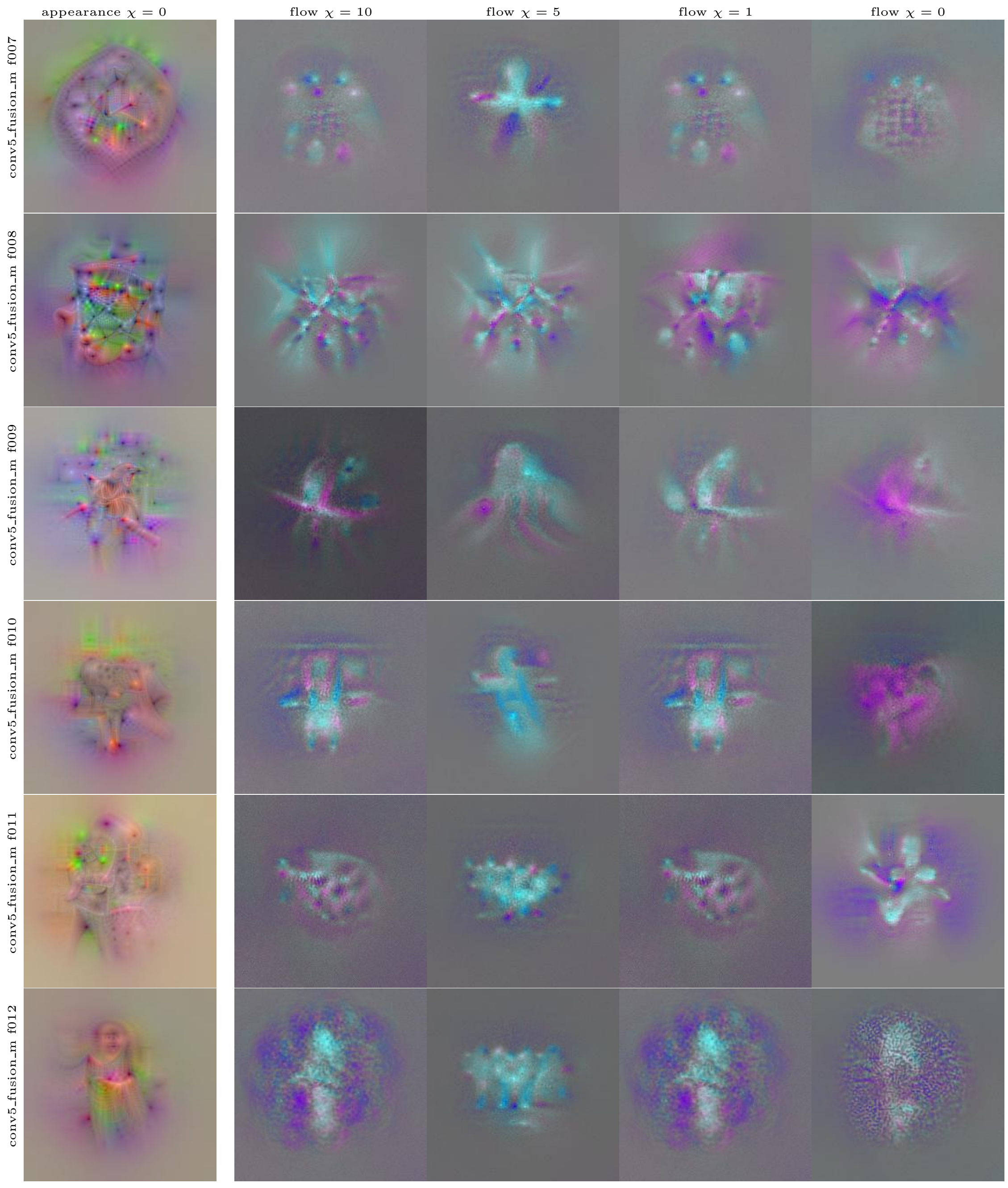


Figure B.13: Visualization of 6 filters of the conv5_fusion (motion stream) layer under different temporal TV regularization. We show the appearance input and the optical flow inputs for slowest $\chi = 10$, slow $\chi = 5$, fast $\chi = 1$, and unconstrained $\chi = 0$, temporal variation regularization.

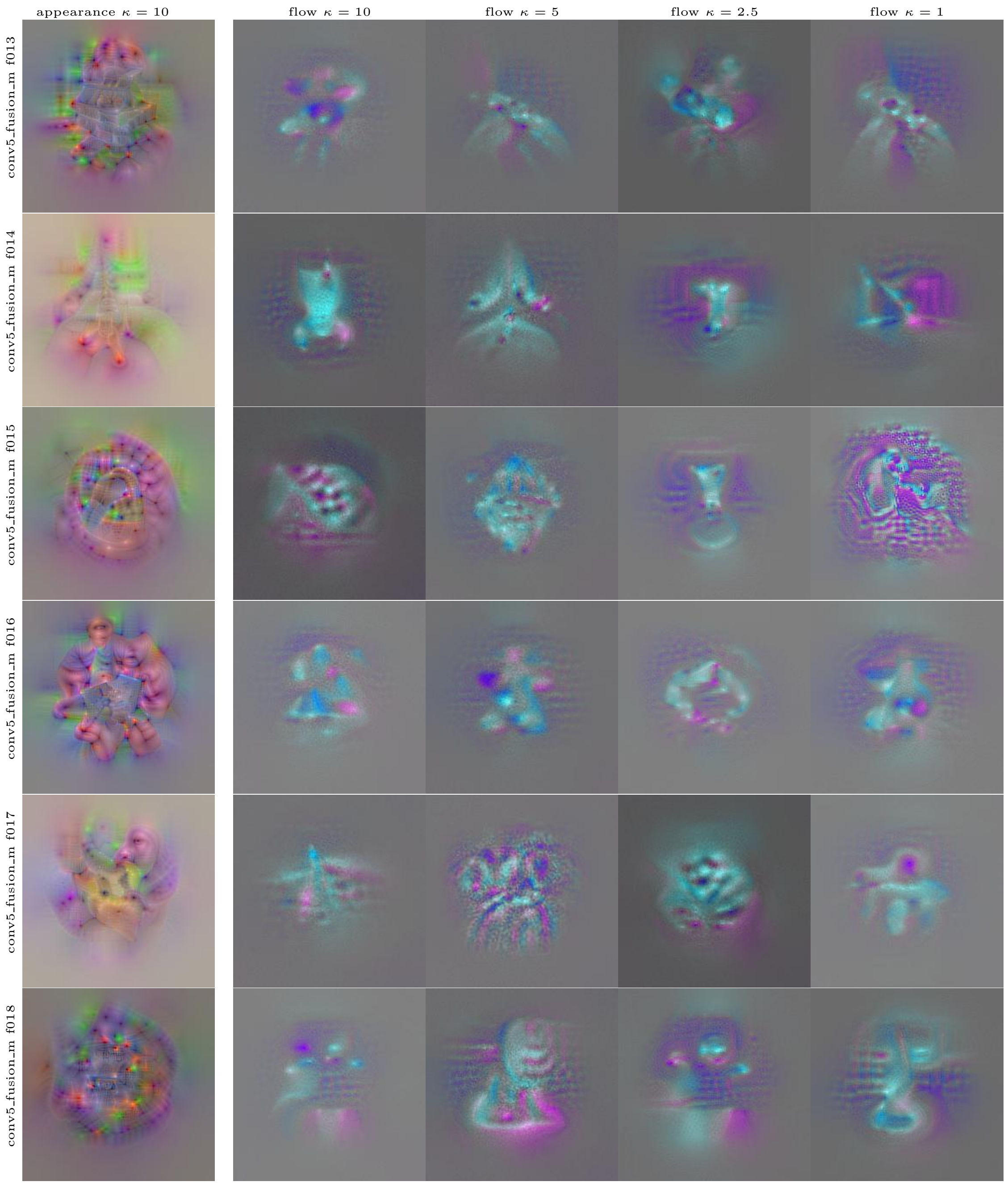


Figure B.14: Visualization of 6 filters of the conv5_fusion (motion stream) layer under different 3D spacetime TV regularization. We show appearance and the optical flow inputs for slowest $\kappa = 10$, slow $\kappa = 5$, fast $\kappa = 2.5$, and faster $\kappa = 1$, spatiotemporal variation.

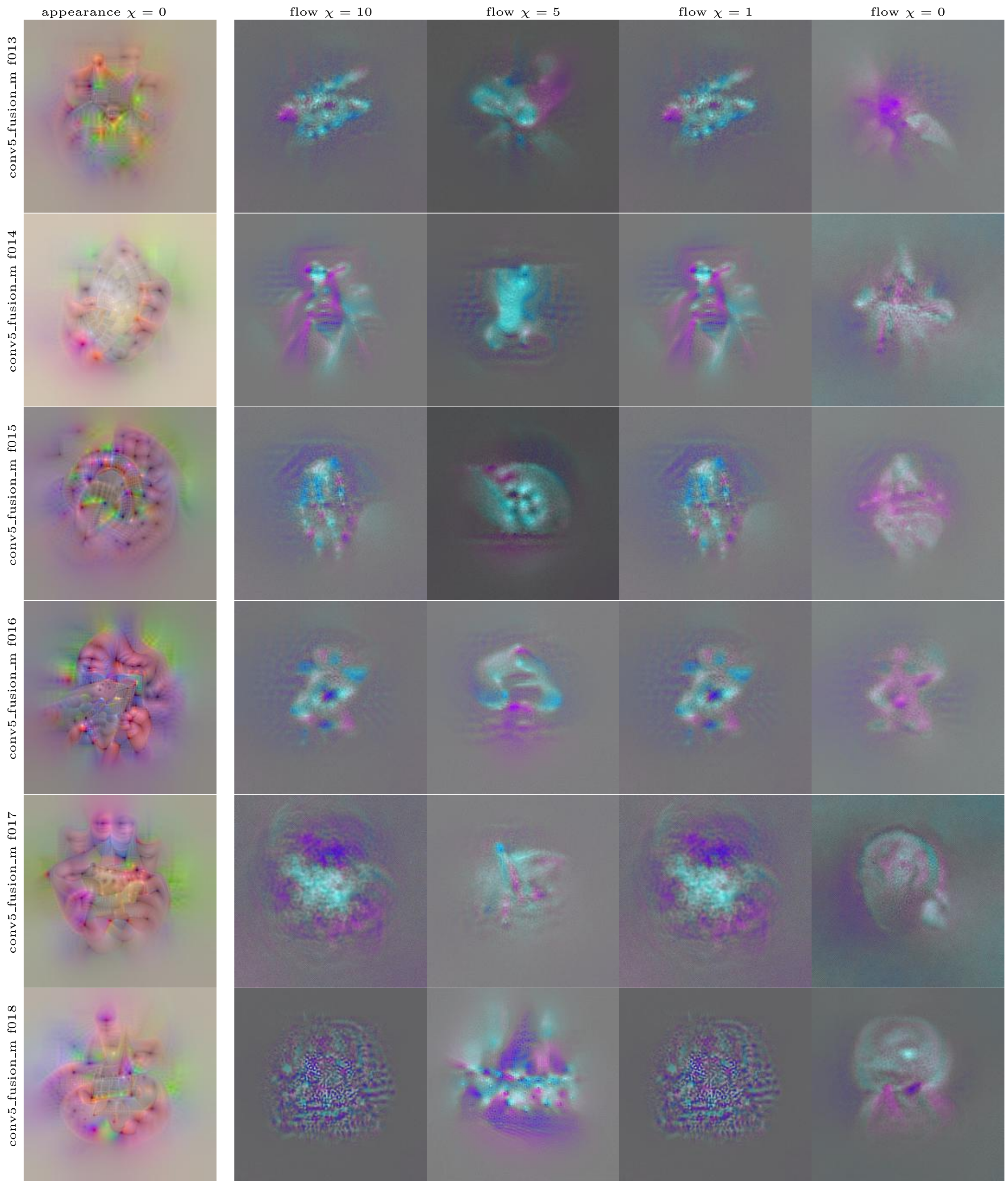


Figure B.15: Visualization of 6 filters of the conv5_fusion (motion stream) layer under different temporal TV regularization. We show the appearance input and the optical flow inputs for slowest $\chi = 10$, slow $\chi = 5$, fast $\chi = 1$, and unconstrained $\chi = 0$, temporal variation regularization.

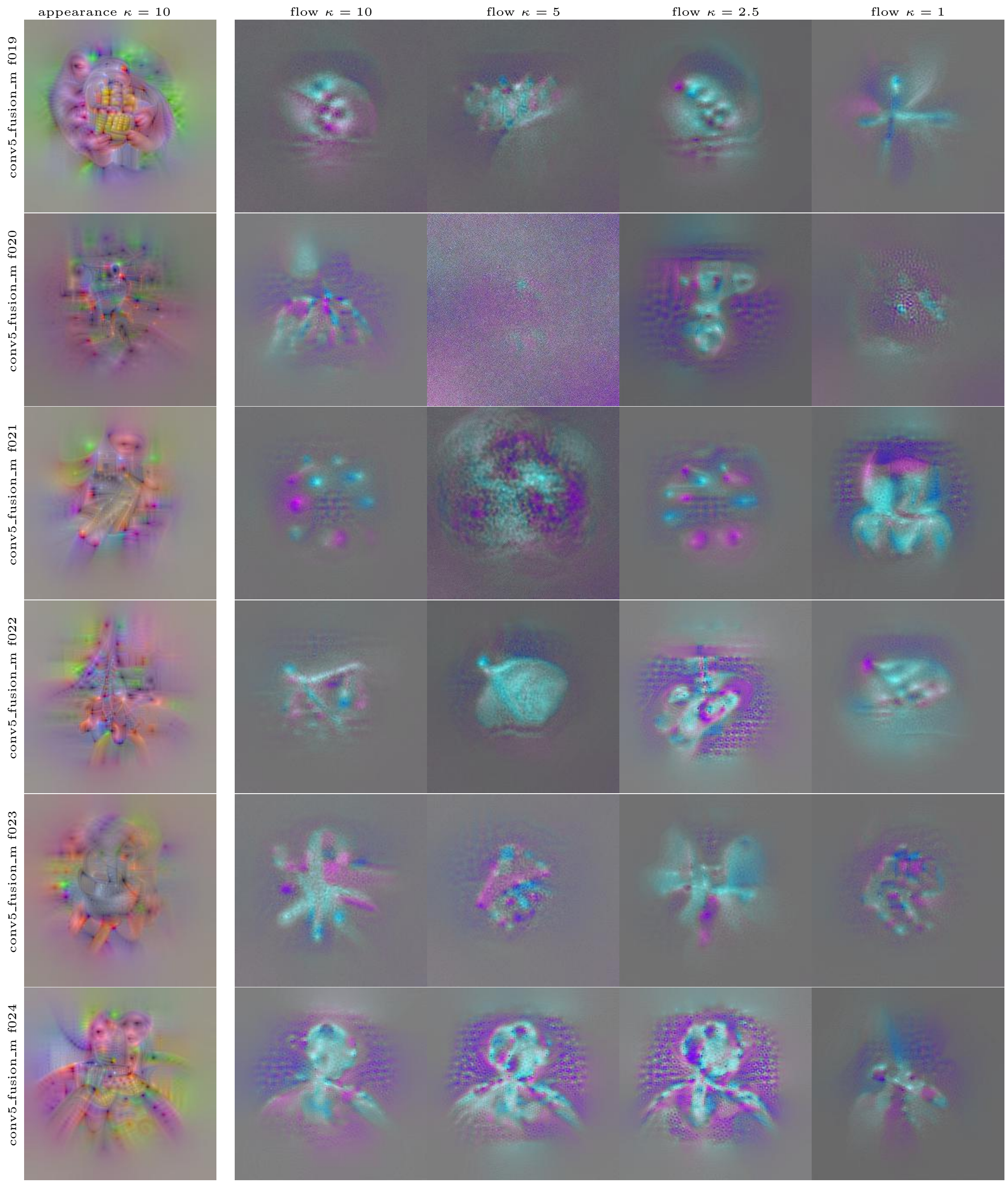


Figure B.16: Visualization of 6 filters of the conv5_fusion (motion stream) layer under different 3D spacetime TV regularization. We show appearance and the optical flow inputs for slowest $\kappa = 10$, slow $\kappa = 5$, fast $\kappa = 2.5$, and faster $\kappa = 1$, spatiotemporal variation.

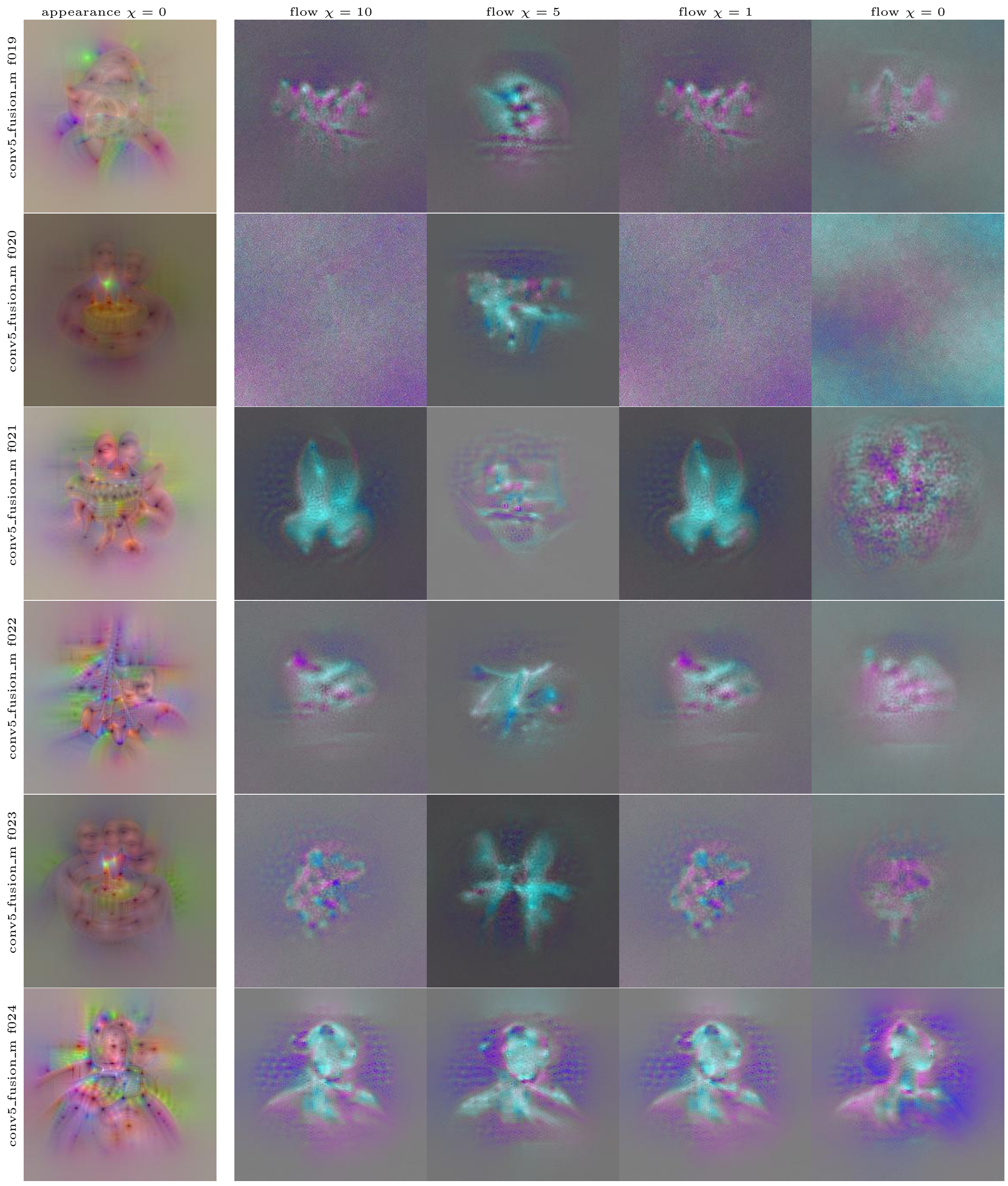


Figure B.17: Visualization of 6 filters of the conv5_fusion (motion stream) layer under different temporal TV regularization. We show the appearance input and the optical flow inputs for slowest $\chi = 10$, slow $\chi = 5$, fast $\chi = 1$, and unconstrained $\chi = 0$, temporal variation regularization.

