Michael Opitz

Efficient Ensembles for Deep Learning

DOCTORAL THESIS

to achieve the university degree of
Doktor der technischen Wissenschaften

submitted to

Graz University of Technology

Supervisor

Univ.Prof. Dipl.Ing. Dr. Horst Bischof
Institute for Computer Graphics and Vision, Graz University of Technology

Prof. Dr. Björn Ommer
Heidelberg Collaboratory for Image Processing, Heidelberg University

Graz, Austria, Dec. 2020

"As humans, we have invented lots of useful kinds of lie. As well as lies-to-children ('as much as they can understand') there are lies-to-bosses ('as much as they need to know') lies-to-patients ('they won't worry about what they don't know') and, for all sorts of reasons, lies-to-ourselves. Lies-to-children is simply a prevalent and necessary kind of lie. Universities are very familiar with bright, qualified school-leavers who arrive and then go into shock on finding that biology or physics isn't quite what they've been taught so far. 'Yes, but you needed to understand that,' they are told, 'so that now we can tell you why it isn't exactly true.' Discworld teachers know this, and use it to demonstrate why universities are truly storehouses of knowledge: students arrive from school confident that they know very nearly everything, and they leave years later certain that they know practically nothing. Where did the knowledge go in the meantime? Into the university, of course, where it is carefully dried and stored."

— Terry Pratchett (1948 - 2015)

Abstract

In the last years, deep learning has been increasingly successful at addressing computer vision problems. These data-driven machine learning algorithms learn large Convolutional Neural Networks (CNNs) from annotated training data to tackle computer vision problems such as object categorization, object detection, image retrieval, semantic segmentation, object tracking, *etc*. However, one problem with such models is over-fitting. To overcome over-fitting and train highly accurate models, several works use ensembles consisting of several diverse *CNNs* to improve performance. While these approaches achieve highly accurate models, their runtime performance is too slow and their memory demand is too high for use in real-world applications.

One way to overcome this problem is to let learners in an ensemble share most parameters with each other. Specifically, we divide a *CNN* at the end into several learners. All learners share the same low- and mid-level feature representation. Consequently, such ensembles have low computational overhead, since we need to compute the shared feature representation only once for all learners. One of the main challenges for such ensembles is to make the learners diverse from each other. As all learners share the same feature representation and train on the same permutation of the training set, they end up highly correlated to each other. Unfortunately, highly correlated learners in an ensemble have no benefits, as they make the same prediction for each input sample.

To increase diversity in an ensemble, we present three contributions. First, we leverage spatial independence to train an ensemble of part-detectors for the problem of face detection. As we show in our experiments, such a *CNN* is more accurate compared to standard *CNNs* and robust to occlusions. Second, we introduce auxiliary loss functions for generic object classification *CNNs*. These loss functions make learners in a parameter-shared ensemble diverse from each other. Consequently, they complement each other well during test time. Third, we extend our generic approach to the problem of metric learning. To this end, we improve our auxiliary loss functions to make feature vectors diverse from each other. Further, we present a gradient boosting-based re-weighting scheme to

make learners focus on different training samples. As we show in our evaluations, such networks have favorable performance compared to standard *CNNs* without introducing computational overhead. Our experiments suggest that such parameter-shared ensemble networks can benefit a wide variety of different computer vision applications.

This work was partially supported by the Austrian Research Promotion Agency (FFG) projects *DARKNET* (858591), and *DIANGO* (840824) as well as the Christian Doppler Laboratory for Embedded Machine Learning (CDL-EML). The GeForce® Titan Xp used for parts of this research was donated by the NVIDIA® Corporation.

Kurzfassung

In den letzten Jahren wurden Deep Learning Verfahren immer erfolgreicher beim Lösen von Problemen im Bereich des Maschinellen Sehens. Diese datengesteuerten Machine Learning Algorithmen lernen große Convolutional Neural Networks (CNNs) von annotierten Trainingsdaten um Probleme des Maschinellen Sehens wie Objekt Kategorisierung, Objekt Lokalisierung, Bildsuche, Semantische Segmentierung, Objekt Tracking, *etc.* zu lösen. Ein Problem dieser Algorithmen ist Overfitting. Um Overfitting zu verringern und sehr genaue Modelle zu trainieren, verwenden manche Methoden ein Ensemble bestehend aus mehreren diversen *CNNs*. Solche Verfahren schaffen es zwar, sehr genaue Modelle zu generieren, sind aber sehr rechenaufwändig und zu speicherintensiv für die meisten Anwendungen.

Eine Möglichkeit dieses Problem zu verbessern ist es, dass sich die Lerner in einem Ensemble die meisten Parameter miteinander teilen. Konkret teilen wir ein *CNN* am Ende auf in mehrere Lerner. Diese Lerner teilen sich eine gemeinsame Low- und Mid-level Feature Repräsentation. Dadurch hat so ein Ensemble einen geringen Rechenaufwand, da die gemeinsame Feature Repräsentation nur einmal für das ganze Ensemble berechnet werden muss. Ein Problem an solchen Netzen ist, dass man ein Verfahren finden muss um die Lerner divers zueinander zu machen. Da sich alle Lerner die gleiche Feature Repräsentation miteinander teilen und alle auf die selbe Permutation der Trainingsdaten trainiert werden, sind diese stark korreliert zueinander. Unglücklicherweise bringen stark korrelierte Lerner in einem Ensemble keine Vorteile, da alle Lerne für ein Eingabebild die gleiche Ausgabe liefern.

Um dieses Problem zu lösen präsentieren wir in dieser Arbeit drei Methoden. Unsere erste Methode verwendet örtliche Unabhängigkeit um ein Ensemble aus diversen Detektoren zu trainieren, die sich auf Teile von Gesichtern spezialisieren. Ein solches Ensemble ist genauer als ein Standard-*CNN* und robust gegenüber Verdeckungen von Gesichtern. Unsere zweite Methode verwendet eine zusätzliche Verlustsfunktion für generische *CNNs*

für das Problem der Objekt Kategorisierung. Während dem Training macht diese Verlustsfunktion die Lerner in einem Ensemble mit gemeinsamer Feature Repräsentation divers voneinander. Dadurch ergänzen sich die Lerner gegenseitig besser zur Testzeit. Unsere dritte Methode erweitert unseren generischen Ansatz für das Lernen von Distanzmetriken. Dazu adaptieren wir unsere Verlustsfunktionen um Vektoren divers zueinander zu machen. Des Weiteren entwickeln wir ein Gradient Boosting basierendes Gewichtungsschema damit sich die Lerner während dem Training auf unterschiedliche Trainingsdaten fokusieren. In unseren Experimenten zeigen wir, dass solche Netze höhere Genauigkeit im Vergleich zu Standard *CNNs* erreichen, ohne zusätzlichen Rechenaufwand zu verursachen. Unsere Experimente suggerieren, dass solche Methoden für sehr viele Bereiche in der Computer Vision anwendbar sind.

Affidavit

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 indicated all material which has been quoted either literally or by content from the sources used.

The text document uploaded to TUGRAZonline is identical to the present doctoral thesis.

Date	Signature

Acknowledgments

I would like to take this opportunity to thank many people, who all supported me. Without them, this thesis would not have been possible. First and foremost I would like to thank Prof. Bischof for giving me the opportunity to work in this exciting research field and for his excellent management skills and scientific advice. Further, I would like to thank my second examiner, Prof. Ommer, for taking the time to read my thesis.

I also want to thank my colleagues in the Learning, Recognition & Surveillance (LRS) group at the Institute of Computer Graphics and Vision (ICG). Specifically, I would like to thank my two long-time office roommates Horst and Georg, which are "mit Abstand" the best colleagues I could wish for. Thank you for sharing your knowledge with me in countless discussions, mentoring me especially in the beginning in writing readable papers, and most importantly bearing my cursing ("sudern") when occasionally some of my experiments failed. Further, I would like to thank Peter, the former Boss, for his discussions with me and for writing the research proposal for the first research project I had the opportunity to work on. I also want to thank the other (former) members of the LRS group, *i.e.* Thomas, Martin, Paul, Samuel, Gernot, Patrick, Georg P, Georg K, David, Christian E, Christian F-R, Wei, Niloofar, Jakub, and Dusan for sharing research ideas and occasionally having after-work beers. Many thanks to the administrative staff of ICG, especially Nicole, Christina, Anna, Daniel, and Andreas for taking care of all the administrative and technical problems which showed up over the year. Further, I would like to thank Prof. Robert Sablatnig, Markus, Stefan, and Florian for getting me interested in computer vision in general during my time as a master student at the Technical University of Vienna.

Outside of work, I am very grateful for countless of happy hours in the last years with my friends from the (former) Bogside group, from the camping group, and from my childhood, who still put up with me. Finally, I want to thank my family, especially my brother Stefan and my parents Roswitha and Klemens, for supporting me throughout my educational career.

Thanks!

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Contributions . . . 6
- 1.3 Outline . . . 7

2 Preliminaries and Related Work **9**
- 2.1 Overview . . . 9
- 2.2 Preliminaries: Learning Theory for Ensembles . . . 11
- 2.3 Preliminaries: Traditional Ensemble Methods . . . 14
  - 2.3.1 Random Forest . . . 14
  - 2.3.2 Boosting . . . 15
  - 2.3.3 Explicit Diversity for Boosting . . . 16
  - 2.3.4 Gradient Boosting . . . 16
  - 2.3.5 On-line Gradient Boosting . . . 17
  - 2.3.6 Negative Correlation Learning . . . 18
- 2.4 Explicit Ensembles for Deep Learning . . . 19
  - 2.4.1 Diversity in Ensembles . . . 20
    - 2.4.1.1 Variation in Training Samples . . . 20
    - 2.4.1.2 Variation in Models . . . 21
    - 2.4.1.3 Variation in Features . . . 22
    - 2.4.1.4 Variation by Auxiliary Loss Functions . . . 22
  - 2.4.2 Parameter Sharing . . . 23
  - 2.4.3 Rapid Training of Ensembles . . . 24
  - 2.4.4 Distillation Approaches . . . 25
- 2.5 Implicit Ensembles with Dropout . . . 25
  - 2.5.1 Inference Gap . . . 26

2.5.2 Sampling Strategy . . . 27
2.5.3 Bayesian Inference . . . 28
2.5.4 Data-Dependent Regularization . . . 28
2.6 Face Detection . . . 29
2.6.1 Traditional Methods . . . 30
2.6.1.1 Boosting Based Methods . . . 30
2.6.1.2 Deformable Parts Methods . . . 30
2.6.2 *CNN* Based Methods . . . 30
2.7 Metric Learning . . . 32
2.7.1 Loss Functions . . . 34
2.7.2 Sampling Strategies . . . 37
2.7.3 Regularization . . . 38
2.7.4 Ensembles . . . 38
2.8 Summary . . . 39

3 Spatial Independence for Ensemble Diversity 41
3.1 Motivation . . . 41
3.2 Grid Loss for *CNNs* . . . 43
3.2.1 Neural Network Architecture . . . 43
3.2.2 Grid Loss Layer . . . 44
3.2.2.1 Regularization Effect . . . 47
3.2.2.2 Deeply Supervised Nets . . . 47
3.2.3 Face Localization Refinement . . . 48
3.3 Evaluation . . . 49
3.3.1 Grid Loss Benefits . . . 50
3.3.2 Spatial Learner Size . . . 52
3.3.3 Weighting Parameter . . . 52
3.3.4 Robustness to Occlusions . . . 53
3.3.5 Effect on Correlation of Features . . . 55
3.3.6 Training Set Size . . . 55
3.3.7 Ellipse Regressor and Layer Skipping . . . 56
3.3.8 Comparison to the State-of-the-Art . . . 57
3.3.9 Qualitative Comparison . . . 58
3.3.10 Computational Efficiency . . . 60
3.4 Summary . . . 60

4 Diversity Loss Functions for Ensembles 65
4.1 Motivation . . . 65
4.2 Efficient Ensembles for Object Categorization . . . 67
4.3 Efficient Ensembles for Metric Learning . . . 69
4.3.1 Metric Learning for Deep Learning . . . 70

4.3.2 On-line Gradient Boosting Efficient Ensembles 72
4.4 Auxiliary Loss Functions 76
4.4.1 DivLoss 76
4.4.2 Activation Loss 77
4.4.3 Adversarial Loss 78
4.5 Loss Function on Multiple Hidden Layers 80
4.6 Image Categorization Experiments 81
4.6.1 Comparison of Diversity Loss Functions 82
4.6.2 Splitting at Shallower Layers 83
4.6.3 Accuracy and Diversity of Learners 84
4.6.4 Comparison with Other Ensemble Methods 85
4.6.5 Efficient ResNet Ensembles 86
4.7 Metric Learning Experiments 88
4.7.1 Strength and Correlation 90
4.7.2 Loss Functions 90
4.7.3 Number of Learners 92
4.7.4 Embedding Sizes 92
4.7.5 Impact of Initialization 93
4.7.6 Impact of Auxiliary Loss Functions 94
4.7.7 Evaluation of the Regularization Parameter 95
4.7.8 Comparison with the State-of-the-Art 97
4.8 Summary 103

5 Conclusion **105**
5.1 Summary 105
5.2 Future Work 107

A List of Acronyms **109**

B List of Publications **111**

Bibliography **119**

List of Figures

1.1 Images[a] are represented as numeric matrix (left). This matrix changes dramatically if the object in the image is occluded, deformed, or subject to illumination changes. Computer vision approaches address this problem with machine learning. Traditional approaches use handcrafted features and feed them into a machine learning classifier (right). More recent methods use deep learning to learn a classifier and a feature representation simultaneously (bottom). 2

1.2 Illustration of several computer vision benchmarks, in which ensemble approaches are successful. From top to bottom: ImageNet [186]: Object Classification, MS-COCO [134]: Instance Segmenation, MS-COCO [134]: Panoptic Segmentation, Waymo Open Dataset [210]: 3D Detection. 4

1.3 Standard ensembles (top) have a large computational overhead per learner (blue, red, ...). By sharing parameters (bottom) we need to compute the shared feature representation (white) only once for the full ensemble (blue, red, green, yellow, ...). Consequently, the per-learner overhead is comparably low. 5

2.1 A categorization of ensemble approaches as implicit and explicit. 10

2.2 A categorization of ensembles for *CNNs* by the parameters individual learners share with each other. 11

2.3 A categorization of methods introducing diversity in *CNN* ensembles. . . . 11

2.4 Training neural networks with dropout during training time. 26

2.5 Overview of important and recent works for face detection. 29

2.6 Overview of deep metric learning systems. 33

2.7 Overview of recent works in deep metric learning. 35

3.1 Schematic overview of (a) standard global loss and (b) the proposed grid loss with an illustrative example on FDDB. 43

3.2 Overview of our method: our detection CNN builds upon Aggregate Channel Features (ACF) [41]. For each window, after pooling, we apply successive convolution filters to the input channels. To distinguish faces from non-faces we use pose-specific classifiers. Instead of minimizing the loss over the last full convolution map, we divide the map into small blocks and minimize a loss function on each of these blocks independently. 43

3.3 Boxplot of 2×2 part activations on the positive training set (i.e. by dividing the detection template into non-overlapping parts, as in Fig. 3.2). Activations trained by regular loss functions can have parts with negative median response. We mark parts whose 25% percentile is smaller than 0 (red) and parts which have significant positive median activations compared to other parts (yellow). 45

3.4 Illustration of our post-hoc regression network. 48

3.5 ROC curve of logistic, hinge, grid + logistic, grid + hinge, grid hidden + hinge and grid hidden + logistic loss on Face Detection Data Set and Benchmark (FDDB). 51

3.6 Comparison of different block sizes on *FDDB*. 53

3.7 Evaluation of our weighting parameter λ. 54

3.8 Impact of different training sub-sets of the positive training set on False Positives Per Image (FPPI) at 0.1 *FPPI* using the hinge loss (Hinge), hinge loss on hidden layers (Hinge-hidden) and our grid loss (Grid) and grid loss on hidden layers (Grid-hidden) 56

3.9 Evaluation of our ellipse regressors on FDDB (continuous score). 58

3.10 Evaluation of our ellipse regressors on FDDB (discrete score) with and without layer skipping. 59

3.11 Discrete evaluation on the FDDB [93] dataset. 60

3.12 Evaluation on the PASCAL Faces [251] dataset. 61

3.13 Our method outperforms state-of-the-art methods on AFW [283]. 62

3.14 Qualitative comparison between a parameter shared spatial independent *CNN* ensemble trained with grid loss and a standard *CNN*. 63

3.15 Comparison of our detection templates learned with our grid loss with our detection templates learned with a standard loss function. 64

4.1 Our ensemble architecture divides the network at the end into several learners. For each learner we add a classification head. We use an auxiliary loss function to make these learners diverse from each other. 68

4.2 BIER divides a large embedding into an ensemble of several smaller embeddings. During training we reweight the training set for successive learners in the ensemble with the negative gradient of the loss function. During test-time we concatenate the individual embeddings of all learners into a single embedding vector. 71
4.3 Illustration of triplet loss, contrastive loss (for negative samples) and binomial deviance loss (for negative samples) and their gradients. Triplet and contrastive loss have a non-continuous gradient, whereas binomial deviance has a continuous gradient. 72
4.4 We divide the embedding (shown as dashed layer) of a metric *CNN* into several weak learners and cast training them as online gradient boosting problem. Each learner iteratively reweights samples according to the gradient of the loss function. Training a metric *CNN* this way encourages successive learners to focus on different samples than previous learners and consequently reduces correlation between learners and their feature representation. 73
4.5 Our Activation Loss mutually suppresses neurons in hidden layers of different learners. 78
4.6 Our adversarial loss function learns auxiliary regressors $g_{(j,i)}$ from the hidden representation of learner j to learner i. To make the hidden representations diverse from each other, we insert a gradient reversal layer between the auxiliary regressor layers (blue) and the network layers (red). 80
4.7 We can apply our method on top of any hidden layer in a neural network. . 81
4.8 Speed vs. accuracy trade-off on the CIFAR-10 dataset. We highlight networks with different widening factor $k \in \{1, 2, 3\}$ in red, green and yellow, respectively. 87
4.9 Speed vs. accuracy trade-off on the CIFAR-100 dataset. We highlight networks with different widening factor $k \in \{1, 2, 3\}$ in red, green and yellow, respectively. 88
4.10 Speed vs accuracy trade-off on the SVHN dataset. We highlight networks with different widening factor $k \in \{1, 2, 3\}$ in red, green and yellow, respectively. 88
4.11 Evaluation of different embedding sizes and group sizes on the CUB-200-2011 [229] dataset. 93
4.12 Evaluation of $\lambda_{\texttt{div}}$ on CUB-200-2011 [229]. 96
4.13 Visualization of the progress in retrieval accuracy over time. We color code the backbone network architecture. 97

List of Tables

3.1 Statistics of our benchmark datasets for face detection. 51

3.2 True positive rates of logistic (L), hinge (H), grid + logistic (G+L), grid + hinge (G+H), grid hidden + hinge (G-h+H) and grid hidden + logistic (G-h+L) loss functions on *FDDB* at a false positive (FP) count of 50, 100, 284 and 500. **Best** and second best results are highlighted. 52

3.3 Comparison of different block sizes on *FDDB*. 52

3.4 True Positive Rate on Caltech Occluded Faces in the Wild (COFW) Heavily Occluded (*COFW*-HO) and Less Occluded (LO) subsets of a grid loss detector (G) and a hinge loss detector (H). 54

3.5 Grid loss reduces correlation in feature maps. 55

3.6 Impact of training on a sub-set (i.e. 0.75 - 0.01) of the positive training set on *FDDB* at 0.1 *FPPI* using the hinge loss (H), hinge loss on hidden layers (H-h) and our grid loss (G) and grid loss on hidden layers (G-h). 57

3.7 Continuous evaluation of the two proposed ellipse loss functions: Numerical PASCAL VOC overlap (NUM) and Sum of Squares Error (SSE) on FDDB. 57

3.8 Effect of numerical loss (NUM), *SSE* loss (SSE) and no ellipse regressor (w/o) applied densely (D) on all pyramid levels or skipping (S) layers on FDDB. 58

4.1 Evaluation of different auxiliary loss functions on the CIFAR-10 dataset. . . 83

4.2 Comparison of standard ensembles and TreeNets [119] with our diversity loss on CIFAR-10 dataset. Each ensemble consists of 4 learners. 84

4.3 Diversity and average accuracy of learners in an ensemble. 85

4.4 Comparison to state-of-the-art ensembling methods on the CIFAR-10 dataset. We split the network at the last hidden layer into 4 learners. . . . 86

4.5 Detailled comparison between our method, ONE [284], EnsembleNet [124] and the combination of both methods. We split the networks after the Pool2 layer into 4 learners. 86
4.6 Speed vs. accuracy trade-off on the CIFAR-10 dataset with a ResNet widening factor of $k \in \{1, 2, 4\}$ and $l \in \{1, 2, 4, 8\}$ learners respectively. 87
4.7 Speed vs. accuracy trade-off on the CIFAR-100 dataset with a ResNet widening factor of $k \in \{1, 2, 4\}$ and $l \in \{1, 2, 4\}$ learners respectively. 88
4.8 Speed vs. accuracy trade-off on the SVHN dataset with a ResNet widening factor of $k \in \{1, 2, 4\}$ and $l \in \{1, 2, 4, 8\}$ learners respectively. 88
4.9 Evaluation of classifier (Clf.) and feature correlation on CUB-200-2011 [229]. **Best** results are highlighted. 91
4.10 Evaluation of loss functions on CUB-200-2011 [229]. 91
4.11 Evaluation of group sizes on CUB-200-2011 [229]. 92
4.12 Evaluation of embedding size on CUB-200-2011 [229]. 93
4.13 Evaluation of Glorot, orthogonal and our Activation Loss and Adversarial Loss initialization method on CUB-200-2011 [229]. 94
4.14 Comparison of auxiliary loss functions on CUB [229]. Our adversarial loss function significantly improves accuracy over our baseline (Boosting Independent Embeddings Robustly (BIER) [167]) and enables higher learning rates and faster convergence. 95
4.15 Impact of auxiliary loss functions on strength and correlation. 96
4.16 Summary of dataset statistics in our experiments. 97
4.17 Comparison with the state-of-the-art on the CUB-200-2011 [229] dataset. **Our** results are highlighted. 98
4.18 Comparison with the state-of-the-art on the Cars-196 [109] dataset. 99
4.19 Comparison with the state-of-the-art on the Stanford Online Products [163] dataset. 100
4.20 Comparison with the state-of-the-art on the In-Shop Clothes Retrieval [141] dataset. 101
4.21 Comparison with the state-of-the-art on VehicleID [136]. 103

1 Introduction

"'Forty-two!' yelled Loonquawl. 'Is that all you've got to show for seven and a half million years' work?' 'I checked it very thoroughly,' said the computer, 'and that quite definitely is the answer. I think the problem, to be quite honest with you, is that you've never actually known what the question is.' "

— Douglas Noel Adams

Contents

1.1 Motivation **1**
1.2 Contributions **6**
1.3 Outline **7**

1.1 Motivation

The human perception system is one of the most valuable things we possess. It enables us to see, hear, smell, taste, touch, *etc.* Consequently, it allows us to perceive and interact with our environment. Mimicking several of these skills for computers is crucial for a large number of real-world applications such as autonomous systems (*e.g.* autonomous driving, intelligent robots, *etc.*), surveillance applications, automated medical procedures, virtual assistants (*e.g.* Alexa, Siri, *etc.*), *etc.* Therefore, there is an increasingly large research effort to enable human perception, interaction, and reasoning skills for computers, *e.g.* [18, 23, 75, 112, 121, 202]. In this work, we focus on computer vision (*i.e.* the visual aspect of perception), which is one of the most important parts in these applications.

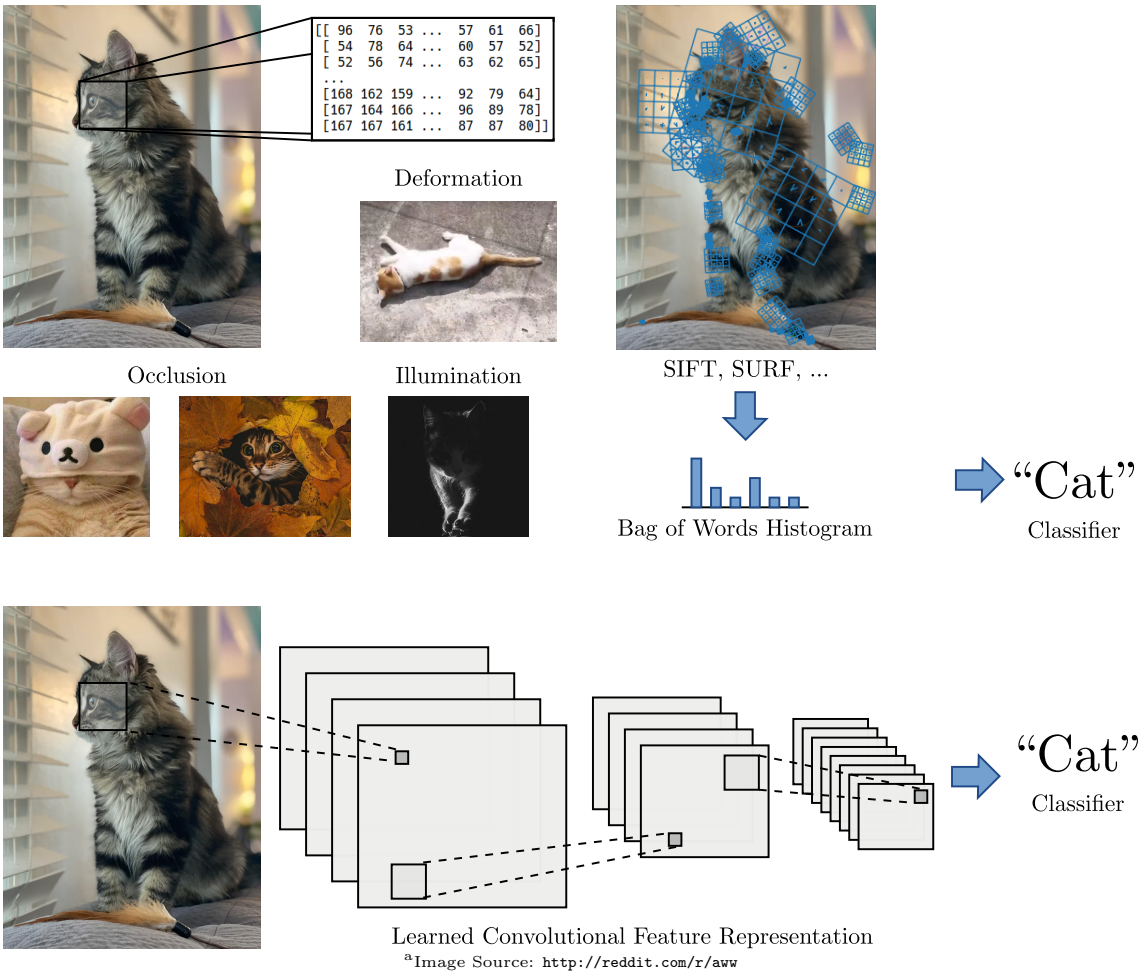


Figure 1.1: Images[a] are represented as numeric matrix (left). This matrix changes dramatically if the object in the image is occluded, deformed, or subject to illumination changes. Computer vision approaches address this problem with machine learning. Traditional approaches use handcrafted features and feed them into a machine learning classifier (right). More recent methods use deep learning to learn a classifier and a feature representation simultaneously (bottom).

Handcrafting such algorithms in this area is typically too complex and therefore not possible. Consider the cat image in Figure 1.1. To detect the cat in an image, a handcrafted algorithm needs to manipulate the pixels, *i.e.* some numeric matrix, in a specific way. Unfortunately, this numeric matrix is subject to large changes if a different cat is visible in the image, other objects occlude the cat, the illumination changes, the cat moves, *etc.* Further, if we want to detect other objects, such as dogs or cars, such algorithms might need a complete redesign. Consequently, algorithms in this line of work typically rely on machine learning techniques to address this problem. These learning algorithms use a dataset in a training phase to learn a model. This trained model should then be able to make decisions on unseen test data (*e.g.* detect cats in new images). To train these

algorithms, computer vision methods extract a feature representation of the input image. Consequently, a lot of research effort was spent on developing better feature representations for visual perception, *e.g.* [7, 36, 41, 143, 227].

In the last years, due to advances in compute and large publicly available datasets, a specific family of machine learning algorithms, *i.e.* deep learning, has achieved great success in mimicking human perception in general *e.g.* [18, 23, 75, 112, 121, 202] and computer vision in particular, *e.g.* [75, 112, 132, 176]. In computer vision, these methods use large Convolutional Neural Networks (CNNs) to simultaneously learn the feature representation and the classifier from large datasets. More specifically, these neural networks are parametrized non-linear functions $f(\boldsymbol{x};\boldsymbol{\theta})$. They take an input image $\boldsymbol{x}$ and map it to an output. The shape of the output is application-specific and might be *e.g.* a vector representing the class label (*e.g.* cat, dog, *etc.*) of the input image. During the training phase, *CNNs* learn the parameters $\boldsymbol{\theta}$, which are typically coefficients of convolution filters.

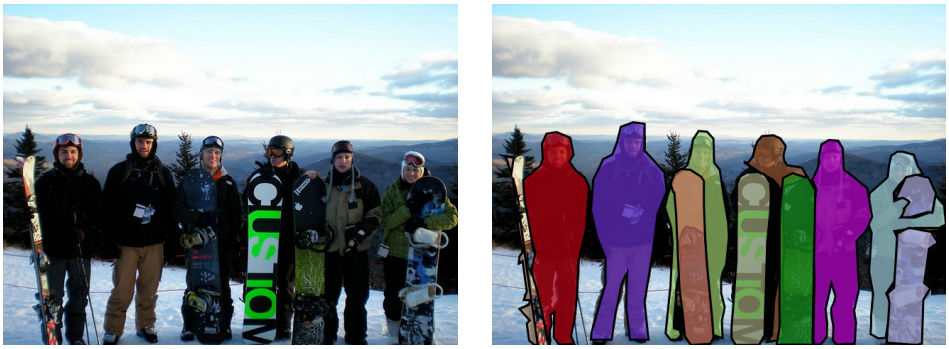
However, one of the main problems in machine learning with powerful non-linear models is over-fitting, *e.g.* [13]. As these neural networks tend to have a huge number of parameters, they can model the training data too well. Consequently, their performance on unseen data suffers. One popular way to reduce over-fitting and improve the accuracy of machine learning methods is training ensembles of diverse models. During test-time, these models then can complement each other. Several of the learners in an ensemble might fail on a particular test sample. But as long as the majority of learners makes correct predictions, the ensemble decision will be correct.

In computer vision, ensembles are popular for achieving highly accurate models. To assess the performance of the state-of-the-art for a wide variety of computer vision problems, there are annual challenges, such as the ImageNet challenge [186], the MS COCO challenge [134], the Waymo Open Dataset challenge [210], *etc.* (see Figure 1.2). Nearly every winning entry in these annual challenges consists of an ensemble of several large neural networks, which improves accuracy compared to using only a single model.

While standard deep learning ensemble approaches are successful for winning these benchmark challenges, their practical use is limited due to their computational expense during test-time. In *CNN*-based ensembles the individual learners already consist of a computationally expensive model. Even though parallel hardware can alleviate some of these problems during inference time, there is a large per-learner overhead of cost in terms of hardware-demand and power-consumption. In recent years the community moved towards increasingly computationally more efficient models, *e.g.* [75, 84, 214], to enable real-world applications which need to run on low-energy devices, such as the Edge-Tensor Processing Unit (TPU), embedded Graphics Processing Units (GPUs), *etc.* Building large ensembles of such models for these hardware platforms is therefore computationally wasteful.

In contrast, many traditional computer vision approaches, which are suitable for real-time applications, use the benefits of ensembles. They achieved state-of-the-art accuracy at the time of publication, *e.g.* [41, 148, 149, 227]. These approaches are based on Boost-

“cat” “dog”



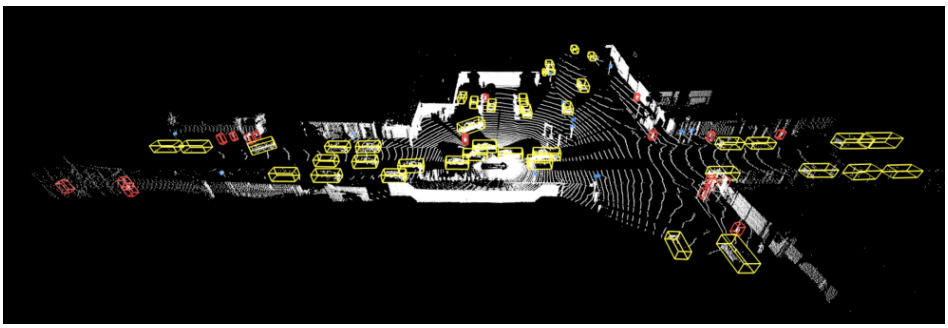


Figure 1.2: Illustration of several computer vision benchmarks, in which ensemble approaches are successful. From top to bottom: ImageNet [186]: Object Classification, MS-COCO [134]: Instance Segmenation, MS-COCO [134]: Panoptic Segmentation, Waymo Open Dataset [210]: 3D Detection.

ing [50] or Random Forest [15] and use very small and cheap learners with low model capacity (*i.e.* weak learners), such as decision-stumps or decision-trees. In contrast to standard *CNN*-based ensembles, the computational per-learner overhead during test-time and training-time is low. Further, especially Boosting-based approaches typically use

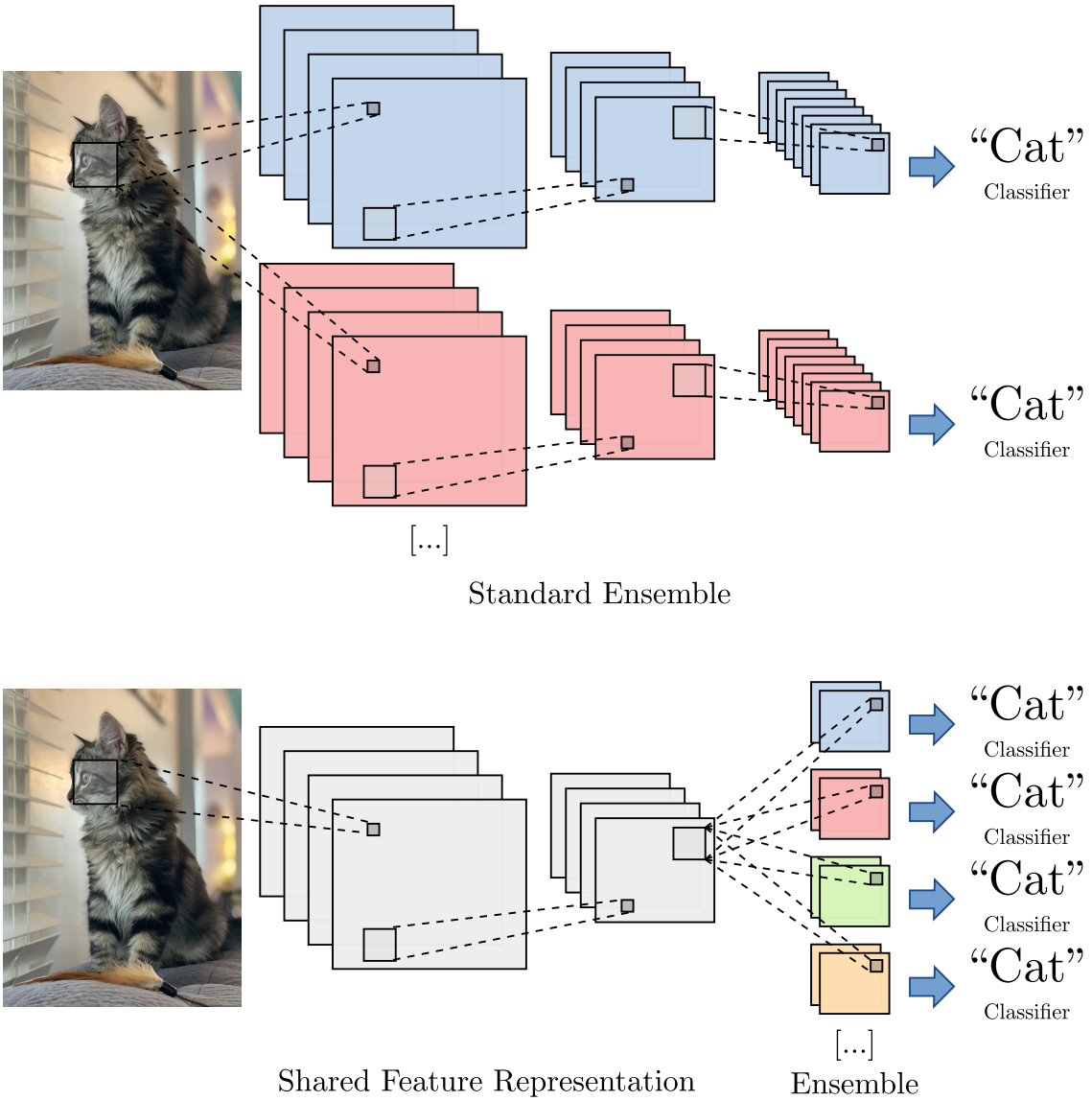


Figure 1.3: Standard ensembles (top) have a large computational overhead per learner (blue, red, ...). By sharing parameters (bottom) we need to compute the shared feature representation (white) only once for the full ensemble (blue, red, green, yellow, ...). Consequently, the per-learner overhead is comparably low.

Therefore, to enable real-world applications to use the benefits of *CNN*-based ensembles, we need to reduce the computational per-learner overhead. One way to achieve this is to let learners share parameters with each other, as we illustrate in Figure 1.3. Specifi-

cally, we split the *CNN* at the end into several learners. The ensemble consists of a shared feature representation (illustrated in white) and several learners (illustrated in blue, red, green, yellow, ...). We need to compute the shared feature representation only once for all layers. Further, we can carefully balance the computational the per-learner overhead of *CNN* ensembles vs the ensemble accuracy.

The main problem of this approach is that it is challenging to introduce diversity in such ensembles. Unfortunately, by sharing most parameters with each other, and training all learners on the same (permutation of) the training set, the individual learners will be highly correlated with each other. For a specific input sample, every learner will make the same prediction. Consequently, there are no benefits of such ensembles. This leads us to our research question, which we address in this thesis, *i.e.* "How can we increase the diversity in parameter shared ensembles?"

1.2 Contributions

In this thesis, we present three contributions to increase the diversity of such ensembles for a variety of different computer vision problems.

Diversity by Spatial Independence: First, in Chapter 3, we introduce a face detection ensemble which introduces diversity by training learners which focus on different spatial parts of the face. These part-detectors share a common feature representation with each other. Specifically, we divide a fully-convolutional face detector at the last hidden layer into several non-overlapping spatial blocks. We set the network up so that the receptive fields of these blocks are small. They are only large enough to cover specific prototypical regions in faces, such as eyes, mouth, nose, *etc.* By optimizing a discriminative loss for each of our learners, we force the network to develop discriminative features for each of these regions separately. In contrast, standard *CNNs* are "lazy" during training. As soon as they develop a feature, which is good enough to classify every training sample into face vs non-face, they have no inclination to develop further features. For example, if the *CNN* develops a feature for the mouth, which can distinguish every face from the background in the training set, it does not develop further features for the eye or nose. Our ensemble overcomes this problem, resulting in more diverse feature representations which are more robust to face occlusions.

Diversity by Auxiliary Loss Functions for Classification: Second, in Chapter 4.2, we improve our method to make it more generally applicable to generic object categorization problems. Specifically, our first contribution requires a specific *CNN* architecture and pose-information of faces during training. To address this problem, we propose diversity encouraging auxiliary loss function to make learners in parameter-shared ensembles diverse from each other. In contrast to our first contribution, this method does not have any specific requirements on the receptive field of the network. We show in our experiments on

small patch datasets, that such ensembles perform favorably compared to another generic ensemble approach, *i.e.* dropout, and competitive to current state-of-the-art approaches on small network architectures. Further, our experiments with the popular ResNet [75] architecture suggest that such ensembles are more pareto-optimal (*i.e.* achieve higher accuracy with the same computational budget) compared to widened ResNet architectures.

Diversity by Re-Weighting and Auxiliary Loss Functions for Metric Learning: Third, we extend our generic method in Chapter 4.2 to large *CNNs* to the application of metric learning for image retrieval. To this end, we improve our auxiliary loss functions to make hidden layer representations of neural networks diverse from each other. Further, in addition to our auxiliary loss functions, we introduce diversity by re-weighting training samples for each learner in the ensemble according to a gradient-boosting algorithm. In our experiments, we show that large *CNNs* for metric learning benefit from such ensembles. Our method achieved state-of-the-art performance at the time of publication. Several successive works use a similar ensemble structure and improve our approach with different diversity encouraging methods.

1.3 Outline

The remainder of this thesis is structured as follows. First, in Chapter 2, we provide an overview of ensemble methods in the context of deep learning for computer vision. Further, we review the preliminaries of learning theory for ensembles and traditional ensemble methods in computer vision. As some of our contributions focus on the application face-detection and metric-learning, we review related work in these areas. Second, in Chapter 3, we discuss our spatial independent ensemble and validate its effectiveness in our experimental evaluation. Third, in Chapter 4 we extend our method to generic *CNN* architectures. We introduce diversity encouraging auxiliary loss functions. Further, we present our gradient-boosting-based metric learning extension. We evaluate our approach on object categorization benchmarks and metric learning for image retrieval benchmarks. Finally, in Chapter 5, we conclude our thesis and discuss possible future research directions.

2

Preliminaries and Related Work

"The current state of knowledge can be summarized thus: in the beginning there was nothing, which exploded."

— Terry Pratchett

Contents

2.1	**Overview**	**9**
2.2	**Preliminaries: Learning Theory for Ensembles**	**11**
2.3	**Preliminaries: Traditional Ensemble Methods**	**14**
2.4	**Explicit Ensembles for Deep Learning**	**19**
2.5	**Implicit Ensembles with Dropout**	**25**
2.6	**Face Detection**	**29**
2.7	**Metric Learning**	**32**
2.8	**Summary**	**39**

2.1 Overview

In this chapter, we first discuss preliminaries for ensembles in machine learning and then give a brief overview of related work in this field. Specifically, in the context of deep learning, we classify ensemble approaches as implicit and explicit (see Figure 2.1). Implicit approaches were pioneered by dropout [208]. During training time these approaches sample certain sub-substructures of the network. Consequently, during training time they implicitly optimize a single network out of a larger number of networks. In contrast, explicit approaches model the ensemble in the network architecture explicitly. They either

use multiple Convolutional Neural Network (CNN) models or multi-head models with a shared low-level feature representation.

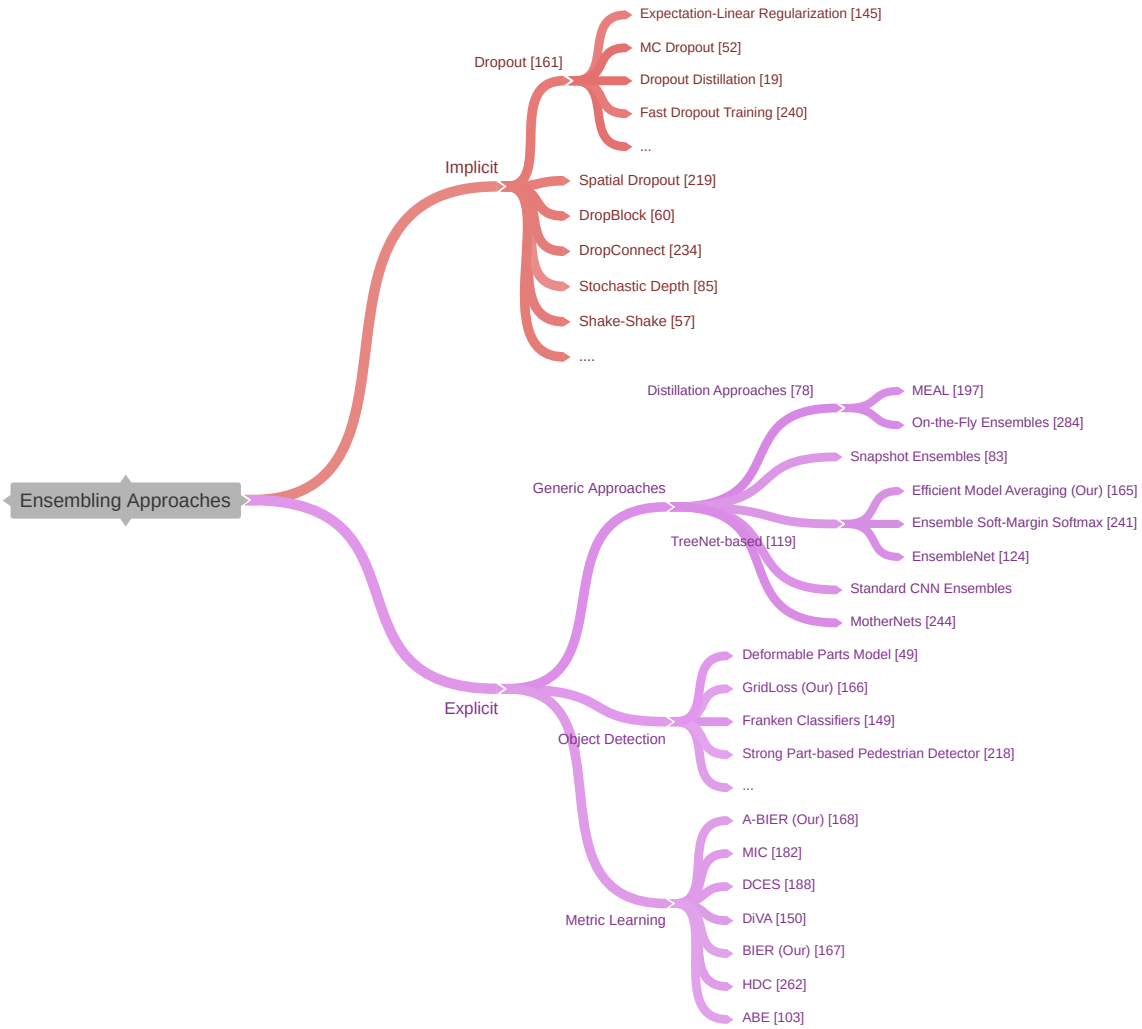


Figure 2.1: A categorization of ensemble approaches as implicit and explicit.

We further characterize ensemble approaches by the number of parameters they share (see Figure 2.2). In implicit ensemble approaches, such as dropout, the individual learners of the ensemble share all their parameters with each other. In contrast, explicit modeling of ensembles allows using different parameters for individual learners, allowing them to balance the speed and accuracy of the ensemble.

Finally, as ensemble accuracy relies on diverse learners, there are several orthogonal methods to increase diversity in ensembles (see Figure 2.3). These methods are not limited to deep learning approaches. Bagging and boosting based approaches typically introduce diversity by sampling or re-weighting the training set. Next, some approaches such as

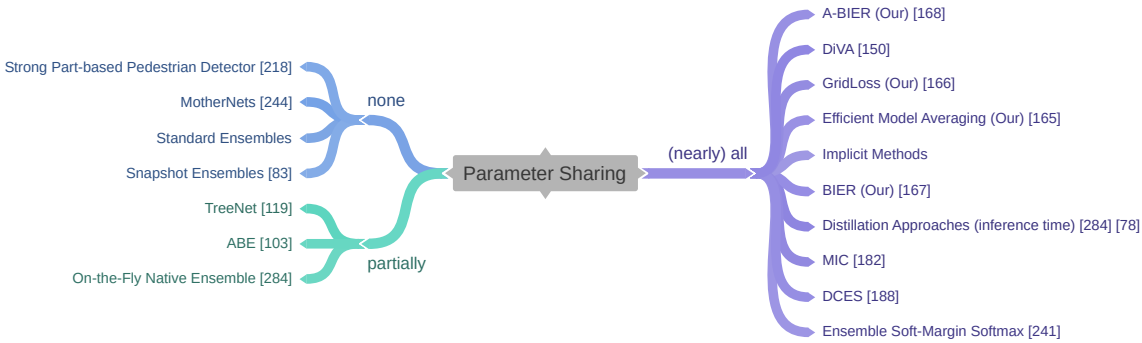


Figure 2.2: A categorization of ensembles for *CNNs* by the parameters individual learners share with each other.

Random Forest, diversify learners by sub-sampling features. Part-based methods, *e.g.* [49], spatially constrain the learners and constrain them to learn specific object parts. Further, especially deep learning based ensemble methods typically try to introduce variation in the model (*e.g.* architecture, initialization, ...) to make learners more diverse to each other.

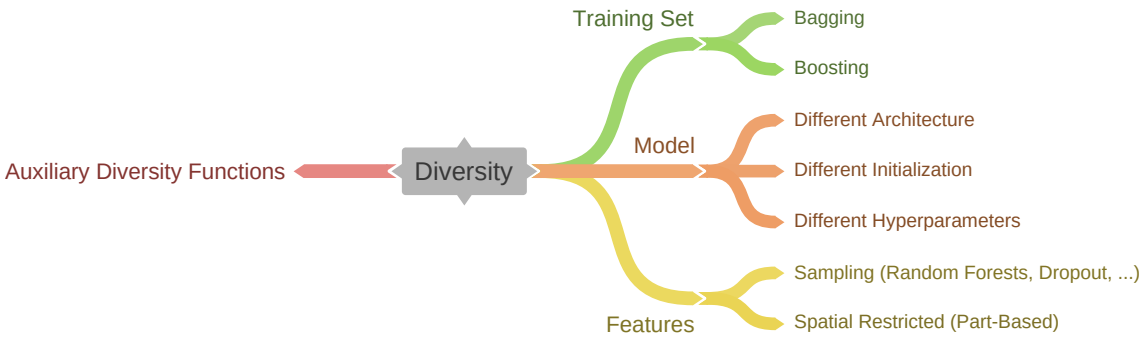


Figure 2.3: A categorization of methods introducing diversity in *CNN* ensembles.

In the remainder of this chapter, we first summarize preliminaries about learning theory for ensembles methods (Section 2.2). Further, we review several traditional ensemble methods, which are popular in computer vision (Section 2.3). Then, we discuss more closely related work on ensembles for deep learning (Section 2.4). Finally, as we address problems of face detection and metric learning for image retrieval, we give an overview of recent works in these areas (Section 2.6 and Section 2.7, respectively).

2.2 Preliminaries: Learning Theory for Ensembles

Ensembles are a popular method to reduce over-fitting in machine learning algorithms for computer vision, *e.g.* [41, 193, 227, 284]. The objective of these methods is to combine

the predictions of multiple machine learning algorithms into a single prediction. Subsequently, this combination typically achieves higher accuracy compared to the individual predictions [164]. However, to show any benefits, an ensemble must consist of a set of diverse classifiers. If the classifiers in an ensemble are highly correlated, *e.g.* make the same prediction for each input sample, the combination of these classifiers cannot show any benefits in terms of accuracy.

The bias-variance-covariance decomposition is a theoretical result which formalizes this effect for the Mean Square Error (MSE) loss [17, 222]. It generalizes the bias-variance decomposition. Formally, let $f(\cdot;\boldsymbol{\theta})$ denote the output of a classifier (*e.g.* a neural network), which is parametrized by $\boldsymbol{\theta}$. $f(\cdot;\boldsymbol{\theta})$ is optimized by a learning algorithm to predict a target y from an input sample $\boldsymbol{x}$. During learning we want to find the parameters $\boldsymbol{\theta}$ which minimize the following expected error:

$$E_{\boldsymbol{x},y}[f(\boldsymbol{x};\boldsymbol{\theta}) - y)^2] = \int (f(\boldsymbol{x};\boldsymbol{\theta}) - y)^2 p(\boldsymbol{x}, y) d\boldsymbol{x} dy, \tag{2.1}$$

where $p(\boldsymbol{x}, y)$ denotes joint true data distribution. We typically do not have access to the true joint data distribution. Therefore, we approximate this integral with a finite sum from samples over a dataset $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_N, y_N)\}$.

$$\approx \frac{1}{N} \sum_{i=1}^{N} (f(\boldsymbol{x}_i;\boldsymbol{\theta}) - y_i)^2. \tag{2.2}$$

However, D is only a finite sample of the true data distribution $p(\boldsymbol{x}, y)$. Therefore, obtaining zero error on the above loss function in Equation (2.2) typically does not yield a classifier which achieves zero generalization error in Equation (2.1). To formally express this generalization error, we look at the expected test *MSE* under different samplings of D. This is the average test *MSE* we would obtain if we repeatedly estimate $f(\cdot)$ using a large number of training sets D. To avoid further notational clutter, we omit $\boldsymbol{\theta}$ in the following, *i.e.* $f(\boldsymbol{x}; D) = f(\boldsymbol{x}; \boldsymbol{\theta}, D)$. Then, we can estimate the generalization error for the *MSE* as:

$$E_{\boldsymbol{x},y,D}[\{f(\boldsymbol{x}; D) - y\}^2], \tag{2.3}$$

where $f(\boldsymbol{x}; D)$ denotes the classifier, which we obtain from our learning algorithm on the dataset D. We sample $\boldsymbol{x}$, y from the true data distribution $p(\boldsymbol{x}, y)$ and take the expectation of the *MSE* w.r.t. to $\boldsymbol{x}$, y and D. This expected *MSE* loss decomposes into two terms, *i.e.* the squared bias and variance, as follows:

$$\begin{aligned} E_{\boldsymbol{x},y,D}[\{f(\boldsymbol{x}; D) - y\}^2] &= E_{\boldsymbol{x},y,D}[\{(f(\boldsymbol{x}; D) - E_D[f(\boldsymbol{x}; D)]) + (E_D[f(\boldsymbol{x}; D)] - y)\}^2] \\ &= E_{\boldsymbol{x},y,D}[\{(f(\boldsymbol{x}; D) - E_D[f(\boldsymbol{x}; D)])\}^2 + \{E_D[f(\boldsymbol{x}; D)] - y\}^2 \\ &\quad +2 \cdot (f(\boldsymbol{x}; D) - E_D[f(\boldsymbol{x}; D)]) \cdot (E_D[f(\boldsymbol{x}; D)] - y)]. \end{aligned}$$

The third term is 0, as:

$$
\begin{aligned}
&E_{\boldsymbol{x},y,D}[2 \cdot (f(\boldsymbol{x};D) - E_D[f(\boldsymbol{x};D)]) \cdot (E_D[f(\boldsymbol{x};D)] - y)] = \\
&E_{\boldsymbol{x},y,D}[2 \cdot (f(\boldsymbol{x};D) \cdot E_D[f(\boldsymbol{x};D)] - f(\boldsymbol{x};D) \cdot y - E_D[f(\boldsymbol{x};D)]^2 + y \cdot E_D[f(\boldsymbol{x};D)])] = \\
&E_{\boldsymbol{x},y}[2 \cdot (E_D[f(\boldsymbol{x};D) \cdot E_D[f(\boldsymbol{x};D)]] - E_D[f(\boldsymbol{x};D) \cdot y] - E_D[E_D[f(\boldsymbol{x};D)]^2] + E_D[y \cdot E_D[f(\boldsymbol{x};D)]])] = \\
&E_{\boldsymbol{x},y}[2 \cdot (E_D[f(\boldsymbol{x};D)]^2 - E_D[f(\boldsymbol{x};D)] \cdot y - E_D[f(\boldsymbol{x};D)]^2 + y \cdot E_D[f(\boldsymbol{x};D)])] = \\
&E_{\boldsymbol{x},y}[0] = 0.
\end{aligned}
\tag{2.4}
$$

Therefore, the expected test *MSE* in Equation (2.3) decomposes into the following two terms

$$
\begin{aligned}
&E_{\boldsymbol{x},y,D}[\{f(\boldsymbol{x};D) - y\}^2] = \\
&\underbrace{E_{\boldsymbol{x},y,D}[\{E_D[f(\boldsymbol{x};D)] - y\}^2]}_{\text{bias}^2} + \underbrace{E_{\boldsymbol{x},y,D}[\{(f(\boldsymbol{x};D) - E_D[f(\boldsymbol{x};D)])\}^2]}_{\text{variance}}.
\end{aligned}
\tag{2.5}
$$

The squared *bias* term indicates the inherent error of the model. It is high if the model is not able to fit the data well. The *variance* term indicates how specialized our individual models are. It can be an indicator of overfitting.

Ueda and Nakano [222] further generalize this decomposition to ensemble methods. Formally, let $\bar{f}(\boldsymbol{x};D) = \sum_{m=1}^{M} f_m(\boldsymbol{x};D)$ denote the ensemble prediction, which is the average of M classifier predictions. We can re-write the bias for the whole ensemble as the average of the individual learner bias terms:

$$
\begin{aligned}
E_{\boldsymbol{x},y,D}\left[\left\{E_D[\bar{f}(\boldsymbol{x};D)] - y\right\}^2\right] &= E_{\boldsymbol{x},y,D}\left[\left\{E_D[\tfrac{1}{M}\textstyle\sum_m f_m(\boldsymbol{x};D)] - y\right\}^2\right] \\
&= E_{\boldsymbol{x},y,D}\left[\left\{\tfrac{1}{M}\textstyle\sum_m (E_D[f_m(\boldsymbol{x};D)] - y)\right\}^2\right].
\end{aligned}
\tag{2.6}
$$

Further, the variance term decomposes into a covariance and a variance term:

$$
\begin{aligned}
&E_{\boldsymbol{x},y,D}\left[\left\{(\bar{f}(\boldsymbol{x};D) - E_D[\bar{f}(\boldsymbol{x};D)])\right\}^2\right] \\
&= E_{\boldsymbol{x},y,D}\left[\left\{(\tfrac{1}{M}\textstyle\sum_m f_m(\boldsymbol{x};D) - E_D[\tfrac{1}{M}\sum_m f_m(\boldsymbol{x};D)])\right\}^2\right] \\
&= E_{\boldsymbol{x},y,D}\left[\left\{\tfrac{1}{M}\textstyle\sum_m (f_m(\boldsymbol{x};D) - E_D[f_m(\boldsymbol{x};D)])\right\}^2\right] \\
&= E_{\boldsymbol{x},y,D}\left[\tfrac{1}{M^2}\left\{\textstyle\sum_m (f_m(\boldsymbol{x};D) - E_D[f_m(\boldsymbol{x};D)])\right\}^2\right] \\
&= E_{\boldsymbol{x},y,D}\left[\tfrac{1}{M^2}\left(\textstyle\sum_m (f_m(\boldsymbol{x};D) - E_D[f_m(\boldsymbol{x};D)])\right) \cdot \left(\textstyle\sum_{m'} (f_{m'}(\boldsymbol{x};D) - E_D[f_{m'}(\boldsymbol{x};D)])\right)\right] \\
&= E_{\boldsymbol{x},y,D}\left[\tfrac{1}{M^2}\left(\textstyle\sum_m \sum_{m'} (f_m(\boldsymbol{x};D) - E_D[f_m(\boldsymbol{x};D)]) \cdot (f_{m'}(\boldsymbol{x};D) - E_D[f_{m'}(\boldsymbol{x};D)])\right)\right] \\
&= E_{\boldsymbol{x},y,D}\left[\tfrac{1}{M^2}\left(\textstyle\sum_m \sum_{m' \neq m} (f_m(\boldsymbol{x};D) - E_D[f_m(\boldsymbol{x};D)]) \cdot (f_{m'}(\boldsymbol{x};D) - E_D[f_{m'}(\boldsymbol{x};D)])\right)\right. \\
&\left. + \tfrac{1}{M^2}\textstyle\sum_m (f_m(\boldsymbol{x};D) - E_D[f_m(\boldsymbol{x};D)])^2\right].
\end{aligned}
\tag{2.7}
$$

If we define the average bias, variance and covariance over the M classifiers as

$$\begin{array}{lcl} \overline{variance} & = & \frac{1}{M}\sum_m variance_{f_m} \\ \overline{covariance} & = & \frac{1}{M(M-1)}\sum_m \sum_{m' \neq m} covariance_{f_m, f_{m'}} \\ \overline{bias} & = & \frac{1}{M}\sum_m bias_{f_m}, \end{array} \tag{2.8}$$

we can re-write the bias-variance decomposition for ensembles as:

$$\begin{array}{l} E_{\boldsymbol{x},y,D}\left[\left\{E_D[\bar{f}(\boldsymbol{x};D)] - y\right\}^2\right] + E_{\boldsymbol{x},y,D}\left[\left\{(\bar{f}(\boldsymbol{x};D) - E_D[\bar{f}(\boldsymbol{x};D)])\right\}^2\right] = \\ \overline{bias}^2 + (1 - \frac{1}{M})\overline{covariance} + \frac{1}{M}\overline{variance}. \end{array} \tag{2.9}$$

Consequently, the generalization performance of an ensemble depends also on the covariance between the learners. Intuitively, learners need to be diverse to each other to improve the expected *MSE* loss. Over the past decades, traditional machine learning ensemble approaches have developed several techniques to increase diversity in ensembles. Therefore, we discuss several popular techniques in the following section.

2.3 Preliminaries: Traditional Ensemble Methods

In this section, we give an overview of traditional ensemble methods, *i.e.* ensemble methods used for "shallow" machine learning models. Traditionally, in computer vision, popular ensemble approaches are built upon computationally cheap classifiers such as decision stumps, trees, histograms, *etc.* The most prominent approaches are Random Forest [15] (Section 2.3.1) and AdaBoost [50] (Section 2.3.2). As one of our works uses online gradient boosting, we review the core concepts of these algorithms. Negative Correlation Learning (NCL) [139] tries to negatively correlate learners in a shallow neural network ensemble with an auxiliary loss function. This method is closely related to several of our contributions. Therefore, we review it in detail in Section 2.3.6.

2.3.1 Random Forest

Random Forest [15] is an ensemble of Decision Trees and uses random feature selection and bootstrap aggregation (Bagging) [14] to introduce diversity in an ensemble. Bagging increases the diversity in ensembles by sub-sampling the training set with replacement for each learner of the ensemble (*i.e.* for each tree). As this trains each learner on a different subset of the dataset, the resulting predictions will be diverse from each other. Random feature selection further reduces the correlation between learners by constraining learners to a different random subset of the features during training. Random Forests typically sub-sample features and training samples at each internal split-node of each of their trees individually.

The amount of sub-sampled features or number of sub-sampled training samples is typically a hyper-parameter which trades off diversity with model strength. For example,

Algorithm 1: Description of the Discrete AdaBoost algorithm.

Data: $(x_1, y_1), \ldots, (x_N, y_N)$, where $x_i \in \mathbb{R}^d, y_i \in \{-1, +1\}$.
Initialize $D_1(i) = \frac{1}{N}$ for $i = 1, \ldots, N$
for $m = 1, \ldots, M$ **do**
  Train weak learner $f_m : \mathbb{R}^d \mapsto \{-1, +1\}$, which minimizes the weighted misclassification error: $\epsilon_m = \sum_{i=1, f_m(\boldsymbol{x}_i) \neq y_i}^{N} D_m(i)$
  Set $\alpha_m = \frac{1}{2} \ln(\frac{1-\epsilon_m}{\epsilon_m})$
  Update $D_{m+1}(i) = \frac{D_i(i) e^{-\alpha_m y_i f_m(\boldsymbol{x}_i)}}{Z_m}$, where Z_m is a normalization factor, so that D_{m+1} is a distribution.
The final ensemble is given by:
$F(\boldsymbol{x}) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m f_m(\boldsymbol{x})\right)$

significantly limiting the features in the split-nodes of the trees leads to underfitting of the individual learners in the ensemble. Consequently, the bias term in the bias-variance-covariance trade-off will be high, as the individual learners will not achieve high enough accuracy to benefit ensemble performance.

2.3.2 Boosting

In contrast to Random Forest, AdaBoost [50] re-weights the dataset for successive learners in the ensemble to compensate for the errors of the previous learners. Subsequently, re-weighting also introduces diversity, as especially the first learners in the ensemble focus on different examples compared to successive learners [200]. Further, methods which apply boosting for computer vision problems also use random feature selection and bagging, as this increases diversity in a boosted ensemble, *e.g.* [41].

More formally, AdaBoost greedily optimizes a global exponential loss function, *i.e.* $\frac{1}{N} \sum_{i=1}^{N} e^{-y_i \cdot F(\boldsymbol{x}_i)}$, where $F(\cdot)$ is the ensemble prediction, $\boldsymbol{x}_i$ the i-th training sample and $y_i \in \{-1, +1\}$ the corresponding ground-truth label. The ensemble consists of several "weak learners" $f_1, f_2, \ldots, f_M$. Its prediction is a weighted combination of these learners $F(\boldsymbol{x}) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m f_m(\boldsymbol{x})\right)$. The algorithm learns weighting factors α_m and the weak learners $f_m(\cdot)$ iteratively, as we illustrate in Algorithm 1.

Several traditional computer vision methods in the field of recognition use boosting, such as real-time object detection, *e.g.* [41, 227], pose estimation, *e.g.* [97], object tracking, *e.g.* [68], *etc.*

There are several variations of this algorithm. They all follow the same structure of AdaBoost (*i.e.* learn a classifier on a weighted dataset, compute the error, re-weight the dataset) but change specific parts of it, such as using a different re-weighting scheme for the dataset. For example, there are variations which extend it to multi-class classification, *e.g.* [50, 74], use the probability outputs of weak learners, *e.g.* [50], try to improve robustness against outliers, *e.g.* [120, 122, 147], adapt it to semi-supervised learning, *e.g.* [146],

transfer-learning, *e.g.* [35], and adapt it to the online learning setting, *e.g.* [68, 169].