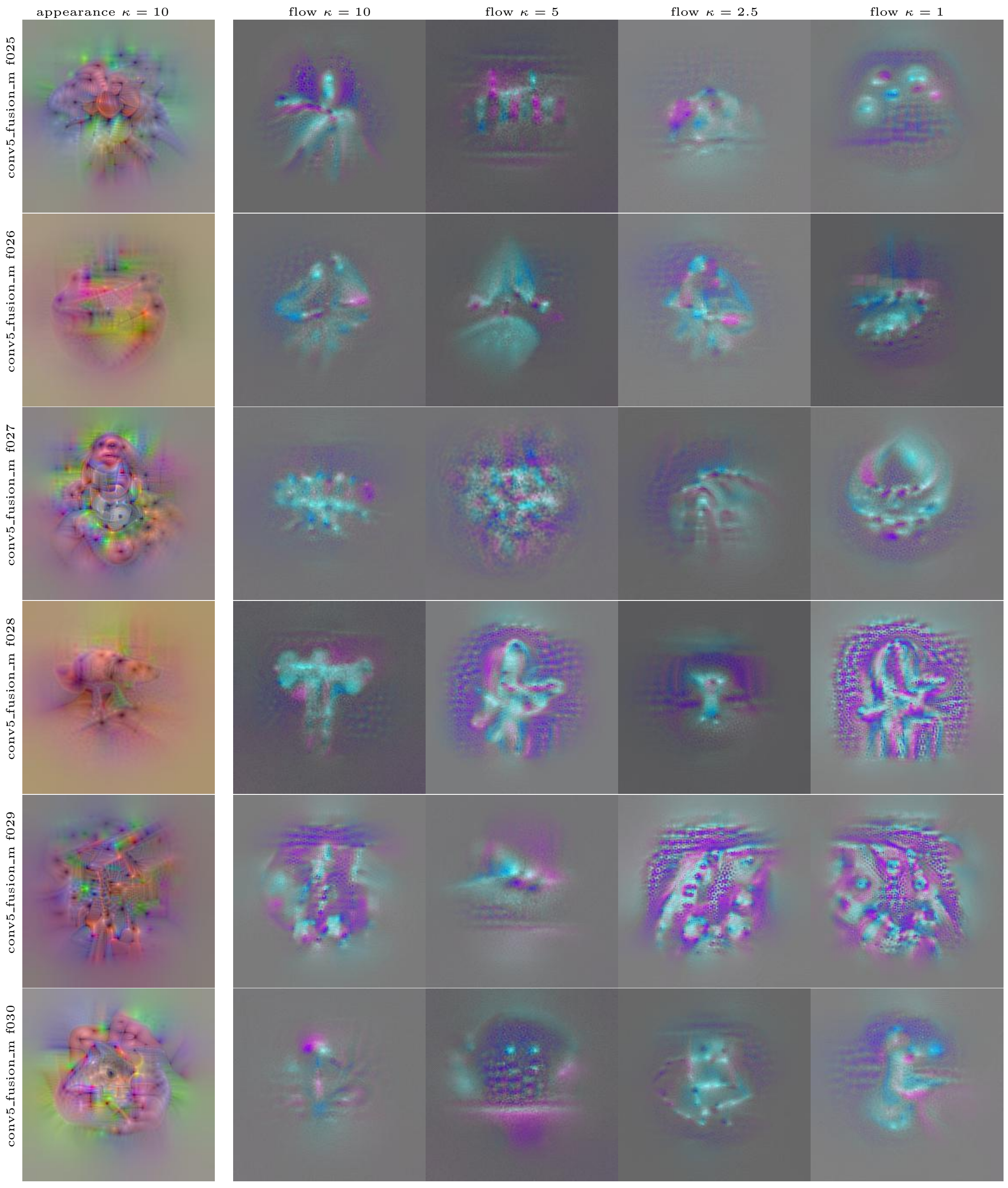


Figure B.18: Visualization of 6 filters of the conv5_fusion (motion stream) layer under different 3D spacetime TV regularization. We show appearance and the optical flow inputs for slowest $\kappa = 10$, slow $\kappa = 5$, fast $\kappa = 2.5$, and faster $\kappa = 1$, spatiotemporal variation.

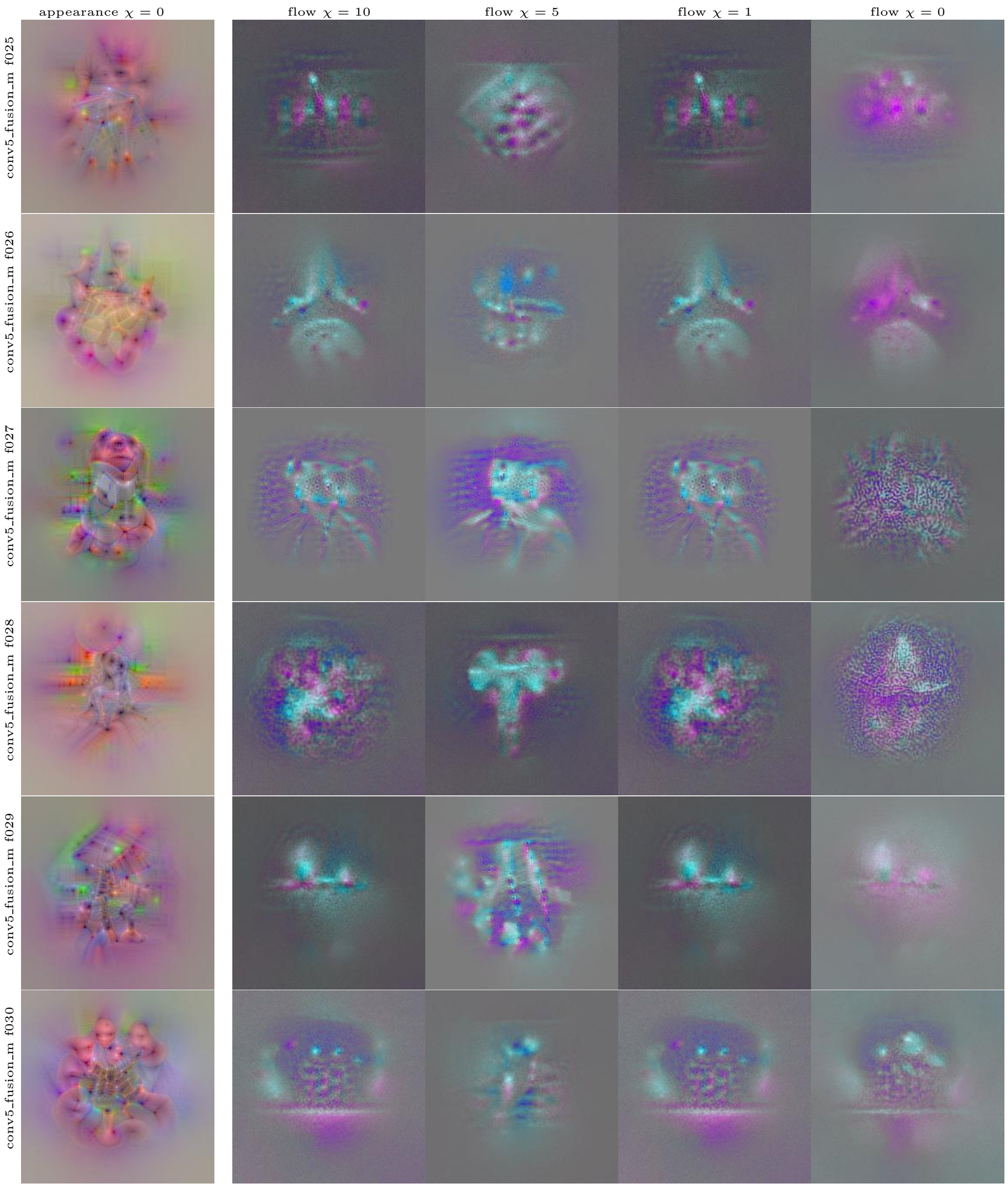


Figure B.19: Visualization of 6 filters of the conv5_fusion (motion stream) layer under different temporal TV regularization. We show the appearance input and the optical flow inputs for slowest $\chi = 10$, slow $\chi = 5$, fast $\chi = 1$, and unconstrained $\chi = 0$, temporal variation regularization.

B.3 Global layer visualizations

This section shows additional visualizations for the fully-connected layers in the motion stream of the VGG-16 fusion architecture, that have global receptive fields by operating on pooled local fusion features. In particular, Fig. B.20 & Fig. B.21 show ten filters of the fully-connected 6 (fc_6) layer while Fig. B.22 & Fig. B.23 show ten filters of the subsequent fully-connected_7 (fc_7) layer.

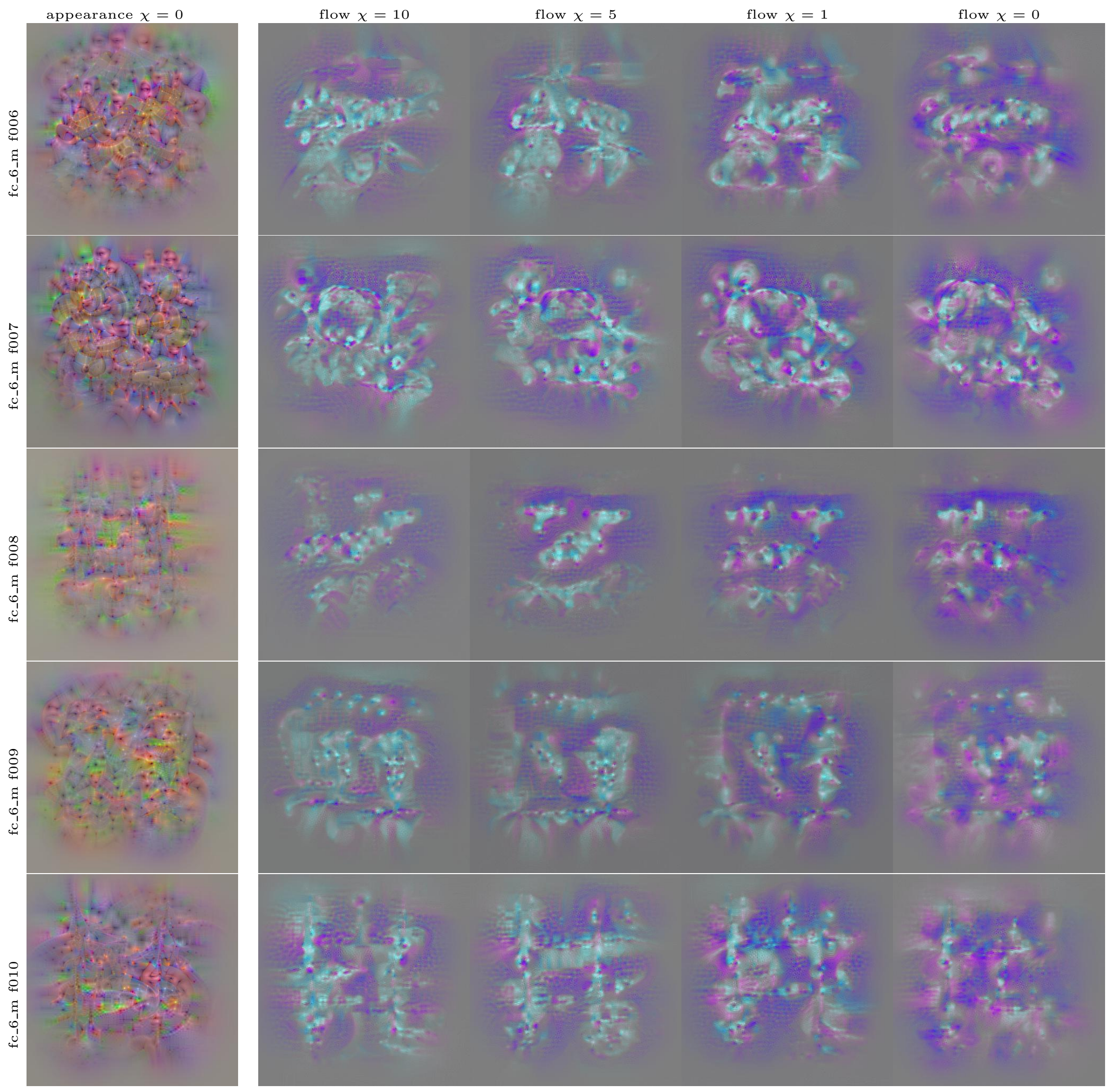


Figure B.20: Visualization of 5 filters of the fc_6 (motion stream) layer under different temporal TV regularization. We show the appearance input and the optical flow inputs for slowest $\chi = 10$, slow $\chi = 5$, fast $\chi = 1$, and unconstrained (fastest) $\chi = 0$, temporal variation regularization.

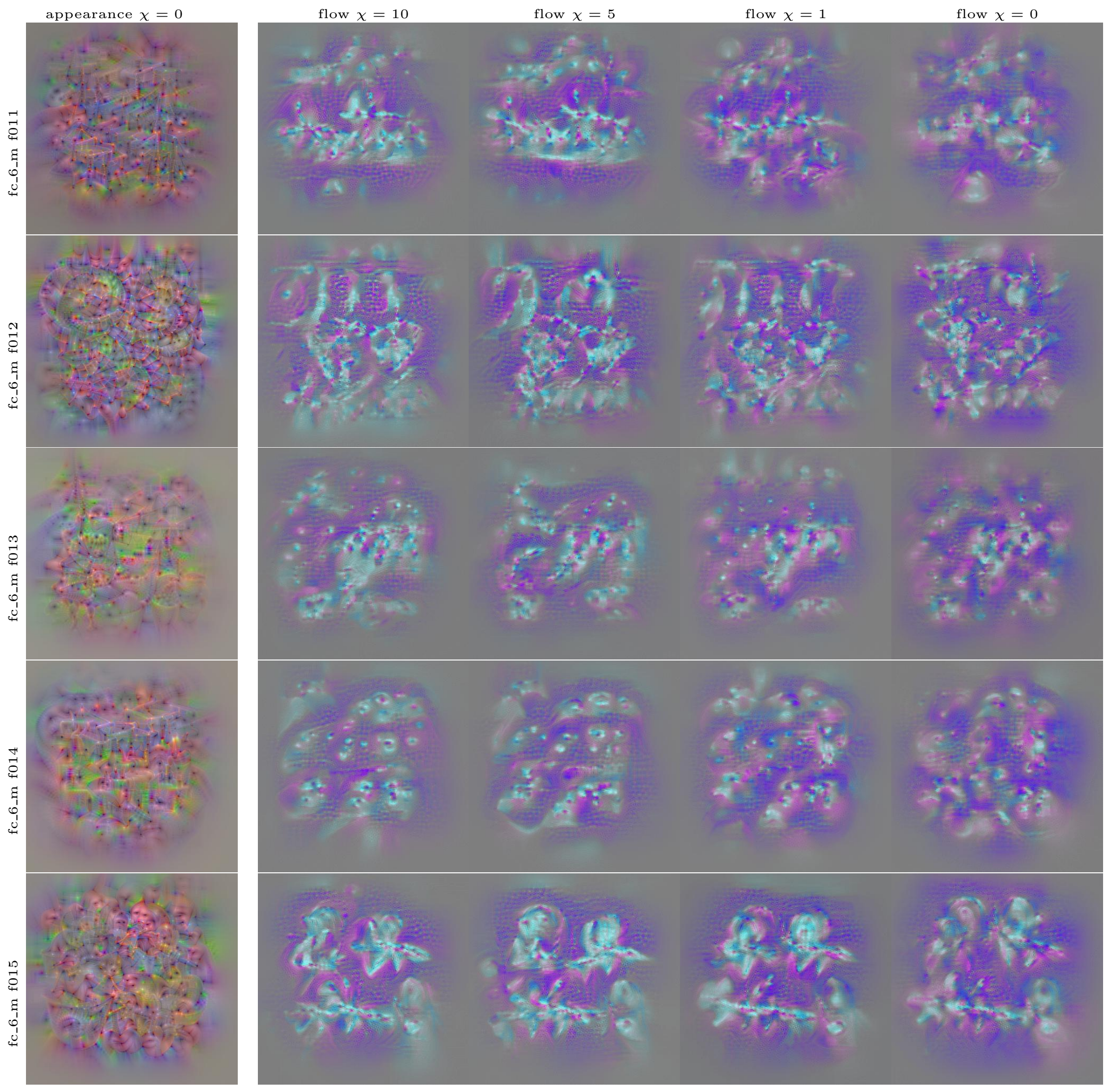


Figure B.21: Visualization of 5 filters of the fc_6 (motion stream) layer under different temporal TV regularization. We show the appearance input and the optical flow inputs for slowest $\chi = 10$, slow $\chi = 5$, fast $\chi = 1$, and unconstrained (fastest) $\chi = 0$, temporal variation regularization.