More closely related to our work are methods which try to explicitly make individual learners in a boosted ensemble diverse to each other, which we review in Section 2.3.3. Another closely related work is Gradient Boosting [51] (Section 2.3.4), which allows minimizing arbitrary (sub-)differentiable functions in the boosting framework. More specifically, our work relates to On-line Gradient Boosting, which extends standard Gradient Boosting to the online learning setting (Section 2.3.5)

2.3.3 Explicit Diversity for Boosting

Some approaches try to explicitly encourage learners to be diverse to each other in a boosted ensemble. They either enforce this during training by altering the re-weighting scheme [239] or post-process a trained ensemble [4].

DivBoosting [4] post-processes a learned AdaBoost ensemble with the Coalition-based Ensemble Design (CED) [3] algorithm. *CED* is a greedy forward selection algorithm to construct an ensemble out of a set of base learners using a diversity measure. It iteratively measures the diversity contribution of the base learners to the current ensemble on a validation set. It then greedily adds the k most promising learners to the current ensemble. *CED* iterates process until the diversity contribution is below a threshold.

In contrast, AdaBoost.NC [239] builds upon negative correlation learning [139]. It directly incorporates diversity during training by introducing an ambiguity term *amb*. This term measures the deviation of the individual learners from the ensemble prediction, *i.e.* $amb = \frac{1}{2M} | \sum_{m=1}^{M} (F(\boldsymbol{x}) - f_m(\boldsymbol{x})) |$ for $f_m : \mathbb{R}^d \mapsto \{-1, +1\}$ and $F : \mathbb{R}^d \mapsto \{-1, +1\}$. It is small if the learner predictions agree with the ensemble prediction, and large otherwise. During re-weighting, AdaBoost.NC multiplies the weights of the training samples by $(1 - amb)^{\lambda}$, where λ is a hyper-parameter. Therefore, successive learners focus more on misclassified samples and on samples where the ambiguity is small.

Our work similarly builds upon ideas of negative correlation learning. However, in contrast to AdaBoost.NC we directly minimize a differentiable loss function between our learners, which are (parts of) a deep *CNN*.

2.3.4 Gradient Boosting

Similar to AdaBoost, Gradient Boosting [51] iteratively builds an ensemble one learner at a time. The main idea is to optimize a global loss function by performing gradient descent in function space. More specifically, during training at iteration m, Gradient Boosting tries to iteratively minimize the following loss function for N training samples $(\boldsymbol{x}_i, y_i)$ w.r.t. to the current to be optimized learner $f_m(\cdot)$:

$$\arg\min_{f_m(\cdot)} \sum_{i=1}^{N} \ell(F_{m-1}(\boldsymbol{x}_i) + f_m(\boldsymbol{x}_i), y_i), \tag{2.10}$$

where $F_{m-1}(\boldsymbol{x})$ is the ensemble prediction until step $m-1$.

Directly optimizing this loss function is too hard. Therefore, Gradient Boosting optimizes the loss function by performing gradient descent in function space. Formally, at step m it updates the ensemble as follows:

$$F_m(\boldsymbol{x}) = F_{m-1}(\boldsymbol{x}) - \nu_m \sum_{i=1}^{N} \nabla_{F_{m-1}} \ell(F_{m-1}(\boldsymbol{x}_i), y_i), \tag{2.11}$$

where ν_m is a shrinkage factor and is typically set constant for all learners $m = 1, \dots M$.

Therefore, the new learner $f_m(\cdot)$ has to make predictions which are proportional to the negative gradient of the loss function. To this end, Gradient Boosting computes so-called pseudo-residuals $r_{im} = -\nabla_{F_{m-1}} \ell(F_{m-1}(\boldsymbol{x}_i), y_i)$, which are the negative gradients of the loss function for all training samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$. Subsequently, Gradient Boosting trains the new learner $f_m(\cdot)$ to predict these pseudo-residuals by augmenting the training set to $\{(\boldsymbol{x}_i, r_{im})\}_{i=1}^N$.

Similar to boosting, we can interpret the negative gradient as a weight for training samples, *i.e.* $w_{im} = |r_{im}|$ [51, 120, 193]. Depending on the loss, samples with high loss typically also have high gradients. Therefore, successive learners in the ensemble tend to focus more on misclassified samples.

2.3.5 On-line Gradient Boosting

The original Random Forest, AdaBoost and Gradient Boosting algorithms assume an offline training setting. In contrast, online algorithms update the model incrementally, typically one sample at a time. Subsequently, online algorithms alleviate the need to hold the entire dataset in memory. Consequently, online algorithms can train on larger datasets compared to offline algorithms. Further, online learning enables some computer vision applications, such as object tracking, which try adapting a classifier to a specific video or scene without storing any additional data. There are several extensions which address these problems of Gradient Boosting. These works are similar to online Boosting methods and adapt Gradient Boosting to the online learning setting, *e.g.* [11, 12, 120].

In contrast to their offline versions, online Gradient Boosting trains a fixed number of online learners. The main requirement of these learners is that they are online trainable. Therefore, online Gradient Boosting can train learners such as Naive Bayes classifiers or gradient-based learning algorithms. The latter group consists of algorithms such as linear Support Vector Machines (SVMs), Neural Networks, *etc*. We summarize a simple online Gradient Boosting algorithm [120] in Algorithm 2.

In some of our works [167, 168], we use several heads on top of a *CNN* as weak learners. They all share the same underlying feature representation. Further, we train them all on the same training set. Unfortunately, naïvely training such an ensemble yields highly correlated classifiers. To address this problem, we optimize our model with batch online

gradient boosting. By re-weighting the training set for each learners, the diversity of the heads increases.

Algorithm 2: Illustration of a simplified version of the online Gradient Boosting Algorithm of Leistner *et al.* [120].

Data: A training sample $(\boldsymbol{x}_i, y_i)$
Set $F_0(\boldsymbol{x}_i) = 0$
Set the initial weight $w_i^{(1)} = -\ell'(0)$
for $m = 1 \dots M$ **do**
  Train m-th weak learner $f_m(\cdot)$ with sample $(\boldsymbol{x}_i, y_i)$ and weight $w_i^{(m)}$.
  Set $F_m(\boldsymbol{x}_i) = F_{m-1}(\boldsymbol{x}_i) + f_m(\boldsymbol{x}_i)$.
  Set the weight $w_i^{(m+1)} = -\ell'(y_i F_m(\boldsymbol{x}_i))$.
Output the final model: $F_M(\boldsymbol{x})$

2.3.6 Negative Correlation Learning

NCL [139] tries to make neural network regression ensembles diverse to each other by explicitly negatively correlating the individual learners from the ensemble prediction during training time. To this end, *NCL* trains the ensemble jointly and uses an additional penalty in the loss function to negatively correlate each learner from the ensemble prediction.

More formally, for regression problems, we typically minimize the squared error loss function

$$\frac{1}{2} \sum_i \left(f_m(\boldsymbol{x}_i) - y_i \right)^2 , \tag{2.12}$$

where $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ denotes training dataset with samples $\boldsymbol{x}_i$ and regression targets y_i. $f_m(\cdot)$ denotes the prediction of the m-th neural network in the ensemble. Typically, the ensemble prediction is the arithmetic mean of the learner prediction and defined as

$$F(\boldsymbol{x}) = \frac{1}{M} \sum_{m=1}^{M} f_m(\boldsymbol{x}). \tag{2.13}$$

NCL adds an additional regularization term to the loss function

$$\frac{1}{2} \sum_i \left(\left(f_m(\boldsymbol{x}_i) - y_i \right)^2 + \lambda \cdot p_m(\boldsymbol{x}_i) \right) , \tag{2.14}$$

where λ is a hyper-parameter which balances the penalty with the standard L_2 loss. $p_m(\boldsymbol{x}_n)$ negatively correlated the learner $f_m(\cdot)$ from the ensemble prediction as follows:

$$p_m(\boldsymbol{x}) = \left(f_m(\boldsymbol{x}) - F(\boldsymbol{x}) \right) \sum_{n \neq m} \left(f_n(\boldsymbol{x}) - F(\boldsymbol{x}) \right). \tag{2.15}$$

If the ensemble prediction is the arithmetic mean of the learners (as defined in Equa-

tion (2.13)), this further simplifies to the following overall loss function:

$$\frac{1}{2}\sum_i \left((f_m(\boldsymbol{x}_i) - y_i)^2 - \lambda(f_m(\boldsymbol{x}_i) - F(\boldsymbol{x}_i))^2\right). \tag{2.16}$$

As we see, the learners should on the one hand predict responses which are close to the targets y_i. On the other hand, they should predict responses which are different from the ensemble prediction $F(\cdot)$. Naturally, this is a trade-off. Setting the hyper-parameter λ too low yields highly correlated learners $f_m(\cdot)$, which show no benefit in an ensemble. Choosing λ too high impairs the model strength, *i.e.* the accuracy of the individual learners $f_m(\cdot)$. Subsequently, every model has to do different predictions from the mean. Therefore, most models will fail to accurately predict the targets y_i. Interestingly, by optimizing this loss function on the training set, *NCL* also achieves diverse learners on the test set, improving the ensemble performance.

As *NCL* trains the full ensemble jointly, it minimizes the penalty term $p_m(\cdot)$ of the loss function with standard Stochastic Gradient Descent (SGD). The main disadvantage of this method is, that during training all learners need to be stored in memory at the same time. Therefore, ensemble training consumes significantly more memory, which is a limiting factor with modern Graphics Processing Unit (GPU) hardware.

NCL is related to our work as we also model the individual learners in an ensemble explicitly and make them explicitly diverse to each other on the training set. Even though this technique has been proposed two decades ago, it did not receive much research attention in the computer vision community recently, presumably due to its computational expense.

2.4 Explicit Ensembles for Deep Learning

In the context of deep learning, explicit ensembles are especially popular in object categorization, detection, and instance segmentation in annual challenges such as the ImageNet [186] challenge or the COCO [134] challenge, where the main objective is to build highly accurate models without regard to speed. More specifically, every winner of the COCO challenge from 2015 to 2018 for object detection and instance segmentation uses the benefits of ensemble learning to improve accuracy.

Besides improving accuracy, ensembles can be beneficial for predicting uncertainty information about a sample. If many learners disagree on a particular input, uncertainty is high. In the context of computer vision, uncertainty information is beneficial for Active Learning [9], Reinforcement Learning [52], detecting novel classes [114], *etc.*

We categorize different explicit ensemble approaches by the diversity method they use and the number of parameters individual learners share with each other in the ensemble. Our works [165–168] focus on using the benefits of ensembles to improve a single *CNN* model. We share most parameters of the individual learners with each other. To make the

learners diverse from each other, we make use of spatial independence [166], boosting [167, 168], and auxiliary loss functions [165, 168].

Therefore, we review in the following sections methods which introduce diversity in *CNN* ensembles (Section 2.4.1) and give an overview of several methods which exploit the benefits of parameter sharing among models (Section 2.4.2). Further, as one of our goals is to speed up ensemble training by parameter sharing, we also review several methods which focus on rapid ensemble training (Section 2.4.3). Finally, as model distillation can also exploit the benefits of ensembles by distilling the knowledge of an ensemble into a student network, we give a brief outline of these methods (Section 2.4.4).

2.4.1 Diversity in Ensembles

As only diverse learners yield any benefits in ensembles, different methods have been proposed to increase diversity for learners in ensembles. We categorize these diversity encouraging models in methods which use training samples (Section 2.4.1.1), different model architectures (Section 2.4.1.2), different features (Section 2.4.1.3), and auxiliary loss functions (Section 2.4.1.4) to increase diversity in ensembles. These methods are not mutually exclusive and ensemble methods typically can employ several of these approaches to increase diversity.

2.4.1.1 Variation in Training Samples

Introducing diversity by varying the training set for each learner is a popular strategy which Bagging [15] and Boosting [50] use to introduce diversity. Bagging sub-samples different samples for individual learners. Boosting re-weights individual samples for different learners. In contrast to these traditional ensemble methods, most recent *CNN* ensemble methods typically do not employ "hard" data sub-sampling, presumably because *CNNs* benefit from large datasets to learn a diverse and discriminative feature representation. There is a trade-off between reducing the dataset size of a specific learner and increasing the diversity of a learner. Training a *CNN* with a small number of samples typically yields a less rich and less diverse feature representation compared to training a *CNN* with all training samples.

Recent work [281] tries to carefully balance this trade-off by assigning different models in the ensemble different samples. To this end, they model the assignment as a bipartite graph between models and training samples. They further introduce two constraints into this assignment problem. These constraints limit the number of samples per model and the number of models assigned to a specific sample.

Some recent approaches, *i.e.* [119, 188, 244], try to overcome this problem, by implicitly or explicitly sharing parameters between learners. Specifically, Wasay *et al.* [244] pre-trains the learners on the full datasets. Consequently, during fine-tuning and sub-sampling, learners can benefit from the rich feature representation learned from all training samples.

Other approaches [119, 188], similar to ours, use parameter-sharing in a multi-head network architecture. Consequently, a large part of the feature representation can benefit from all training samples. Subsequently, the network can develop a rich and diverse feature representation. The individual learners then can specialize on a certain part of the training set. TreeNets [119] assign samples in a mini-batch to learners based on the loss of the learners on this sample. The method then assigns to a given training sample the k learners with the lowest loss. This typically yields to a split of the dataset per object category, as learners tend to specialize on certain categories.

In the context of deep metric learning, several works employ sampling and weighting to increase diversity. Most of these works, *i.e.* [167, 168, 188], use parameter-sharing to allow the *CNN* feature representation to benefit from all training samples. Our works [167, 168] use online gradient boosting to re-weight the training samples for learners which all share the same feature representation [167]. As our learners use parameter-sharing and our weighting is a "soft" weighting, our feature representation can benefit from all training samples. Similarly, Sanakoyeu *et al.* [188] also leverage the benefits of parameter sharing. They use a two-step divide and conquer training approach to cluster the training set and train individual learners of a parameter-shared ensemble to be discriminative for the specific cluster. In the end, they fine-tune the ensemble on the full dataset.

Deep Randomized Ensembles for Metric Learning (DREML) [248] does not employ feature sharing, but randomly groups class labels of samples for each learner into different small groups. Consequently, all learners train on the full dataset, but focus on separating different categories from each other. However, the main disadvantage of this approach is, that it needs to train several independent learners and therefore cannot enjoy the computational benefits of parameter sharing.

2.4.1.2 Variation in Models

Another way to make learners in a *CNN* ensemble diverse from each other is to vary the models of the individual learners. This includes the *CNN* architecture (*i.e.* number of layers, size of convolutions, *etc.*), the random initialization, and different hyper-parameters of the models, *e.g.* [75]. The learners are then typically trained on the full training dataset (with a different random permutation), without sub-sampling for the individual learners. The reason for this is presumably due to the benefits of large datasets for deep neural networks compared to traditional, shallow approaches such as Random Forests, which do not need to learn the feature representation of the model.

Current research [9, 158] suggests that difference in initialization contributes more to variation of network models compared to bagging the training set or random permutations of the training set. Interestingly, variance due to initialization increases as the depth of the network increases [158]. Consequently, as training of *CNNs* is a non-convex optimization problem, due to different initializations, these networks converge to a different local minimum of the loss function, which introduces diversity in the ensemble.

2.4.1.3 Variation in Features

One popular way to increase diversity in an ensemble is to restrict different learners to different features of the network, *e.g.* [15]. Deep learning approaches typically apply feature sampling on a hidden layer of the network. This approach is popular for implicit ensemble methods such as dropout [208] and its extensions (Section 2.5).

Explicit methods can employ attention as a form of soft feature selection for individual learners to introduce diversity [103]. Each learner focuses on different features, which introduce diversity among learners. A separate "expert" network predicts this attention mask. Consequently, the expert decides which learner should be active for a given training sample.

Several multi-modality (*e.g.* RGB-D, NIR, *etc.*) *CNN* approaches, *e.g.* [30, 70, 142, 226], can be interpreted as an ensemble, where learners operate on separate modalities. Traditionally, these approaches perform late- or deep-fusion of these multiple learners. Typically, they have one copy of the same network per input modality, and then fuse their outputs or an intermediate feature representation. These copies are traditionally trained separately and then fine-tuned together to build an ensemble which consists of two learners operating on RGB and depth input, respectively. Recent methods, *e.g.* [226], fuse modalities on multiple layers and use auxiliary loss functions to make the individual modality networks discriminative alone. In contrast to these methods, we do not explore the benefits of multi-modal inputs in our works.

Another way to introduce feature diversity are part-based models, *e.g.* [49, 59, 149, 250, 283]. Traditionally, these models have been successful in object detection. Learners in these models focus on different object parts, *i.e.* spatial sub-regions of the input feature representation. For example, such parts might correspond to the torso or legs of a person.

Consequently, if one part is missing due to occlusion, the ensemble can recover. Modern *CNN* based detectors are typically holistic detectors and have large receptive fields, *e.g.* [132, 133, 176]. Consequently, dividing them into part based detectors is a challenging problem.

There are several attempts to integrate object parts into the *CNN* based detection frameworks, *e.g.* [218, 255, 270, 271]. Early works in this area use several large *CNNs* to model each object part, *i.e.* [218]. Similarly, one of our works is a part-based *CNN* detector [166]. However, our detector shares all hidden layer features with all part detectors. Consequently, our method is computationally more efficient. Successive works address this problem with additional occlusion or part-location supervision, *e.g.* [255, 271], in combination with attention [270].

2.4.1.4 Variation by Auxiliary Loss Functions

Introducing diversity in an ensemble by an auxiliary loss function has not received much research attention in the recent years, presumably because during training the ensemble has to be jointly trained. Consequently, each learner has to be stored concurrently in

memory, which makes these approaches computationally demanding.

For neural networks, this approach has been pioneered by *NCL* [139], which tries to make regression ensembles diverse from each other. It negatively correlates the prediction from the learner to the ensemble prediction during training.

The benefits of this technique have also been used to improve boosting ensembles [239] and gradient boosting ensembles [235].

With the availability of more computational resources and improvements in model architecture (*i.e.* parameter sharing between learners), these approaches are also applicable to modern deep neural networks. These methods have been applied to simple object categorization [165, 241], crowd counting [199], metric learning [150, 168, 182], *etc.* Similar to dropout, these methods typically employ parameter sharing between the individual learners. Typically, only the last hidden layer has different parameters, which results in a computationally fast training and inference time compared to standard ensembles.

Several of our contributions, *i.e.* [165, 167, 168], fall into this category. Compared to standard *NCL* we employ parameter sharing between learners and use a different auxiliary loss function on the hidden layer of the network to make the learners diverse from each other.

Successive works in this category use ideas from of self-supervised learning to provide different auxiliary supervision signals to learners. They use pseudo labels from clustering, *e.g.* [182, 188], or instance-augmentation [150] to train learners on different supervision signals.

2.4.2 Parameter Sharing

In the context of deep learning, one of the main disadvantages of ensembles is the computational expense. During test time inference has to be done for each member of the ensemble. One way to overcome this problem is to share parameters between learners of the ensemble.

The most popular approach which employs parameter sharing is dropout [208], which we review in Section 2.5. Dropout randomly omits neurons in the hidden layer during training time from a large neural network. Consequently, we can interpret these "thinned" networks as learners from a larger ensemble, which all share parameters with each other.

Another way to address this issue is to explicitly divide or replicate the network at one hidden layer. These networks have multiple heads which correspond to the individual learners in an ensemble. Compared to dropout, these types of networks have not received much research attention.

We can interpret some coarse-to-fine approaches, *e.g.* [21, 266], as an ensemble, where individual learners refine the predictions of previous ones. This is useful for pose estimation and localization tasks. Some of these approaches also use parameter sharing. For example, Cascade R-CNN [21] uses a cascade of multiple object detection heads to refine detections, yielding more accurate bounding boxes around objects. Lower level features for Cascade

R-CNN are shared between learners. To introduce diversity, successive heads operate on inputs of previous heads to refine the object localization. In contrast to these approaches, our work typically does not use the benefits of coarse-to-fine refinement and is therefore more generally applicable.

Most similar to our work is the unpublished work of Lee *et al*. [119], which investigates the effect of splitting networks for object categorization tasks. They found that jointly training an ensemble with multiple heads actually harms performance, as it reduces diversity. To overcome this problem, they assign a random subset of categories to different classification heads, which introduces diversity in their ensemble. However, this approach has the disadvantage that gradient updates for learners become unstable, as due to the label assignment individual learners receive only a few training samples per batch. Compared to their work, we explore auxiliary loss functions [165, 168], and soft re-weighting of samples [167, 168] to introduce diversity. Successive works further combine the benefits of parameter-sharing with attention [103], pseudo labels obtained by clustering [182, 189], and self-supervised auxiliary loss functions [150]. However, while most of these approaches focus on the problem of metric learning, we also show that our contributions extend to object categorization problems.

Further, recently Zhu *et al*. [284] independently also propose a multi-head architecture, where they use knowledge distillation to convert the multiple heads back into a single head model. They use a gating network to decide which of their heads is responsible for classifying a sample. Compared to their work, we focus more on how to introduce diversity in such a multi-head setup. We argue that more diverse heads could be beneficial for distillation, consequently improving the accuracy of the student network.

Recently, Li *et al*. [124] propose a multi-head ensemble architecture during training and show that such an ensemble setup can improve the accuracy of ResNet-50 and ResNet-101 on the ImageNet dataset. They just rely on random initialization to increase diversity and might benefit from additional methods to increase diversity between learners.

2.4.3 Rapid Training of Ensembles

As training several large *CNNs* is computationally expensive, several works focus on speeding up training of ensembles for *CNNs*. These approaches typically focus on reducing the training time of the ensemble. However, during test time, they consist of several individual *CNN* models. Consequently, inference time is similar to standard *CNNs* ensembles.

Snapshot ensembles, *e.g*. [56, 83, 254], combine several different snapshots from training a single model for building an ensemble. During training, they use a cyclical learning rate schedule. Specifically, they use a cosine annealing and periodically let the model converge into a local minimum, store a snapshot, and then re-start training with a higher learning rate. The final ensemble consists then of the last models.

Another line of work [244] trains a large mother network and transforms it into multiple other networks by *e.g*. inserting or widening layers. They preserve the predictions of

the mother network by function preserving transformations. Finally, they fine-tune the transformed networks on bagged datasets. Due to parameter sharing with the mother network and bagging, the training time of these transformed networks is significantly faster compared to ensembles trained from random initializations.

In contrast to these works, we focus on approaches which allow comparably fast inference. Our objective is that the models of the ensemble share most parameters with each other, so that the inference time is comparable to a single model.

2.4.4 Distillation Approaches

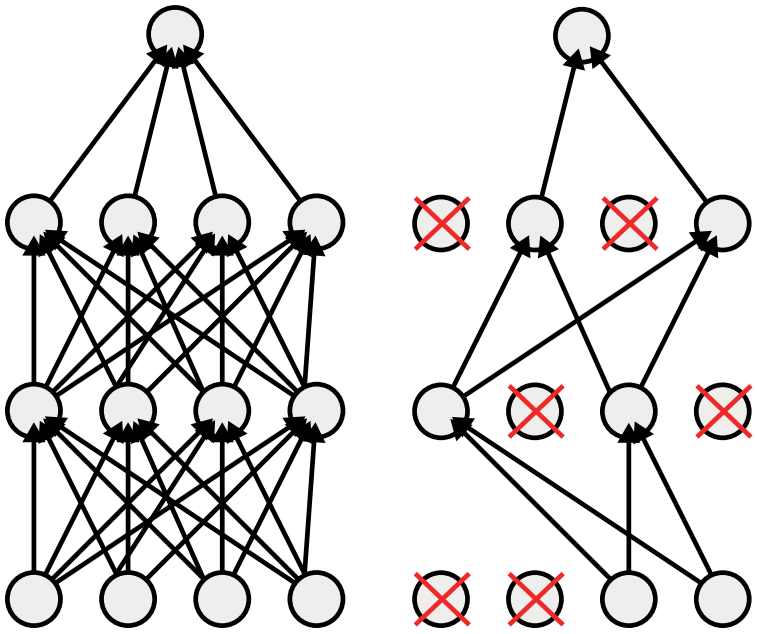
Distillation approaches try to train a smaller so-called student network from larger teacher networks, *e.g.* [78, 181, 284]. They typically use soft-labels obtained from the teachers [78] to train the student network. These approaches obtain these soft-labels by dividing the logits by a temperature scaling factor and then normalize them with a softmax activation function. The student network minimizes the KL divergence between its predictions and these soft-labels. Consequently, it learns the relative importance between the class labels (according to the teachers) for a given input image. Further approaches extend this by also trying to train auxiliary regressors between feature maps of student and teacher networks [181], or using adversarial loss function and using different multiple different teacher architectures [197]. Consequently, the learned features of the student and teachers are similar.

These methods typically have a fast inference time, as the student network is a standard *CNN*. Compared to our work, however, these distillation approaches have to train several different models during training time. Further, their objective is to mimic a larger and stronger model with a smaller model. In contrast, we directly aim to train a model without teachers.

2.5 Implicit Ensembles with Dropout

Dropout [208] is a simple way to prevent neural networks from overfitting. The main idea is to sample a binary mask from a Bernoulli distribution during training for each sample and multiply it with the neurons of a hidden layer (see Figure 2.4). The remaining neurons, which are multiplied by one, can be interpreted as a "thinned" version of the larger network. Therefore, dropout samples for each training sample a different "thinned" neural network. Consequently, one interpretation of dropout is that it trains an ensemble of an exponential number of neural networks during training, which all share parameters with each other.

During test time it is computationally impossible to correctly average this exponential number of neural networks. Therefore, dropout approximates this intractable inference by scaling the activations during test time to match the expected output of the layer during training time [208]. More specifically, dropout scales the activations by $1.0 - p$, where p



(a) Full network.

(b) Thinned network after applying the dropout mask.

Figure 2.4: Training neural networks with dropout during training time.

is the dropout probability. Consequently, standard dropout has a so-called *inference gap*, as the inference during training time differs from the inference during test time.

There is another variation of dropout, which add multiplicative Gaussian noise $\mathcal{N}(1, \sigma^2)$ to the hidden layers as opposed to Bernoulli noise, where σ is a hyper-parameter [208]. As the expectation of the hidden layers does not change during test time, Gaussian dropout does not need to re-scale the weights during test time. However, standard (Bernoulli) dropout is more popular. Further, Gaussian dropout is more related to data augmentation than ensembles. Therefore, we focus our discussion on the Bernoulli variant of dropout unless otherwise stated.

There are several works which extend standard dropout. One family of work tries to overcome this inference gap between training and test time (Section 2.5.1). Another line of work aims to alter the sampling strategy during training time (Section 2.5.2). Further, there is a line of work which interprets dropout as Bayesian inference (Section 2.5.3). Finally, a some works interpret dropout as form of data dependent regularization (Section 2.5.4).

2.5.1 Inference Gap

Dropout suffers from an inference gap, as the method approximates the ensemble prediction during test time by re-scaling the activations. However, this hampers the performance

of the method. Therefore, there are several works which try to overcome this inference gap.

One way to approximate this inference is Monte Carlo sampling, *e.g.* [52]. These methods try to close this inference gap by averaging the prediction of several "thinned" networks during test time. Subsequently, this typically yields more accurate results at additional computational expense during test-time, as for each sample several forward passes with different dropout masks are necessary to compute the ensemble prediction. To address this computational problem, distillation [78] can distill the Monte Carlo predictions from a teacher network trained with dropout back into a single student network [19]. Consequently, inference during test time for the student network only requires a single forward pass through the network.

Similarly, Expectation-Linear regularization tries to make the output of dropout layers close to the standard dropout mean approximation (*i.e.* by scaling weights) during training [145].

Another approach is to make some assumptions of the distribution of activations, *e.g.* inputs to hidden units are Gaussian distributed. With this assumption, a Gaussian approximation of the loss under dropout training can be derived, which can be optimized directly [240]. During test-time, this method does not need to re-scale the weights.

2.5.2 Sampling Strategy

One further line of work tries to change the dropout sampling strategy during training time. Standard dropout samples a random dropout mask and multiplies it with the activations of the hidden layers. In contrast, DropConnect [234] applies dropout on the weights. During test time DropConnect has to resort to Monte Carlo sampling to make the inference.

ModDrop [160] and Modout [123] apply dropout on an input modality (*e.g.* the depth-branch of an RGB-D network). This makes the individual modality networks discriminative on their own and reduces co-adaptations between the individual per-modality networks.

Some works try to improve dropout specifically for *CNNs*. Standard dropout typically does not show many benefits when it is applied directly to convolutional channels. To overcome this problem these works try to integrate a form of structure in the dropout masks. They drop entire convolution channels [219] or larger rectangular blocks from the inputs or feature maps, *e.g.* [40, 60].

There are also several extensions to dropout, which try to adapt it to ResNets [75], *e.g.* [57, 85, 249]. As ResNets consists of two branches, these dropout variants try to stochastically omit the non-residual branch [85], or scale them with a random scaling factor during the forward and backward pass [249]. Further, ResNeXt [247] is a ResNet extension, which introduces several smaller parallel branches next to the residual branch. For these types of networks Shake-Shake [57] does a random convex combination between

the non-residual connections of the networks during training time. During test-time, these variations typically scale the layers with the expectation of this random scaling factor. Interestingly, applying Shake-Shake regularization between branches decreases the correlation of features between branches [57]. Therefore, the authors hypothesize that the Shake-Shake benefits are due to the increased diversity between sub-networks.

2.5.3 Bayesian Inference

Another line of work interprets dropout as Bayesian inference, *e.g.* [52, 53, 81, 105, 140]. Bayesian approaches model the parameters $\boldsymbol{w}$ of a neural network as distribution $p(\boldsymbol{w}|D)$, where D denotes the dataset. However, to compute this posterior, *i.e.* $p(\boldsymbol{w}|D) = p(\boldsymbol{w})p(D|\boldsymbol{w})/P(D)$, intractable integrals are required. To overcome this problem, variational inference methods approximate this posterior with a simpler distribution $q(\boldsymbol{w})$. During training, it maximizes the evidence lower bound: $E_{q(\boldsymbol{w})}[\log p(y|\boldsymbol{x}, \boldsymbol{w})] - KL(q(\boldsymbol{w})\|p(\boldsymbol{w}))$, where the second term is a regularizer which makes the approximation similar to a prior distribution $p(\boldsymbol{w})$. During inference, the predictive distribution has to be marginalized w.r.t. $\boldsymbol{w}$, *i.e.* $\int_{\boldsymbol{w}} p(y|x, \boldsymbol{w})q(\boldsymbol{w})d\boldsymbol{w}$, typically by Monte Carlo integration.

Variational Gaussian dropout approaches typically parametrize the approximate distribution $q(\boldsymbol{w})$ with mean $\overline{\boldsymbol{w}}$ and standard deviation σ and assume that the weights factorize (*i.e.* entries of the weight matrix are independent) [81, 105, 140]. Similarly, for Bernoulli dropout, variational methods also assume independence and use the weight matrix $\overline{\boldsymbol{w}}$ and dropout probability p to parametrize for $q(\boldsymbol{w}) = \overline{\boldsymbol{w}} \cdot \text{diag}[\text{Bernoulli}(1-p)]$ [53].

During training, all methods learn the variational parameters, *i.e.* p and σ, and model parameters $\overline{\boldsymbol{w}}$ jointly. During training, these methods evaluate the log-likelihood term by a single Monte Carlo sample from the distribution $q(\boldsymbol{w})$, which corresponds to dropout training.

2.5.4 Data-Dependent Regularization

Dropout is also a form of data-dependent regularization, *e.g.* [152, 208, 228]. For shallow models, it is possible to derive a deterministic regularization. For example, in linear regression, dropout corresponds to scaling the weights w_i in the weight decay by the standard deviation of the i-th dimension of the data [208, 240]. Therefore, the regularizer penalizes dimensions with high variance more compared to standard weight decay. Intuitively, this also explains why batch normalization [89] reduces the need for dropout. Batch normalization transforms the input for each layer to zero mean and unit variance. Therefore, applying L_2 on the weights operating on this transformed input space has a similar effect to dropout [152]. Consequently, it is beneficial to reduce dropout rates in batch normalized networks.

However, for neural networks with multiple hidden layers, there is less understanding of the effect of dropout as a data-dependent regularizer. Helmbold and Long [76] discovered some properties of dropout in deep networks, *i.e.* invariance to scale of inputs and weights

and exponential power of the regularizer with increasing depth of the network. Further, Mou *et al.* [152] derive an error bound on the generalization error for dropout under the data-dependent regularization interpretation.

Our work is related to dropout, as we also try to train an ensemble of networks which share most parameters with each other. However, in contrast to our work, dropout typically uses a form of random noise on the hidden layers of a network to sample a "thinned" sub-network during training time, which all share parameters which each other. Each sample typically trains a different sub-network. In our work, we use a more explicit ensemble model. We train only a few "thinned" networks and make them different from each other with an auxiliary loss function. We fix these networks during training and train them with all the training samples.

2.6 Face Detection

We evaluate our spatial independent ensemble for the problem of face detection. Therefore, we will give a brief overview of face detection methods in this section. Since there is a vast number of works in this area, we only give a brief overview of seminal work in this area and the most recent works in the field. We illustrate the most important historic milestones and recent works in Figure 2.5. We categorize face detection methods into traditional methods (Section 2.6.1) and *CNN* based methods (Section 2.6.2).

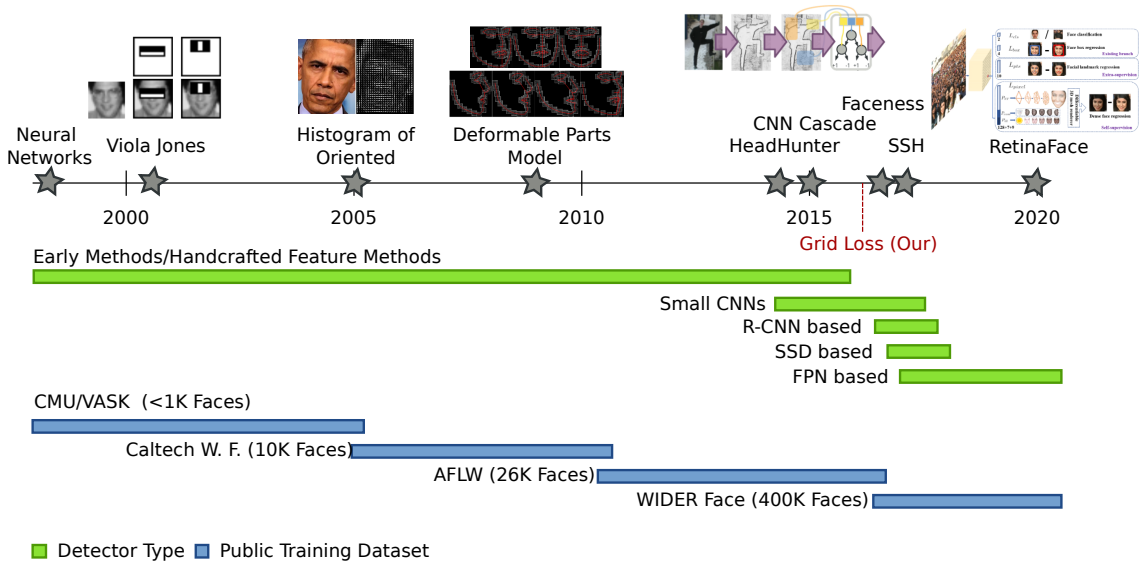


Figure 2.5: Overview of important and recent works for face detection.

2.6.1 Traditional Methods

Face detectors typically have real-time requirements. Therefore, traditional methods typically were designed to be computationally efficient. We categorize traditional methods into boosting based methods (Section 2.6.1.1) and Deformable Parts Model (DPM) (Section 2.6.1.2 based methods.

Further, there are complementary approaches which improve existing detectors by domain adaption techniques [125]; and exemplar based methods which use retrieval techniques to detect and align faces [113, 196].

2.6.1.1 Boosting Based Methods

One seminal work in the field of boosting based face detection is the realtime detector of Viola and Jones [227]. This detector uses an integral image representation to compute rectangular Haar features. The integral image enables computing Haar feature responses in constant time independent of their size. Further, the detector uses a cascade of several classifiers. The first classifier in this cascade can already discard most "easy" background examples, reducing the computational load of successive classifiers. Both the integral image and the cascade structure contribute to the realtime speed of the method.

More modern boosting based detectors typically use more discriminative feature representations, as increasingly large datasets and more computational capacity gets available. They use linear classifiers as weak learners with SURF based features [129], exemplars [126] or leverage landmark information with shape-indexed features for classification [26]. Successive works [148, 252] use oriented gradient images and LUV images as feature representation. HeadHunter [148] uses integral images on top of this representation to train tree-based weak learners in a cascade. Further, several works apply filters on top of these oriented gradient images, *e.g.* [157, 269], to make these features more discriminative. Finally, Yang *et al.* [253] propose *CNN* features for the boosting framework.

2.6.1.2 Deformable Parts Methods

In parallel, Histogram of Oriented Gradients (HOG) [36] features and *DPM* [49] based detectors became popular. Standard *HOG* based detectors use a linear *SVM* to learn a detector on oriented gradient features. *DPM* extends standard *SVMs* detectors to learn one root and multiple part templates. The responses of these detectors are then combined with a deformation cost function to compute the final detection score. Extensions to *DPMs* have been proposed which handle occlusions [59], improve runtime speed [250] and leverage manually annotated part positions in a tree structure [283].

2.6.2 *CNN* Based Methods

With the success of *CNNs* for computer vision, *e.g.* [61, 112], the first small *CNN* based detectors became popular. Early works in this era (*i.e.* pre-2012), apply a small num-

ber of convolution filters followed by sum pooling or average pooling on the image, *e.g.* [55, 185, 224]. The first modern *CNN* based methods exploit the benefits of multi-scale cascades, *e.g.* [127, 174]. Further works improve these detectors by including context information [268]) and multitask learning (by predicting landmarks) to detect faces [267]. The main benefit of these types of detectors is that they can run in realtime on standard Central Processing Unit (CPU) hardware. Our spatial independent ensemble, *i.e.* grid loss [165], also falls into this family of small *CNN* based detectors. In contrast to these other works, our spatial independent ensemble especially benefits detecting occluded faces (see Chapter 4).

With the availability of larger datasets, *i.e.* WIDER face [256], larger *GPUs*, and advances in generic object detectors for *CNNs*, *e.g.* [132, 133, 137, 176], an increasingly larger amount of works focus on adapting these generic detectors to face detection.

These generic detectors typically tile anchor boxes over one or more convolutional feature maps of different scales of a *CNN*. The detector classifies each anchor box as an object or background and regresses the object bounding box relative to the anchor box. R-CNN based methods [176] use a two-step approach. In the first step, they use a Region Proposal Network (RPN) to predict object proposals from anchors on the last convolutional map of a *CNN*. In the second step, they crop a Region of Interest (RoI) around detected object proposals (*i.e.* foreground anchors) and feed it into a second neural network to re-score and refine the proposal to the final prediction. In contrast, single-shot methods [132, 133, 137] directly predict the object class and bounding box from anchors, which are tiled over convolutional maps on different scales.

One of the main differences between generic object detection datasets and face detection datasets is the size distribution of objects [82]. Face datasets typically contain many smaller objects compared to generic object detection datasets. However, face bounding boxes typically only have a very limited aspect ratio, as faces are rigid objects. Therefore, a lot of research effort focused on adapting generic detectors to detecting smaller objects.

The first family of these detectors built upon the Faster R-CNN [156, 176, 255] framework, *e.g.* [72, 96, 236]. As single shot detectors, *e.g.* [137], became more popular due to their speed, several detectors adopted this framework, *e.g.* [257, 273, 274]. Single-shot detectors especially benefit from Feature Pyramid Networks (FPNs) [132]. These detector architectures combine the high level (but coarse) semantic features with lower level (but fine-grained) features by upsampling and element-wise operations. Consequently, these types of detectors are especially beneficial for detecting small objects. Therefore, face detectors quickly adopted this structure, *e.g.* [31, 38, 128, 130, 138, 155, 215, 258, 265, 272, 277, 282].

In contrast to generic object detection datasets (*e.g.* COCO [134], PASCAL VOC [47], *etc.*), face detection datasets consist of a larger amount of tiny objects. Therefore, *FPN* based face detectors typically additionally use the stride 4 *FPN* levels (*i.e.* higher resolutions) to detect faces. In contrast, the lowest level in standard detectors is the stride 8 *FPN* level. However, this introduces a large number of negative anchor boxes during training. To address this problem, recent methods improve the mining of anchors [128,

130, 138, 274, 282], use cascades on the upward pass on the pyramid to reduce easy background samples [31, 272, 277], introduce several latent background and foreground classes [215, 274, 277] and typically use Online Hard Example Mining (OHEM) [201]. Further, some works add explicit context modules to the *CNN*, which are useful in detecting small faces by detecting body parts, *e.g.* [128, 130, 155, 215]. Finally, some works introduce multitask landmarks or mesh regression losses [38, 258]. Even though *FPN* based detectors improve upon detecting small objects, state-of-the-art methods still rely on traditional scale-space pyramids (*i.e.* test-time augmentation) to achieve good results [38, 155, 274].

2.7 Metric Learning

We evaluate our parameter shared ensemble with diversity auxiliary loss functions for the problem of deep metric learning for image retrieval. Therefore, we provide a brief summary of related work in this field. For a more complete overview of metric learning methods, we refer to a survey of Bellet *et al.* [8].

The main objective of metric learning approaches is to learn a distance function $d(\cdot,\cdot) : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}^+$, which maps two inputs $\boldsymbol{x} \in \mathcal{X}, \boldsymbol{y} \in \mathcal{X}$ to a non-negative distance. This distance should be "small" for inputs which are semantically similar to each other, and "big" for inputs which are semantically dissimilar to each other. Such distance functions are useful in computer vision applications where there are no pre-defined number of categories such as face verification (*e.g.* [86]), image retrieval (*e.g.* [163]), object tracking (*e.g.* [216]), person Re-ID (*e.g.* [198]), *etc.*

Traditional methods in computer vision typically rely on handcrafted features to represent $\boldsymbol{x}, \boldsymbol{y}$, *e.g.* [28, 79, 107]. Therefore, $\mathcal{X} = \mathbb{R}^d$, where d represents the feature dimensionality. These methods typically learn a Mahalanobis distance $M \in \mathcal{S}^+$, where $\mathcal{S}^+$ denotes the space of positive semidefinite matrices. The squared distance is then defined as: $d(\boldsymbol{x}, \boldsymbol{y})^2 = (\boldsymbol{x} - \boldsymbol{y})^\top M (\boldsymbol{x} - \boldsymbol{y})$.

As M is positive semidefinite, it is possible to factorize it into $M = L^\top L$. Consequently, we can interpret the Mahalanobis distance between two feature vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ as Euclidean distance of $\boldsymbol{x}$ and $\boldsymbol{y}$ in a linear transformed feature space. Formally,

$$\begin{aligned} d(\boldsymbol{x}, \boldsymbol{y})^2 &= (\boldsymbol{x} - \boldsymbol{y})^\top M (\boldsymbol{x} - \boldsymbol{y}) \\ &= (\boldsymbol{x} - \boldsymbol{y})^\top L^\top L (\boldsymbol{x} - \boldsymbol{y}) \\ &= (L\boldsymbol{x} - L\boldsymbol{y})^\top (L\boldsymbol{x} - L\boldsymbol{y}). \end{aligned} \tag{2.17}$$

$L \in \mathbb{R}^{\tilde{d} \times d}$ maps $\boldsymbol{x} \in \mathbb{R}^d$ and $\boldsymbol{y} \in \mathbb{R}^d$ into a $\tilde{d}$-dimensional vector via a linear projection. Traditional metric learning approaches either learn M or the factorized transform L from training data, *e.g.* [37, 63, 65, 245].

Successive works introduce kernels in these frameworks, to make non-linear distance functions possible, *e.g.* [24, 220, 245]. Kernel methods express decision functions with

so-called kernel functions $k(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x})^\top \phi(\boldsymbol{y})$, where $\phi(\boldsymbol{x})$ is a non-linear function which maps $\boldsymbol{x}$ into a high-dimensional space. Rather than explicitly computing $\phi(\cdot)$, kernel methods only need to model the kernel function $k(\cdot, \cdot)$. Some common examples are $k_{\text{rbf}}(\boldsymbol{x}, \boldsymbol{y}) = e^{\frac{-||\boldsymbol{x}-\boldsymbol{y}||^2}{2\sigma^2}}$, or $k_{\text{poly}}(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}^\top \boldsymbol{y} + c)^d$. Consequently, kernel methods do not need to explicitly compute the possibly expensive non-linear map $\phi(\cdot)$.

In contrast to these kernel methods, several works try to model the non-linear map $\phi(\cdot)$ directly. They replace the linear function L with a parametrized non-linear function $\phi(\cdot; \theta)$ in Equation (2.17). θ are parameters of the function which are learned during training. Early works in this area model $\phi(\cdot; \theta)$ with a gradient boosted ensemble of decision trees, *e.g.* [98], or neural networks, *e.g.* [32, 187].

Modern deep metric learning approaches for computer vision follow the early works in metric learning with neural networks, *e.g.* [32, 187]. They use *CNNs* to model the non-linear function $\phi(\cdot; \theta)$. Further, rather than using a feature vectors as input, they take images as input. *CNNs* learn a suitable feature representation during training. Therefore, for *CNN* based methods, $\mathcal{X}$ denotes the space of images.

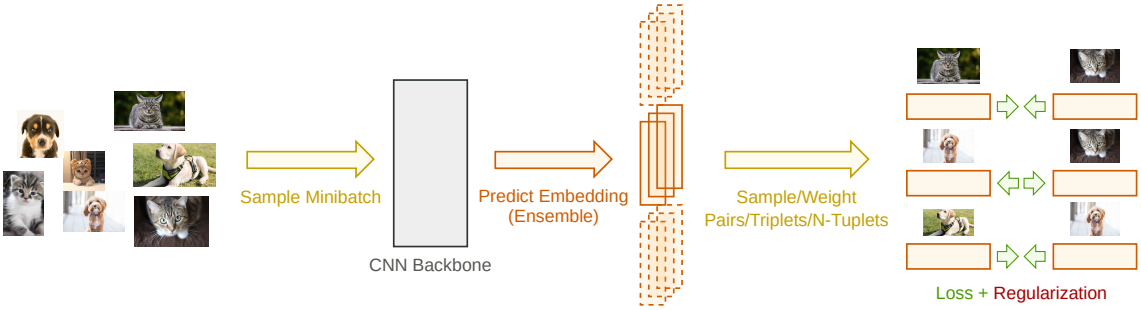


Figure 2.6: Overview of deep metric learning systems.

We illustrate a framework of modern deep metric learning systems in Figure 2.6. In contrast to image classification problems, deep metric learning systems predict feature vectors and use special loss functions to make feature vectors of similar images close to each other and dissimilar images far apart from each other. As large neural networks tend to overfit, some works use regularization techniques to reduce overfitting. Further, in contrast to traditional metric learning methods, modern deep learning are trained on large datasets with mini-batch gradient descent. However, random sampling mini-batches typically yields image pairs which are easy to categorize as similar or dissimilar. Consequently, the loss function results in too small gradients, which makes training highly accurate deep *CNNs* impossible. Therefore, modern architectures use sampling strategies to explicitly construct informative mini-batches with "hard" training examples resulting in high gradients. Finally, some works exploit the benefits of ensembles and split the embedding up into an embedding ensemble.

We summarize recent contributions in the field in Figure 2.7. In the following, we give a brief overview of loss functions (Section 2.7.1), sampling strategies (Section 2.7.2),

regularization approaches (Section 2.7.3) and embedding ensembles (Section 2.7.4).

2.7.1 Loss Functions

One of the main focus of recent research is loss functions for deep metric learning. The main objective of loss functions is to guide the network to make distances between feature vectors of similar images close to each other and dissimilar images far apart from each other. Most of these functions can be categorized into loss functions operating on n-tuples, prototype-based loss functions, and ranking based loss functions.

N-Tuples N-tuple-based loss functions use pairs, triplets, or quadruplets of similar or dissimilar images to learn the embedding. One of the most prominent works in this area is the contrastive loss, *e.g.* [32, 71], and triplet loss, *e.g.* [191, 245], which use pairs or triplets of similar and dissimilar images to optimize the loss. Further work generalizes this to quadruplets, *e.g.* [116, 279], n-tuples [206] or uses a structural loss function [163].

The margin loss [246] extends the contrastive loss by learning a class and image specific boundary threshold between positive and negative pairs. It combines the benefits of triplet loss, *i.e.* a per image distance threshold, with the simplicity of contrastive loss, *i.e.* it only requires pairs to evaluate the loss.

The hierarchical triplet loss [58] incorporates hierarchies into the triplet loss. It performs a hierarchical clustering over classes and encodes this clustering into an adaptive margin in a triplet loss function. Therefore, the loss forces semantically dissimilar classes to lie further away in the embedding space compared to semantically similar classes. Similarly, the log-ratio loss [102] incorporates label distance into the loss function.

As standard triplet loss and contrastive loss are not differentiable everywhere, as they use a $\max(0, \cdot)$ function. Several loss functions propose smooth approximations, *e.g.* [163, 206, 237, 243, 259, 260], with the log-sum-exp approximation. With these approximations, the loss functions are differentiable everywhere and consequently easier to optimize.

Further, several works L_2 normalize feature vectors, *e.g.* [237, 243, 260]. Computing distances or similarities with L_2 normalized vectors corresponds to computing angles between vectors. Subsequently, this empirically performs favorably compared to non-normalized vectors. Successive works try to introduce additional angular margins [260] into these loss formulations.

Loss functions on pairs, triplets, *etc.* are typically closely tied to sampling strategies, *e.g.* [77, 175, 191]. For example, the performance of networks trained with triplet loss is largely impacted by which samples are chosen to evaluate the loss function. Consequently, the samples determine the landscape of the loss function of the neural network. The multi-similarity loss [243] casts the sample mining and weighting strategy in combination with the loss function into a theoretical framework. It combines an elaborate mining method with a simple binomial deviance loss [259] and achieves state-of-the-art accuracy.

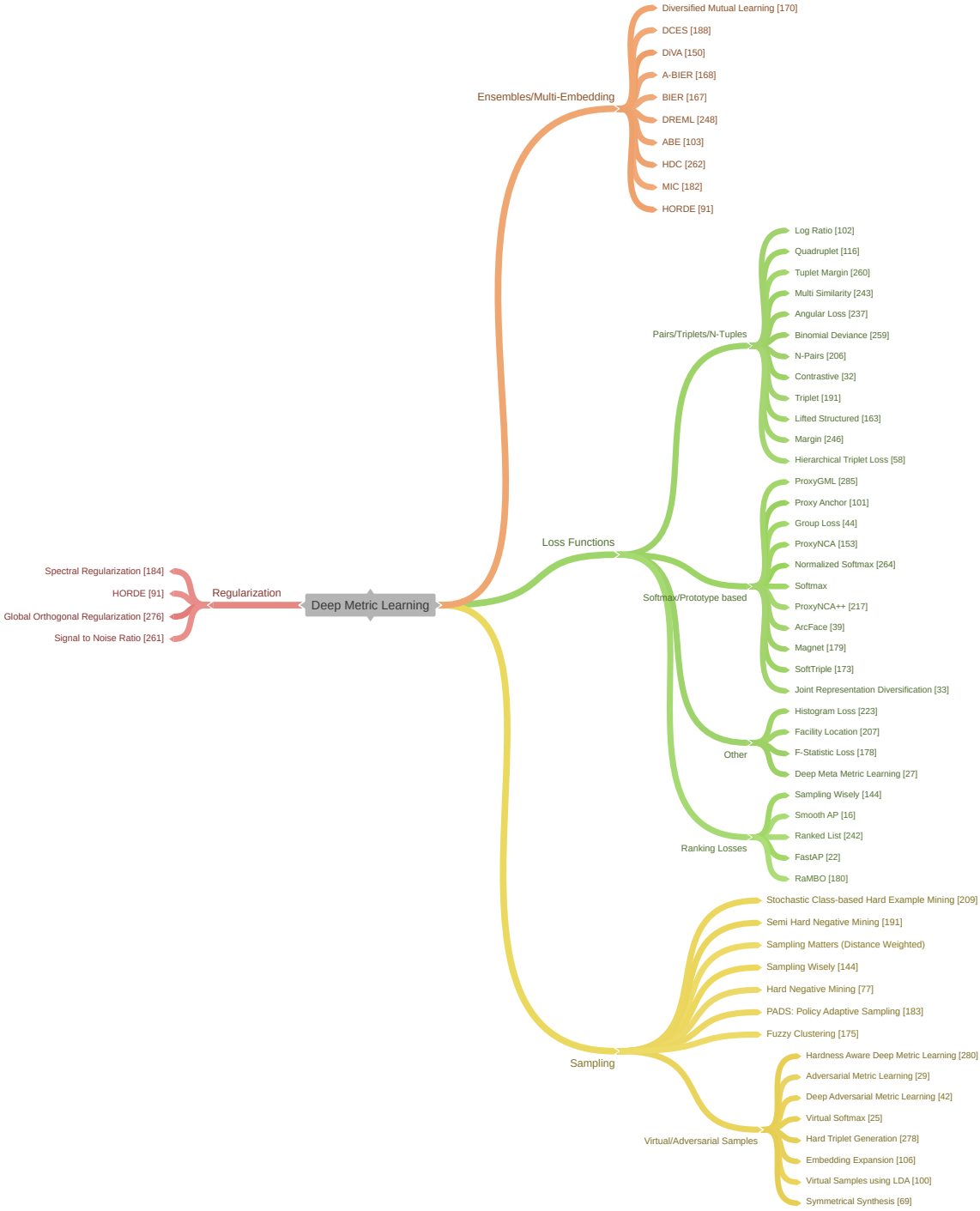


Figure 2.7: Overview of recent works in deep metric learning.

Prototypes Prototype-based loss functions use prototype vectors to learn an embedding space. One of the simplest prototype-based loss functions is the softmax-cross-entropy loss function. This loss function is the standard loss function for object categorization problems. The last layer consists of a weight matrix $\boldsymbol{W}$, which maps the last hidden layer $\boldsymbol{x}$, *i.e.* the embedding space, to the output classification scores, *i.e.* $\boldsymbol{W}\boldsymbol{x}$. We can also be interpret this as computing the similarity between feature vectors of the hidden layers to learned per class prototype vectors with the dot product. These prototype vectors are the row-vectors of the weight matrix $\boldsymbol{W}$. During inference time, prototype-based methods discard the classification layer and prototypes. They then use the embedding vectors $\boldsymbol{x}$ of the last hidden layer for retrieval.

One of the main benefits of prototype-based approaches is the fast convergence time, *e.g.* [153]. One of the main drawbacks of prototype-based approaches is that these approaches have a memory overhead proportional to the number of classes during training time. Further, they need datasets with class labels for training. However, in metric learning problems, the number of classes is typically large. Further, some problems do not have a pre-defined number of classes during training time. For example, several self-supervised learning approaches dynamically augment object instances in images [94]. These approaches use metric learning methods to learn an embedding space in which augmented object instances are closer to each other than images from different objects.

Modern approaches in this line of work use L_2 normalization of the weights and prototype vectors, *e.g.* [39, 264]. Similarly to the n-tuplet based loss functions, under this normalization, similarities between vectors correspond to angles between feature vectors and prototype vectors. Successive work introduces an additional margin in the loss function, *e.g.* [39].

Another line of work learns multiple prototype vectors per class, *e.g.* [101, 153, 173, 179, 217, 285]. Early works in this area repeatedly cluster the dataset after several training epochs to create these prototype vectors [179].

More modern approaches try to learn prototype vectors during training with backpropagation. Several of these works use the Neighborhood Component Analysis (NCA) loss to learn prototype-based embeddings, *e.g.* [153, 217]. Compared to the softmax loss function, the ProxyNCA loss allows a flexible allocation of prototype vectors to classes. It can share prototypes among classes or allocate multiple prototypes to a single class. Consequently, it allows balancing the memory footprint and the accuracy of the loss. Successive works propose graphs to further improve the loss functions, which use multiple prototype vectors per class [285].

Several multi-prototype-based works combine prototypes with pairwise loss functions (*e.g.* [33, 101, 173]) such as the triplet loss [173] or the binomial deviance loss [101]. Consequently, they try to exploit the benefits of prototype-based loss functions, *i.e.* fast convergence time, with the benefits of n-tuple based losses, *i.e.* take fine-grained sample-to-sample distances into account.

Finally, some approaches try to combine prototypes with graphs over samples or other

prototypes, *e.g.* [44, 285]. These works use the benefits of label propagation to improve prototype-based methods for metric learning.

Ranking N-tuple-based loss functions and prototype-based loss functions typically focus largely on retrieving the nearest neighbor of a query image correctly. A recent line of work addresses this problem by also trying to retrieve successive neighbors correctly, *i.e.* [16, 22, 144, 180, 242] They directly optimize the ranking of the retrieved samples, *i.e.* the mean Average Precision (mAP), of a metric learning system. One of the main challenges in this line of work is that mAP is a non-differentiable function. Consequently, it is hard to integrate these loss functions directly into the end-to-end training of *CNNs*.

State-of-the-art works use different strategies to approximate mAP so that the loss function is differentiable. Several works employ sampling and weighting strategies in combination with simple pairwise losses to optimize mAP, *e.g.* [144, 242]. Another strategy is to quantize the distances between all pairs in a mini-batch, similar to [223], and optimize mAP, *e.g.* [22]. Further, rather than sampling and quantizing, it is possible to approximate mAP with a smooth function, by replacing indicator functions with sigmoid functions, *i.e.* [16]. Finally, another method to minimize mAP is to use blackbox differentiation [180].

Other Apart from n-tuple, prototype, and ranking-based loss functions, there are also loss functions which do not fall into these prototypical categories. The histogram loss [223] operates on the histogram of distances between similar and dissimilar samples. Further, it is possible to pose metric learning based on the F-statistics of distributions [178] and as facility location problem [162]. Finally, there are approaches which use meta learning to improve loss functions for metric learning [27].

2.7.2 Sampling Strategies

Another design choice in deep metric learning is the sampling, mining, or weighting strategy. These methods are relevant, as we optimize deep neural networks with mini-batch gradient descent on large datasets. Consequently, the gradient estimates are noisy. Sampling methods help to overcome this problem. In contrast, traditional methods typically use standard gradient descent or non-gradient based optimization methods during training. Therefore, these approaches do not suffer from noisy gradients.

More specifically, when we naïvely sample mini-batches uniformly randomly, chances are high that the samples are non-informative. Subsequently, if the number of classes is high in the training set, the probability is high that all of them belong to different classes. However, especially loss functions operating on n-tuples rely on positive pairs of samples (*i.e.* two samples corresponding to the same class) in a mini-batch for training. Further, most pairs of samples are easy to distinguish as either positive pairs or negative pairs. Consequently, when we sample uniformly randomly, the probability is low that the network sees hard to classify image pairs.

Sampling and mining strategies in deep metric learning try to overcome this problem. These methods sample more informative samples by considering class information and distance information between samples. N-tuple based losses typically first uniformly sample a set of classes and then several samples for each sampled class. Subsequently, this ensures that evaluating these loss functions is possible. Additionally, several strategies incorporate distances between samples in their mining strategy, *e.g.* [73, 77, 191, 209, 246].

Most of these sampling methods focus on the triplet loss, *e.g.* [73, 77, 191, 209]. They try to find semi-hard [191] or the hardest [77] triplets in a mini-batch. Further, some extensions try to find hard triplets offline over the full training set with approximate nearest neighbor search methods [73]. Another line of work first mines a set of similar classes. From these classes, they sample a several hard triplets [209]. As similar classes are close in feature space to each other, they tend to have harder triplets compared to randomly sampled classes.

Wu *et al.* [246] propose to sample pairs uniformly according to their relative distance to each other. They design their method to work with a pairwise loss, *i.e.* the margin loss. Further, subtype clustering [175] re-weights the triplets according to a fuzzy clustering approach. Finally, Roth *et al.* [183] pose sampling as a reinforcement learning problem, which allows drawing samples conditioned on the current training status.

Another line of work tries to synthetically generate "hard" samples directly in the embedding space, *e.g.* [25, 29, 42, 69, 100, 106, 278, 280]. In contrast to traditional Generative Adversarial Network (GAN) based approaches, this line of work generates feature vectors rather than images. Several of these works, *e.g.* [25, 69, 106] use simple algebraic operations, such as interpolation, to generate samples. More complex works use adversarial loss functions with generators to generate "realistic" synthetic but "hard" examples, *e.g.* [29, 42, 278, 280]. Further, Kim *et al.* [100] use Linear Discriminant Analysis (LDA) to generate synthetic "hard" samples between decision boundaries.

2.7.3 Regularization

Several works in metric learning add an auxiliary loss function to regularize the learned embedding, *e.g.* [91, 184, 261, 276]. Several of these loss functions make feature vectors orthogonal [276] or use spectral regularizers to encourage a large number of directions of variance in the embeddings. Consequently, these methods reduce redundancies in embeddings. Further, Higher Order Regularizer for Deep Embeddings (HORDE) [91] encourages higher-order statistics of similar images to be similar to each other, and dissimilar images to be dissimilar to each other. Finally, the Signal to Noise Ratio (SNR) [261] loss is motivated by the signal to noise ratio and penalizes feature vectors to be zero-mean.

2.7.4 Ensembles

Recently, another line of work became popular which use embedding ensembles to predict multiple embeddings for an input image, *e.g.* [103, 150, 167, 168, 182, 189, 248, 248, 262].

DREML [248] uses bagging to increase diversity in a large embedding ensemble. Rather than bagging training samples, the method uses bagging to sample the training labels into small subsets. Each learner then learns to categorize a different random grouping of the label-space. In this ensemble the learners do not share parameters with each other. Consequently, this method has large computational demands.

Most other works in this line of work use parameter sharing to reduce computational complexity and memory overhead. Hardness Aware Deeply Cascaded Embedding (HDC) uses the benefits of deeply supervised networks [118, 212] and predicts embedding vectors at multiple hidden layers at the *CNN*. *HDC* trains each layer to handle harder examples.

Successive works split the network in the end into several embedding heads. The main challenge with these approaches is to make the embeddings diverse from each other, as all learners share the same feature representation and are trained on the same dataset. We use re-weighting [167] and auxiliary loss functions [168] to increase diversity in such ensembles.

Successive works improve this method by also including feature level attention [103]. Similar to Random Forest, different learners are encouraged to focus on different features. To this end, the authors introduce an attention mask per learner. To make the mask diverse, they use an auxiliary loss function on the learners.

HORDE [91] can also be interpreted as an ensemble approach, which iteratively composes several learners by randomly projecting and attending previous learners with a form of attention mechanism. Consequently, *HORDE* builds an ensemble which introduces diversity by random initialization and projection of its learners.

Further, several works [182, 188] repeatedly cluster the data and make individual heads focus on certain clusters. Finally, Diverse Visual Feature Aggregation for Deep Metric Learning (DiVA) [150] uses different training supervision signals such as supervised learning and self-supervised learning to diversify the embedding.

2.8 Summary

In this section we gave a brief overview of ensemble methods in the context of computer vision and more specifically in the context of neural networks and deep learning. We categorized ensemble methods for *CNNs* according to three criteria. First, we categorize ensembles for *CNN* into implicit (*i.e.* dropout based methods) and explicit (*i.e.* traditional ensembles). Second, we consider the number of parameters learners share in an ensemble with each other. Third, we classified the diversity method they use to make the learners diverse from each other. Further, as we address the problem of face detection and metric learning for image retrieval, we also gave an overview over these fields. Our works fall into the category of explicit parameter shared ensembles. In the following chapters we outline a ensemble which exploits spatial independence to increase diversity (Chapter 3) and a method which uses auxiliary loss functions and sampling to increase diversity (Chapter 4).

3

Spatial Independence for Ensemble Diversity

"The fact is, I don't know where my ideas come from. Nor does any writer. The only real answer is to drink way too much coffee and buy yourself a desk that doesn't collapse when you beat your head against it."

— Douglas Noel Adams

Contents

3.1 Motivation 41
3.2 Grid Loss for Convolutional Neural Networks (CNNs) 43
3.3 Evaluation 49
3.4 Summary 60

3.1 Motivation

In this chapter, we show the benefits of spatial independence for parameter shared *CNN* ensembles. Specifically, we show this on the problem of face detection.

Traditional approaches for face detection are based on boosting [10, 41, 148, 192, 227, 253, 269] and Deformable Parts Models (DPMs) [49, 148]. After the success of deep learning for computer vision, *e.g.* [112], *CNN* based object detection methods became more popular, *e.g.* [48, 80, 127, 137, 176, 195] (see Section 2.6).

One of the most challenging problems for standard *CNNs* in this field are partial occlusions of faces. Occluding objects can have arbitrary shape and texture. Subsequently, this introduces a significant variation in facial appearance. Standard *CNNs* can only

overcome this problem by collecting large datasets which contain a wide variety of occluded faces. Unfortunately, collecting such datasets is prohibitively expensive.

The main problem in this context is that standard *CNNs* are "lazy" in learning diverse features. As soon as they learn a few good features, *e.g.* mouth, which are good enough to distinguish faces from the background in the training set, they have no inclination to learn more features for solving this task. Consequently, they do not need to develop features for nose or eyes, even though they might be useful for detecting occluded faces. However, such *CNNs* fail during test-time if these few prototypical regions, *e.g.* mouth, are occluded.