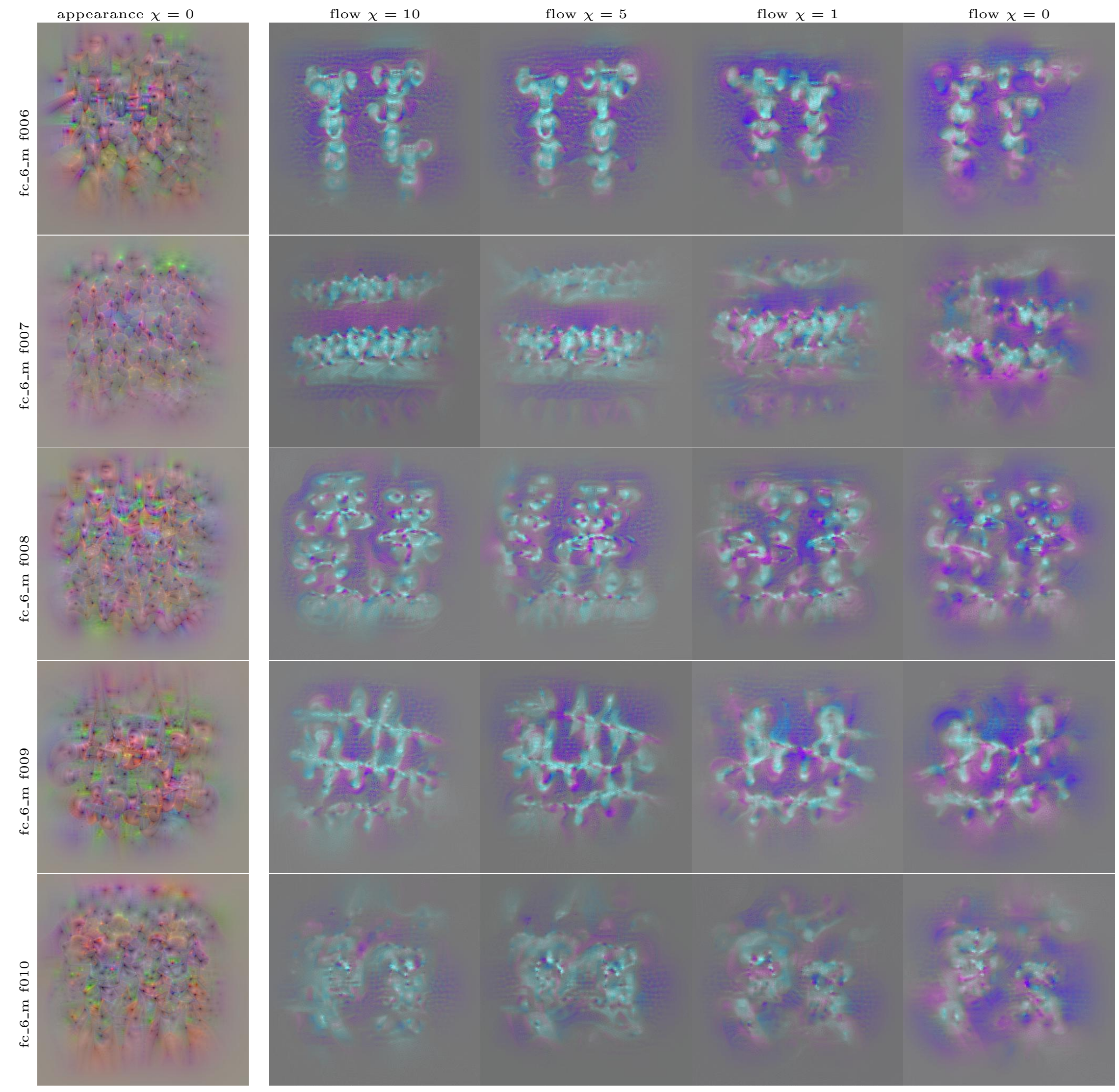


Figure B.22: Visualization of 5 filters of the fc_7 (motion stream) layer under different temporal TV regularization. We show the appearance input and the optical flow inputs for slowest $\chi = 10$, slow $\chi = 5$, fast $\chi = 1$, and unconstrained (fastest) $\chi = 0$, temporal variation regularization.

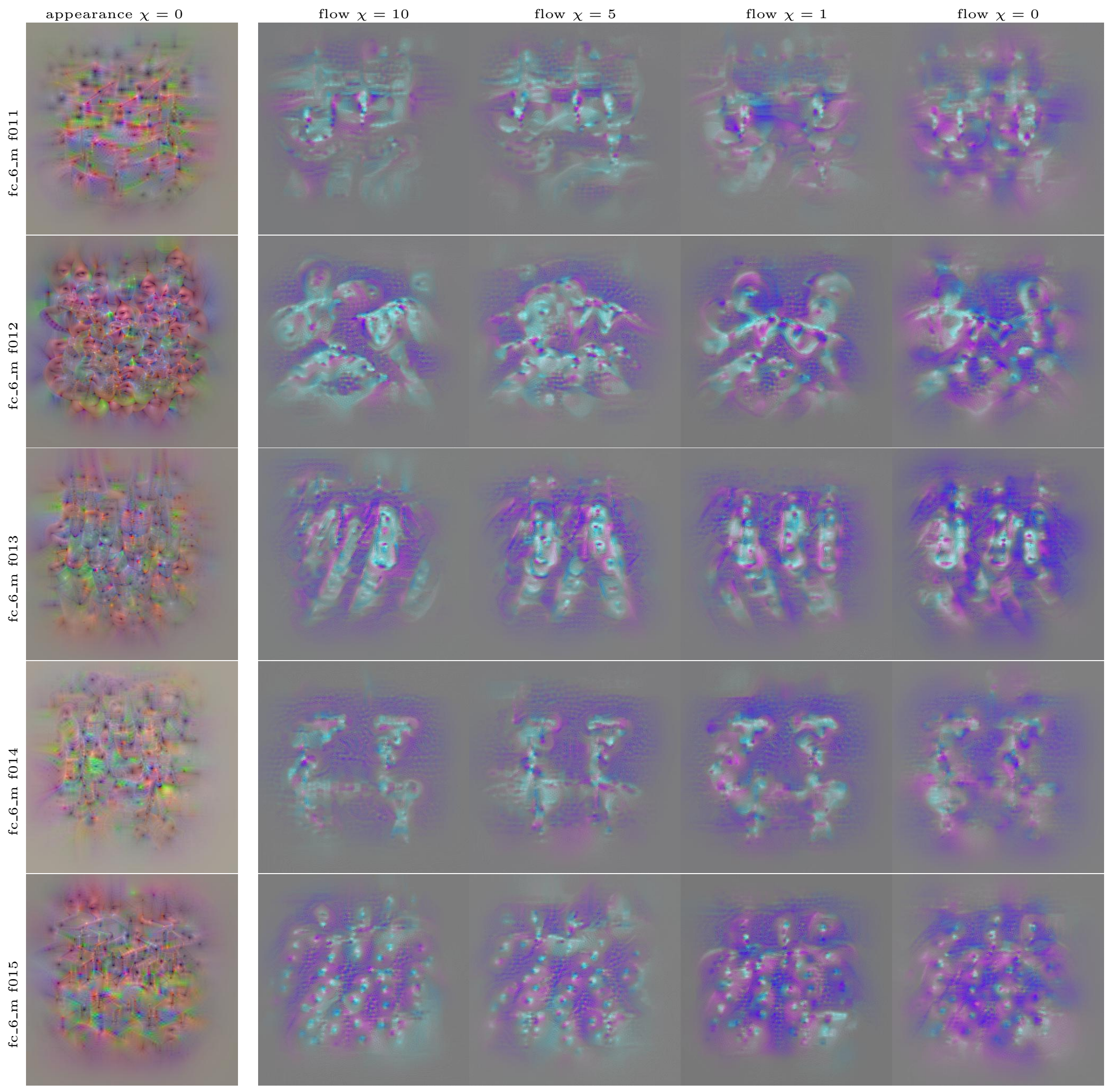


Figure B.23: Visualization of 5 filters of the fc_7 (motion stream) layer under different temporal TV regularization. We show the appearance input and the optical flow inputs for slowest $\chi = 10$, slow $\chi = 5$, fast $\chi = 1$, and unconstrained (fastest) $\chi = 0$, temporal variation regularization.

B.4 Prediction layer visualizations

In Figures B.24-B.40, we show additional visualizations for the learned VGG-16 fusion model on the classes present in UCF101. Again, we visualize appearance and optical flow inputs for different temporal regularization strengths (χ). Notably, in some cases, the fast temporal variation case ($\chi = 0$) reveals large scale motion that is not present in the slower cases. For example, for the Billiards class the fastest visualization indicates player movement around the pool table, etc.

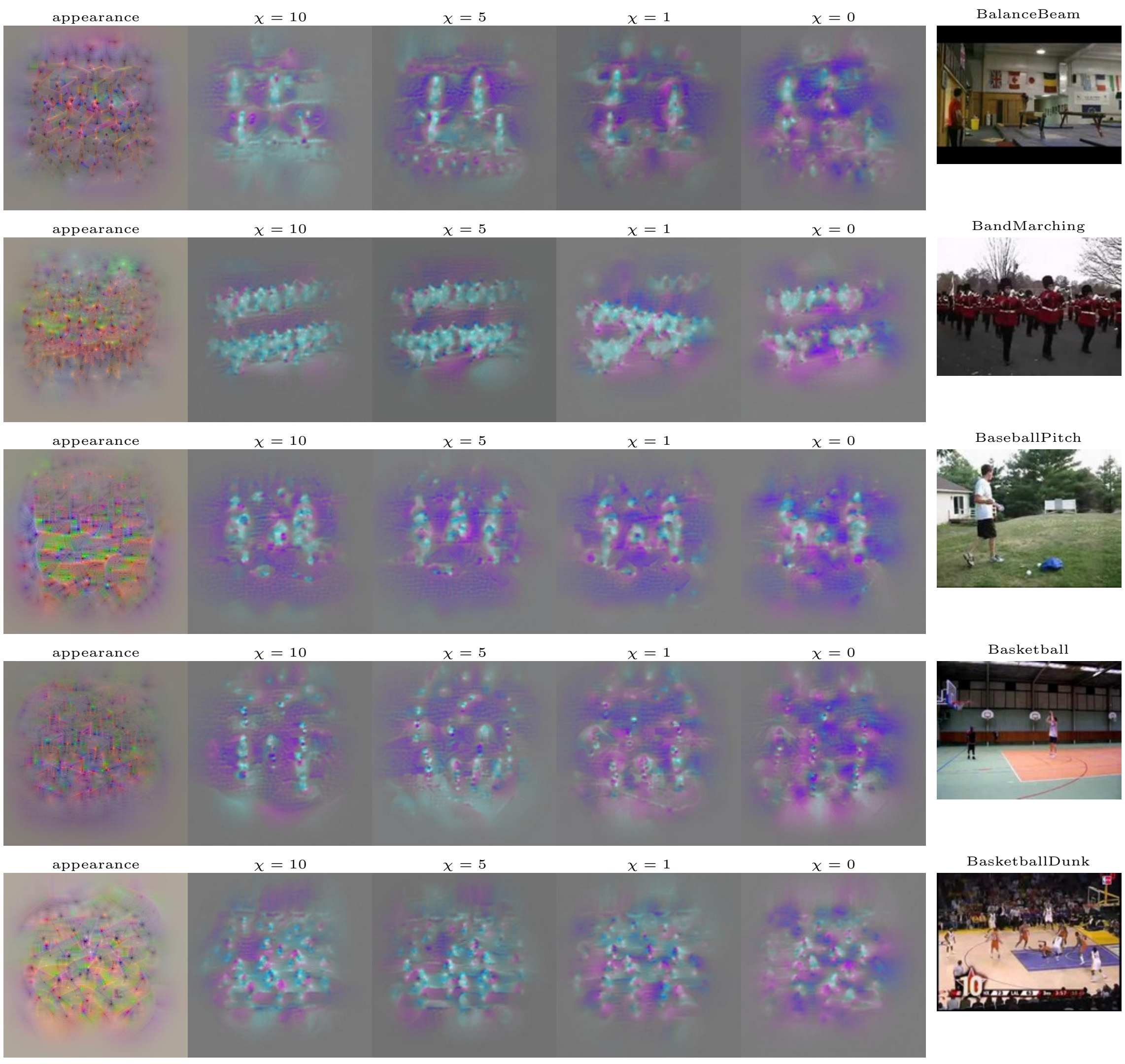


Figure B.24: Visualizations of classification units at the last layer of the network. The first column shows the appearance and the second to fifth column the motion input generated by maximizing the prediction layer output for the respective classes, while using different degrees of temporal variation regularization (χ). The last column shows 40 sample frames from the first video of the corresponding class in the test set. Best viewed electronically.

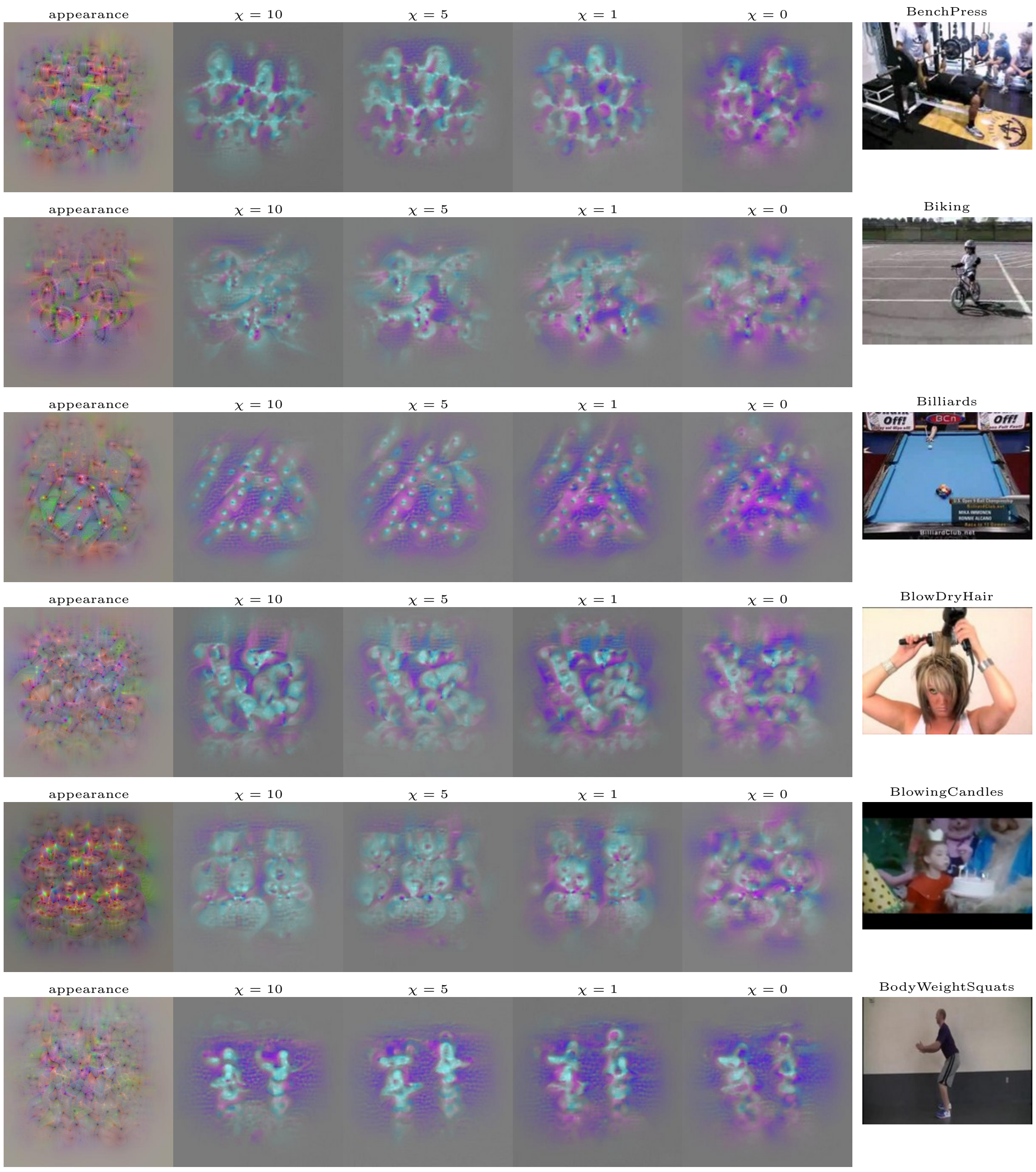


Figure B.25: Visualizations of classification units at the last layer of the network. The first column shows the appearance and the second to fifth column the motion input generated by maximizing the prediction layer output for the respective classes, while using different degrees of temporal variation regularization (χ). The last column shows 40 sample frames from the first video of the corresponding class in the test set. Best viewed electronically.

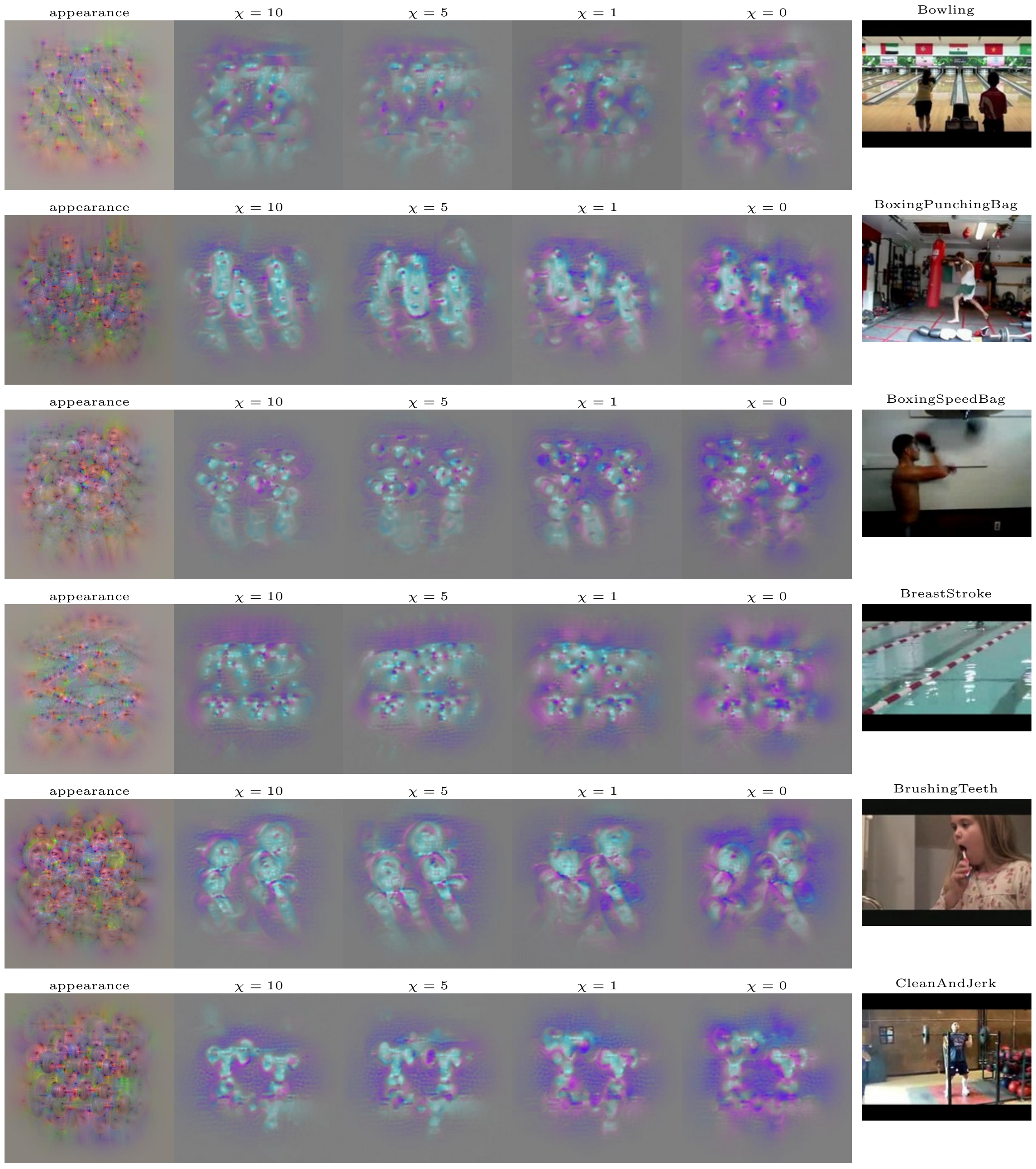


Figure B.26: Visualizations of classification units at the last layer of the network. The first column shows the appearance and the second to fifth column the motion input generated by maximizing the prediction layer output for the respective classes, while using different degrees of temporal variation regularization (χ). The last column shows 40 sample frames from the first video of the corresponding class in the test set. Best viewed electronically.