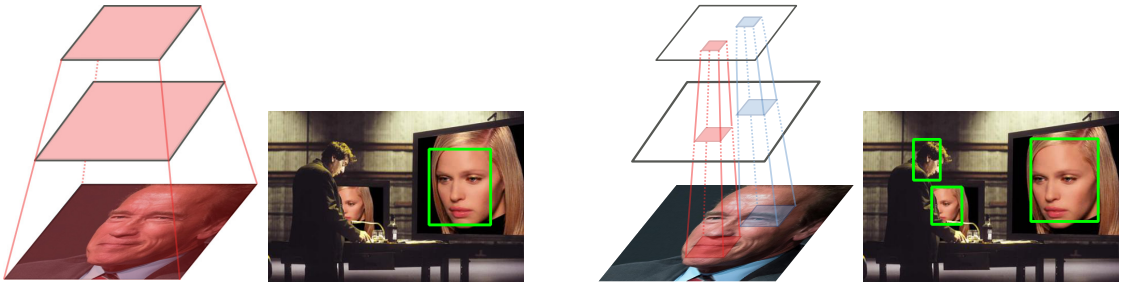
To address this problem, we propose a novel loss function called *grid loss*. This loss function divides the last convolutional layer of a *CNN* into a grid of non-overlapping blocks. We optimize a linear classifier for each of these groups separately. Each learner shares parameters with a global linear classifier over the full ensemble.

A single learner in our ensemble has a small receptive field, which *e.g.* just contains an eye, nose, mouth, *etc.* (see Figure 3.1). During training, each learner needs to learn how to distinguish faces from the background from the information in its receptive field alone. Consequently, by applying our grid loss, the *CNN* ensemble has to learn a more diverse feature representation. Our grid loss forces *CNNs* to learn discriminative features in all subregions (*e.g.* eye, mouth, nose, *etc.*). As we show in our experiments (Section 3.3), even if some learners in the ensemble fail due to occlusions, the full ensemble can still detect the face.

As all our learners share weights with a global linear layer, we can map our learners back to a regular *CNN* after training. Therefore, our grid loss does not introduce any additional runtime overhead during test time. Further, as computing the activations of the convolutional layers dominates training time, our method only introduces a negligible overhead during training time in terms of memory and computational demand.

In our experiments (Section 3.3) we make a detailed ablation study of our grid loss and compare it to the state-of-the-art. Specifically, we show that grid loss compares favorably against standard *CNNs* (Section 3.3.1). Further, on a dataset with vast object occlusions, it outperforms *CNNs* trained with standard loss functions (Section 3.3.4). Finally, *CNNs* trained with our grid loss develop a more diverse feature representation compared to standard *CNNs* (Section 3.3.5). As we show in our evaluation, this also helps to make our method more robust to over-fitting (Section 3.3.5). Further, as we use a very small network architecture, our detector can run in real-time on standard Central Processing Units (CPUs).

The remainder of this chapter is structured as follows. First, we present our method (Section 3.2), where we first outline our neural network architecture (Section 3.2.1). Then, we discuss our novel grid loss layer (Section 3.2.2). Next, we introduce a post-hoc bounding box and ellipse regressor (Section 3.2.3), which refines the predictions of our face detector. Finally, we show a detailed evaluation of grid loss (Section 3.3) and conclude our findings (Section 3.4).



(a) Global loss layer. **(b)** Our grid loss layer.

Figure 3.1: Schematic overview of (a) standard global loss and (b) the proposed grid loss with an illustrative example on FDDB.

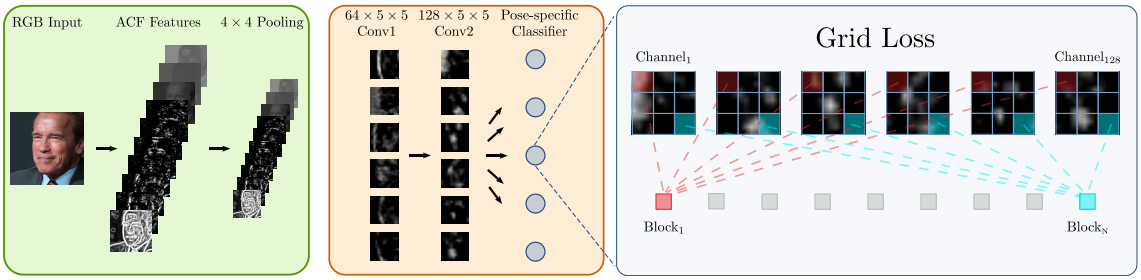


Figure 3.2: Overview of our method: our detection CNN builds upon Aggregate Channel Features (ACF) [41]. For each window, after pooling, we apply successive convolution filters to the input channels. To distinguish faces from non-faces we use pose-specific classifiers. Instead of minimizing the loss over the last full convolution map, we divide the map into small blocks and minimize a loss function on each of these blocks independently.

3.2 Grid Loss for *CNNs*

Holistic object detectors for faces or pedestrians typically have realtime requirements, as they are commonly used in video surveillance applications. Therefore, we design a small and fast network architecture, which meets these requirements (see Figure 3.2). We apply only two convolution layers on top of handcrafted input features, as we explain in Section 3.2.1. During test time, we apply our detector "fully convolutional" over an image pyramid, similar to [194]. In Section 3.2.2, we detail how we train this *CNN* with our grid loss layer to obtain highly accurate part-based pose-specific classifiers. Finally, we apply a post-hoc regressor to refine the initial bounding box locations (Section 3.2.3). This regressor allows our face detector to skip intermediate octave levels and therefore improves runtime performance.

3.2.1 Neural Network Architecture

We illustrate the architecture of our *CNN* in Figure 3.2. Rather than using raw images as input, our network builds upon *ACF* [41] as low-level inputs. Compared to convolu-

tion layers, these features are faster to compute on the CPU while achieving competitive accuracy when combined with tree ensembles [148].

We subsample this *ACF* pyramid by a factor of 4, which reduces the computational cost of the successive layers. Then, we feed these low-level features into two 5×5 convolution layers, which extract mid-level features. After each convolution layer, we apply a Rectified Linear Unit (ReLU) non-linearity. Further, we normalize the responses across layers with a Local Contrast Normalization (LCN) layer in between the two convolution layers. Finally, we apply a small amount of dropout [161] of 0.1 after the last convolution layer. On top of these shared feature representation, we train several pose-specific classifiers ("detection templates") which distinguish faces with a specific pose (*e.g.* front-view, side-view, *etc.*) from the background. To combine the predictions from these pose-specific classifiers, we use their maximum score.

Unless otherwise stated, we train the network from scratch. We initialize the weights randomly with a Gaussian of zero mean and 0.01 standard deviation. We train our network on patches of faces and patches of background regions of an image. We use the pose information of our training dataset [108] to match a face patch to its corresponding detection template.

At test-time, we apply our network fully convolutional over the feature pyramid at several scales. We then perform greedy Non Maxima Suppression (NMS) of pairs of bounding boxes B_a and B_b using the area of intersection over min-area overlap score. Formally, we define their overlap score as

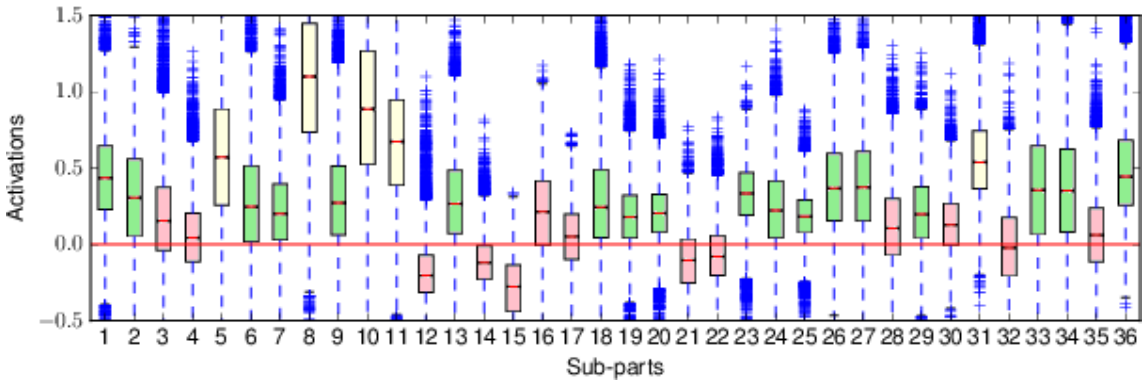
$$o_{\text{NMS}}(B_a, B_b) = \frac{|B_a \cap B_b|}{\min(|B_a|, |B_b|)}, \tag{3.1}$$

where $|\cdot|$ denotes the area of a bounding box and $\cdot \cap \cdot$ denotes the intersection of two bounding boxes. After detection, we sort all bounding boxes by their confidence scores and greedily suppress them if their overlap threshold exceeds 0.3, following [148]. As face detectors tend to detect sub-regions of faces, normalizing the overlap by the minimum area outperforms normalization by union area of the two boxes [148].

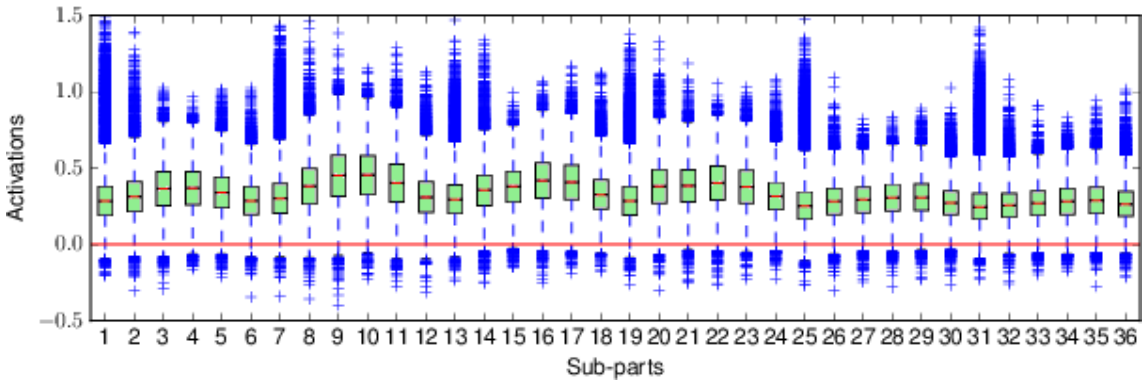
In our evaluation, we also show the benefits of grid loss with larger neural network architectures. To this end, similar to [205], we replace each 5×5 convolution layer with two 3×3 convolution layers. Further, we increase the number of convolution channels to 64, 256, 512, 512, respectively. We denote this architecture as *Big* in our experiments in Section 3.3.8.

3.2.2 Grid Loss Layer

When we train one of our pose-specific detection templates with a standard loss function (*e.g.* cross-entropy loss, hinge loss, *etc.*), they tend to develop non-discriminative subparts. These parts have negative median responses over the positive training set, as we illustrate in Figure 3.3a (top). The main reason for this is that standard *CNNs* are "lazy".



(a) Regular loss



(b) Grid loss

Figure 3.3: Boxplot of 2×2 part activations on the positive training set (i.e. by dividing the detection template into non-overlapping parts, as in Fig. 3.2). Activations trained by regular loss functions can have parts with negative median response. We mark parts whose 25% percentile is smaller than 0 (red) and parts which have significant positive median activations compared to other parts (yellow).

As soon as one region develops a discriminative feature, which works on all the training samples, the *CNN* has no inclination to develop more features. Consequently, during test-time, the *CNN* heavily relies on these few sub-parts on the feature map to make positive predictions. However, if these parts are occluded during test-time, the detector tends to miss the face.

We address this problem by dividing the last convolutional layer into small $n \times n$ blocks and optimize a loss on each of these blocks for each of our pose-specific detection templates separately. Consequently, we force the *CNN* to learn discriminative parts for all sub-regions separately. The median response for all these parts is non-negative over the positive training set, as we illustrate in Figure 3.3a (bottom). When a face is partially occluded, only the corresponding sub-set of the part-classifiers will fail. As all the non-occluded part-classifiers are unaffected by such partial occlusions, the detector has a chance to recover.

Formally, let $\boldsymbol{x} = g(\boldsymbol{a})$ denote a $c \times h \times w$ dimensional tensor, which represents the last convolutional layer map, consisting of c channels, h rows, and w columns. $g(\cdot)$ represents our *CNN*, which takes as input an *ACF* tensor $\boldsymbol{a} \in \mathbb{R}^{10\times20\times20}$. To apply our grid loss, for each pose-specific detection template, we divide $\boldsymbol{x}$ into several small $c \times n \times n$ non-overlapping regions $\boldsymbol{r}_i$, $i = 1 \dots N$, where $N = \lceil \frac{h}{n} \rceil \cdot \lceil \frac{w}{n} \rceil$. Then, we optimize a hinge loss on each of these regions

$$\ell_{\text{local}}(\boldsymbol{\theta}) = \sum_{i=1}^{N} \max(0, m - y \cdot (\boldsymbol{w}_i^\top \boldsymbol{r}_i + b_i)), \tag{3.2}$$

where $m = \frac{1}{N}$ denotes the margin, $y \in \{-1, 1\}$ the class label and $\boldsymbol{w}_i$ and b_i are weights and bias for the i-th region. $\boldsymbol{\theta} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \dots, \boldsymbol{w}_N, b_1, b_2, \dots, b_N]$ are the parameters of the layer.

Some of these regions might correspond to non-informative areas of the face (*e.g.* flat regions such as the forehead). Such regions are typically not discriminative for detecting faces. Therefore, we propose to weight each of these regions, allowing the *CNN* to reduce the impact of non-informative regions in its final decision. To this end, we introduce a global classifier which shares weights with these local classifiers. Similar to our local classifiers, we train this global classifier with a hinge loss. Specifically, as our local classifiers are non-overlapping and span over the last convolutional layer, we just concatenate their weights $\boldsymbol{w}_i$, $i = 1, \dots, N$ and sum their biases b_i to obtain the global weight $\boldsymbol{w}$ and bias b. Formally, $\boldsymbol{w} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \dots, \boldsymbol{w}_N]$ and $b = \sum_{i=1}^{N} b_i$. Consequently, we compute the output of one pose-specific detection template from the *ACF* inputs as $f(\boldsymbol{a}) = \boldsymbol{w}^\top g(\boldsymbol{a}) + b = \sum_i (\boldsymbol{w}_i^\top \boldsymbol{r}_i + b_i)$.

During training, we minimize the following multitask loss

$$\ell(\boldsymbol{\theta}) = \max(0, 1 - y \cdot (\boldsymbol{w}^\top \boldsymbol{x} + b)) + \lambda \cdot \sum_{i=1}^{N} \max(0, m - y \cdot (\boldsymbol{w}_i^\top \boldsymbol{r}_i + b_i)), \tag{3.3}$$

where λ is a hyper-parameter balancing the influence of the global and local loss. We empirically set $\lambda = 1$, which we validate in our experiments (Section 3.3.3). During training, this setup only introduces a small number of additional parameters (*i.e.* $N - 1$ biases). Due to this setup, during test time, we only need to use the global classifier to detect faces. Consequently, compared to standard *CNNs*, we do not introduce any additional computational overhead with our grid loss.

During training, local and global classifiers which do not classifies a training sample correctly backpropagate errors back to the hidden layer of the *CNN*. They encourage the *CNN* to learn mid-level features with which the classifier can categorize the sample correctly. On the other hand, if the global or a local classifier classify a sample correctly, for this specific classifier no error is backpropagated to the hidden layer. Consequently, poorly performing local classifiers are strengthened during training, as we force the *CNN*

to develop features which help them to make local decisions. Subsequently, our region classifiers have a more uniform activation pattern, which we illustrate in Figure 3.3. As the detection ensemble does not focus on a small number of regions for classification, it is more robust to occlusions.

In contrast, with only a global loss function, the global classifier would stop backpropagating errors as soon as it finds a single feature which helps to distinguish the training sample from the background. Subsequently, our part-classifiers have a non-uniform activation patterns. Consequently, the classifier is less robust to occlusions. Our grid loss, however, encourages the *CNN* to develop further features as long as there are local classifiers which misclassify a sample.

3.2.2.1 Regularization Effect

We can interpret our grid loss is as an efficient model averaging of several part-based detectors. All detectors share a common feature representation. Consequently, our method exploits the benefits of ensembles to reduce over-fitting. We experimentally show in Section 3.3.6 that with decreasing training set size the performance of our method increases compared to standard loss functions.

Another interpretation of grid loss is that it creates better feature representations for *CNNs*. Ideally, features are discriminative and have a low correlation from each other. Consequently, they complement each other well when we use them in a classifier. We experimentally verify in Section 3.3.5 that our grid loss significantly reduces correlation in *CNN* features. This is due to the fact, that we force the *CNN* during training time to develop features for every facial region independently to distinguish faces from the background. In contrast, standard *CNNs* might solve this problem by developing only a few features. As *CNNs* are "lazy", once they find features with which they can classify the training set correctly, there is no inclination for them to learn new features. Consequently, they develop a highly redundant feature representation in which activations have a high correlation.

3.2.2.2 Deeply Supervised Nets

Deeply supervised networks [126, 212] apply the cross-entropy or hinge loss as an auxiliary loss during training on hidden layers of a network. Consequently, they encourage hidden layers of a *CNN* to be discriminative on their own. As hidden layers are already discriminative, successive layers in the network have a higher chance of distinguishing foreground vs background images. Inspired by this success, we apply our grid loss on top of hidden layers as auxiliary loss. As we show in our experiments in Section 3.3.1, this further improves the performance of our method without sacrificing speed during test time, as we use these auxiliary loss functions only during training time.

3.2.3 Face Localization Refinement

Standard fully convolutional detectors, which only rely on classification to predict the location of faces, tend to make a large number of localization errors. They predict high confidence boxes, which are not well aligned with the face in the image. Unfortunately, this results in high confidence false positive predictions. We address this problem by applying a post-hoc regressor, which refines the initial localization of the face. This allows our method to recover from such errors.

Further, detectors tend to detect parts of faces on higher or lower scales in the image pyramid. By applying our regressor, we can recover the correct face pose from also from such mislocalized detections. Consequently, when we apply our post-hoc regressor, we need to process a fewer number of intermediate scales in the image pyramid with our fully convolutional detector. This further increases the computational efficiency of our detector.

We illustrate our regressor in Figure 3.4. Similar to our detector, our regressor uses an ACF representation as input. We apply several 5×5 convolution operations with stride 1 followed by max-pooling operations of size 3×3 and stride 2. After 3 of such convolution and pooling layers, we apply a 2048 dimensional fully connected layer and an output layer. Similar to our classification network, we use $ReLUs$ as activation functions. On face datasets, where we only have bounding box annotations, our regressor just predicts bounding boxes (*i.e.* the x and y offsets and width and height of the bounding boxes). Some datasets also provide ellipse annotations. For such datasets, we train our regressor to predict ellipses. To this end, we parametrize our regressor to predict the length of the major and minor axis, the offset of the centroid, and the orientation of the ellipse.

The input size of our regressor is bigger compared to our detection network (*i.e.* 40×40 vs 20×20), as this helps our refinement network to better localize the face boundaries. We do not need to recompute the ACF pyramid to obtain these high-resolution inputs. We just project the initial bounding box location from our fully convolutional classification network into one octave lower in the ACF pyramid. We then crop the corresponding patch and feed it into our regression network to obtain the refined face location.

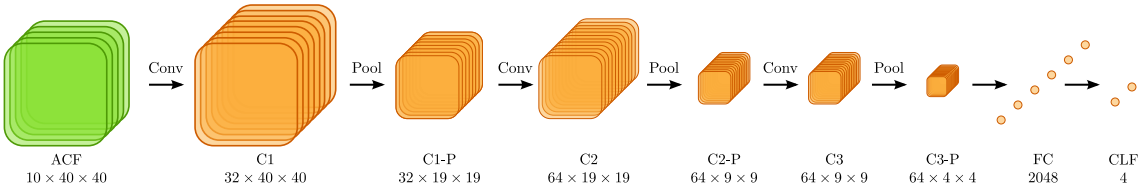


Figure 3.4: Illustration of our post-hoc regression network.

During training, we initialize the weights of our regressor from a Gaussian distribution with zero mean and 0.01 standard deviation. We apply dropout with a dropout rate of 0.5 on our 2048 dimensional fully connected layer. We train our network with Stochastic Gradient Descent (SGD) and momentum with an initial learning rate of 0.01 and a momentum of 0.9. Following standard training protocols, we anneal the learning rate after

50 and 100 epochs by one order of magnitude and stop training after 150 epochs.

For evaluation object detectors we use the PASCAL VOC overlap criterion to determine if a predicted bounding box or ellipse matches with the ground truth bounding box or ellipse. Specifically, in the evaluation, we define the overlap o_{VOC} for two faces F_a and F_b as

$$o_{\text{VOC}}(F_a, F_b) = \frac{|F_a \cap F_b|}{|F_a \cup F_b|}, \tag{3.4}$$

where $\cdot \cap \cdot$ denotes the intersection of two bounding boxes or ellipses, and $\cdot \cup \cdot$ denotes the union of two bounding boxes or ellipses. $|\cdot|$ denotes the area of the bounding box or ellipse.

When we predict ellipses as opposed to bounding boxes, the sensitivity of the overlap with the ground truth varies for different ellipse parameters. For example, changing the orientation of our prediction by one radiant impacts the overlap more than changing the minor axis length by one pixel. Optimizing the standard Sum of Squares Error (SSE) loss between predicted and ground-truth parameters (minor axis length, major axis length, orientation, centroid position), does not take this sensitivity into account. Similar to [177], we try to address this problem by optimizing the Intersection over Union (IoU) of prediction and ground truth directly.

An analytical estimation of the area of intersection between two ellipses requires complex algorithms, *e.g.* [88]. Therefore, we approximate the IoU numerically by central differences:

$$g_i(\boldsymbol{r}) \approx \frac{o_{\text{VOC}}(\boldsymbol{r} + \epsilon \cdot \boldsymbol{a}_i, \boldsymbol{y}) - o_{\text{VOC}}(\boldsymbol{r} - \epsilon \cdot \boldsymbol{a}_i, \boldsymbol{y})}{2 \cdot \epsilon}, \tag{3.5}$$

where g_i, $i = 1, \ldots, 5$ is the gradient of the 5 ellipse parameters. $\boldsymbol{r}$ and $\boldsymbol{y}$ is the prediction and ground truth of the ellipse, respectively. $\boldsymbol{a}_i$ denotes a one-hot vector where only the i-th entry is 1 and the remaining entries are 0. ϵ denotes the step size we use to approximate the gradient. To compute the area of intersection, we rasterize the predicted ellipse and the ground truth ellipse in a 40×40 pixel window. We choose ϵ big enough so that this rasterization changes at least by one pixel.

3.3 Evaluation

In this section, we do a detailed ablation study of our grid loss ensemble and compare it to the state-of-the-art. Specifically, we first compare our grid loss ensemble with holistic loss functions (*i.e.* a standard learner trained with hinge loss or logistic loss, see Section 3.3.1). We then evaluate the choice of the spatial size of our learners in the ensemble (Section 3.3.2) and the choice of our weighting parameter λ (Section 3.3.3). Next, we show the benefits of our grid loss ensemble on a dataset with occluded faces (Section 3.3.4). Further, we analyze how our grid loss ensemble reduces overfitting by increasing the diversity of the *CNN* feature representations (Section 3.3.5) and achieving higher accuracy on smaller training set sizes compared to a holistic learner trained with standard loss functions (Section 3.3.6).

Then, we evaluate the impact of our regressor (Section 3.3.7).

Finally, we compare our method to the state-of-the-art (Section 3.3.8). We also show a qualitative comparison of our detection ensemble trained with grid loss and our baseline model (Section 3.3.9).

In our experiments, we collect images from the Annotated Facial Landmarks in the Wild (AFLW) [108] dataset for training our detector. Specifically, we crop $15,106$ faces from the dataset, and resize them to 80×80 pixel windows in which 60×60 faces are visible. We follow the preprocessing and data augmentation steps of Mathias *et al.* [148]. We cluster faces into 5 discrete poses by yaw angle, which have pitch and roll between -22 and +22 degrees. As rotated faces are underrepresented in the dataset, we create rotated versions of each pose by rotating the images by 35 degrees. Further, we mirror faces and add the mirrored versions to the appropriate pose cluster to increase the number of positive faces in our dataset.

For feature extraction, we feed *ACF* to our *CNN*. We set the pre-smoothing radius to 1, the subsampling factor to 4, and the post-smoothing parameter to 0. As we subsample our features by a factor of 4 the input faces of our *CNN* have a size of 20×20. Further, as *ACF* features have 1 luminance channel, 2 color channels, 6 gradient orientation channels, and one gradient magnitude channel, our input tensor is of size $10 \times 20 \times 20$. Unless otherwise stated, our pyramid consists of 8 intermediate-scales per octave.

As we train our detector on positive and negative image patches, we perform hard negative mining to find negative face windows. Following [148], we first randomly sample $10,000$ easy negative samples form the non-person images of the PASCAL VOC dataset [47]. From this preliminary dataset, we train an initial detector. We use this detector to collect $10,000$ further hard negative windows on the non-person PASCAL VOC dataset and add them to our training set. We then finetune our detector on this enlarged dataset. After repeating this process 3 times, our detector cannot find any hard negatives on our negative images anymore and training converges.

To train our regressor, we use the face image patches of our training set. We apply data augmentation, *i.e.* mirroring the face and random translation, scaling on the input to make our regressor robust to the small localization errors of our detector. Further, we train our regressor on twice the resolution of our detector. During test-time, as we do not apply any post-smoothing to our feature pyramid, we just crop detected face windows from one octave lower than they are initially detected.

For evaluation, we benchmark our method on three public datasets, *i.e.* Face Detection Data Set and Benchmark (FDDB) [93], Annotated Faces in the Wild (AFW) [283] and PASCAL Faces [251]. We summarize the dataset statistics in Table 3.1.

3.3.1 Grid Loss Benefits

In this section, we show the benefits of our parameter shared ensemble trained with our grid loss compared to a standard *CNN* with the same architecture. We evaluate our

Dataset	Number of Images	Number of Faces	Annotation Type
FDDB [93]	2,845	5,171	Ellipse + Bounding Box
PASCAL Faces [251]	1,635	851	Bounding Box
AFW [283]	205	545	Bounding Box

Table 3.1: Statistics of our benchmark datasets for face detection.

CNN on the *FDDB* [93] dataset. For a fair comparison, we omit here our regression refinement network and only evaluate the detection *CNNs* in isolation. In our experiments, we evaluate two baseline loss functions, *i.e.* the logistic loss (cross-entropy) and the hinge loss. We then extend these loss functions by dividing our network at the last hidden layer into a detection ensemble. Further, we also explore the benefits of applying our loss function "deeply supervised", on the hidden layers as an auxiliary loss function to a *CNN*.

We report the true positive rate of all our detectors at a false positive count of $\{50, 100, 284(\approx 0.1\,FPPI), 500\}$ face windows. As we see in Table 3.2 and Figure 3.5, detection *CNNs* significantly benefit from our grid loss. Our ensemble improves the true positive rate over the standard hinge or logistic loss by 3.2% at 0.1 *FPPI* ($\widehat{=}$ 284 false positives). Finally, when we apply our grid loss additionally as an auxiliary loss on hidden layers (deeply supervised), we further can increase the true positive rate over the baseline by 1%.

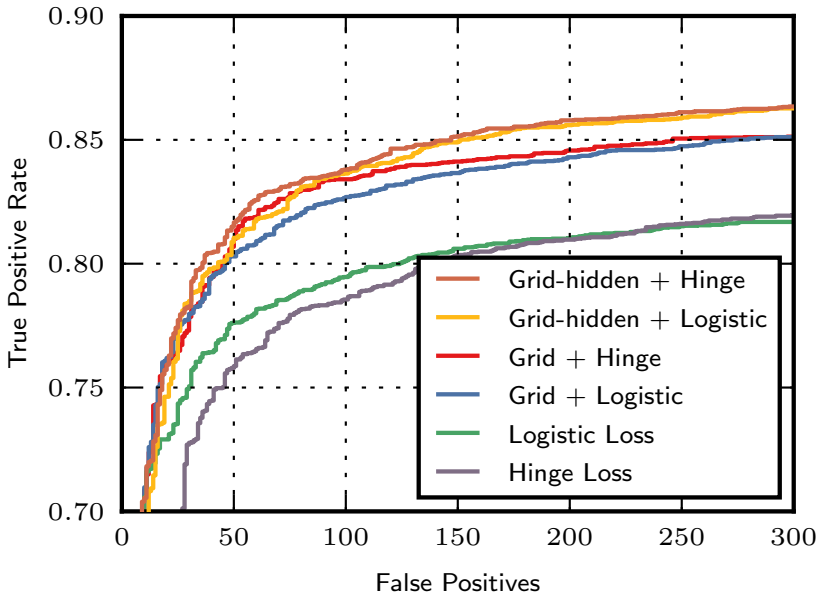


Figure 3.5: ROC curve of logistic, hinge, grid + logistic, grid + hinge, grid hidden + hinge and grid hidden + logistic loss on *FDDB*.

Method	50 FP	100 FP	284 FP	500 FP
L	0.776	0.795	0.817	0.824
H	0.758	0.786	0.819	0.831
G+L	0.803	0.827	0.851	0.859
G+H	0.807	0.834	0.851	0.858
G-h+L	<u>0.809</u>	<u>0.836</u>	<u>0.862</u>	<u>0.869</u>
G-h+H	**0.815**	**0.838**	**0.863**	**0.871**

Table 3.2: True positive rates of logistic (L), hinge (H), grid + logistic (G+L), grid + hinge (G+H), grid hidden + hinge (G-h+H) and grid hidden + logistic (G-h+L) loss functions on *FDDB* at a false positive (FP) count of 50, 100, 284 and 500. **Best** and <u>second best</u> results are highlighted.

3.3.2 Spatial Learner Size

In this section, we evaluate the spatial size of the learners in our parameter shared ensemble. To this end, we vary the spatial input of the input regions $\boldsymbol{r}_i$ of our learners. We apply our method also deeply supervised on hidden layers of our *CNN* and consider spatial sizes of $n = 2^{\{1,2,3,4\}}$. We summarize our results in Table 3.3 and Figure 3.6. Similar to our previous experiment, we report the true positive rates at $\{50, 100, 284, 500\}$ false positive detections. In our evaluation, we see that by constraining our learners to smaller spatial feature sizes, our detector achieves higher accuracy. In contrast, when we enlarge the region size to 16, our ensemble consists only of a small number of learners, which have access to the full receptive input field of the face. This model is similar to *CNN* trained with auxiliary loss functions on hidden layers, similar to [126, 212]. Consequently, this model cannot benefit from spatial independence and achieves lower detection accuracy.

Method	50 FP	100 FP	284 FP	500 FP
Block-2	**0.815**	**0.838**	**0.863**	**0.871**
Block-4	<u>0.812</u>	<u>0.834</u>	<u>0.852</u>	<u>0.861</u>
Block-8	0.790	0.809	0.830	0.838
Block-16	0.803	0.816	0.834	0.843

Table 3.3: Comparison of different block sizes on *FDDB*.

3.3.3 Weighting Parameter

In this section, we evaluate our weighting parameter λ, which balances the impact of our regional classifier to our global classifier (see Equation (3.3)). In our experiments, we jointly vary $\lambda = \{5, 1, 0.1, 0.05, 0.01, 0.005, 0.001\}$ and the spatial size of our learners $n = 2^{\{1,2,3,4\}}$ In Figure 3.7 we report the true positive rate of our detector at a positive count of 284 (≈ 0.1 *FPPI*) on the *FDDB* [93] dataset.

We see that our method works best with small block sizes of $2 - 4$ and $\lambda \approx 1$. When λ decreases to 0, the benefits of our parameter shared ensemble diminish, and our detector

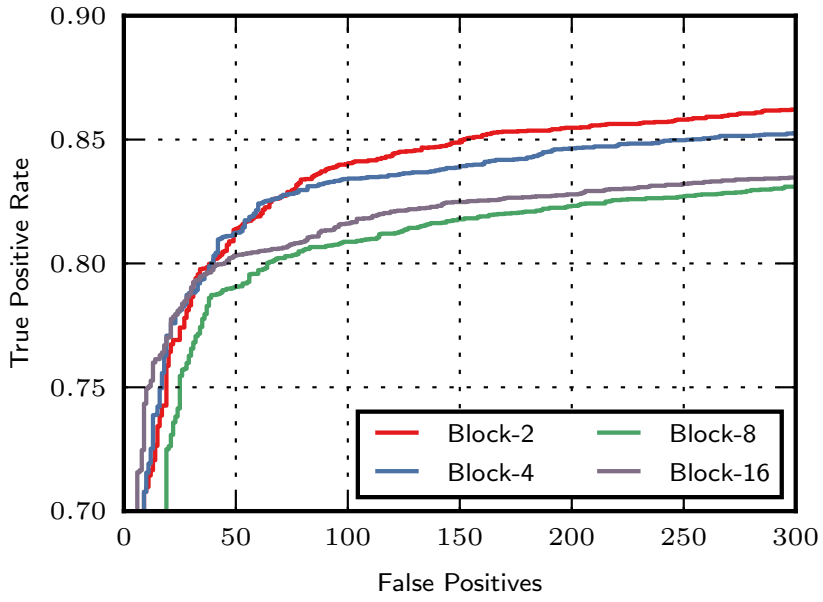


Figure 3.6: Comparison of different block sizes on *FDDB*.

degrades toward a standard *CNN* detector. Further, when we set λ too high, training becomes unstable, and performance degrades as well.

3.3.4 Robustness to Occlusions

In this section, we evaluate the robustness of our spatial independent ensemble to occlusions. Therefore, we need a dataset with occlusion annotations. As *FDDB* does not provide such annotations, we use the Caltech Occluded Faces in the Wild (COFW) dataset [20] to assess the performance of our detector on occluded faces. *COFW* provides occlusion annotations for facial landmarks. It is a benchmark dataset for landmark detection under occlusions. It consists of $1,852$ faces with landmarks and occlusion annotations. We split this dataset into 329 heavily occluded faces where at least 30% of all landmarks are occluded (*COFW*-HO) and $1,523$ faces where less than 30% of all landmarks are occluded (*COFW*-LO).

The *COFW* dataset is not designed for benchmarking face detection approaches, as there is no large background variation and the scale of the faces in the dataset is similar. Therefore, to evaluate our detectors, we use a high confidence threshold, which corresponds to a low false-positive rate on the *FDDB* [93] dataset. In contrast to *COFW*, *FDDB* is a benchmark designed for face detection approaches. Consequently, it has a large variation in scale and background. To compare our grid loss with standard *CNN*, we report the true positive rate on *COFW* at 0.1 *FPPI* on *FDDB*.

We report our results in Table 3.4. We see that both detectors achieve similar per-

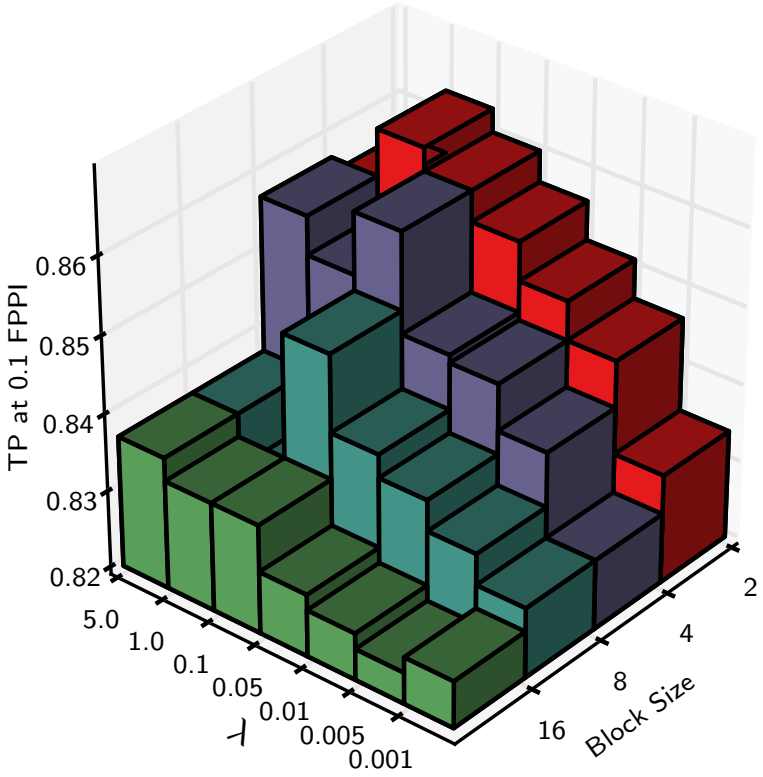


Figure 3.7: Evaluation of our weighting parameter λ.

formance on faces which are less than 30% occluded. However, on faces which have large occlusions (*i.e.* more than 30%), grid loss detectors outperform standard detectors by a large margin (*i.e.* by 7%). Therefore, this validates that our spatial independent ensemble is especially beneficial for detecting occluded faces compared to standard *CNNs*.

Even though our regional detectors may fail to detect parts of faces due to occlusions, the remaining detectors remain unaffected by occlusions. In contrast, standard *CNNs* tend to rely on certain prototypical regions to distinguish faces from the background. If these regions are missing due to occlusions, the *CNN* fails to detect the face.

Method	*COFW*-HO	*COFW*-LO
G	**0.979**	**0.998**
H	0.909	0.982

Table 3.4: True Positive Rate on *COFW* Heavily Occluded (*COFW*-HO) and Less Occluded (LO) subsets of a grid loss detector (G) and a hinge loss detector (H).

3.3.5 Effect on Correlation of Features

In this section, we analyze the benefits of grid loss for the diversity of our *CNN* features. Ideally, feature representations are highly discriminative and lowly correlated to each other (*i.e.* diverse), so that they complement each other well. By creating a spatial independent ensemble, we force each region classifier to develop discriminative features on their own. As regional classifiers have different receptive fields, the learned features are different from each other.

For example, one of our regional classifier might only have access to the eye region of the face. Therefore, our *CNN* has to learn an eye feature. A different regional classifier might have only access to the mouth region of the face. Therefore, our *CNN* has to also learn a mouth feature in addition to an eye feature.

In contrast, standard *CNNs* are lazy. As soon as they find a suitable feature, which is good enough to distinguish a face from the background (*e.g.* eye) on the training set, it has no inclination to develop further features.

More diverse features result in less correlated feature activations. For a given sample, different feature channels should be active for different mid-level features. In contrast, non-diverse features are highly correlated, as for a given input sample with a discriminative mid-level feature (*e.g.* eye), multiple activations are always active at the same time.

To measure this effect of grid loss on the feature representation we measure the correlation of our activations. As the last hidden layer of our *CNN* has a spatial size of 12×12 with 128 channels, we compute for each coordinate a normalized 128×128 correlation matrix. We then sum the absolute values of the off-diagonal elements of these matrices. If two features are independent of each other, their correlation is 0. If they are positively or negatively correlated, their corresponding off-diagonal elements are positive or negative, respectively. Therefore, a diverse (lowly correlated) feature representation has ideally a score of 0.

We report our results in Table 3.5. Our grid loss ensemble achieves two orders of magnitudes lower correlation score. Therefore, it significantly increases the diversity of feature representations compared to standard *CNNs*.

Method	Correlation ↓
Grid loss	**225.96**
Hinge loss	22500.25

Table 3.5: Grid loss reduces correlation in feature maps.

3.3.6 Training Set Size

In this section, we emphasize how our ensemble performs as a regularizer for face detectors. To this end, we subsample the positive training set by a factor of $0.75 - 0.01$ and evaluate the performance on the *FDDB* [93] dataset. We compare a detector trained with our grid

loss (*Grid*) to a standard *CNN* detector (*Hinge*). As deeply supervised networks [126, 212] also have a regularization effect, we also compare our loss to deeply supervised *CNNs*. Specifically, we train a apply a standard loss function as an auxiliary loss on the hidden layers of a standard *CNN* (*Hinge-hidden*). Further, we combine our grid loss with the idea of deeply supervised networks and apply grid loss on top of the hidden layers as well as on the output layer of a *CNN* (*Grid-hidden*).

We report the true positive rate of our models at 0.1 *FPPI* (284 false positives) in Table 3.6 and Figure 3.8. Interestingly, the performance gap between our spatially independent ensembles models trained with grid loss increases compared to standard *CNN* models trained with hinge loss as the training set gets smaller. Specifically, the gap increases from 3.2% to 10.2%, suggesting that our grid loss especially benefits *CNNs* when the training set size is small. Further, we see that *CNNs* also benefit from grid loss applied as an auxiliary loss function on hidden layers.

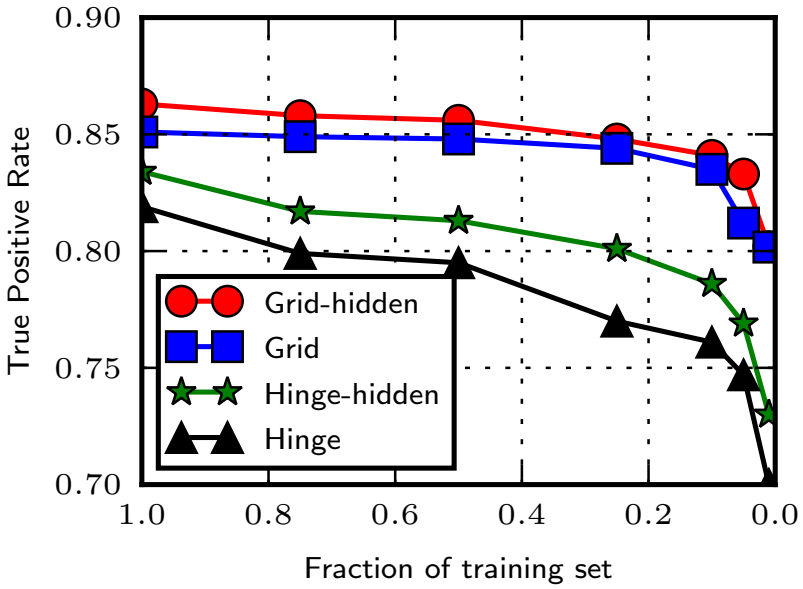


Figure 3.8: Impact of different training sub-sets of the positive training set on *FPPI* at 0.1 *FPPI* using the hinge loss (Hinge), hinge loss on hidden layers (Hinge-hidden) and our grid loss (Grid) and grid loss on hidden layers (Grid-hidden)

3.3.7 Ellipse Regressor and Layer Skipping

Fully convolutional detectors tend to make localization errors. They predict high confidence bounding boxes, which miss the ground truth face by a small margin. In this section, we show that post-hoc refinement regressors can overcome this problem. Further, we show that applying such regressors also allows us to reduce the image pyramid size by

M	1.00	0.75	0.50	0.25	0.10	0.05	0.01
G-h	**0.863**	**0.858**	**0.856**	**0.848**	**0.841**	**0.833**	**0.802**
G	0.851	0.849	0.848	0.844	0.835	0.812	0.802
H-h	0.834	0.817	0.813	0.801	0.786	0.769	0.730
H	0.819	0.799	0.795	0.770	0.761	0.747	0.700

Table 3.6: Impact of training on a sub-set (i.e. 0.75 - 0.01) of the positive training set on *FDDB* at 0.1 *FPPI* using the hinge loss (H), hinge loss on hidden layers (H-h) and our grid loss (G) and grid loss on hidden layers (G-h).

omitting several intermediate scales. Detectors are also sensitive to faces which are smaller or bigger than the input receptive field. Therefore, they make predictions on scales which are above or below the ground truth face. Regressors can also overcome these localization errors. Consequently, we can reduce the number of intermediate scales in the scale-space pyramid, when we use a regressor.

In Table 3.8 and Figure 3.10, we report the true-positive rate of detectors when we apply them densely across all scales (D) or skip intermediate scales (S) on the *FDDB* [93] dataset. Further, we also compare a regressor optimized with *SSE* loss and a numerical IoU loss (NUM). We see that especially at a low false-positive count our regressor improves the true-positive rate. The main reason for this is that our regressors can realign high confidence localization errors from our detector. Further, even when we omit every second intermediate scale in the image pyramid, our post-hoc regressor improves over a densely applied detector.

We also evaluate our regressor on the continuous evaluation protocol of *FDDB*. In this evaluation protocol, matches of ground truth faces and predicted faces are weighted bey their PASCAL IoU score. In Table 3.7 and Figure 3.9, we see that by optimizing the IoU score numerically, we achieve a small improvement (*i.e.* 0.1% to 0.2%) compared to optimizing the *SSE* loss.

Method	50 FP	100 FP	284 FP	500 FP	1000 FP
NUM (D)	**0.680**	**0.690**	**0.702**	**0.708**	**0.714**
SSE (D)	0.679	0.688	0.700	0.706	0.713

Table 3.7: Continuous evaluation of the two proposed ellipse loss functions: Numerical PASCAL VOC overlap (NUM) and *SSE* on FDDB.

3.3.8 Comparison to the State-of-the-Art

We compare our detector to the state-of-the-art at time of publication on the FDDB dataset [93], the AFW dataset [283] and PASCAL Faces dataset [251], see Figs. 3.11, 3.12 and 3.13. For evaluation on AFW and PASCAL Faces we use the evaluation toolbox provided by [148]. For evaluation on FDDB we use the original evaluation tool provided by [93]. We report the accuracy of our small fast model and our large model. On FDDB

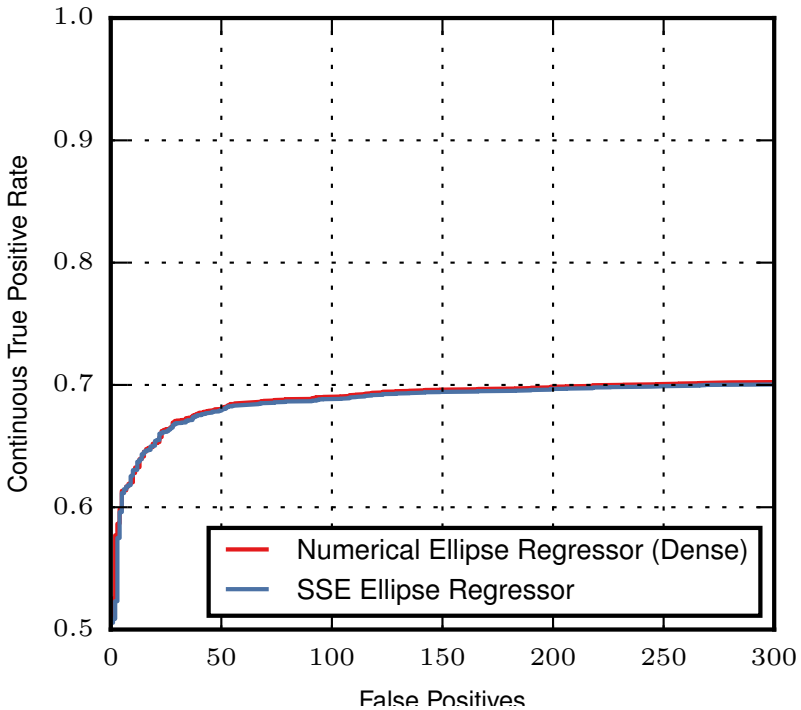


Figure 3.9: Evaluation of our ellipse regressors on FDDB (continuous score).

Method	50 FP	100 FP	284 FP	500 FP
NUM (D)	0.843	**0.857**	**0.872**	**0.879**
NUM (S)	0.835	0.851	0.867	0.874
SSE (D)	**0.844**	**0.857**	**0.872**	0.878
SSE (S)	0.835	0.848	0.866	0.873
w/o (D)	0.815	0.838	0.863	0.871

Table 3.8: Effect of numerical loss (NUM), *SSE* loss (SSE) and no ellipse regressor (w/o) applied densely (D) on all pyramid levels or skipping (S) layers on FDDB.

our fast network combined with our regressor retrieves 86.7% of all faces at a false positive count of 284, which corresponds to about 0.1 *FPPI* on this dataset. With our larger model we can improve the true positive rate to 89.4% at 0.1 *FPPI*, outperforming the state-of-the-art at the time of publication by 0.7%. On PASCAL Faces and AFW we outperform the state-of-the-art by 1.38% and 1.45% Average Precision (AP) respectively.

3.3.9 Qualitative Comparison

In this section, we qualitatively compare our spatial independent ensemble with standard *CNNs*. We first show several qualitative results which show that grid loss compares favorably to standard *CNNs*. As we see in Figure 3.14, our grid loss *CNN* achieves higher detection rates for occluded faces compared to standard *CNNs*.

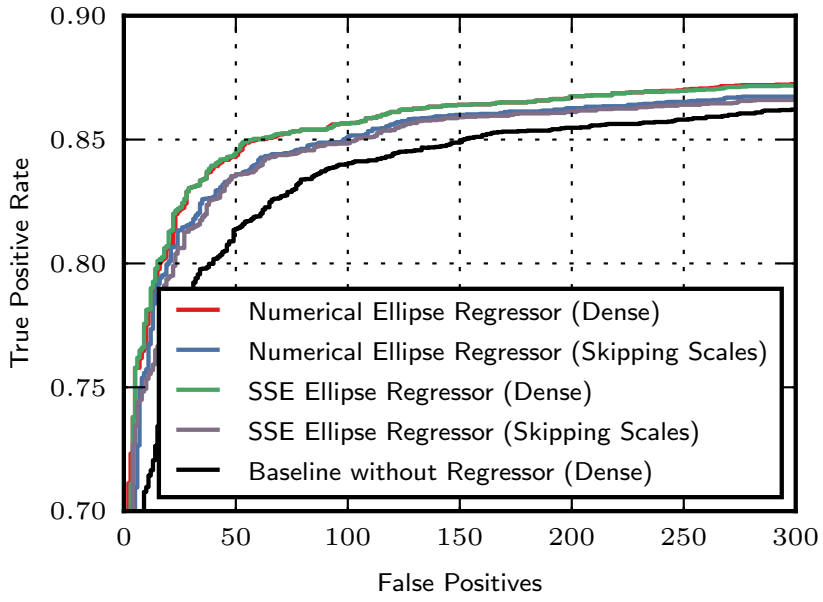


Figure 3.10: Evaluation of our ellipse regressors on FDDB (discrete score) with and without layer skipping.

Further, we visualize the learned detection templates of a baseline *CNN* and with a *CNN* trained with our grid loss. To this end, we use a visualization method [204] to analyze what our networks have learned. Specifically, we seek pixels in the input, which maximize the prediction of the respective pose-specific classifiers. Formally, as we train our network on an *ACF* representation, we seek the input tensor $\boldsymbol{a} \in \mathbb{R}^{10\times20\times20}$ which maximizes

$$\hat{\boldsymbol{a}} = \arg\max_{\boldsymbol{a}} f(\boldsymbol{a}) - \lambda_{\text{viz}} ||\boldsymbol{a}||_2^2 , \tag{3.6}$$

where $f(\cdot)$ denotes the output of a pose-specific classifier and λ_{viz} is a regularization parameter.

As the *ACF* representation has 10 channels, we visualize the channel-wise maximum, *i.e.*

$$\boldsymbol{I}(i,j) = \max_{c=1,\ldots,10} \hat{\boldsymbol{a}}(c,i,j). \tag{3.7}$$

$\boldsymbol{I} \in \mathbb{R}^{20\times20}$ denotes the visualization image for a single pose specific detection template. It has high pixel values, if the corresponding pixel contributes to a positive detection result.

We illustrate our visualizations in Figure 3.15. Each detection template corresponds to a different face pose. Our grid loss templates focus especially on cheeks and the nose of the face, suggesting that these regions are most important for distinguishing faces from the background. In contrast, the detection templates trained with a standard loss function tend to be noisier compared to templates trained with grid loss.

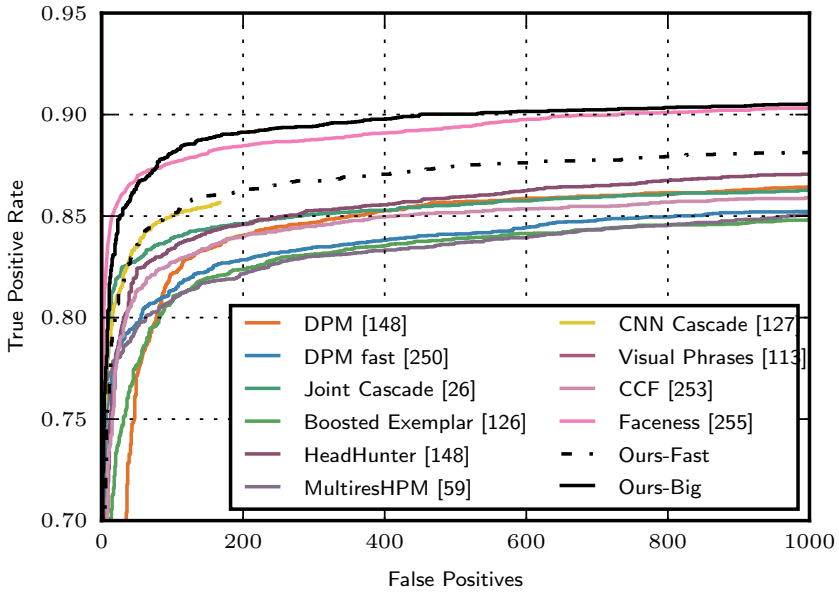


Figure 3.11: Discrete evaluation on the FDDB [93] dataset.

3.3.10 Computational Efficiency

As we use only very small neural networks, our detector can run in realtime on a standard CPU. Specifically, we implement our method with Theano [6] and Python. For our experiments, we use a desktop machine with a 3.20 GHz Intel Core i5 CPU. Our small dense model with skipping intermediate scales runs in 170 ms on 640×480 pixels. This is competitive to fast tree-based boosting methods, *e.g.* [148, 252], while outperforming them in terms of accuracy.

On a NVIDIA GTX 770 GPU, our method runs in 200 ms when we do not use a post-hoc regressor and apply our network densely on all intermediate scales. When we add a regressor and skip intermediate scales, it runs in about 50 ms per image using non-optimized Python code.

Our method can also benefit from recent speedup techniques such as image patchwork [43, 62], factorization methods, *e.g.* [92, 275], or detector cascades [127].

3.4 Summary

In this chapter, we presented a novel parameter shared spatial independent ensemble for face detection. Compared to standard *CNNs*, our method performs favorably, especially on occluded faces.

The main reason for this is that standard *CNNs* are lazy in developing diverse features. As soon as they develop a few discriminative features, which can correctly classify the

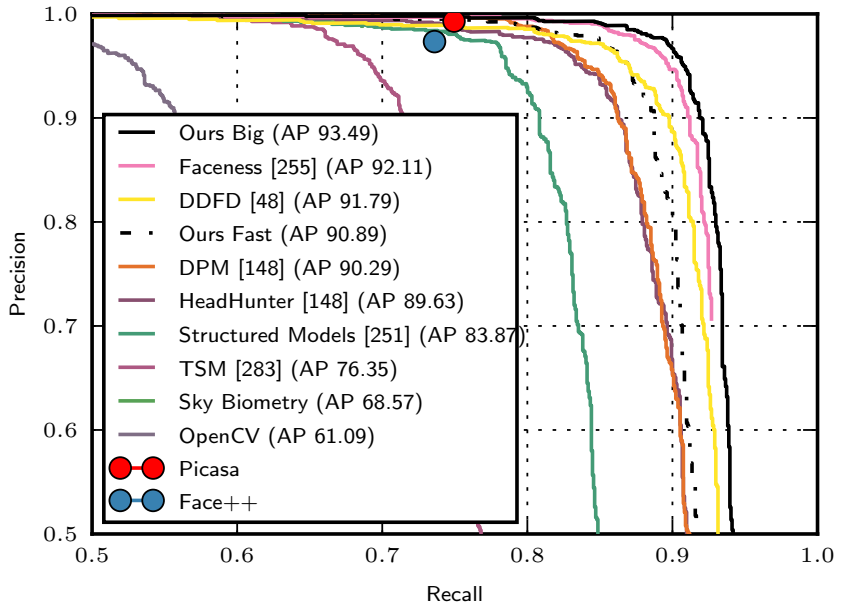


Figure 3.12: Evaluation on the PASCAL Faces [251] dataset.

training set, they have no inclination to develop further features. However, if these few prototypical features are missing during test time due to *e.g.* occlusions, standard *CNNs* fail.

In contrast, our spatial independent *CNN* ensemble forces a *CNN* to develop more diverse features. It uses a simple loss function, called grid loss, which optimizes a discriminative loss on spatial sub-regions of a *CNN* feature map. Each of these sub-regions has a spatial different receptive field. Consequently, each of these regions needs to be discriminative on its own. Therefore, grid loss forces our *CNN* ensemble to learn a set of diverse features. If some of these features are missing during test time due to *e.g.* occlusions, our ensemble can still recover.

At the time of publication, our method achieved state-of-the-art performance on public face detection benchmarks, while using only very small network architectures.

One of the main limitations of our work is that we need to restrict the receptive field of each of our learners in the ensemble. Consequently, the receptive field of neurons in the last hidden layer of the *CNN* needs to be small. Further, specific object parts such as eye, nose, *etc.* need to be in the receptive field of the same learner for all training samples. This is typically only possible when the pose-information of objects is available and objects are not deformable.

However, modern *CNN* architectures trained on ImageNet [186] typically have very large receptive fields. Subsequently, this allows them to use rich context information of the scene more effectively. Further, pose annotations for objects might be expensive to collect

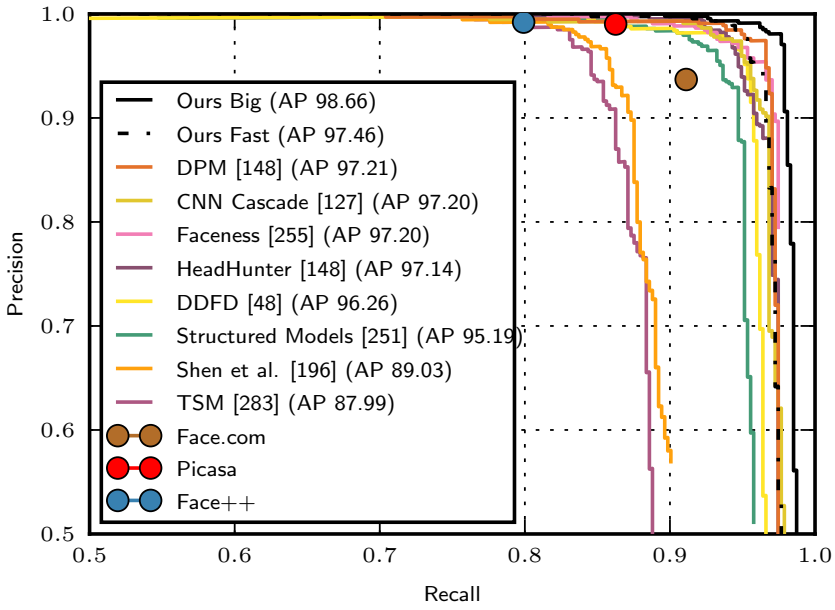
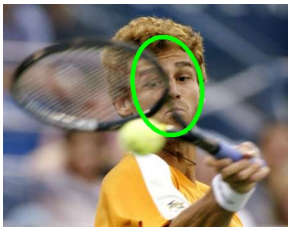


Figure 3.13: Our method outperforms state-of-the-art methods on AFW [283].

and some problems might require detecting highly deformable objects. To address this issue, we propose a different strategy in Chapter 4. We use auxiliary loss functions and re-weighting to encourage different learners in the ensemble to be diverse from each other. As we show in our experiments, these methods can benefit a wide variety of different *CNN* architectures.

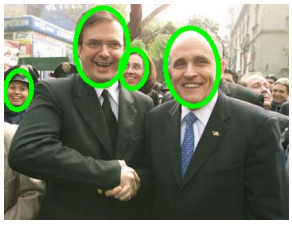
Another way to overcome these limitations might be spatial attention, *e.g.* [225], combined with auxiliary loss functions to encourage each learner to use different attention masks for their predictions. However, to ensure that learners are spatially independent, these attention masks need to be inserted after one of the first layers of the *CNN*. Otherwise, the receptive fields of these attention masks would be too large. Consequently, there is a large computational overhead per additional learner, as we need to split the shared ensemble after these attention masks.



(a) Grid loss.

(b) Standard loss.

(c) Grid loss.



(d) Standard loss.

(e) Grid loss.

(f) Standard loss.

(g) Grid loss.

(h) Standard loss.

(i) Grid loss.

(j) Standard loss.

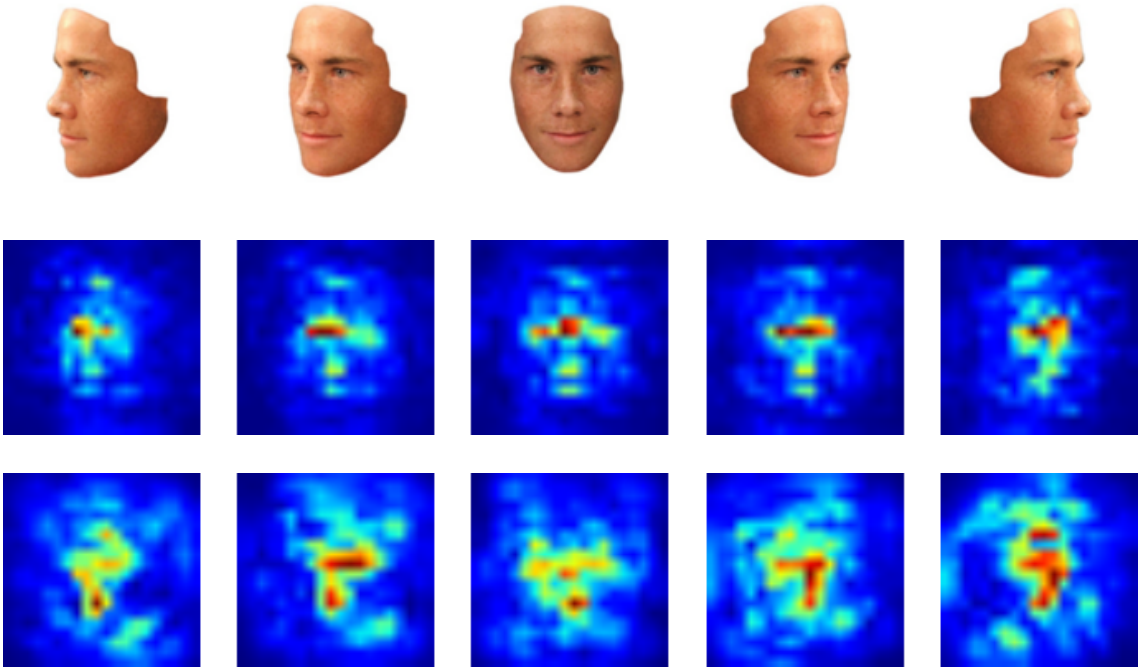
(k) Grid loss.

(l) Standard loss.

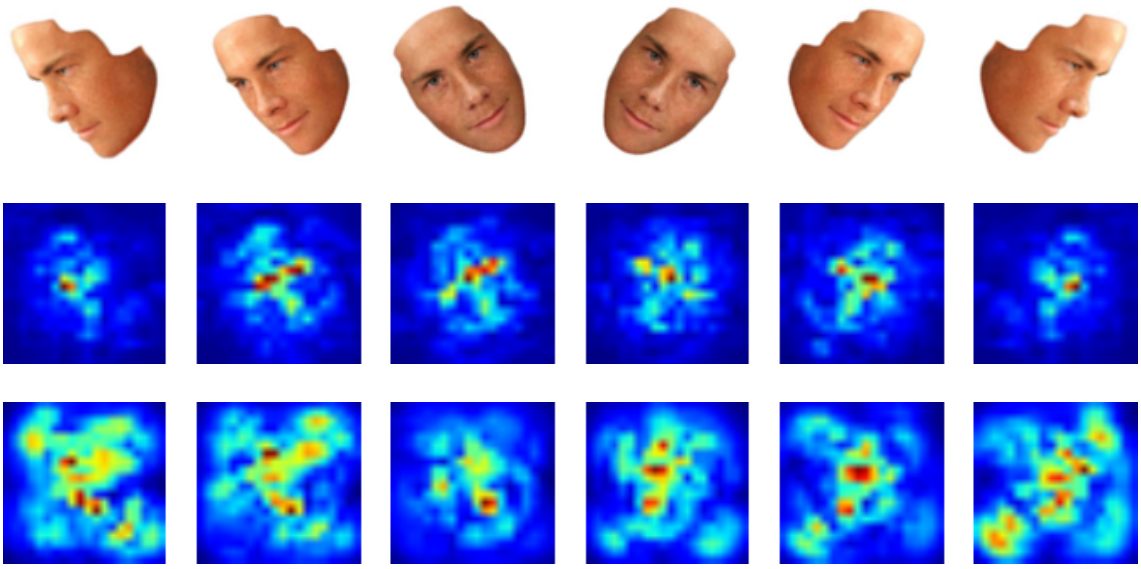
(m) Grid loss.

(n) Standard loss.

Figure 3.14: Qualitative comparison between a parameter shared spatial independent *CNN* ensemble trained with grid loss and a standard *CNN*.



(a) Detection templates for 5 different face poses (top) using our grid loss (middle) vs standard loss (bottom)



(b) Detection templates for ±35° rotated poses (top) using our grid loss (middle) vs standard loss (bottom)

Figure 3.15: Comparison of our detection templates learned with our grid loss with our detection templates learned with a standard loss function.