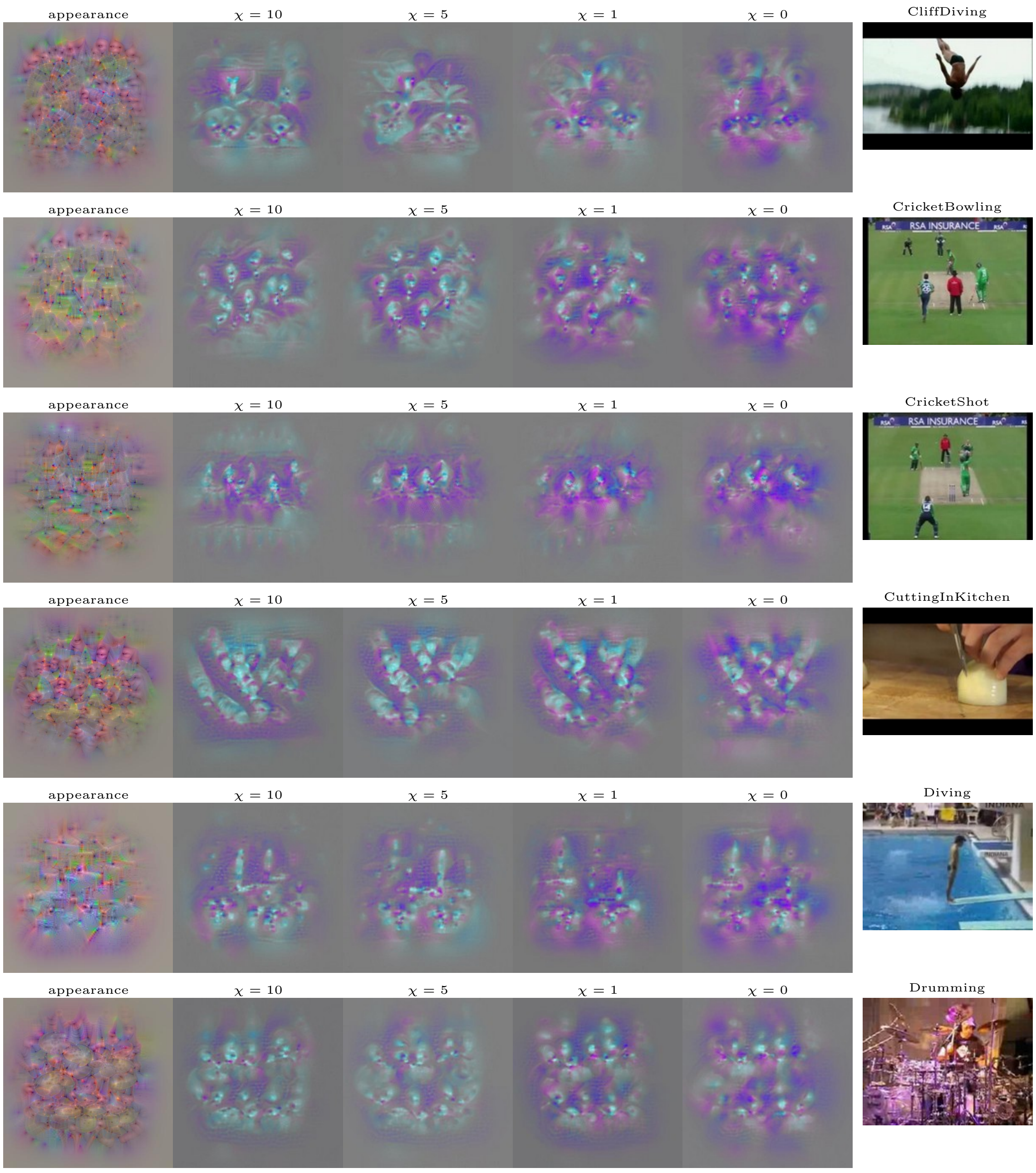


Figure B.27: Visualizations of classification units at the last layer of the network. The first column shows the appearance and the second to fifth column the motion input generated by maximizing the prediction layer output for the respective classes, while using different degrees of temporal variation regularization (χ). The last column shows 40 sample frames from the first video of the corresponding class in the test set. Best viewed electronically.

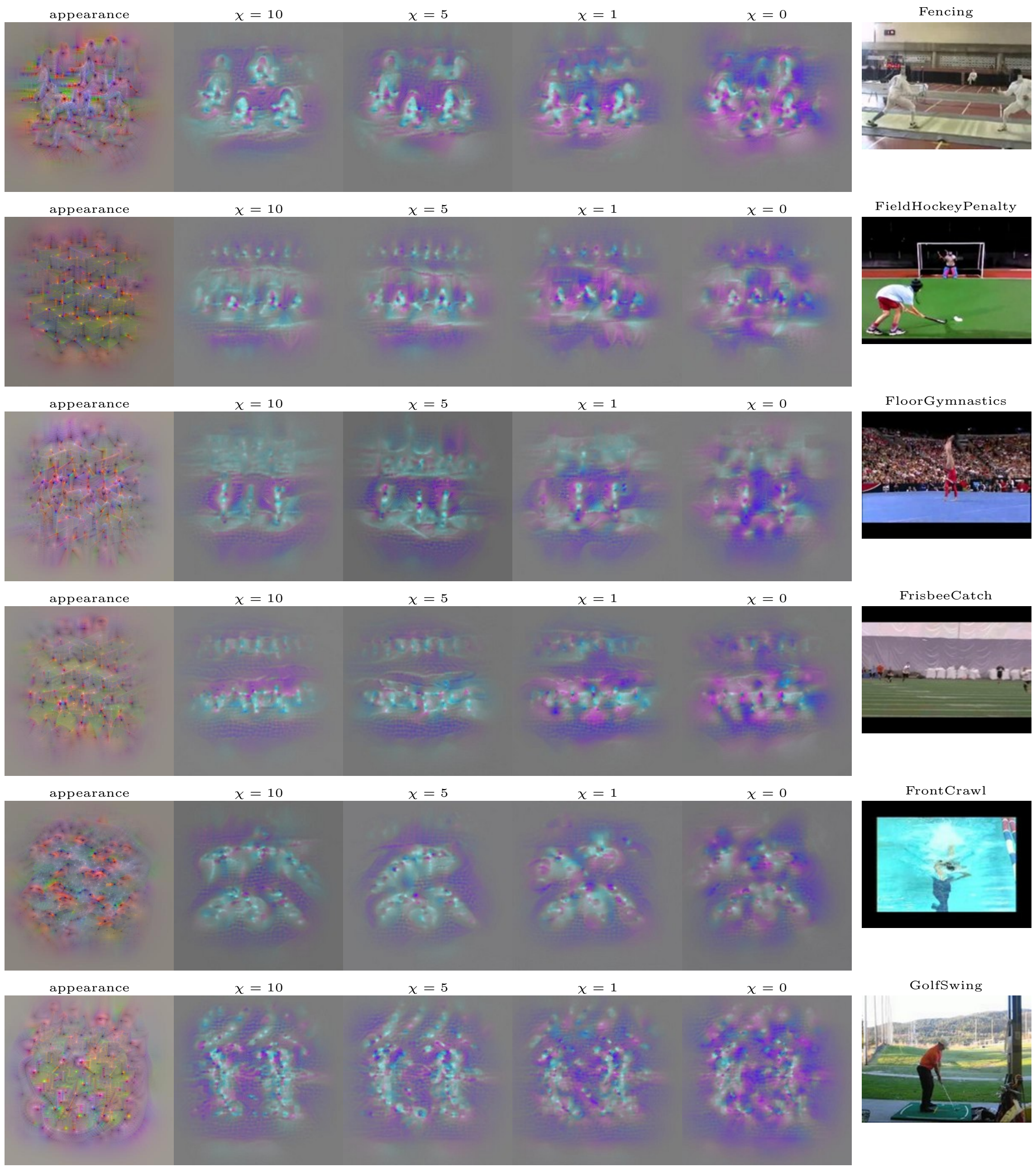


Figure B.28: Visualizations of classification units at the last layer of the network. The first column shows the appearance and the second to fifth column the motion input generated by maximizing the prediction layer output for the respective classes, while using different degrees of temporal variation regularization (χ). The last column shows 40 sample frames from the first video of the corresponding class in the test set. Best viewed electronically.

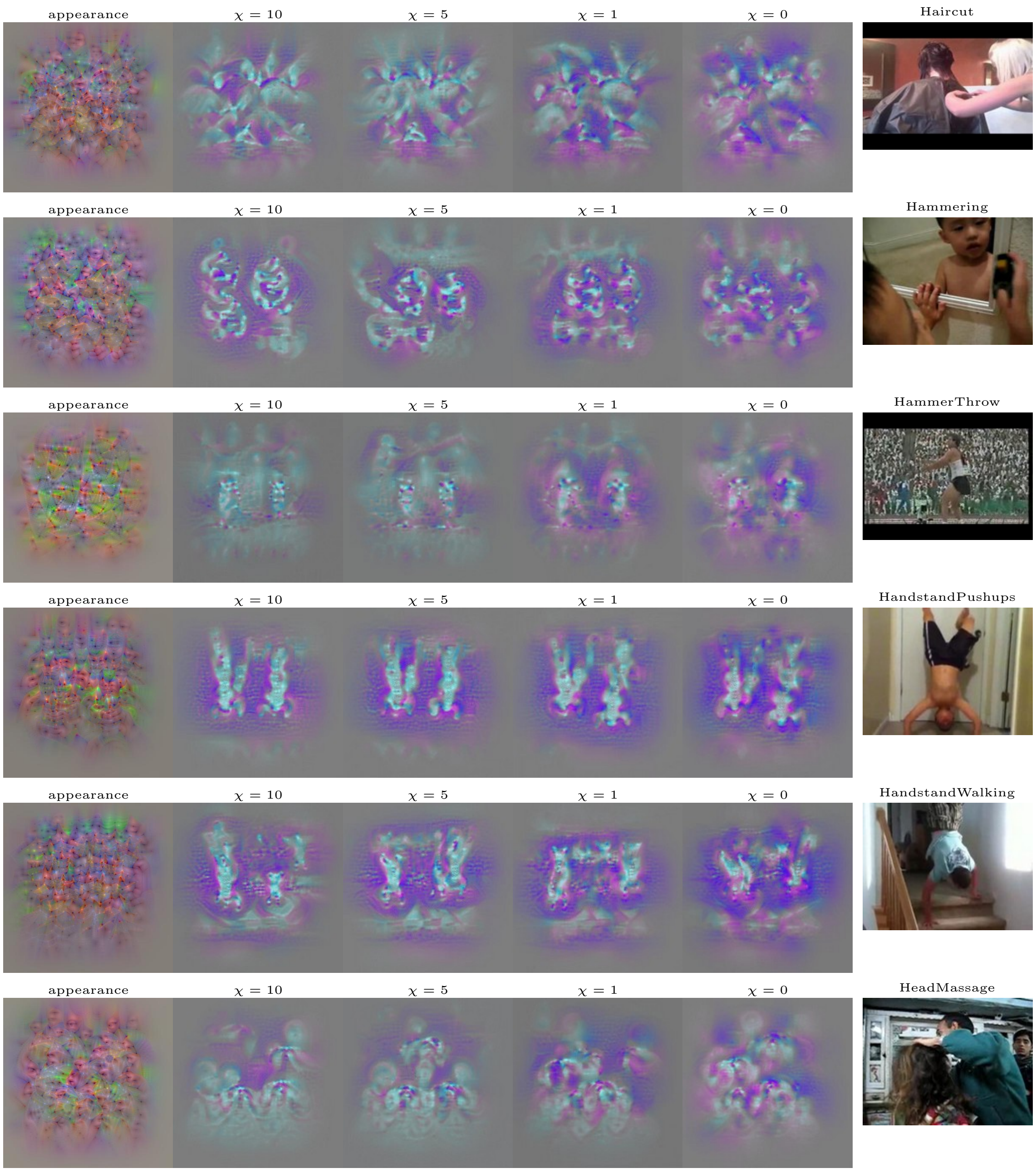


Figure B.29: Visualizations of classification units at the last layer of the network. The first column shows the appearance and the second to fifth column the motion input generated by maximizing the prediction layer output for the respective classes, while using different degrees of temporal variation regularization (χ). The last column shows 40 sample frames from the first video of the corresponding class in the test set. Best viewed electronically.

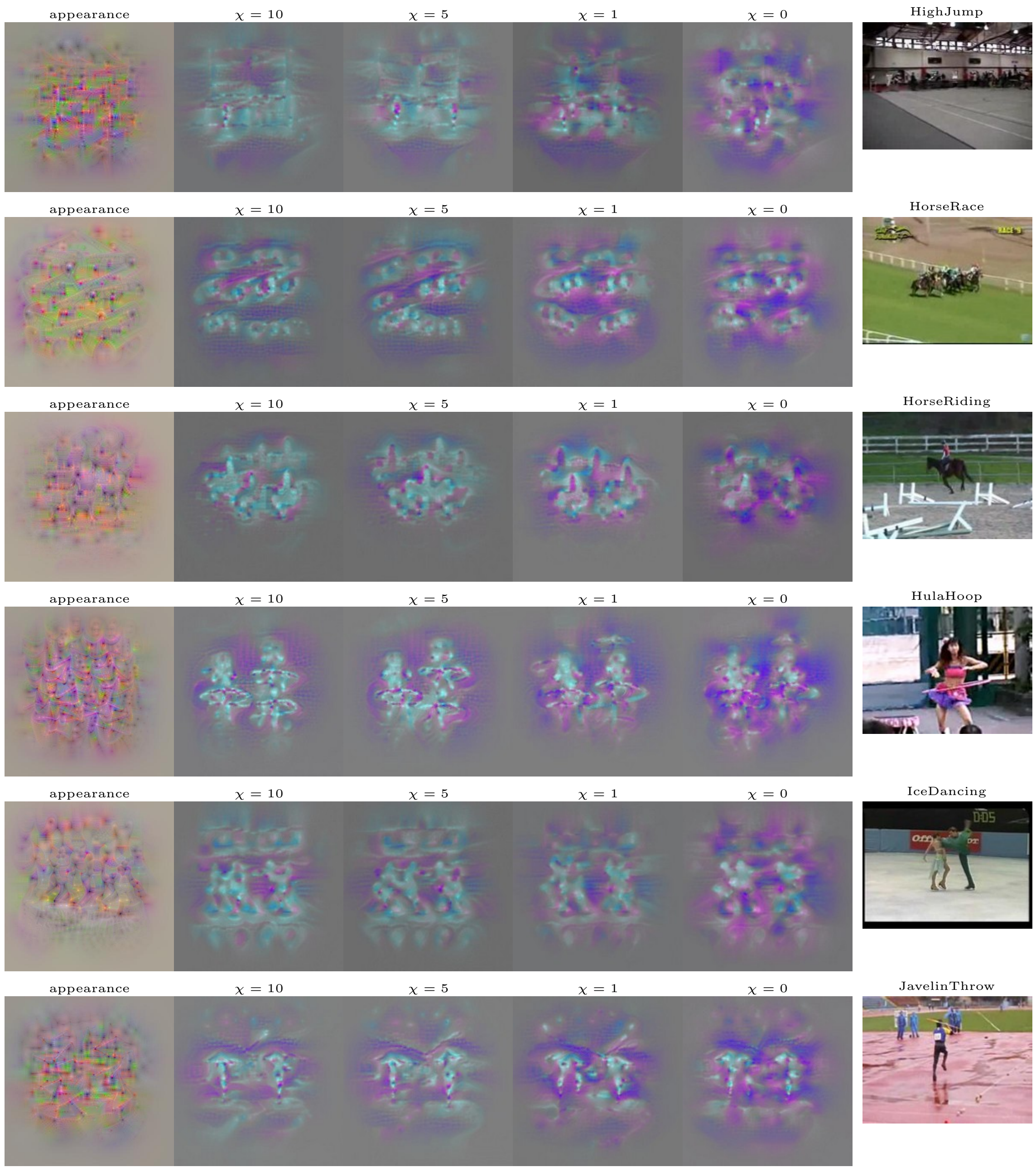


Figure B.30: Visualizations of classification units at the last layer of the network. The first column shows the appearance and the second to fifth column the motion input generated by maximizing the prediction layer output for the respective classes, while using different degrees of temporal variation regularization (χ). The last column shows 40 sample frames from the first video of the corresponding class in the test set. Best viewed electronically.