4

Diversity Loss Functions for Ensembles

"Solutions nearly always come from the direction you least expect, which means there's no point trying to look in that direction because it won't be coming from there."

— Douglas Noel Adams

Contents

4.1 Motivation 65
4.2 Efficient Ensembles for Object Categorization 67
4.3 Efficient Ensembles for Metric Learning 69
4.4 Auxiliary Loss Functions 76
4.5 Loss Function on Multiple Hidden Layers 80
4.6 Image Categorization Experiments 81
4.7 Metric Learning Experiments 88
4.8 Summary 103

4.1 Motivation

In this chapter, we introduce several diversity encouraging auxiliary loss functions for Convolutional Neural Network (CNN) ensembles with shared feature representation. Further, we extend our method to real-world metric learning applications (Section 4.3). We originally presented these loss functions in our previous works [165, 167, 168].

Traditional *CNN* ensembles use variation in model architecture and random initialization, *e.g.* [75, 203], to make learners diverse from each other. However, such ensembles

typically consists of several large *CNNs*, which makes training and inference time computationally expensive.

To overcome these issues TreeNets [119] and dropout [161] use parameter sharing to reduce the computational expense of *CNN* ensembles. Similar to these works we propose to train an ensemble by sharing most parameters of the learners with each other. More specifically, we divide the network at a hidden layer at the end into multiple groups and optimize a classification head for each of these groups separately. All heads share a common low-level feature representation. During training- and test-time, such an ensemble is computationally very competitive to a single *CNN*, as the learners share the computationally most expensive intermediate responses during training- and test-time. In fact, if we divide the network on the last hidden layer, we can map it back to a single neural network during test-time, causing no computational overhead at all.

However, especially for pre-trained neural networks, such a setup typically yields highly correlated learners, as, during training, all learners are optimized on the same permutation of the training set and share all their features. In contrast to dropout, which overcomes this issue by randomly omitting features during training time, we introduce diversity by auxiliary loss functions. With this loss function, we can balance the discriminativeness of the learners and their correlation to each other. To this end, we propose *DivLoss* [165] (Section 4.4.1), which introduces diversity on the classifiers.

Further, we extend our method to metric learning for image retrieval on large pre-trained neural networks (Section 4.3). The goal of image retrieval methods is to retrieve semantically similar images from a database of images. To this end, they map an image to a feature vector (also called "embedding"). In this feature space, these methods use a query image to retrieve similar images from a database (*e.g.* images of the same product category or the same car) by computing distances.

Existing metric learning approaches typically learn highly correlated embeddings. Consequently, they do not utilize the capacity of the embedding layer effectively. Our method addresses this problem by reducing the correlation of the final embedding layer. To this end, we divide the embedding layer into several learners. We pose training this ensemble as an online gradient boosting problem. Further, we also apply diversity encouraging auxiliary loss functions. As our *DivLoss* for classification is not able to make generic vectors or hidden layer representations diverse from each other, we propose two novel loss functions which address this problem. Our *ActLoss* [167, 168] (Section 4.4.2) imposes a group sparsity constraint on the hidden layer activations of the ensemble. Our *AdvLoss* [168] (Section 4.4.3) is an adversarial loss function which makes the distribution of hidden layer representations of learners dissimilar. Consequently, these loss functions are more general than our *DivLoss* and also applicable to classification problems.

As we show in our experiments for object categorization (Section 4.6), our auxiliary loss functions perform favorably compared to dropout regularization for simple object categorization tasks. The main reason for that is that the learners in our ensemble are stronger (*i.e.* more accurate) and less correlated compared to the learners of dropout.

Further, we show that our method also benefits recent ResNet architectures on the CIFAR-10 [111], CIFAR-100 [111] and SVHN [159] datasets (Section 4.6.5). As we see in our experiments, by incorporating our ensemble approaches, efficient ResNet ensembles are computationally more efficient compared to widened ResNets (*i.e.* increasing the convolutional channels by a factor $k = \{2, 3, 4\}$). Consequently, with a given computational budget, we typically can achieve higher accuracy with efficient ResNet ensembles.

In our metric learning experiments, we show that gradient boosting-based re-weighting combined with our auxiliary loss functions reduce the correlation in the embedding layer and consequently significantly improve the retrieval performance of metric learning networks (Section 4.7).

In the remainder of this chapter, we first present our ensemble approach for object categorization (Section 4.2). Then, we present our metric learning extension (Section 4.3). Next, we discuss our auxiliary loss function (Section 4.4). We show the effectiveness of our approaches in a detailed evaluation for both classification problems Section 4.6) and metric learning problems (Section 4.7). Finally, we conclude this chapter with a brief summary (Section 4.8).

4.2 Efficient Ensembles for Object Categorization

In this section, we apply our efficient ensembles to the problem of object categorization. To reduce the computational costs of the ensembles, we propose an architecture in which we share most parameters among learners (see Figure 4.1). All our learners share most hidden layers with each other, which reduces parameters and computational cost during training- and test-time. For most of our experiments, we divide the last hidden layer into several non-overlapping groups. With this setup, we can map our ensemble back to a regular neural network at test-time. Therefore, our method does not impair any computational costs during test-time. In general, however, groups might overlap with each other, or might consist of several hidden layers. This allows more diverse learners at a modest increase in computational cost compared to standard ensembles.

For each group, we add a separate classification head, which represents the learners in our ensemble. During training time, we optimize a loss function for each of the learners. In contrast to standard ensembles, where learners are traditionally trained independently from each other, we train our model jointly end-to-end.

We formally discuss here the simplest and most efficient architectural choice, *i.e.* dividing the last hidden layer into several learners, to avoid cluttering the notation. Let $\boldsymbol{x} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_M] \in \mathbb{R}^H$ denote the last hidden layer of our neural network, which we divide into M non-overlapping groups $\boldsymbol{x}_m \in \mathbb{R}^C$, $1 \leq m \leq M$. The hidden layer consists of H neurons, which we divide into M groups consisting of C neurons, therefore $C = \frac{H}{M}$. Further, let $\boldsymbol{W} \in \mathbb{R}^{H \times D}$ denote the output weight matrix, which predicts the logits for

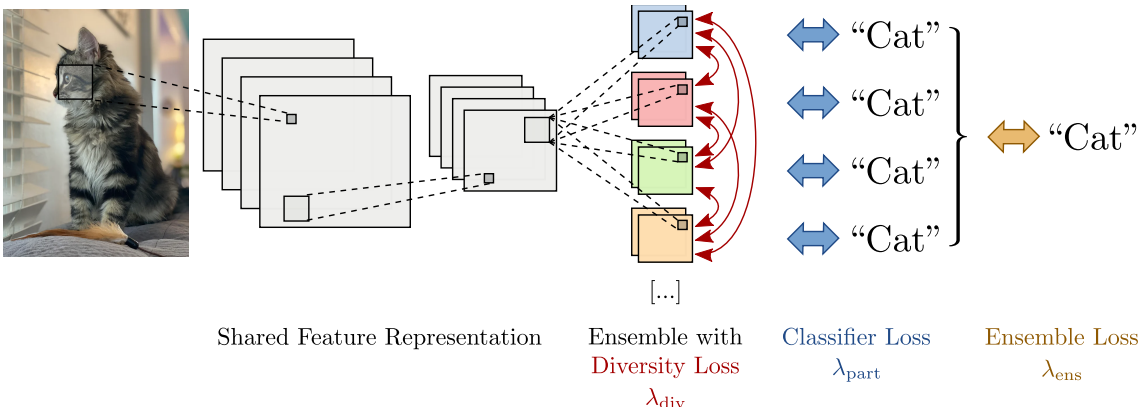


Figure 4.1: Our ensemble architecture divides the network at the end into several learners. For each learner we add a classification head. We use an auxiliary loss function to make these learners diverse from each other.

the D categories. Similar to $\boldsymbol{x}$, we can partition $\boldsymbol{W}$ into several sub-matrices as follows:

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{W}_1 \\ \boldsymbol{W}_2 \\ \cdots \\ \boldsymbol{W}_M \end{bmatrix}, \tag{4.1}$$

where $\boldsymbol{W}_m \in \mathbb{R}^{C \times D}$ is the weight matrix for the m-th learner. We then can compute the logits $\boldsymbol{c}_m$ (*i.e.* the inputs to the last softmax layer) for the m-th learner as

$$\boldsymbol{c}_m = \boldsymbol{x}_m \boldsymbol{W}_m + \boldsymbol{b}_m, \tag{4.2}$$

where $\boldsymbol{b}_m$ denotes the bias for the m-th learner.

Further, the ensemble logit is the average of the individual learners, *i.e.*

$$\boldsymbol{o} = \frac{1}{M} \sum_{m=1}^{M} \boldsymbol{c}_m. \tag{4.3}$$

We compute the final classifier output with the softmax over the ensemble logits $\boldsymbol{o}$. This is proportional to the geometric mean of the individual classifier softmax predictions. Let $\boldsymbol{o}_j$ denote the output for the j-th class, then the following holds:

$$\sigma(\boldsymbol{o}_j) = \frac{e^{\frac{1}{M}\sum_{m=1}^{M} \boldsymbol{c}_{mj}}}{Z} = \frac{\left(\prod_{m=1}^{M} e^{\boldsymbol{c}_{mj}}\right)^{\frac{1}{M}}}{Z} \tag{4.4}$$

$$= \frac{\left(\prod_{m=1}^{M} \sigma(\boldsymbol{c}_m)_j \cdot Z_m\right)^{\frac{1}{M}}}{Z} = \frac{\left(\prod_{m=1}^{M} \sigma(\boldsymbol{c}_m)_j\right)^{\frac{1}{M}}}{\hat{Z}}, \tag{4.5}$$

where Z denotes the ensemble softmax normalization factor, Z_m the softmax normalization factor for the m-th learner and

$$\hat{Z} = \frac{Z}{\left(\prod_{m=1}^{M} Z_m\right)^{\frac{1}{M}}} = \frac{\sum_{j=1}^{D}\left[\left(\prod_{m=1}^{M} Z_m\right)^{\frac{1}{M}}\left(\prod_{m=1}^{M}\sigma(\boldsymbol{c}_m)_j\right)^{\frac{1}{M}}\right]}{\left(\prod_{m=1}^{M} Z_m\right)^{\frac{1}{M}}} \tag{4.6}$$

$$= \sum_{j=1}^{D}\left(\prod_{m=1}^{M}\sigma(\boldsymbol{c}_m)_j\right)^{\frac{1}{M}}. \tag{4.7}$$

The normalizations Z_m can be pulled out of the sum and then cancels with the denominator of the equation. Therefore, we see that the ensemble output is proportional to the geometric mean of the predictions of the individual learners.

To train our ensemble, we make the full ensemble discriminiative as well as the individual learners discriminative. To this end, we optimize the following loss function:

$$\mathcal{L}_{\text{discr}} = \underbrace{\lambda_{\text{ens}}\cdot\mathcal{H}(\boldsymbol{y},\sigma(\boldsymbol{o}))}_{\text{ensemble loss}} + \lambda_{\text{part}}\cdot\left(\frac{1}{M}\sum_{m=1}^{M}\underbrace{\mathcal{H}(\boldsymbol{y},\sigma(\boldsymbol{c}_m))}_{m\text{-th learners loss}}\right), \tag{4.8}$$

where $\boldsymbol{y}$ denotes the one-hot encoded ground-truth label and $\boldsymbol{c}_m$, $\boldsymbol{o}$ are computed from the hidden layer response $\boldsymbol{x}$. Further, $\mathcal{H}(\cdot,\cdot)$ denotes the cross entropy loss:

$$\mathcal{H}(\boldsymbol{y},\boldsymbol{p}) = -\sum_{j=1}^{D}\boldsymbol{y}_j\cdot\log(\boldsymbol{p}_j). \tag{4.9}$$

The parameters $\lambda_{\text{part}}, \lambda_{\text{ens}}$ are hyper-parameters which can be set via (cross-)validation. They balance the discriminativeness of the individual learners and the discriminativeness of the full ensemble prediction.

4.3 Efficient Ensembles for Metric Learning

In this section, we extend our ensemble approach to the problem of metric learning for image retrieval. We originally presented this extension in [167, 168]. Metric learning approaches learn a semantic similarity function between images, where "similar" images should be close to each other in metric space and "dissimilar" images should be far apart from each other. Typically, these methods use specific loss functions operating on image pairs (*e.g.* [32, 71]), triplets (*e.g.* [191, 245]) or quadruples (*e.g.* [116, 279]) to learn such embeddings. Another line of work poses metric learning as a classification problem (*e.g.* [39, 264]). They learn a large classifier over thousands of categories with specialized softmax loss functions. After training, they remove the classification layer and use the last hidden layer as an embedding layer.

To avoid cluttering the notation, we focus our discussions in this section on loss functions operating on pairs. However, our method is also compatible with loss functions operating on triplets, quadruplets, or n-tuples as well as classification loss functions.

We illustrate our learning approach, called Boosting Independent Embeddings Robustly (BIER), in Figure 4.2. Similar to our ensemble method for categorization, for metric learning, we divide a network in the end into several learners. Specifically, we use a pre-trained ImageNet model and add an embedding layer at the end. We split this embedding layer into several smaller layers, where each one corresponds to a learner in a parameter shared ensemble. To make this ensemble diverse, we employ two strategies.

First, we propose online gradient boosting, which introduces weights for each training sample. During training, each learner reweights the training samples for successive learners in the ensemble with the negative gradient of the loss function. Consequently, successive learners focus on samples which are misclassified by the previous learners.

Second, we employ one of our auxiliary loss functions, which can operate on the hidden layer, to make learners diverse from each other (see Section 4.4). Specifically, we either use our Activation Loss (Section 4.4.2) or our Adversarial Loss (Section 4.4.3) on the last embedding layer of the network to make our learners diverse from each other. Our standard *BIER* [167] uses these auxiliary loss functions only during initialization for finding the initial weights of the embedding layers of the individual learners. We later also include these loss functions as auxiliary loss during training [168].

During test-time, the final embedding is the concatenation of the embeddings of the individual learners. Consequently, our method only introduces marginal computational overhead compared to standard metric learning *CNNs* (*i.e.* only concatenation).

In the following, we briefly review our baseline *CNN* based metric learning system (Section 4.3.1) and discuss our boosting based extension (Section 4.3.2).

4.3.1 Metric Learning for Deep Learning

In this section, we give an overview of our baseline metric learning system. The main objective of *CNNs* for metric learning is to map an image $\boldsymbol{x}$ with a *CNNs* $f(\cdot)$ to a feature space $\mathbb{R}^d$. In this space similar images should be close to each other and dissimilar images should be far apart from each other. The last layer of such *CNN* is called embedding layer and maps the last hidden layer of size h to the embedding space $\mathbb{R}^d$.

To measure similarity $s(\cdot, \cdot)$ between two images $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$, we use the cosine distance between the two feature vectors, *i.e.*

$$s(f(\boldsymbol{x}^{(1)}), f(\boldsymbol{x}^{(2)})) = \frac{f(\boldsymbol{x}^{(1)})^\top f(\boldsymbol{x}^{(2)})}{\left|\left|f(\boldsymbol{x}^{(1)})\right|\right| \cdot \left|\left|f(\boldsymbol{x}^{(2)})\right|\right|}. \tag{4.10}$$

Compared to Euclidean distance, the cosine similarity has the advantage that the similarity score is bounded between $[-1, +1]$.

During training, we follow recent work, *e.g.* [163, 191, 223], and sample a minibatch

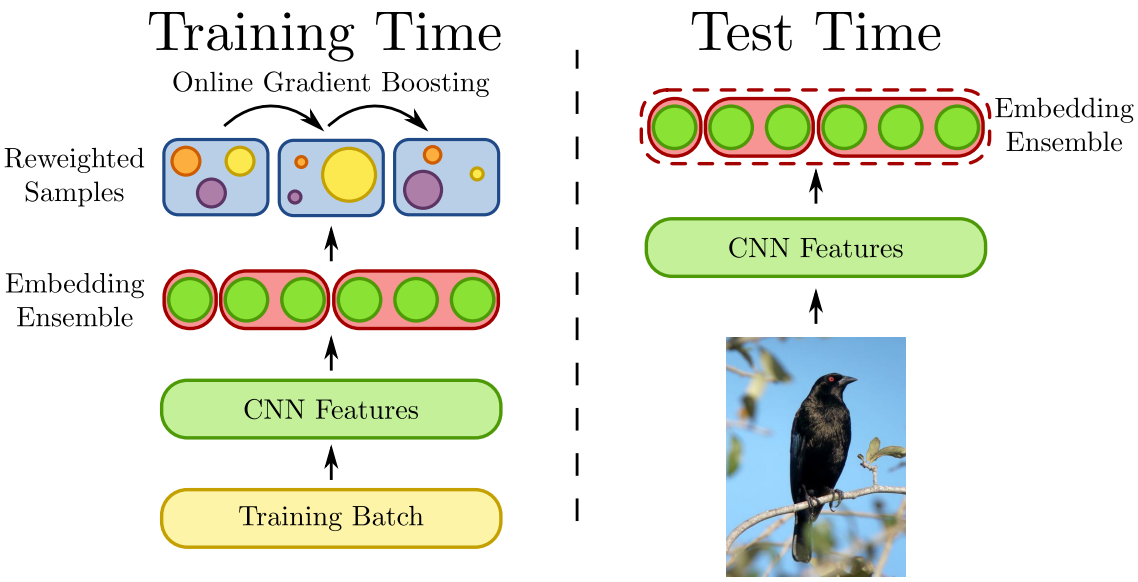


Figure 4.2: BIER divides a large embedding into an ensemble of several smaller embeddings. During training we reweight the training set for successive learners in the ensemble with the negative gradient of the loss function. During test-time we concatenate the individual embeddings of all learners into a single embedding vector.

of N samples. We compute the similarity score $s(\cdot,\cdot)$ between all pairs of samples in the batch. We then use this similarity matrix $S \in \mathbb{R}^{N\times N}$ to optimize a loss function between pairs or triplets of samples. In contrast to previous works, *e.g.* [32, 71], which employ a Siamese *CNN* architecture, this approach does not need to keep several copies of the *CNN* in memory. Therefore, we can decrease memory consumption and increase computational efficiency.

In our experiments (Section 4.7) we show that our method can benefit several popular metric learning loss functions. Specifically, we consider the binomial deviance loss $\ell_{BD}(\cdot,\cdot)$, the contrastive loss $\ell_C(\cdot,\cdot)$ and the triplet loss $\ell_T(\cdot,\cdot)$. We illustrate these functions and their gradients in Figure 4.3 and formally define them as follows:

$$\ell_{BD}(s,y) = \begin{cases} \log(1+e^{-\beta_1(s-\beta_2)}) & \text{if } y=1 \\ \log(1+e^{\beta_1(s-\beta_2)C}) & \text{otherwise,} \end{cases} \tag{4.11}$$

$$\ell_C(s,y) = \begin{cases} (s-1)^2 & \text{if } y=1 \\ \max(0,s-m) & \text{otherwise,} \end{cases} \tag{4.12}$$

$$\ell_T(s^+,s^-) = \max(0,s^- - s^+ + m), \tag{4.13}$$

where $s = s(f(\boldsymbol{x}^{(1)}), f(\boldsymbol{x}^{(2)}))$ denotes the similarity score between two samples, y is 1 if the two samples are similar and 0 if they are dissimilar. β_1, β_2, and C are scaling, shifting, and class-balancing hyper-parameters for the binomial deviance loss function. We follow previous work [223] and set these parameters to 2, 0.5, and 25, respectively. m denotes

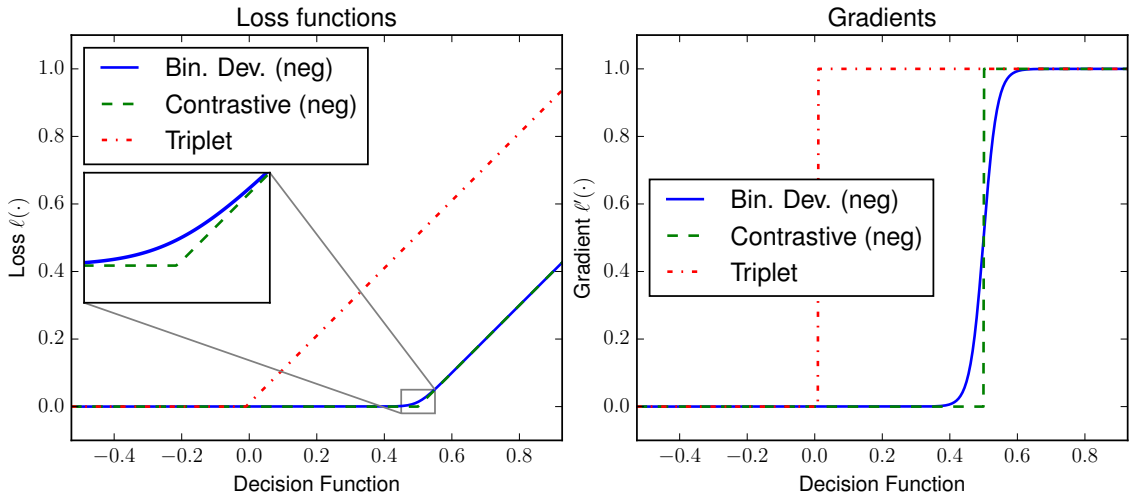


Figure 4.3: Illustration of triplet loss, contrastive loss (for negative samples) and binomial deviance loss (for negative samples) and their gradients. Triplet and contrastive loss have a non-continuous gradient, whereas binomial deviance has a continuous gradient.

the margin for the contrastive and triplet loss, which we set to 0.5 and 0.01, respectively. Finally, s^+ denotes the similarity score of a positive (similar) training pair, and s^- denotes the similarity score of a negative (dissimilar) training pair for the triplet loss.

As we show in our experiments in Section 4.7.2, our method benefits the binomial deviance loss more than the triplet loss and the contrastive loss. We hypothesize that this because the gradient of the binomial deviance loss function is smooth compared to the triplet loss and the contrastive loss. Specifically, in Figure 4.3 we see that the binomial deviance loss is a smooth version of the contrastive loss. The gradient of the triplet and contrastive loss is non-continuous and either 0 or 1. In contrast, the gradient of the binomial deviance loss is continuous.

As we employ online gradient boosting for training our *CNN*, we use the negative gradients of the loss function as weights for our training samples. Consequently, the weights of the binomial deviance loss function are also smooth, which conveys more information to successive learners in the ensemble compared to non-smooth weights of the contrastive or triplet loss.

4.3.2 On-line Gradient Boosting Efficient Ensembles

In this section, we introduce our boosting based framework for training metric learning *CNNs* (see Figure 4.4). Rather than optimizing a global loss function for the full ensemble, as we did in Section 4.2 for object categorization, we use online gradient boosting for training our learners. In contrast to optimizing a global loss function, boosting makes it easier for individual learners to be more diverse from each other.

To train our network with online gradient boosting, *e.g.* [11, 12, 28, 120], we split the last embedding layer of a metric learning *CNN* into M weak learners (see Figure 4.4). Our

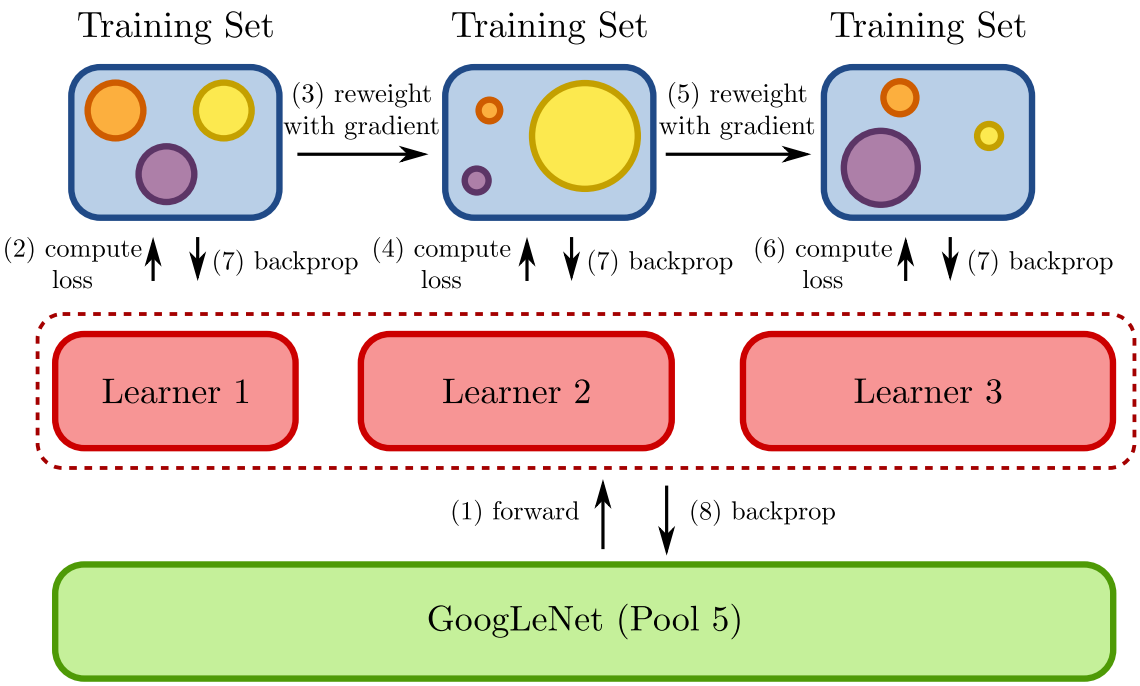


Figure 4.4: We divide the embedding (shown as dashed layer) of a metric *CNN* into several weak learners and cast training them as online gradient boosting problem. Each learner iteratively reweights samples according to the gradient of the loss function. Training a metric *CNN* this way encourages successive learners to focus on different samples than previous learners and consequently reduces correlation between learners and their feature representation.

learners can have different embedding sizes. Inspired by traditional cascade based boosting approaches [227], we typically split our ensemble into a small, medium-sized, and a large learner. During training, we sample a mini-batch of training images and compute the responses of all the embedding layers. Next, we compute the loss function for the first learner in the ensemble. Then, we use the negative gradient of the loss function as weights for each sample to compute the loss function of the successive learner. We iterate this process until we evaluate all loss functions. Finally, we backpropagate the loss for each learner to the backbone *CNN*.

As misclassified samples have a higher gradient than correctly classified loss functions, successive learners in the ensemble focus on different samples compared to previous learners. Consequently, our learners get more diverse from each other. Further, all our learners share a common feature representation. The backbone *CNN* dominates the computational and memory demands. Consequently, the computational and memory overhead imposed by our method is negligible.

More formally, we want to find M weak learners $\{f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_M(\boldsymbol{x})\}$ and their

corresponding weights $\alpha_1, \alpha_2, \dots \alpha_M$. We define output $F(\cdot, \cdot)$ of the boosting model as:

$$F\left(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}\right) = \sum_{m=1}^{M} \alpha_m s(f_m(\boldsymbol{x}^{(1)}), f_m(\boldsymbol{x}^{(2)})). \tag{4.14}$$

During training, we optimize a loss function $\ell(\cdot)$ for each of the M learners. To optimize the m-th learner, we reweight the mini-batch with the negative gradient $-\ell'(\cdot)$ of the $m-1$-th learner.

To train our learners we adapt an online gradient boosting algorithm [11]. This algorithm uses fixed weights for α_m and supports weak learners which can be trained by gradient-based optimization such as Stochastic Gradient Descent (SGD) with momentum. As our learners correspond to the last embedding layer of a *CNN*, we can train the full *CNN* end-to-end with this approach. We extend this algorithm to work on pairs or triplets and illustrate this training procedure in Algorithm 3. In practice, our algorithm works on mini-batches.

During the forward pass, we first sample a batch of images $\boldsymbol{x}$ and forward propagate it through the backbone *CNN*. For each learner m and each pair of samples n we iteratively compute the similarity score s_n^m, which is a weighted combination of the current learner prediction $s(f_m(\boldsymbol{x}_n^{(1)}), f_m(\boldsymbol{x}_n^{(2)}))$ and the previous learners' predictions s_n^{m-1}.

During the backward pass, we backpropagate the gradients of the re-weighted loss $w_n^m \ell(s_n^m)$ to the embedding layer of the m-th learner. Then, we compute the weights w_n^{m+1} of the $m+1$-th learner in the ensemble with the negative gradient of the loss function. We iterate this until we evaluate and backpropagate all gradients of the loss functions for all our M learners in the ensemble. Finally, we backpropagate the gradients from the M embeddings to the backbone *CNN*. The computation of the convolutional responses of the backbone *CNN* dominates the computational and memory demand of such a network. Consequently, as we split the *CNN* only at the end into several learners, we do not impose any significant overhead during training and test-time with our approach.

Our online boosting method learns a convex combination of M weak learners. As successive learners in the ensemble tend to focus on harder examples due to the re-weighting, we also assign them a larger embedding size (*i.e.* a larger model capacity). Specifically, we set the embedding size of the m-th learner proportional to its similarity weight $\alpha_m = \eta_m \cdot \prod_{n=m+1}^{M}(1-\eta_n)$ in the training algorithm, where $\eta_m = \frac{2}{m+1}$. We experimentally verify this design choice in our evaluation in Section 4.7.1.

During test-time, we predict a single feature-vector for an input image $\boldsymbol{x}$. To this end, we L_2 normalize the individual weak learner predictions $f_m(\boldsymbol{x})$, $1 \leq m \leq M$, scale them according to their weight $\sqrt{\alpha_m}$ and concatenate all responses to a single vector. Formally, we define the ensemble prediction $F(\boldsymbol{x})$ as:

$$F(\boldsymbol{x}) = \left[\sqrt{\alpha_1}\frac{f_1(\boldsymbol{x})}{||f_1(\boldsymbol{x})||_2} \middle\| \sqrt{\alpha_2}\frac{f_2(\boldsymbol{x})}{||f_2(\boldsymbol{x}||_2} \middle\| \dots \middle\| \sqrt{\alpha_m}\frac{f_m(\boldsymbol{x})}{||f_m(\boldsymbol{x})||_2}\right], \tag{4.15}$$

Algorithm 3: Online gradient boosting algorithm for our *CNN*.

Let $\eta_m = \frac{2}{m+1}$, for $m = 1, 2, \ldots, M$,
M = number of learners, I = number of iterations
for $n = 1$ **to** I **do**
  `/* Forward pass */`
  Sample pair $(\boldsymbol{x}_n^{(1)}, \boldsymbol{x}_n^{(2)})$ and corresponding label y_n
  $s_n^0 := 0$
  for $m = 1$ **to** M **do**
    $s_n^m := (1 - \eta_m) s_n^{m-1} + \eta_m s(f_m(\boldsymbol{x}_n^{(1)}), f_m(\boldsymbol{x}_n^{(2)}))$
  Predict $s_n = s_n^M$
  `/* Backward pass */`
  $w_n^1 := 1$
  for $m = 1$ **to** M **do**
    Backprop $w_n^m \ell(s(f_m(\boldsymbol{x}_n^{(1)}), f_m(\boldsymbol{x}_n^{(2)})), y_n)$
    $w_n^{m+1} := -\ell'(s_n^m, y_n)$

where $\cdot \| \cdot$ denotes vector concatenation.

When we compute the similarity between two ensemble vectors with the dot product, we see that the ensemble prediction corresponds to the weighted sum of the cosine similarities of the individual learners. Formally, $F(\boldsymbol{x}^{(1)})^\top F(\boldsymbol{x}^{(2)}) =$

$$
\begin{aligned}
&\left[\sqrt{\alpha_1}\frac{f_1(\boldsymbol{x}^{(1)})}{||f_1(\boldsymbol{x}^{(1)})||_2}\middle\| \ldots \middle\| \sqrt{\alpha_m}\frac{f_m(\boldsymbol{x}^{(1)})}{||f_m(\boldsymbol{x}^{(1)})||_2}\right]^\top \left[\sqrt{\alpha_1}\frac{f_1(\boldsymbol{x}^{(2)})}{||f_1(\boldsymbol{x}^{(2)})||_2}\middle\| \ldots \middle\| \sqrt{\alpha_m}\frac{f_m(\boldsymbol{x}^{(2)})}{||f_m(\boldsymbol{x}^{(2)})||_2}\right] \\
=&\sqrt{\alpha_1}\frac{f_1(\boldsymbol{x}^{(1)})^\top}{||f_1(\boldsymbol{x}^{(1)})||_2}\sqrt{\alpha_1}\frac{f_1(\boldsymbol{x}^{(2)})}{||f_1(\boldsymbol{x}^{(2)})||_2} + \ldots + \sqrt{\alpha_m}\frac{f_m(\boldsymbol{x}^{(1)})^\top}{||f_m(\boldsymbol{x}^{(1)})||_2}\sqrt{\alpha_m}\frac{f_m(\boldsymbol{x}^{(2)})}{||f_m(\boldsymbol{x}^{(2)})||_2} \\
=&\alpha_1 s(f_1(\boldsymbol{x}^{(1)}), f_1(\boldsymbol{x}^{(2)})) + \ldots + \alpha_m s(f_m(\boldsymbol{x}^{(1)}), f_m(\boldsymbol{x}^{(2)})).
\end{aligned}
\tag{4.16}
$$

Consequently, our method can be plugged into any off-the-shelf retrieval system and is compatible with fast approximate search methods such as [154].