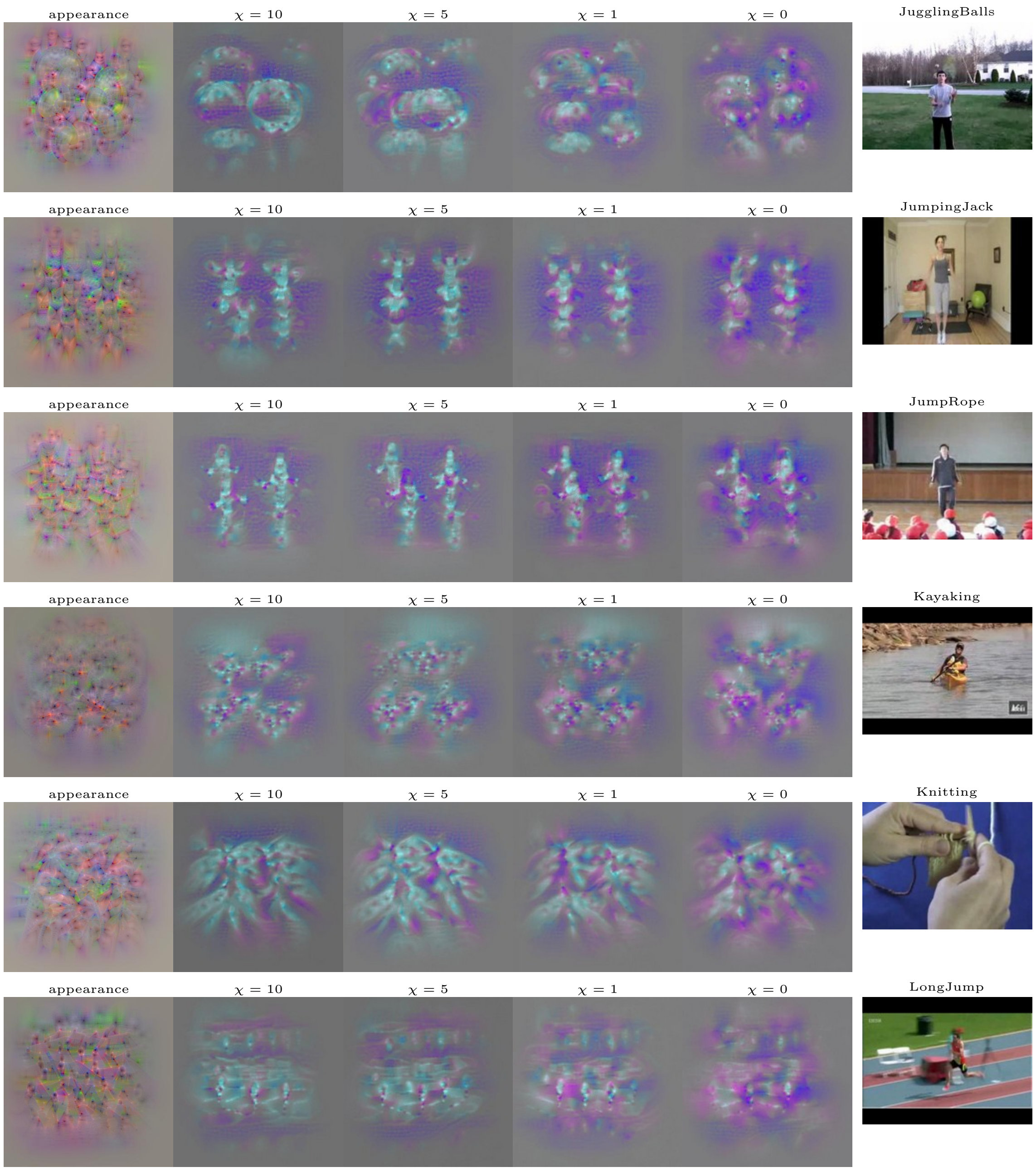


Figure B.31: Visualizations of classification units at the last layer of the network. The first column shows the appearance and the second to fifth column the motion input generated by maximizing the prediction layer output for the respective classes, while using different degrees of temporal variation regularization (χ). The last column shows 40 sample frames from the first video of the corresponding class in the test set. Best viewed electronically.

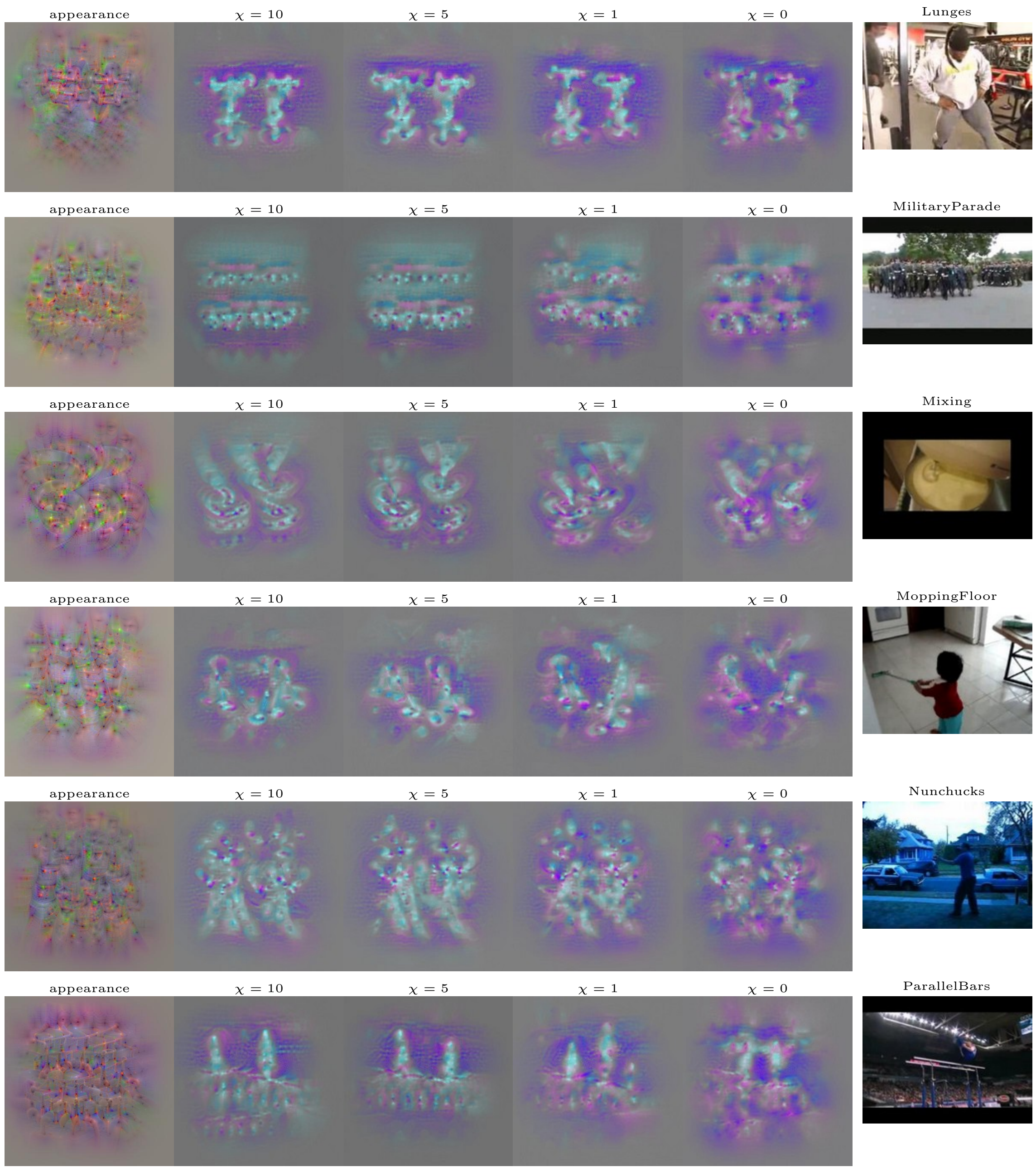


Figure B.32: Visualizations of classification units at the last layer of the network. The first column shows the appearance and the second to fifth column the motion input generated by maximizing the prediction layer output for the respective classes, while using different degrees of temporal variation regularization (χ). The last column shows 40 sample frames from the first video of the corresponding class in the test set. Best viewed electronically.

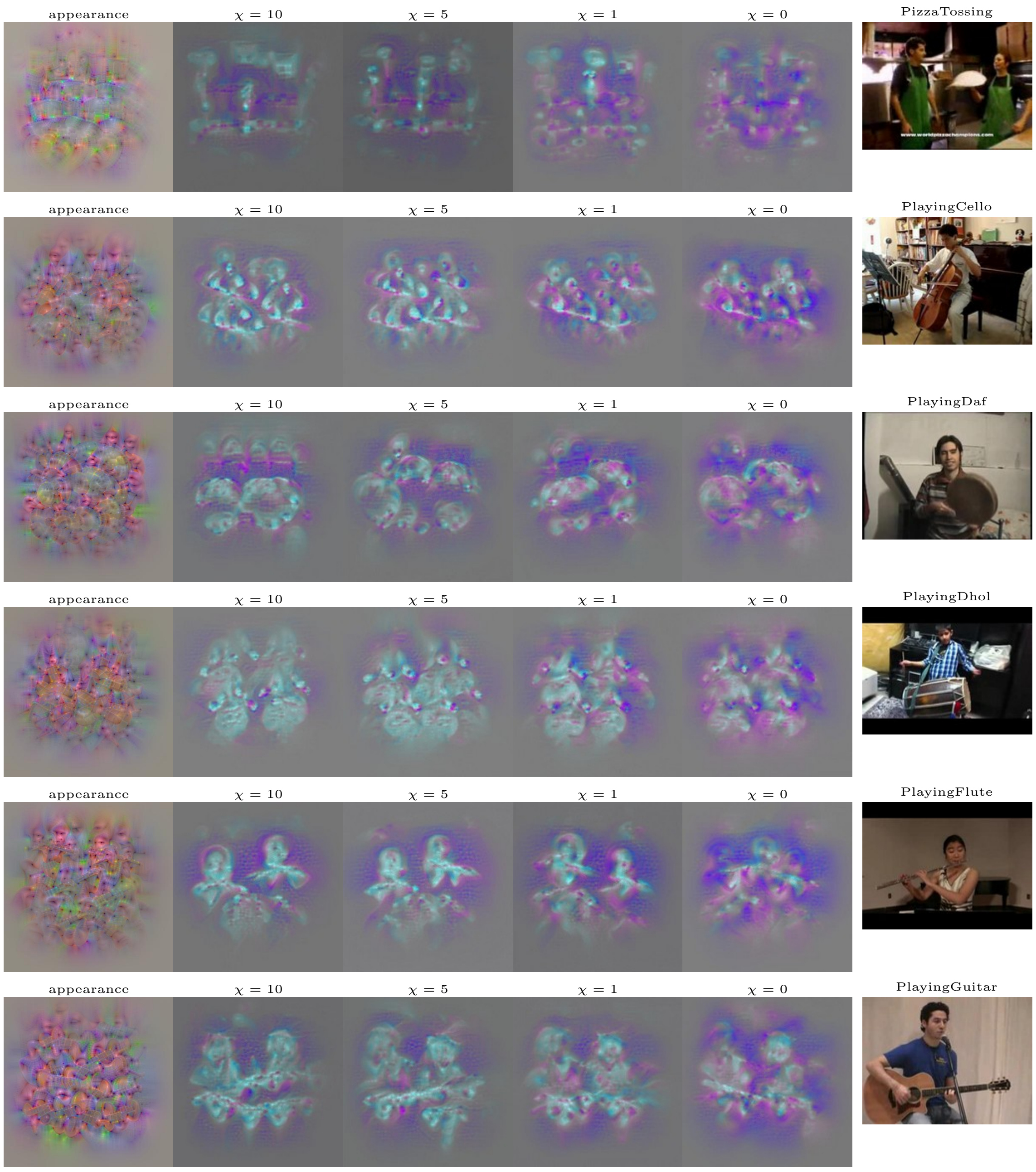


Figure B.33: Visualizations of classification units at the last layer of the network. The first column shows the appearance and the second to fifth column the motion input generated by maximizing the prediction layer output for the respective classes, while using different degrees of temporal variation regularization (χ). The last column shows 40 sample frames from the first video of the corresponding class in the test set. Best viewed electronically.

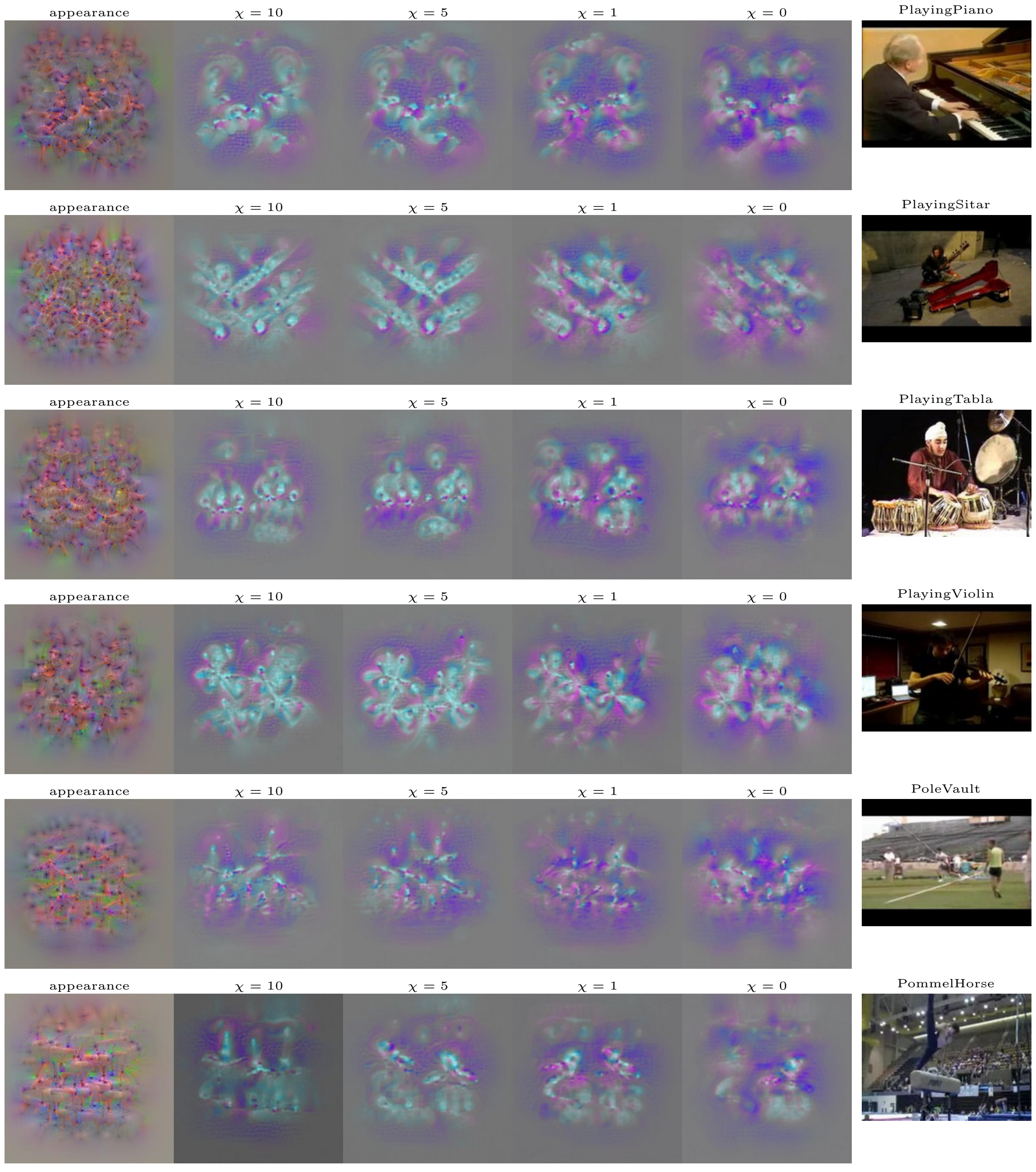


Figure B.34: Visualizations of classification units at the last layer of the network. The first column shows the appearance and the second to fifth column the motion input generated by maximizing the prediction layer output for the respective classes, while using different degrees of temporal variation regularization (χ). The last column shows 40 sample frames from the first video of the corresponding class in the test set. Best viewed electronically.

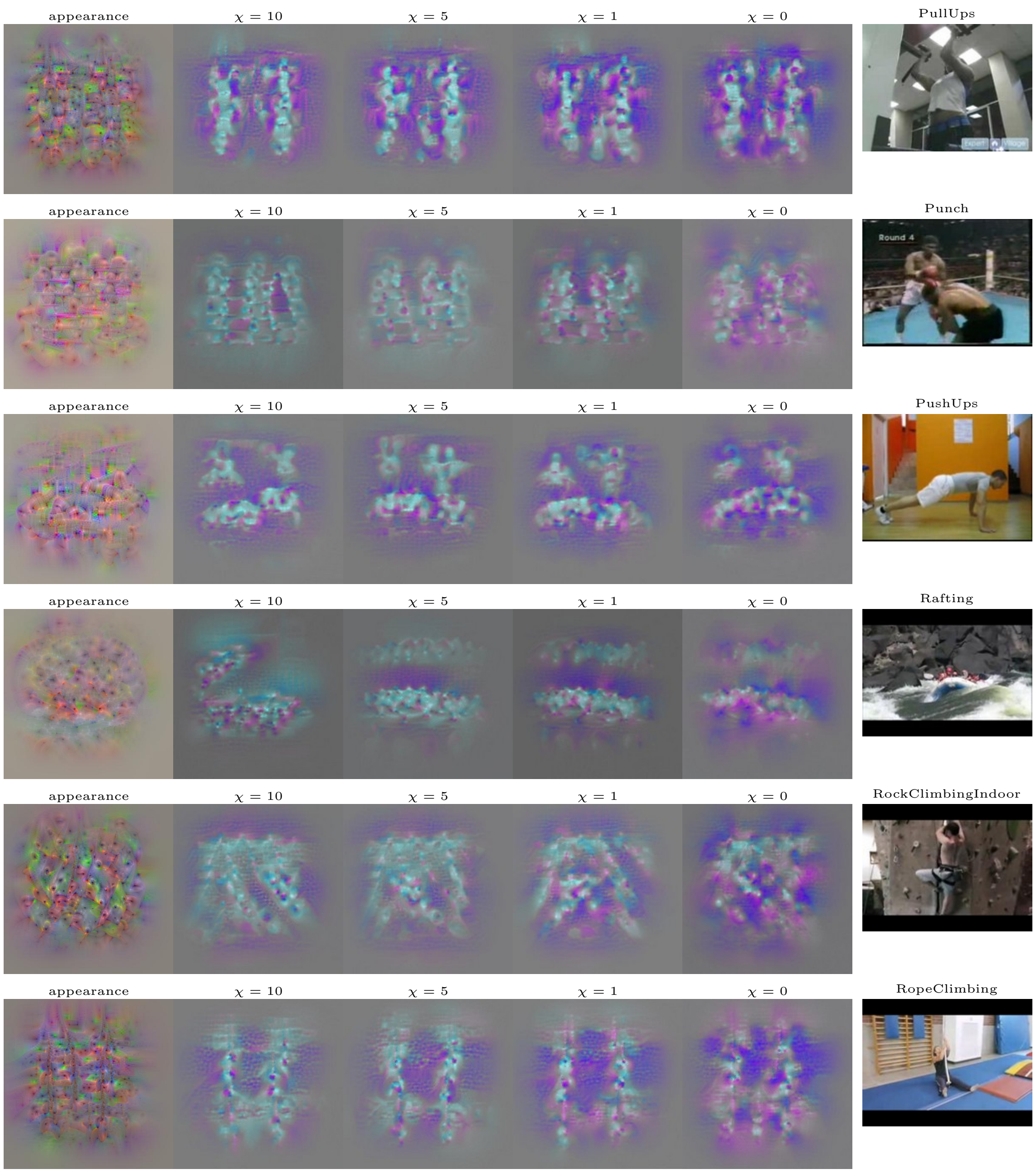


Figure B.35: Visualizations of classification units at the last layer of the network. The first column shows the appearance and the second to fifth column the motion input generated by maximizing the prediction layer output for the respective classes, while using different degrees of temporal variation regularization (χ). The last column shows 40 sample frames from the first video of the corresponding class in the test set. Best viewed electronically.

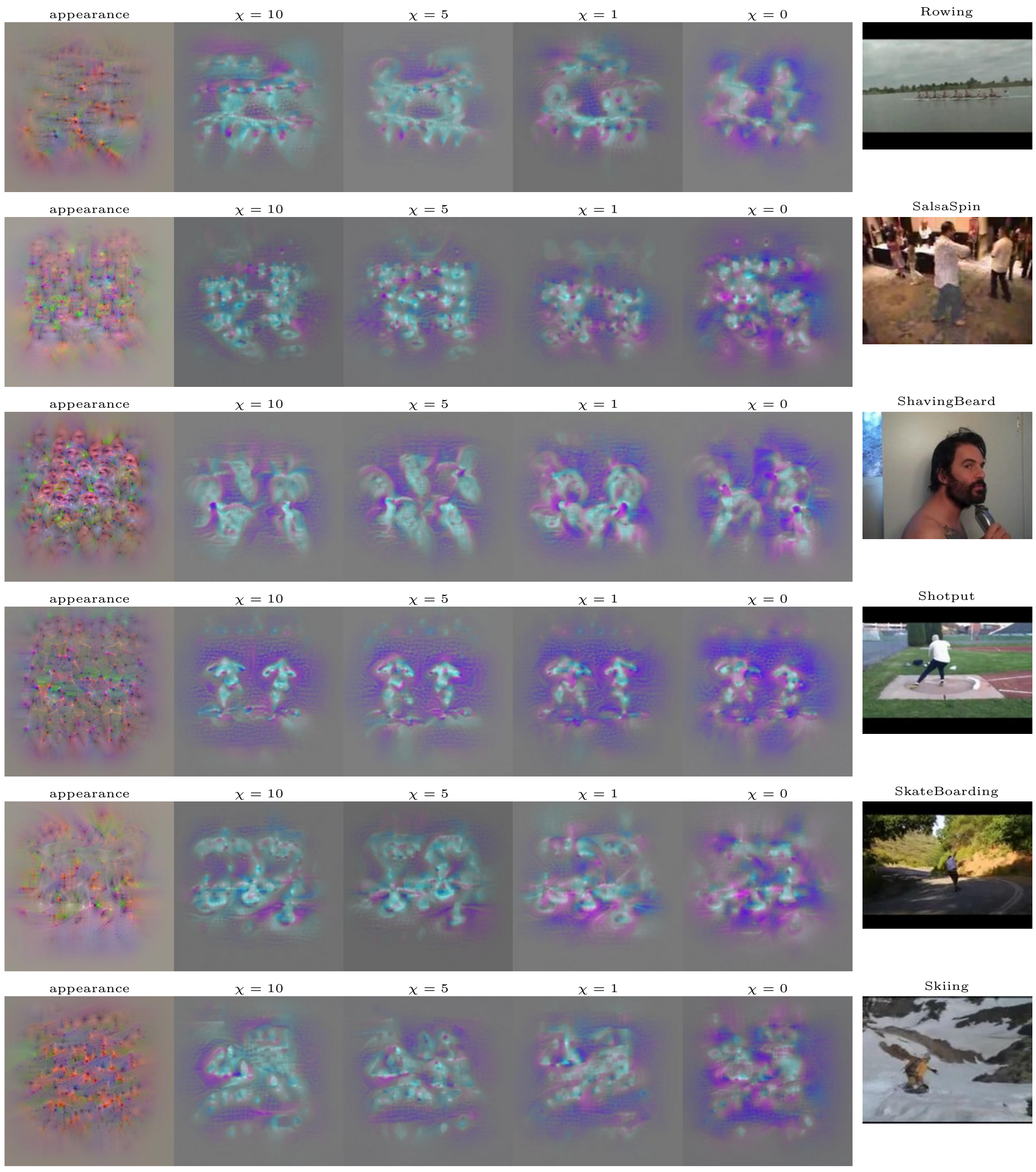


Figure B.36: Visualizations of classification units at the last layer of the network. The first column shows the appearance and the second to fifth column the motion input generated by maximizing the prediction layer output for the respective classes, while using different degrees of temporal variation regularization (χ). The last column shows 40 sample frames from the first video of the corresponding class in the test set. Best viewed electronically.

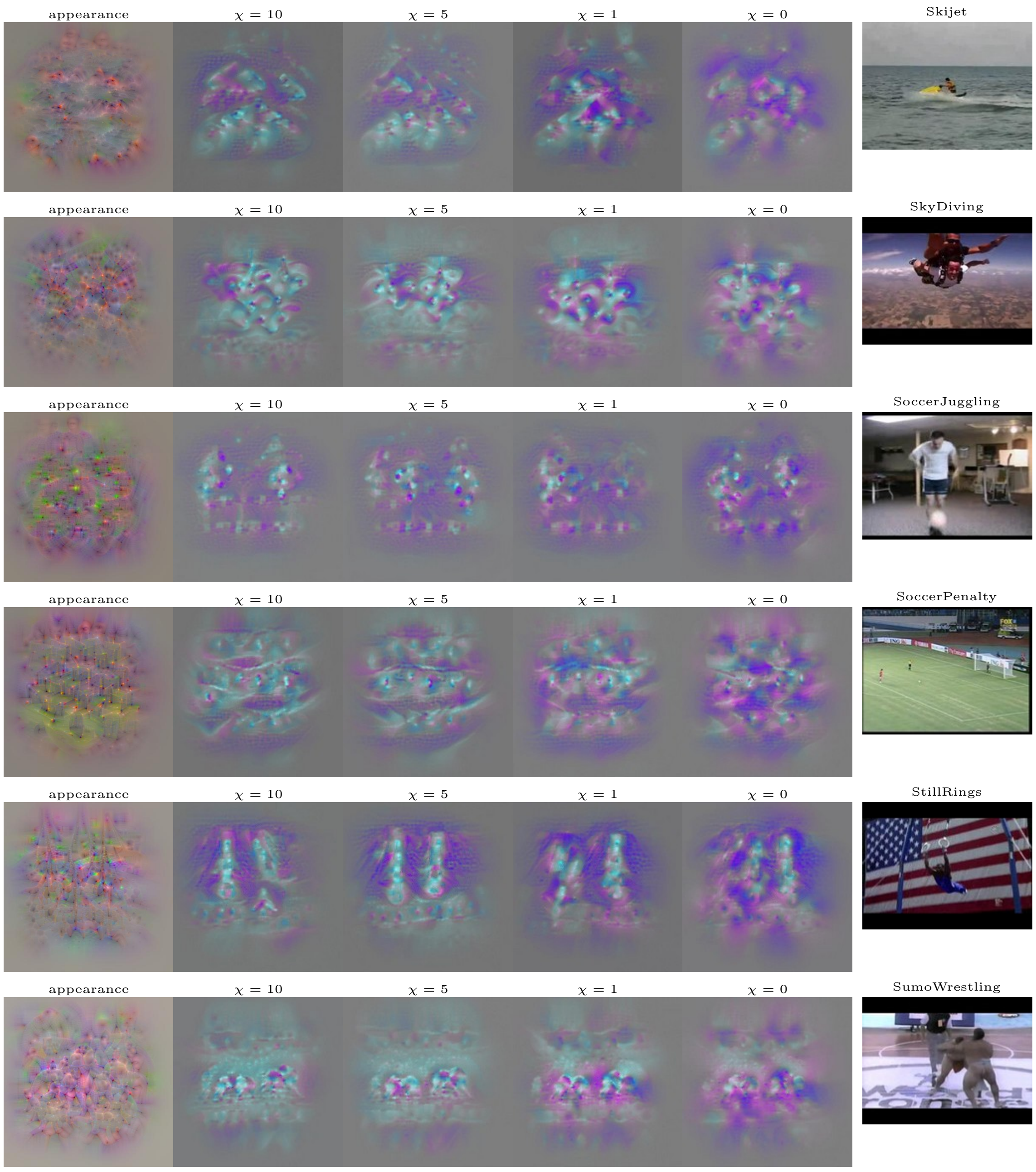


Figure B.37: Visualizations of classification units at the last layer of the network. The first column shows the appearance and the second to fifth column the motion input generated by maximizing the prediction layer output for the respective classes, while using different degrees of temporal variation regularization (χ). The last column shows 40 sample frames from the first video of the corresponding class in the test set. Best viewed electronically.

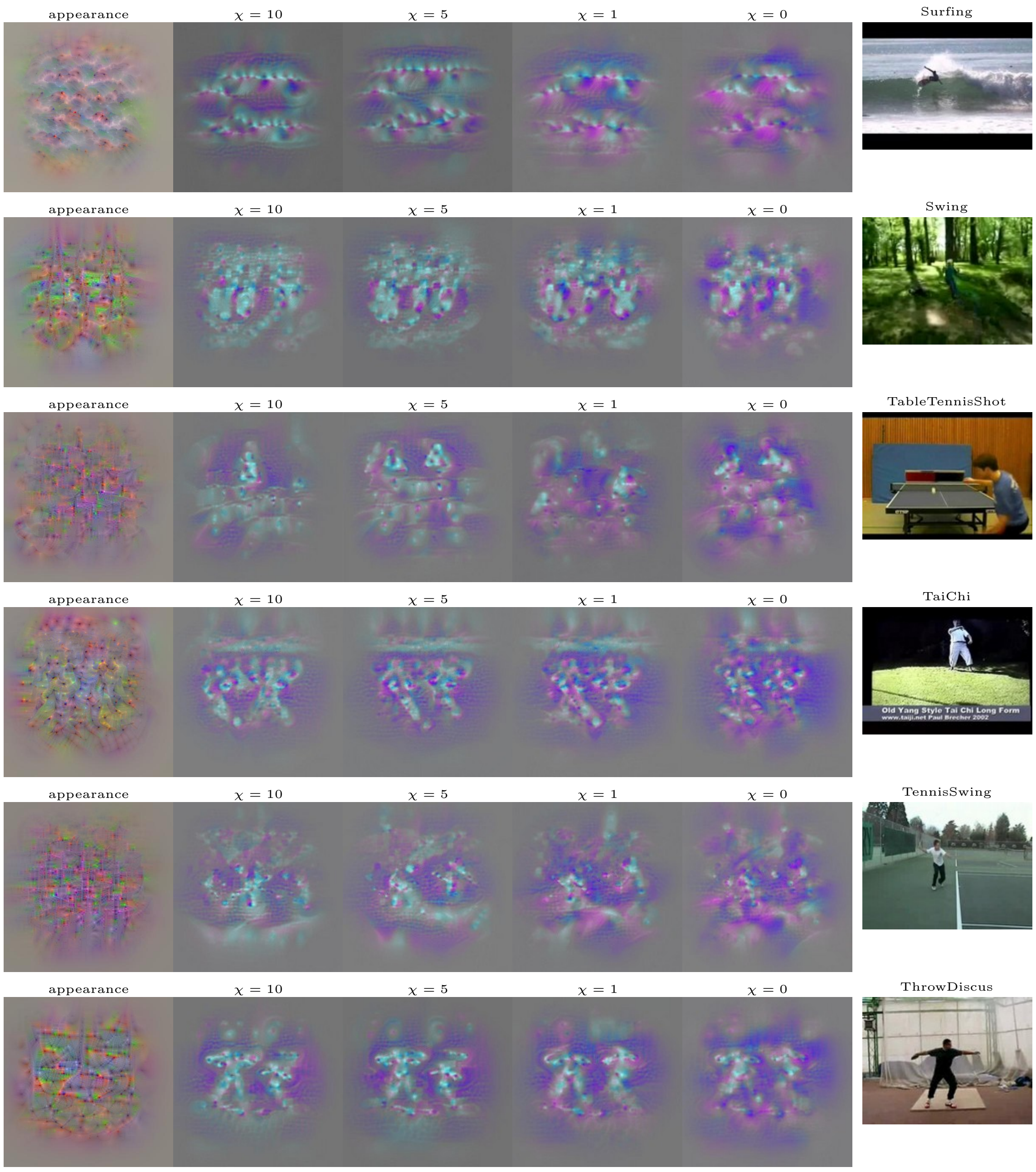


Figure B.38: Visualizations of classification units at the last layer of the network. The first column shows the appearance and the second to fifth column the motion input generated by maximizing the prediction layer output for the respective classes, while using different degrees of temporal variation regularization (χ). The last column shows 40 sample frames from the first video of the corresponding class in the test set. Best viewed electronically.

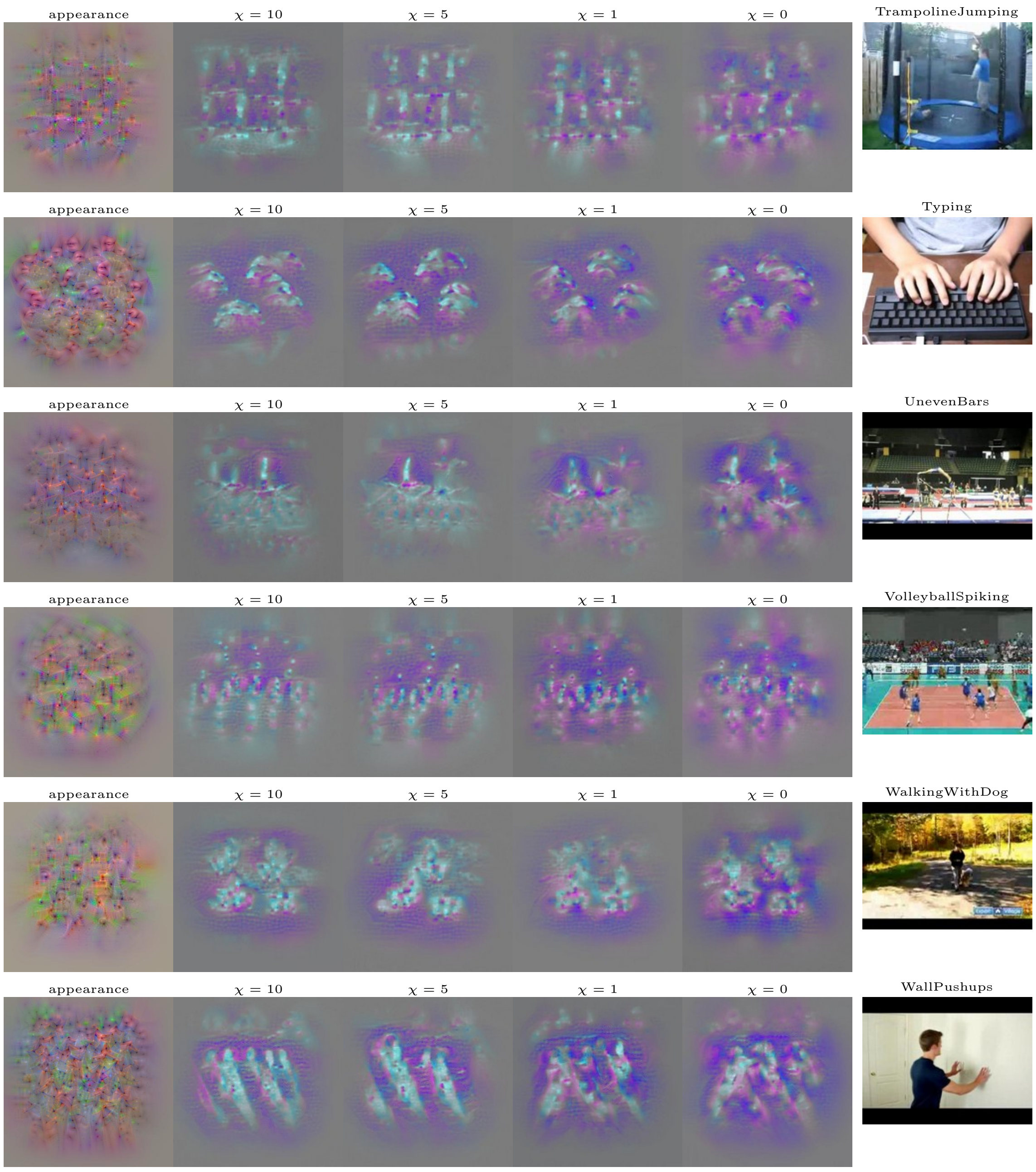


Figure B.39: Visualizations of classification units at the last layer of the network. The first column shows the appearance and the second to fifth column the motion input generated by maximizing the prediction layer output for the respective classes, while using different degrees of temporal variation regularization (χ). The last column shows 40 sample frames from the first video of the corresponding class in the test set. Best viewed electronically.

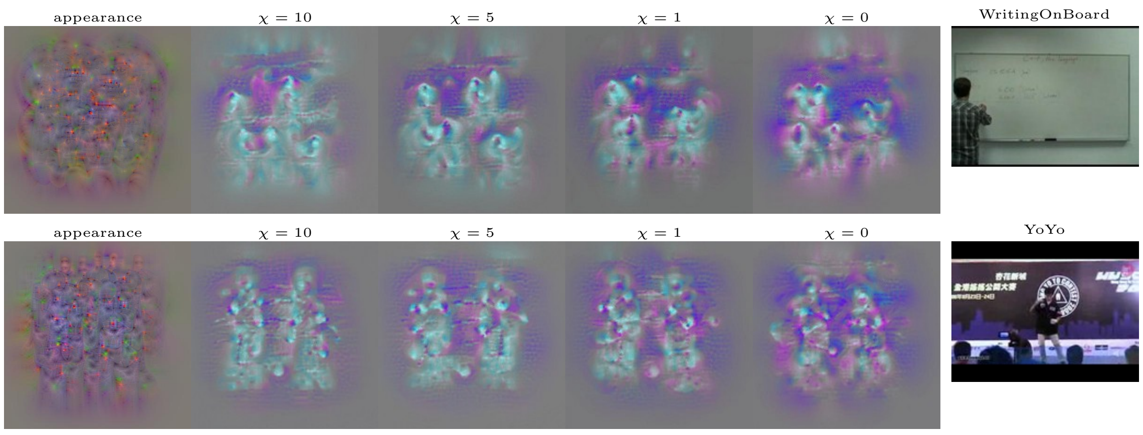


Figure B.40: Visualizations of classification units at the last layer of the network. The first column shows the appearance and the second to fifth column the motion input generated by maximizing the prediction layer output for the respective classes, while using different degrees of temporal variation regularization (χ). The last column shows 40 sample frames from the first video of the corresponding class in the test set. Best viewed electronically.

C

Baseline algorithms for Dynamic Scene Recognition

Slow feature analysis (SFA) approaches analyze temporal data to extract features that vary most slowly over time, taking those to be most indicative of the stable properties of the input (Wiskott and Sejnowski, 2002). The approach has been applied to dynamic scene recognition (Theriault et al., 2013) by extracting features from filter responses that are reputed to model primate V1 cortical operations, as they result from local maxima of spatially oriented, multiscale Gabor filters (Serre et al., 2007). The slowest varying features among those are identified by taking their temporal derivatives and subsequently are encoded via soft assignment with respect to a dictionary built with unsupervised sampling. Following encoding, the features are pooled into a feature vector via application of max-pooling to the entire video in spatial pyramid regions (Lazebnik et al., 2006).

Bags of spacetime energies (BoSE) (Feichtenhofer et al., 2014) is the penultimate version of spatiotemporal energy approaches applied to dynamic scenes (Derpanis et al., 2012, Feichtenhofer et al., 2013). The approach extracted dense measurements of spatiotemporal energy across a range of scales and orientations as well as CIE-LUV colour measurements. Here, the spatiotemporal features are augmented with dense SIFT measurements (Lowe, 2004) to more finely capture spatial orientation. The descriptors are encoded by Improved Fisher Vector (IFV) (Perronnin and Dance, 2007, Perronnin et al., 2010b) encoding with a visual word dictionary represented by a Gaussian Mixture Model (diagonal covariance) with 64 centres. We average the frame-level BoSE encodings over a video which simplifies the the temporal slice-based SVM prediction of the original BoSE system (Feichtenhofer et al., 2014). The simplification is employed for equality in comparison to other baselines which also train a single one-vs-rest SVMs for video classification.

Trajectory features (IDT) have been investigated with respect to a variety of video understanding tasks, e.g., (Moore et al., 2011, Raptis and Soatto, 2010, Sand and Teller, 2006, Wang et al., 2013). Curiously, it appears that they have not previously been applied to dynamic scene recognition. Recently, however, they have provided the basis for a number of outstanding approaches to action recognition as instantiated in improved Dense Trajectories (IDTs) (Wang and Schmid, 2013); therefore, it is of interest to evaluate their performance on scene recognition, as follows. Trajectory features are extracted across stabilized video sequences by concatenating a series of optical flow vectors for densely extracted interest points. Feature descriptors are aggregated across each trajectory in terms of trajectory shape (Wang et al., 2013), HOG (Dalal and Triggs, 2005), HOF (Laptev et al., 2008b) and MBH (Dalal et al., 2006) measurements. Following extraction, the features are encoded using (improved) Fisher Vectors (FVs) (Perronnin et al., 2010b) with dictionary represented by a Gaussian Mixture model (diagonal covariance) having 256 centres. Before training the GMM, all features are augmented with their normalized (x, y) image coordinates as an efficient way to capture location information. Details of extraction of the trajectories, their descriptors and encoding are exactly as in their original application to action recognition (Wang and Schmid, 2013). All DT and IDT parameters are used as in (Wang and Schmid, 2013, Wang et al., 2013) and their publicly available code is used to extract the descriptors.

Spatial convolutional network (S-CNN) features (Cimpoi et al., 2015) are generated from the last convolutional layer of a VGG-16 network (Simonyan and Zisserman, 2014b). The model is pre-trained on ImageNet (Deng et al., 2009). It has been shown that the features from such pre-trained CNNs are transferable to many other vision domains (Chatfield et al., 2014b, Cimpoi et al., 2015, Fang et al., 2014, Gupta et al., 2014). This approach derives its features from the last convolutional layer of a VGG-16, which uses features from the last conv-layer of VGG-16. The resulting 512-dimensional features are encoded using (improved) Fisher Vectors (FVs) (Perronnin et al., 2010b) with dictionary represented by a Gaussian Mixture model (diagonal covariance) having 64 centres. Features from a single video are extracted with a stride of 16 frames. Before encoding the features are augmented with their normalized (x, y) image coordinates, as with the above IDT approach.

Temporal convolutional network (T-CNN) uses a stack of 10 optical flow frames as input, with optical flow extracted by a standard algorithm (Brox et al., 2004) and is first pre-trained on the UCF101 action recognition dataset (Khurram Soomro and Shah, 2012). The final model is a CNN-M-2048 network (Chatfield et al., 2014b). (In our preliminary evaluation with this implementation, a recognition accuracy of 82.6% on UCF101 (split 1) was achieved, which compares favourably to the 81.2% reported originally (Simonyan and Zisserman, 2014a).) The same IFV encoding procedure as used for the spatial CNN above is employed, since this approach is common practice in state-of-the-art video action recognition (Gorban et al., 2015) and provided slightly better performance than using the output of the last fully connected layer.

Spatiotemporal convolutional network (C3D) provides a spatiotemporal analogue to the spatial S-CNN. As a generalization of spatial convolutional neural networks, 3D spatiotemporal networks working over image spacetime, (x, y, t), have potential to more directly capture temporal aspects of the data even while maintaining spatial information. Various previous efforts have been mounted to consider this potential (Ji et al., 2013, Karpathy et al., 2014, Tran et al., 2015a). Here, C3D is considered, as it has previously been applied to dynamic scene recognition (Tran et al., 2015a). Features are extracted by applying the C3D network model, pretrained on the Sports-1M dataset (Karpathy et al., 2014), densely to 16-frame snippets of the input video. As in (Tran et al., 2015a), the fully connected layer 6 outputs of each 16-frame clip are averaged across the video into a 4096-dimensional descriptor.

Classification is performed as in the original approaches (Cimpoi et al., 2015, Feichtenhofer et al., 2014, Simonyan and Zisserman, 2014a, Theriault et al., 2013, Tran et al., 2015a, Wang and Schmid, 2013), with a linear SVM (Cortes and Vapnik, 1995). Before training, the descriptors are L2-normalized. All feature vectors extracted from the training set are used to train one-vs-rest linear SVM classifiers. The SVM's regularization loss trade-off parameter is set to $C = 100$. During classification, each feature type is classified by its one-vs-rest SVM to yield SVM scores for a test video and an overall classification of the video according to the maximum score.

D

Qualitative results for Video Detection & Tracking

Qualitative examples for our D&T approach are shown in Figures D.1 and D.2. We show challenging samples from the ImageNet VID validation set. Our method could improve for multi-object occlusions (*e.g.* the cars in the second row of Fig. D.1 or the monkeys in the penultimate row of Fig. D.2), missed or incorrect classifications (*e.g.* the cars in the ultimate row of Fig. D.1 or the squirrel classified as hamster in the first row of Fig. D.2); another source of failure are occlusions as *e.g.* seen in the background of the frames shown in the ultimate row of Fig. D.1.

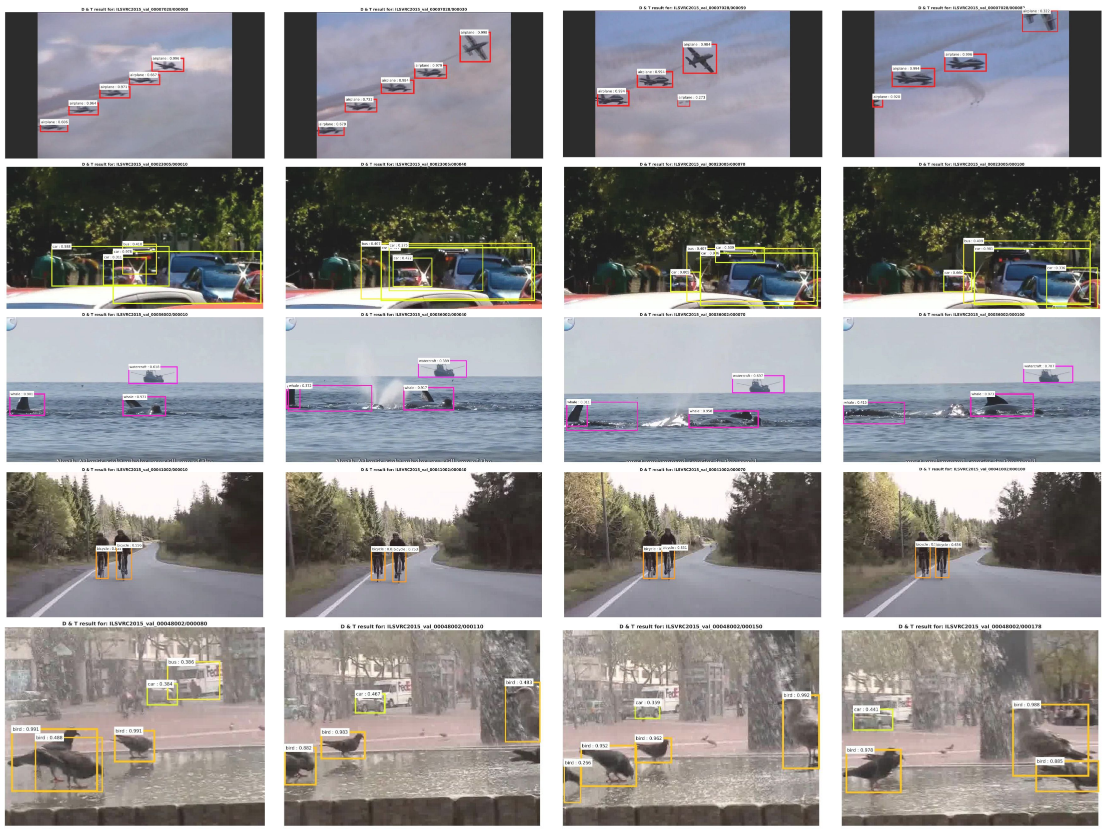


Figure D.1: Qualitative results for consecutive frames of videos from the ImageNet VID validation set. Failures can be attributed to scale, occlusion, misclassification, or NMS issues.

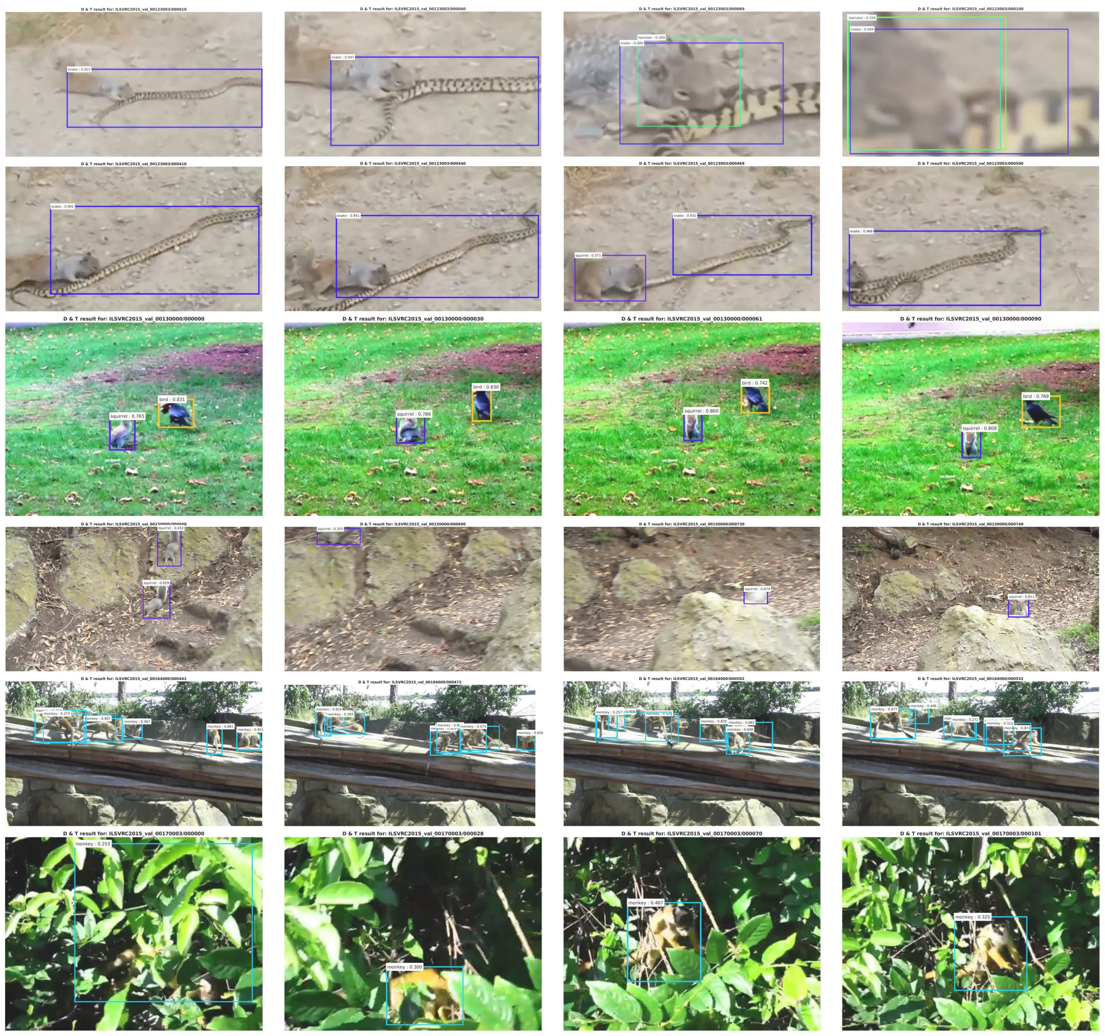


Figure D.2: Qualitative results for consecutive frames where our D&T approach could improve.

Bibliography

J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 14

N. Ballas, L. Yao, C. Pal, and A. Courville. Delving deeper into convolutional networks for learning video representations. In *Proc. ICLR*, 2016. 44, 47, 50, 51, 53, 57, 63, 99

S. Bell, C. L. Zitnick, K. Bala, and R. Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In *Proc. CVPR*, 2016. 101

L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr. Fully-convolutional siamese networks for object tracking. In *ECCV VOT Workshop*, 2016. 114, 115, 117

H. Bilen, B. Fernando, E. Gavves, A. Vedaldi, and S. Gould. Dynamic image networks for action recognition. In *Proc. CVPR*, 2016. 44, 53, 57, 68

D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui. Visual object tracking using adaptive correlation filters. In *Proc. CVPR*, 2010. 117

R. T. Born and R. B. Tootell. Segregation of global and local motion processing in primate middle temporal visual area. *Nature*, 357(6378):497–499, 1992. 21, 43

L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010. 12

T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *Proc. ECCV*, 2004. 34, 186

J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proc. CVPR*, 2017. 133

J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu. Semantic segmentation with second-order pooling. In *Proc. ECCV*, pages 430–443, 2012. 27

K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *Proc. BMVC.*, 2014a. 11

K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *Proc. BMVC*, 2014b. 17, 34, 36, 186

M. Cimpoi, S. Maji, and A. Vedaldi. Deep filter banks for texture recognition and segmentation. In *Proc. CVPR*, 2015. 38, 107, 186, 187

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995. 5, 187

N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. CVPR*, 2005. 186

N. Dalal, B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. In *Proc. ECCV*, 2006. 5, 56, 186

M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *Proc. ECCV*, 2016. 117

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proc. CVPR*, 2009. 2, 186

K. Derpanis and R. P. Wildes. Spacetime texture representation and recognition based on a spatiotemporal orientation analysis. *PAMI*, 34(6):1193–1205, 2012. 4

K. Derpanis, M. Lecce, K. Daniilidis, and R. P. Wildes. Dynamic scene understanding: The role of orientation features in space and time in scene classification. In *Proc. CVPR*, 2012. 56, 98, 99, 105, 107, 108, 134, 185

K. Derpanis, M. Sizintsev, K. Cannons, and R. P. Wildes. Action spotting and recognition based on a spatiotemporal orientation analysis. *PAMI*, 35(3):527–540, 2013. 5

P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *Second Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance, In conjunction with the ICCV*, pages 65–72, 2005. 56

J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proc. CVPR*, 2015. 5, 6, 39, 47, 68, 99

G. Doretto, A. Chiuso, Y. N. Wu, and S. Soatto. Dynamic textures. *IJCV*, 51(2):91–109, 2003. 4, 98

A. Dosovitskiy and T. Brox. Generating images with perceptual similarity metrics based on deep networks. In *Proc. NIPS*, 2016. 74

A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *Proc. ICCV*, 2015. 120

J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011. 13, 78

C. J. Duffy. MST neurons respond to optic flow and translational movement. *Journal of neurophysiology*, 80(4):1816–1827, 1998. 73

C. J. Duffy and R. H. Wurtz. Sensitivity of MST neurons to optic flow stimuli. i. a continuum of response selectivity to large-field stimuli. *Journal of neurophysiology*, 65(6):1329–1345, 1991. 73

S. Eifuku and R. H. Wurtz. Response to motion in extrastriate area MSTl: center-surround interactions. *Journal of Neurophysiology*, 80(1):282–296, 1998. 73

D. Erhan, Y. Bengio, A. Courville, and P. Vincent. Visualizing higher-layer features of a deep network. Technical Report 1341, University of Montreal, Jun 2009. 73, 74

H. Fang, S. Gupta, F. Iandola, R. Srivastava, L. Deng, P. Dollár, J. Gao, X. He, M. Mitchell, J. Platt, et al. From captions to visual concepts and back. In *Proc. CVPR*, 2014. 186

L. Fei-Fei and P. Perona. A Bayesian hierarchical model for learning natural scene categories. In *Proc. CVPR*, 2005. 4, 98

C. Feichtenhofer, A. Pinz, and R. P. Wildes. Spacetime forests with complementary features for dynamic scene recognition. In *Proc. BMVC*, 2013. 5, 99, 106, 185

C. Feichtenhofer, A. Pinz, and R. P. Wildes. Bags of spacetime energies for dynamic scene recognition. In *Proc. CVPR*, 2014. 5, 98, 99, 106, 107, 185, 187

C. Feichtenhofer, A. Pinz, and R. Wildes. Dynamically encoded actions based on spacetime saliency. In *Proc. CVPR*, 2015. 5, 56

C. Feichtenhofer, A. Pinz, and R. P. Wildes. Dynamic scene recognition with complementary spatiotemporal features. *PAMI*, 38(12):2389–2401, 2016a. 5

C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *Proc. CVPR*, 2016b. 98, 99, 103

D. J. Felleman and D. C. Van Essen. Distributed hierarchical processing in the primate cerebral cortex. *Cerebral cortex (New York, NY: 1991)*, 1(1):1–47, 1991. 2, 72

P. Földiák. Learning invariance from transformation sequences. *Neural Computation*, 3(2):194–200, 1991. 76

K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36:193–202, 1980. 14

S. Gidaris and N. Komodakis. Object detection via a multi-region and semantic segmentation-aware cnn model. In *Proc. CVPR*, 2015. 124

R. B. Girshick. Fast R-CNN. In *Proc. ICCV*, 2015. 6, 22, 23, 115, 117, 119

R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proc. CVPR*, 2014. 6, 15, 22, 42, 115, 116, 119

G. Gkioxari and J. Malik. Finding action tubes. In *Proc. CVPR*, 2015. 115, 122

M. A. Goodale and A. D. Milner. Separate visual pathways for perception and action. *Trends in Neurosciences*, 15(1):20–25, 1992. 2, 20, 42, 72

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Proc. NIPS*, 2014. 74

I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. 83

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`. 11

A. Gorban, H. Indrees, Y. Jiang, A. R. Zamir, I. Laptev, M. Shah, and R. Sukthankar. Thumos challenge: Action recognition with a large number of classes. `http://wwwthumos.info/`, 2015. 6, 186

R. Goroshin, J. Bruna, J. Tompson, D. Eigen, and Y. LeCun. Unsupervised feature learning from temporal data. In *Proc. ICCV*, 2015. 47, 77

M. Goudreau, C. Giles, S. Chakradhar, and D. Chen. First-order versus second-order single-layer recurrent neural networks. *IEEE Transactions on Neural Networks*, 5:511–513, 1994. 60

S. Gupta, R. Girshick, P. Arbelaez, and J. Malik. Learning rich features from RGB-D images for object detection and segmentation. In *Proc. ECCV*. 2014. 186

D. A. Gusnard and M. E. Raichle. Searching for a baseline: functional imaging and the resting human brain. *Nature Reviews Neuroscience*, 2(10):685–694, 2001. 73

K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proc. CVPR*, 2015. 13

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. CVPR*, 2016a. 6, 7, 15, 22, 23, 42, 43, 44, 45, 48, 49, 51, 57, 58, 61, 62, 67, 99, 100, 101, 102, 103, 107, 114, 117, 118

K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *Proc. ECCV*, 2016b. 42, 44, 57, 58, 65, 100, 103

D. Held, S. Thrun, and S. Savarese. Learning to track at 100 FPS with deep regression networks. In *Proc. ECCV*, 2016. 114, 115, 117, 124

J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *PAMI*, 37(3):583–596, 2015. 117

G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012a. 15

G. Hinton, N. Srivastava, and K. Swersky. Neural networks for machine learning lecture 6a overview of mini–batch gradient descent. 2012b. 13

G. E. Hinton, S. Osindero, and Y. W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006. 74

D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, 160:106–154, 1962. 14

Y. Ioannou, D. Robertson, J. Shotton, R. Cipolla, and A. Criminisi. Training cnns with low-rank filters for efficient image classification. In *Proc. ICLR*, 2016. 47

S. Ioffe. Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. *arXiv preprint arXiv:1702.03275*, 2017. 14

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. ICML*, 2015. 13, 35, 42, 44, 48, 49, 50, 57, 62, 63, 103

H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid. Aggregating local image descriptors into compact codes. *PAMI*, 2012. 5

H. Jhuang, T. Serre, L. Wolf, and T. Poggio. A biologically inspired system for action recognition. In *Proc. ICCV*, pages 1–8, 2007. 5

S. Ji, W. Xu, M. Yang, and K. Yu. 3D convolutional neural networks for human action recognition. *PAMI*, 35(1):221–231, 2013. 26, 27, 43, 57, 187

A. Joulin, K. Tang, and L. Fei-Fei. Efficient image and video co-localization with frank-wolfe algorithm. In *Proc. ECCV*, 2014. 116

K. Kang, H. Li, J. Yan, X. Zeng, B. Yang, T. Xiao, C. Zhang, Z. Wang, R. Wang, X. Wang, and W. Ouyang. T-CNN: tubelets with convolutional neural networks for object detection from videos. *arXiv preprint*, 2016a. 114, 116, 123, 124, 125, 126

K. Kang, W. Ouyang, H. Li, and X. Wang. Object detection from video tubelets with convolutional neural networks. In *Proc. CVPR*, 2016b. 116, 123, 124, 125, 126

K. Kang, H. Li, T. Xiao, W. Ouyang, J. Yan, X. Liu, and X. Wang. Object detection in videos with tubelet proposal networks. In *Proc. CVPR*, 2017. 116, 123, 124, 125, 126

H.-O. Karnath. New insights into the functions of the superior temporal cortex. *Nature Reviews Neuroscience*, 2(8):568–576, 2001. 73

A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proc. CVPR*, 2014. 5, 6, 26, 27, 39, 43, 57, 68, 99, 100, 187

A. R. Z. Khurram Soomro and M. Shah. Ucf101: A dataset of 101 human actions calsses from videos in the wild. Technical Report CRCV-TR-12-01, UCF Center for Research in Computer Vision, 2012. 6, 36, 51, 64, 106, 107, 186

D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proc. ICLR*, 2015. 13

A. Kläser, M. Marszałek, and C. Schmid. A spatio-temporal descriptor based on 3d-gradients. In *Proc. BMVC*, 2008. 5, 56

Z. Kourtzi and N. Kanwisher. Activation in human mt/mst by static images with implied motion. *Journal of cognitive neuroscience*, 12(1):48–55, 2000. 2, 72, 73

D. J. Kravitz, K. S. Saleem, C. I. Baker, and M. Mishkin. A new neural framework for visuospatial processing. *Nature Reviews Neuroscience*, 12(4):217–230, 2011. 73

A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, pages 1106–1114, 2012a. 1, 6, 13, 15, 17, 42, 44, 50, 57

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. NIPS*, 2012b. 2, 3, 100, 102, 103

H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *Proc. ICCV*, 2011. 6, 36, 51, 64, 106, 107

S. Kwak, M. Cho, I. Laptev, J. Ponce, and C. Schmid. Unsupervised object discovery and tracking in video collections. In *Proc. CVPR*, 2015. 116

I. Laptev, M. Marszałek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *Proc. CVPR*, 2008a. 5

I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *Proc. CVPR*, 2008b. 5, 56, 186

S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proc. CVPR*, 2006. 4, 5, 98, 185

Q. Le, M. Ranzato, R. Monga, M. Devin, K. Chen, G. Corrado, J. Dean, and A. Ng. Building high-level features using large scale unsupervised learning. In *Proc. ICML*, 2012. 74

Q. V. Le, W. Y. Zou, S. Y. Yeung, and A. Y. Ng. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *Proc. CVPR*, 2011. 43, 57

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989. 6, 14

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 14

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. 1

G. Lev, G. Sadeh, B. Klein, and L. Wolf. RNN Fisher vectors for action recognition and image annotation. In *Proc. ECCV*, 2016. 68

F. F. Li, R. VanRullen, C. Koch, and P. Perona. Rapid natural scene categorization in the near absence of attention. *Proceedings of the National Academy of Sciences*, 99(14):9596–9601, 2002. 4

Y. Li, K. He, J. Sun, et al. R-FCN: Object detection via region-based fully convolutional networks. In *Proc. NIPS*, 2016a. 6, 11, 22, 23, 114, 115, 117, 118, 119, 120, 123

Z. Li, E. Gavves, M. Jain, and C. G. Snoek. VideoLSTM convolves, attends and flows for action recognition. *arXiv preprint arXiv:1607.01794*, 2016b. 57, 68

M. Lin, Q. Chen, and S. Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013. 102

T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *Proc. ECCV*, 2014. 114

T.-Y. Lin, A. RoyChowdhury, and S. Maji. Bilinear CNN models for fine-grained visual recognition. In *Proc. ICCV*, 2015. 27, 29, 37, 38

W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *Proc. ECCV*, 2016. 22, 115

J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proc. CVPR*, 2015. 15, 42, 118

D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004. 5, 185

C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang. Hierarchical convolutional features for visual tracking. In *Proc. ICCV*, 2015. 114, 115, 117, 126

B. Mahasseni and S. Todorovic. Regularizing long short term memory with 3D human-skeleton sequences for action recognition. In *Proc. CVPR*, 2016. 44, 47, 57

A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. In *Proc. CVPR*, 2015. 37

A. Mahendran and A. Vedaldi. Visualizing deep convolutional neural networks using natural pre-images. *IJCV*, pages 1–23, 2016. 74, 75, 76, 78

M. Marszalek, I. Laptev, and C. Schmid. Actions in context. In *Proc. CVPR*, 2009. 4, 98

J. H. Maunsell and D. C. Van Essen. Functional properties of neurons in middle temporal visual area of the macaque monkey. i. selectivity for stimulus direction, speed, and orientation. *Journal of neurophysiology*, 49(5):1127–1147, 1983. 72, 73

L. Wang et al. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, 2016. 68, 69, 132

R. Memisevic and G. E. Hinton. Learning to represent spatial transformations with factored higher-order boltzmann machines. *Neural Computation*, 22(6):1473–1492, 2010. 59

M. Mishkin and L. G. Ungerleider. Contribution of striate inputs to the visuospatial functions of parieto-preoccipital cortex in monkeys. *Behavioural brain research*, 6(1):57–77, 1982. 20

M. Mishkin, L. G. Ungerleider, and K. A. Macko. Object vision and spatial vision: two cortical pathways. *Trends in neurosciences*, 6:414–417, 1983. 72

B. Moore, S. Ali, R. Mehran, and M. Shah. Visual crowd surveillance through a hydrodynamics lens. *Commun. ACM*, 54(12):64–73, 2011. 186

A. Mordvintsev, C. Olah, and M. Tyka. Inceptionism: Going deeper into neural networks. Google Research Blog. Retrieved June 20, 2015. `https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html`. 74

H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. In *Proc. CVPR*, 2016. 115

J. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *Proc. CVPR*, 2015a. 98

J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *Proc. CVPR*, 2015b. 5, 6, 27, 32, 39, 44, 47, 52, 53, 57, 68, 98

A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Proc. NIPS*, 2016. 74

A. Nguyen, J. Yosinski, Y. Bengio, A. Dosovitskiy, and J. Clune. Plug & play generative networks: Conditional iterative generation of images in latent space. 2017. 74

J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. Singh. Action-conditional video prediction using deep networks in atari games. In *Proc. NIPS*, 2015. 30, 59

A. Oliva and A. Torralba. Modeling the shape of the scene: a holistic representation of the spatial envelope. *IJCV*, 2001. 4, 98

P. Over, G. Awad, M. Michael, J. Fiscus, G. Sanders, W. Kraaij, A. Smeaton, and Q. Quenot. Trecvid 2013 - an overview of the goals, tasks, data, evaluation mechanisms and metrics. In *Proc. TRECVID*, 2013. 98

D. Park, C. L. Zitnick, D. Ramanan, and P. Dollár. Exploring weak stabilization for motion feature extraction. In *Proc. CVPR*, 2013. 98

X. Peng and C. Schmid. Multi-region two-stream R-CNN for action detection. In *Proc. ECCV*, 2016. 116, 122

X. Peng, L. Wang, X. Wang, and Y. Qiao. Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice. *CoRR*, abs/1405.4506, 2014. 39

F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In *Proc. CVPR*, 2007. 5, 185

F. Perronnin, J. Sánchez, and T. Mensink. Improving the Fisher kernel for large-scale image classification. In *Proc. ECCV*, 2010a. 6

F. Perronnin, J. Sánchez, and T. Mensink. Improving the Fisher kernel for large-scale image classification. In *Proc. ECCV*, 2010b. 185, 186

Y. Poleg, A. Ephrat, S. Peleg, and C. Arora. Compact CNN for indexing egocentric videos. In *Proc. CVPR*, 2015. 98, 99

M. Potter. Recognition and memory for briefly presented scenes. *Frontiers in Psychology*, 3(32): 1–9, 2012. 98

M. C. Potter and E. I. Levy. Recognition memory for a rapid sequence of pictures. *Journal of experimental psychology*, 81(1):10, 1969. 4

A. Prest, C. Leistner, J. Civera, C. Schmid, and V. Ferrari. Learning object class detectors from weakly annotated video. In *Proc. CVPR*, 2012. 116

A. Quattoni and A. Torralba. Recognizing indoor scenes. In *Proc. CVPR*, 2009. 106

M. Raptis and S. Soatto. Tracklet descriptors for action modeling and video analysis. In *Proc. ECCV*, 2010. 186

E. Real, J. Shlens, S. Mazzocchi, X. Pan, and V. Vanhoucke. YouTube-BoundingBoxes: A Large High-Precision Human-Annotated Data Set for Object Detection in Video. *ArXiv e-prints*, 2017. 116

J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proc. CVPR*, 2016. 22, 115

S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *PAMI*, 2016. 6, 11, 22, 23, 114, 115, 117, 118, 119, 123

G. A. Rousselet, S. J. Thorpe, M. Fabre-Thorpe, et al. How parallel is visual processing in the ventral pathway? *Trends in cognitive sciences*, 8(8):363–370, 2004. 4

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986. 12

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, S. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. Berg, and F. Li. Imagenet large scale visual recognition challenge. *IJCV*, 2015a. 14, 15

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 115(3):211–252, 2015b. 114, 123

S. Saha, G. Singh, M. Sapienza, P. H. Torr, and F. Cuzzolin. Deep learning for detecting multiple space-time action tubes in videos. In *Proc. BMVC*, 2016. 116, 122

K. Saleem, W. Suzuki, K. Tanaka, and T. Hashikawa. Connections between anterior inferotemporal cortex and superior temporal sulcus regions in the macaque monkey. *Journal of Neuroscience*, 20(13):5083–5101, 2000. 2, 72

T. Salimans and D. P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Proc. NIPS*, 2016. 14

P. Sand and S. Teller. Particle video: Long-range motion estimation using point trajectories. In *Proc. CVPR*, 2006. 186

F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proc. CVPR*, 2015. 6, 42

P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. In *Proc. ICLR*, 2014. 11

T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio. Robust object recognition with cortex-like mechanisms. *PAMI*, 29(3):411–426, 2007. 185

S. Sharma, R. Kiros, and R. Salakhutdinov. Action recognition using visual attention. In *NIPS workshop on Time Series*. 2015. 6, 44, 47, 57

E. Shechtman and M. Irani. Space-time behavior-based correlation-or-how to tell if two underlying motion fields are similar without computing them? *PAMI*, 29(11):2045–2056, 2007. 5

A. Shrivastava, A. Gupta, and R. Girshick. Training region-based object detectors with online hard example mining. In *Proc. CVPR*, 2016. 123

N. Shroff, P. Turaga, and R. Chellappa. Moving vistas: Exploiting motion for describing scenes. In *Proc. CVPR*, 2010. 98, 99, 105, 107, 134

K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014a. 5, 6, 7, 11, 13, 19, 20, 26, 27, 30, 31, 34, 35, 36, 37, 38, 39, 42, 45, 49, 51, 52, 53, 57, 58, 60, 64, 68, 98, 99, 107, 115, 186, 187

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proc. ICLR*, 2014b. 6, 27, 34, 38, 42, 43, 44, 47, 100, 102, 103, 186

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015. 11, 15, 16, 17, 23

K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Workshop at International Conference on Learning Representations*, 2014. 74, 75, 76

N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using LSTMs. In *Proc. ICML*, 2015. 39, 68, 99

J. Stone. *Vision and Brain - How we perceive the world.* MIT press, 2012. 5

L. Sun, K. Jia, D.-Y. Yeung, and B. Shi. Human action recognition using factorized spatio-temporal convolutional networks. In *Proc. ICCV*, 2015. 27, 39

I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *Proc. ICML*, 2013. 12

C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014. 17, 74, 75

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proc. CVPR*, 2015a. 6, 11, 13, 15, 38, 39, 42, 61, 100, 102

C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. *arXiv preprint arXiv:1512.00567*, 2015b. 42, 44, 47, 52, 57, 101

C. Szegedy, S. Ioffe, and V. Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv preprint arXiv:1602.07261*, 2016. 42, 49, 101, 102

M. Szummer and R. Picard. Indoor-outdoor image classification. In *IEEE International Workshop on Content-Based Access of Image and Video Database*, 1998. 4

K. Tanaka and H.-A. Saito. Analysis of motion of the visual field by direction, expansion/contraction, and rotation cells clustered in the dorsal part of the medial superior temporal area of the macaque monkey. *Journal of neurophysiology*, 62(3):626–641, 1989. 73

G. W. Taylor and G. E. Hinton. Factored conditional restricted boltzmann machines for modeling motion style. In *Proc. ICML*, 2009. 59

G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler. Convolutional learning of spatio-temporal features. In *Proc. ECCV*, pages 140–153, 2010. 5, 32, 43, 57, 99

C. Theriault, N. Thome, and M. Cord. Dynamic scene classification: Learning motion descriptors with slow features analysis. In *Proc. CVPR*, 2013. 99, 106, 107, 185, 187

J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler. Efficient object localization using convolutional networks. In *Proc. CVPR*, 2015. 6, 42

A. Torralba, K. Murphy, and W. Freeman. Using the forest to see the trees: Object recognition in context. *Commun. ACM*, 53:107–114, 2010. 98

D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3D convolutional networks. In *Proc. ICCV*, 2015a. 5, 26, 32, 39, 43, 53, 57, 68, 98, 99, 100, 103, 106, 107, 187

D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proc. ICCV*, 2015b. 133

L. G. Ungerleider and R. Desimone. Cortical connections of visual area MT in the macaque. *Journal of Comparative Neurology*, 248(2):190–222, 1986. 72, 73

L. G. Ungerleider and J. V. Haxby. 'what'and 'where' in the human brain. *Current opinion in neurobiology*, 4(2):157–165, 1994. 20

A. Vailaya, M. Figueiredo, A. Jain, and H. Zhang. Image classification for content-based indexing. *TIP*, 10:117–130, 2001. 98

D. C. Van Essen and J. L. Gallant. Neural mechanisms of form and motion processing in the primate visual system. *Neuron*, 13(1):1–10, 1994. 21, 43

G. Varol, I. Laptev, and C. Schmid. Long-term temporal convolutions for action recognition. *arXiv:1604.04494*, 2016. 57, 68

A. Vedaldi and K. Lenc. MatConvNet – convolutional neural networks for MATLAB. In *Proceeding of the ACM Int. Conf. on Multimedia*, 2015. 11, 26, 49, 99

H. Wang and C. Schmid. Action recognition with improved trajectories. In *Proc. ICCV*, 2013. 5, 6, 39, 40, 53, 56, 68, 69, 100, 186, 187

H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Dense trajectories and motion boundary descriptors for action recognition. *IJCV*, pages 1–20, 2013. 5, 107, 186

J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *Proc. CVPR*, 2010. 5

L. Wang, W. Ouyang, X. Wang, and H. Lu. Visual tracking with fully convolutional networks. In *Proc. ICCV*, 2015a. 114, 115, 126

L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *Proc. CVPR*, 2015b. 39, 53

X. Wang, A. Farhadi, and A. Gupta. Actions ˜ transformations. In *Proc. CVPR*, 2016a. 44, 50, 51, 52, 53, 57, 63, 68

Y. Wang, S. Wang, J. Tang, N. O'Hare, Y. Chang, and B. Li. Hierarchical attention network for action recognition in videos. *arXiv preprint arXiv:1607.06416*, 2016b. 57, 68

L. Wiskott and T. J. Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural computation*, 14(4):715–770, 2002. 76, 77, 185

Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016a. 15

Y. Wu, S. Zhang, Y. Zhang, Y. Bengio, and R. Salakhutdinov. On multiplicative integration with recurrent neural networks. In *Proc. NIPS*, 2016b. 60

Z. Xu, Y. Yang, and A. G. Hauptmann. A discriminative CNN video representation for event detection. 2015. 98

J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *Proc. CVPR*, 2009. 5

J. Yang, H. Shuai, Z. Yu, R. Fan, Q. Ma, Q. Liu, and J. Deng. ILSVRC2016 object detection from video: Team NUIST. `http://image-net.org/challenges/talks/2016/Imagenet%202016%20VID.pptx`, 2016. 125, 126

J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson. Understanding neural networks through deep visualization. In *Deep Learning Workshop, International Conference on Machine Learning*, 2015. 74, 75, 76

C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime TV-L1 optical flow. In *Proc. DAGM*, pages 214–223, 2007. 34, 38, 50, 63, 69

M. D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012. 78

M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013. 17, 30, 42, 74

B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Object detectors emerge in deep scene cnns. In *Proc. ICLR*, 2014. 102

X. Zhou, K. Yu, T. Zhang, and T. Huang. Image classification using super-vector coding of local image descriptors. In *Proc. ECCV*, 2010. 5

W. Zhu, J. Hu, G. Sun, X. Cao, and Y. Qiao. A key volume mining deep framework for action recognition. In *Proc. CVPR*, 2016. 57, 68

X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei. Deep feature flow for video recognition. In *Proc. CVPR*, 2017. 116, 123, 124