In contrast to object classification (Section 4.2), our boosting based algorithm (Algorithm 3) directly optimizes $\mathcal{L}_{\text{discr}}$ in Equation (4.8) without introducing additional hyperparameters for balancing the ensemble and learner losses. The gradient boosting algorithm balances the diversity of the learners and the discriminativenes of the ensemble.

To make our ensembles more diverse, we also introduce several auxiliary loss functions in Section 4.4, *i.e.* *DivLoss*, *ActLoss*, and *AdvLoss*. We found that our boosting-based ensemble benefits from such auxiliary loss functions, which operate on the embedding layer of a metric learning *CNN*. Standard *BIER* [167] uses such loss function only initialization, to find the initial weight for the embedding layer. Our extension [168] also uses one of these functions as auxiliary loss during training. As we show in our evaluation (Section 4.7), this

allows us to train our models with higher learning rates and improves retrieval accuracy.

4.4 Auxiliary Loss Functions

When we naïvely apply the loss in Equation (4.8) for object categorization, we found that the individual learners in the ensemble typically make highly correlated predictions. Therefore, the ensemble has no benefits compared to standard *CNNs*. We also show this more formally in Section 2.2. To address this issue, we propose several loss functions to make our learners explicitly diverse from each other. The *DivLoss* (Section 4.4.1), which we originally proposed in [165], increases diversity by making the classifier outputs dissimilar to each other. In contrast, our *ActLoss* (Section 4.4.2), which we originally proposed in [167, 168], operates on the hidden layer of a classifier or the embedding layer of a metric learning *CNN* and mutually suppresses their responses. Finally, our *AdvLoss* (Section 4.4.3), which we originally proposed in [168], uses an adversarial loss function to make the hidden layer or embedding layer of our learners diverse from each other.

During training time, we jointly optimize an auxiliary diversity loss $\mathcal{L}_{\text{div}}$ with our discriminative loss function $\mathcal{L}_{\text{discr}}$, where div $\in \{\text{DivLoss}, \text{ActLoss}, \text{AdvLoss}\}$, as follows:

$$\mathcal{L} = \mathcal{L}_{\text{discr}} + \lambda_{\text{div}} \cdot \mathcal{L}_{\text{div}}, \tag{4.17}$$

where λ_{div} is a hyper-parameter which is typically set via (cross-)validation. There is a natural trade-off between optimizing for discriminative learners and diverse learners on the same training set. On the one hand, classifiers should agree on the class label during training time. Consequently, they will make increasingly more correlated predictions as training progresses, and the training accuracy of the learners increases. On the other hand, the learners' predictions in an ensemble should be diverse. The diversity loss increases diversity by reducing the training accuracy, as learners are encouraged to be less confident in their predictions and make different predictions compared to other learners.

For object categorization problems, we define $\mathcal{L}_{\text{discr}}$ in Equation (4.8) in Section 4.2. For metric learning problems, we use online gradient boosting to optimize $\mathcal{L}_{\text{discr}}$, as we explain in the previous Section 4.3.2. In the following Sections 4.4.1, 4.4.2, and 4.4.3 we give an overview of our auxiliary loss functions.

4.4.1 DivLoss

Our DivLoss is designed for object categorization and operates on the outputs of the learners. It tries to make their predictions diverse from each other. One popular way to measure agreement between two probability distributions for supervised learning is cross-entropy. In this setting, we want the predictive distribution of a classifier to agree with the ground-truth label distribution. To this end, neural networks typically minimize the cross-entropy loss $\mathcal{H}(\cdot, \cdot)$ between ground-truth $\boldsymbol{y}$ and prediction $\sigma(\boldsymbol{o})$ during training-time.

In contrast, in our loss function, we want to make the distribution of individual classifiers dissimilar to each other. To this end, we *maximize* the cross-entropy loss $\mathcal{H}(\cdot,\cdot)$ between pairs of our M learners to make them diverse from each other. Formally, we define the DivLoss as

$$\mathcal{L}_{\text{DivLoss}} = \frac{1}{M \cdot (M-1)} \sum_{m=1}^{M} \sum_{n \neq m} -\mathcal{H}(\sigma(\boldsymbol{c}_m), \sigma(\boldsymbol{c}_n)), \tag{4.18}$$

where $\sigma(\boldsymbol{c}_m)$ denotes the m-th classifier prediction (*i.e.* softmax activation) of our ensemble. As the cross-entropy is asymmetric, *i.e.* $\mathcal{H}(\boldsymbol{p}, \boldsymbol{q}) \neq \mathcal{H}(\boldsymbol{q}, \boldsymbol{p})$, for learners i and j we minimize both, $-\mathcal{H}(\sigma(\boldsymbol{c}_i), \sigma(\boldsymbol{c}_j))$ and $-\mathcal{H}(\sigma(\boldsymbol{c}_j), \sigma(\boldsymbol{c}_i))$.

4.4.2 Activation Loss

The Activation Loss (*ActLoss*) introduces diversity on the hidden layers of the individual learners or the embedding layers of a metric learning *CNN*. Compared to the previous DivLoss, which requires that the predictions of the classifiers are dissimilar, this loss imposes a weaker constraint. More specifically, it mutually suppresses the activations in the hidden layer of each learner (see Figure 4.5). Ideally, for a given sample only a single learner is active, *i.e.* has non-zero activations. To that end, we define a suppression loss between learner i and j as:

$$\mathcal{L}_{\text{sup}_{(\text{i,j})}}(\boldsymbol{x}) = \sum_{k=1}^{C} \sum_{l=1}^{C} \left((\boldsymbol{x}_i)_k \cdot (\boldsymbol{x}_j)_l\right)^2 \tag{4.19}$$

$$= \sum_{k=1}^{C} \sum_{l=1}^{C} \left((\boldsymbol{x}_i)_k^2 \cdot (\boldsymbol{x}_j)_l^2\right) \tag{4.20}$$

$$= \left(\sum_{k=1}^{C} (\boldsymbol{x}_i)_k^2\right) \cdot \left(\sum_{l=1}^{C} (\boldsymbol{x}_j)_l^2\right) \tag{4.21}$$

$$= ||\boldsymbol{x}_i||_2^2 \cdot ||\boldsymbol{x}_j||_2^2. \tag{4.22}$$

As we see, this is a squared group sparsity loss on the activations. Typically, when we initialize the weights of $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ randomly, we can optimize this loss function with standard *SGD*. However, if both hidden layers have the same parameters, *e.g.* due to initialization from the same pre-trained network, we have to break this symmetry. To this end, we alternatingly optimize for either $\boldsymbol{x}_i$ or $\boldsymbol{x}_j$ during the first iterations, keeping the other learner fixed.

Another problem with this loss is, that it allows a trivial solution by setting the weight matrix of the hidden layer to zero. In practice, for supervised learning problems, when we initialize the weights from scratch, we found that the supervised loss function is enough to encourage that the weight matrix is non-zero. For metric learning problems (Section 4.3),

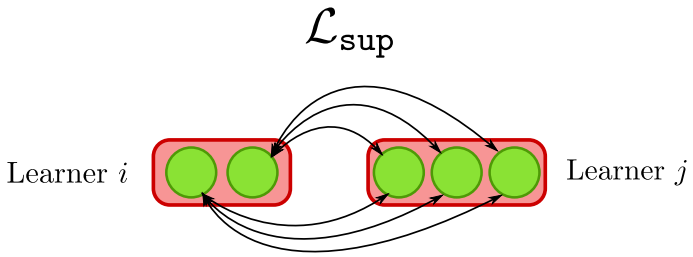


Figure 4.5: Our Activation Loss mutually suppresses neurons in hidden layers of different learners.

we found that it is necessary to constrain the weight matrix to be non-zero with an additional auxiliary loss on the weights. Formally, we add the following penalty to our diversity loss function:

$$\mathcal{L}_{\text{weight}} = \sum_{i=1}^{d} (\boldsymbol{w}_i^\top \boldsymbol{w}_i - 1)^2, \tag{4.23}$$

where $\boldsymbol{w}_i$ (with $1 \leq i \leq d$) are the row vectors of the weight matrix of the last embedding layer $\boldsymbol{W}$.

We define the final diversity loss function as average over all pairs of learners:

$$\mathcal{L}_{\text{ActLoss}}(\boldsymbol{x}) = \frac{2}{M \cdot (M-1)} \sum_{m \neq n} \mathcal{L}_{\text{sup}_{(\text{m,n})}}(\boldsymbol{x}) + \lambda_{\text{weight}} \cdot \mathcal{L}_{\text{weight}}. \tag{4.24}$$

For metric learning problems, we set λ_{weight} large enough, so that our weight vectors have a squared norm of $1 \pm 1e^{-3}$. For classification problems, we do not need this constraint and set λ_{weight} to 0.

4.4.3 Adversarial Loss

Another way to impose diversity on a hidden representation is to make the distribution of the activations dissimilar. Compared to the ActLoss and DivLoss, this is a weaker constraint, as we do not require zero activations for all but one hidden layer among our learners. This way, the hidden representations can be diverse, but nonetheless discriminative for all learners.

As hidden layer feature vectors $\boldsymbol{x}_i$, $\boldsymbol{x}_j$ between learner i and learner j can *e.g.* be a permutation from each other, we typically cannot increase diversity between them by just increasing the distance between them during training. Therefore, we train a regressor which tries to make the two hidden features $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ as similar as possible to each other. For this regressor, we use a non-linear projection, which maps $\boldsymbol{x}_j$ to $\boldsymbol{x}_i$. Therefore, such a regressor can *e.g.* easily learn to overcome simple permutations of our feature vectors, or cope with hidden layer representations of different dimensionality. The latter is beneficial, if we want to exploit the benefits of introducing variation by different model architectures between different learners.

To make the feature vectors dissimilar to each other, we add a gradient reversal layer [54] after our hidden layers $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. Therefore, during training, the hidden representations will get diverse from each other. Compared to standard Generative Adversarial Networks (GANs) [66], the gradient reversal layer is computationally cheaper. It not need an alternating optimization of the generator network (*i.e.* the backbone *CNN*) and the discriminator network (*i.e.* the regressors). Therefore, it is computationally much more efficient during training time.

Formally, let $\boldsymbol{x}_i \in \mathbb{R}^{d_i}$ and $\boldsymbol{x}_j \in \mathbb{R}^{d_j}$ denote the hidden layer representation of learner i and learner j with dimensionality d_i and d_j, respectively. Further, let $g_{(j,i)}(\cdot)$ be a projection from the hidden layer representation of learner j, to the hidden layer representation of learner i, *i.e.* $g_{(j,i)} \colon \mathbb{R}^{d_j} \mapsto \mathbb{R}^{d_i}$. Our regressor aims to make the projection $g_{(j,i)}(\cdot)$ of $\boldsymbol{x}_j$ as similar as possible to $\boldsymbol{x}_i$ by minimizing the following loss function

$$\mathcal{L}_{\text{sim}_{(\text{i,j})}}(\boldsymbol{x}) = -\frac{1}{d_i} \sum \left(\boldsymbol{x}_i \odot g_{(j,i)}(\boldsymbol{x}_j) \right)^2 , \tag{4.25}$$

where $\odot$ denotes the Hadamard (*i.e.* elementwise) product. On the other hand, the hidden representations of our learners should maximize the above loss function, making $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ dissimilar to each other, w.r.t. to the regressor $g_{(j,i)}(\cdot)$. Therefore, we have a min-max optimization problem similar to *GANs*:

$$\max_{\boldsymbol{\theta}_i, \boldsymbol{\theta}_j} \min_{g_{(j,i)}} \mathcal{L}_{\text{sim}_{(\text{i,j})}}(\boldsymbol{x}), \tag{4.26}$$

where $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_j$ denote the parameters for learners i and j, respectively. To optimize this loss function, we insert a gradient reversal layer after the hidden layers $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, as we illustrate in Fig. 4.6.

The similarity between hidden layer representations $\boldsymbol{x}_i$ and $g_{(j,i)}(\boldsymbol{x}_j)$ can be made arbitrarily large by scaling the weights of $g_{(j,i)}(\cdot)$ and the weights of the hidden layer, where we apply the loss. To address this problem, we constrain the weights $\boldsymbol{W}$ and biases $\boldsymbol{b}$ of the regressor and the hidden layers, where we apply the loss by

$$\mathcal{L}_{\text{weight}} = \max(0, \boldsymbol{b}^\top \boldsymbol{b} - 1) + \sum_i (\boldsymbol{w}_i^\top \boldsymbol{w}_i - 1)^2, \tag{4.27}$$

where $\boldsymbol{w}_i$ denotes the i-th row of the weight matrix $\boldsymbol{W}$.

Our final AdvLoss is the sum over all pairs of learners and adds the penalty on the weights of the regressor:

$$\mathcal{L}_{\text{AdvLoss}}(\boldsymbol{x}) = \frac{2}{M \cdot (M-1)} \sum_{m \neq n} \mathcal{L}_{\text{sim}_{(\text{m,n})}}(\boldsymbol{x}) + \lambda_{\text{w}} \cdot \mathcal{L}_{\text{weight}}, \tag{4.28}$$

where λ_{w} is typically set high enough so that our weight vectors have a squared norm close to $1 \pm 1e^{-3}$. In our experiments, we typically use non-linear regressors with a single

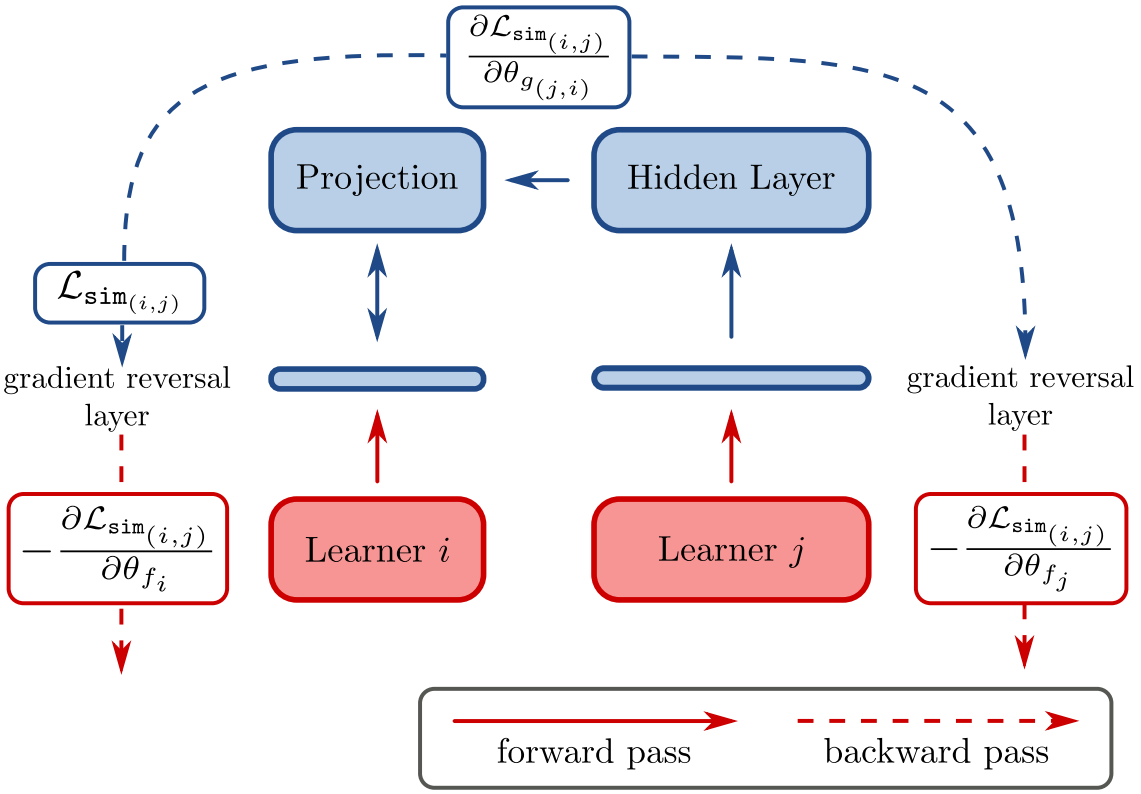


Figure 4.6: Our adversarial loss function learns auxiliary regressors $g_{(j,i)}$ from the hidden representation of learner j to learner i. To make the hidden representations diverse from each other, we insert a gradient reversal layer between the auxiliary regressor layers (blue) and the network layers (red).

hidden layer and a Rectified Linear Unit (ReLU) activation function for $g_{(i,j)}(\cdot)$. We set the hidden layer size of these regressors to the same size as the hidden layers $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$.

4.5 Loss Function on Multiple Hidden Layers

Typically, we apply our diversity loss function only on the last hidden layer of a *CNN* to make its representation diverse. However, implicit ensemble methods such as dropout typically can exploit the benefits of ensembles for every hidden layer in a *CNN*. To this end, inspired by deeply supervised networks [117, 213], we apply our loss functions on arbitrary hidden layers by dividing them into multiple groups and applying the corresponding diversity auxiliary loss. The successive layer in the network receives the full hidden layer (*i.e.* without the groups) as input. Therefore, we do not change the network architecture by this method.

As our DivLoss does not operate on the hidden representation directly, but rather on classifiers on top of these representations, we also add several auxiliary classifiers during training time for this loss. During test-time, we discard these classifiers. Therefore, our

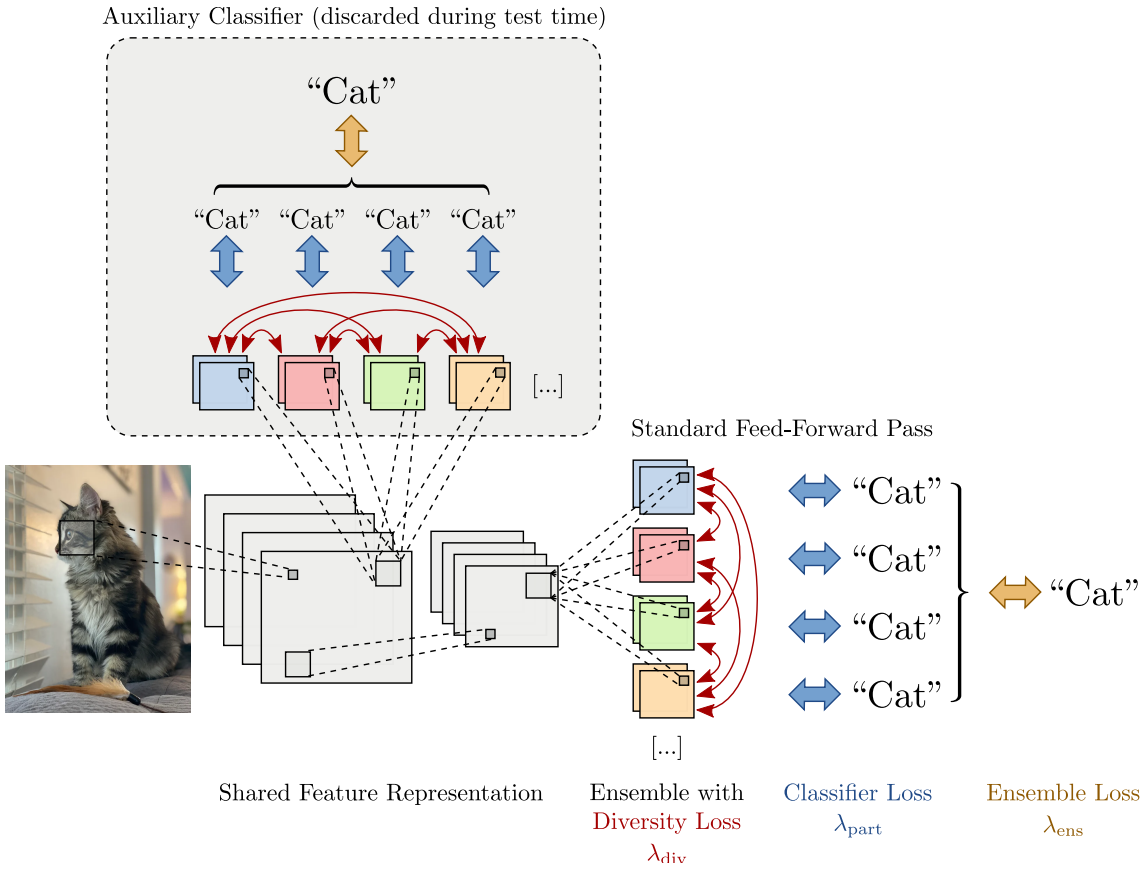


Figure 4.7: We can apply our method on top of any hidden layer in a neural network.

method does not impair any computational overhead during test-time, and only a small additional computational overhead during training time.

4.6 Image Categorization Experiments

In this section, we evaluate the benefits of our ensembling approach for image categorization. To this end, we run several experiments on the CIFAR-10, CIFAR-100 [111], and SVHN [159] datasets. The CIFAR-10 and CIFAR-100 dataset consist of $50,000$ training color images and $10,000$ test color images of size 32×32. The main difference between these two datasets is the number of classes. CIFAR-10 consists of 10 classes, and CIFAR-100 consists of 100 classes. The SVHN dataset consists of $73,257$ street view digits for training and $26,032$ street view samples for testing. Similar to the CIFAR datasets, the image size is also 32×32. Unless otherwise stated, we only use simple mean subtraction as pre-processing and do not use data augmentation.

We implement our method with TensorFlow [1] and run our experiments on an NVIDIA Titan Xp GPU. As network architecture, we adopt the CIFAR-10 Quick architecture [95]

and widen it by a factor 2, similar to Cogswell *et al.* [34]. We train our networks with *SGD* and momentum for 100 epochs. We start training with a learning rate of 0.01, and we anneal the learning rate after 50 and 90 epochs by 0.1, respectively. After each epoch, we shuffle the training dataset. We use 10% of the training set as validation set for hyper-parameter search and early stopping.

Unless otherwise stated, we use an ensemble consisting of 4 learners by splitting up the last hidden layer into 4 equally sized groups. Therefore, our method is computationally as expensive as a single network. Such networks have the same number of parameters as a standard network. Further, we can map them back to a standard neural network. For our ablation studies, we typically use the CIFAR-10 dataset.

In Section 4.6.1, we show the effectiveness of several of our diversity encouraging loss functions. We further compare our loss functions with Negative Correlation Learning (NCL) and an adapted version of *NCL*. Next, in Section 4.6.2, we analyze the impact of splitting the network at a shallower hidden layer into multiple learners, similar to TreeNets [119]. Further, we analyze the accuracy and diversity of the learners of our ensemble in Section 4.6.3, and compare it with the strength and correlation of implicit ensemble methods (*i.e.* dropout [161]). In Section 4.6.4, we compare our method against other ensembling methods in terms of accuracy, number of parameters, and FLOPS. Finally, in Section 4.6.5, we show that our approach can also benefit recent ResNet [75] and Wide ResNet [263] architectures on the CIFAR-10, CIFAR-100, and SVHN datasets. Specifically, we show that with our method we can train more pareto-optimal networks compared to (Wide) ResNets.

4.6.1 Comparison of Diversity Loss Functions

In this section, we compare our diversity loss functions to directly minimizing the correlation between learners, *i.e.* *NCL* [139]. Therefore, we minimize the following auxiliary loss function:

$$\frac{1}{D}\sum_{j=1}^{D}\sum_{m\neq n}^{M}\left(\sigma(\boldsymbol{c}_m)_j-\sigma(\boldsymbol{o})_j\right)\cdot\left(\sigma(\boldsymbol{c}_n)_j-\sigma(\boldsymbol{o})_j\right), \tag{4.29}$$

where D denotes the number of classes, M the number of learners, $\sigma(\boldsymbol{c}_m)_j$ the m-th learner prediction for the j-th class, and $\sigma(\boldsymbol{o})$ the ensemble prediction.

NCL is designed for regression problems. It tries to negatively correlate the classifier predictions. We hypothesize that for classification problems, it is more beneficial to have learners which have zero correlation (*i.e.* are "independent"). Therefore, we also try to adapt the loss function by penalizing the absolute correlation. This loss encourages learners to have zero correlation (as opposed to negative correlation):

$$\frac{1}{D}\sum_{j=1}^{D}\sum_{m\neq n}^{M}\left|\left(\sigma(\boldsymbol{c}_m)_j-\sigma(\boldsymbol{o})_j\right)\cdot\left(\sigma(\boldsymbol{c}_n)_j-\sigma(\boldsymbol{o})_j\right)\right|. \tag{4.30}$$

We summarize our results in Table 4.1. We see that diversity encouraging loss functions can benefit *CNN* performance for object categorization. Further, they do not introduce computational overhead. When we penalize the absolute correlation of learners rather than the signed correlation, we already improve standard *NCL* by a large margin. Further, our novel loss functions outperform *NCL*, presumably because *NCL* addresses regression problems as opposed to classification problems.

Finally, we see that making the representation of the classifier diverse directly performs favorably compared to making the classifiers predictions diverse from each other. Directly forcing the classifiers to be non-correlated to each other is a constraint which typically is directly the opposite of being discriminative. On the training set, *CNNs* typically achieve close to perfect accuracy. Consequently, each learner typically always predicts the correct ground truth label for a training sample. If we require them to be weakly correlated to each other, the prediction of several classifiers must be wrong. However, if we only require the feature representation to be different from each other, the individual classifiers still can make correct predictions by focusing on different high-level features. If during test-time, some of these features are missing (*e.g.* due to severe pose variation, occlusion, *etc.*), only some classifiers in the ensemble are affected.

Method	Test Accuracy	Parameters	FLOPS
Baseline	80.86	$0.59 \cdot 10^6$	$84 \cdot 10^6$
Negative Correlation	80.90	$0.59 \cdot 10^6$	$84 \cdot 10^6$
Absolute Correlation	81.28	$0.59 \cdot 10^6$	$84 \cdot 10^6$
DivLoss (Ours) [165]	82.30	$0.59 \cdot 10^6$	$84 \cdot 10^6$
ActLoss (Ours) [167, 168]	83.10	$0.59 \cdot 10^6$	$84 \cdot 10^6$
AdvLoss (Ours) [168]	**83.21**	$0.59 \cdot 10^6$	$84 \cdot 10^6$

Table 4.1: Evaluation of different auxiliary loss functions on the CIFAR-10 dataset.

4.6.2 Splitting at Shallower Layers

In this section, we study the effect of splitting the network at shallower layers. Compared to splitting the network at the last layer, this increases the parameters and computational complexity of the network. In this setup, we can only share the shallow layers between the learners. Further, we cannot map the network back to a regular neural network during test-time. On the other hand, such a setup allows the individual learners to be more diverse from each other, as they share fewer parameters with each other.

We train an ensemble of 4 learners with different splitting points and compare the accuracy with and without our ActLoss regularization. Training such networks without an auxiliary loss function is similar to TreeNets [119]. The only way learners will get diverse from each other is due to their random initialization.

We report our results in Table 4.2. We see that accuracy typically increases with an increase in parameters. One of the reasons for this is that with more parameters the

ensemble benefits more from diversity by random initialization. However, if we split the network at the last hidden layer, the diversity between learners is low from the beginning of the training. Consequently, the learners converge in a similar local minimum and are highly correlated to each other. By introducing an auxiliary loss function, we explicitly make the learners diverse from each other. Subsequently, especially parameter shared ensembles, which share a significant amount of parameters, benefit from our auxiliary loss function.

Further, ensembles in which learners share a small number of parameters, also benefit from our auxiliary loss function. Notably, a network with shared low-level representation achieves comparable accuracy to an ensemble which consists of 4 separate networks (*i.e.* 4 times more parameters and FLOPS), which only rely on random initialization and random permutation of the training set to introduce diversity.

Split Point	No Auxiliary Loss [119]	ActLoss (Ours) [167, 168]	Parameters	FLOPS
Input	82.54	84.25	$2.38 \cdot 10^6$	$336 \cdot 10^6$
Pool1	82.98	84.95	$2.35 \cdot 10^6$	$322 \cdot 10^6$
Pool2	82.80	84.60	$2.05 \cdot 10^6$	$164 \cdot 10^6$
Pool3	81.15	83.44	$1.43 \cdot 10^6$	$86 \cdot 10^6$
FC2	80.69	83.10	$0.59 \cdot 10^6$	$84 \cdot 10^6$

Table 4.2: Comparison of standard ensembles and TreeNets [119] with our diversity loss on CIFAR-10 dataset. Each ensemble consists of 4 learners.

4.6.3 Accuracy and Diversity of Learners

In this section, we compare the accuracy and diversity of our learners to standard ensembles and dropout ensembles. We measure the average disagreement between the learners. The more learners disagree with each other, the higher diversity in the ensemble. As dropout trains an exponential number of networks, we sub-sample 16 sub-networks and measure their correlation and accuracy. As there might be noise due to sampling in this experiment, we measure the standard deviation of the correlation and accuracy by repeating this experiment 10 times.

In Table 4.3 we see that without an auxiliary loss function, networks in a parameter shared ensemble tend to make highly correlated predictions, as their best prediction is always the same. In contrast, when we introduce our auxiliary loss function, learners tend to disagree with each other. Consequently, our ensemble is more diverse.

Compared to dropout ensembles, learners in explicit parameter shared ensembles typically achieve higher average accuracy. The main reason for this is that dropout dynamically samples a learner from the ensemble for each training sample during training-time. In contrast, in explicit ensembles, the learners are fixed during the whole training time. Therefore, learners in explicit ensembles can achieve higher accuracy. On the other hand, due to the random training dynamics, learners in dropout ensembles are more diverse from

each other.

Method	Avg. Disagreement	Avg. Sub-Network Acc.	Ensemble Acc.
Dropout	0.071 ± 0.0025	79.9 ± 0.00073	81.07
No Auxiliary Loss	0.0	80.69 ± 0	80.69
ActLoss [167, 168]	0.0035	83.01 ± 0.0002	83.10

Table 4.3: Diversity and average accuracy of learners in an ensemble.

4.6.4 Comparison with Other Ensemble Methods

In this section, we compare our method against other ensemble methods on the CIFAR-10 dataset. For a fair comparison, we use the same architecture (*i.e.* CIFAR-10-Quick wider) for all our experiments. Further, we use an ensemble consisting of 4 learners. As different methods also vary in the way they share parameters (*i.e.* all parameters are shared, some parameters are shared, no parameters are shared), we also report the number of parameters and FLOPS for each method. We summarize our results in Table 4.4. We see that our ensemble method achieves competitive performance to state-of-the-art approaches over a variety of different FLOPS and parameter budgets.

When we replicate the architecture at the Pool2 layer (*i.e.* some parameters are shared), our method outperforms standard ensembles with more parameters, as well as an on the fly distillation ensemble approach (ONE-E) [284]. We also provide a detailed comparison of the on the fly distillation method and our method in Table 4.5. We see that while in our ensemble, the individual learners have lower accuracy compared to the on the fly distillation ensemble, our correlation is significantly lower due to our diversity loss function. Consequently, our ensemble accuracy is higher, as learners complement each other better during test-time. If we combine both methods, we see that the accuracy of the individual learners increases, as they benefit more from distilling the knowledge of a better ensemble. Compared to the on the fly ensemble, this combination has a lower correlation, which yields significantly higher accuracy.

We also compare our method in this setup to EnsembleNet [124], which introduces diversity by making the prediction of the learners deviate from the ensemble prediction. We find that penalizing the features (*i.e.* the hidden layer), rather than the classifier achieves more diversity (*i.e.* lower correlation), yielding more accurate ensembles.

In the setting where all parameters are shared, our method achieves competitive performance to the student network in an on the fly ensemble distillation approach [284]. Further, when we combine our method with a distillation method (ONE), we see that performance of the student network (ONE) benefits from our diversity loss functions. We hypothesize that the main reason for this is, that our loss function improves the ensemble accuracy from 83.65% to 85.27% by increasing the diversity of the individual learners (ONE-E + ActLoss). The ONE method only encourages learners to be diverse from each

Method	Year	Accuracy	Parameters	FLOPS
Standard Ensembles		83.14	$2.38 \cdot 10^6$	$336 \cdot 10^6$
Standard CNN		80.14	$0.59 \cdot 10^6$	$84 \cdot 10^6$
Dropout [161]	JMLR'14	81.07	$0.59 \cdot 10^6$	$84 \cdot 10^6$
Dropout on FC1+FC2 [161]	JMLR'14	81.83	$0.59 \cdot 10^6$	$84 \cdot 10^6$
TreeNets [119]	arXiv'15	81.17	$0.59 \cdot 10^6$	$84 \cdot 10^6$
E-Softmax [241]	IJCAI'18	81.12	$0.59 \cdot 10^6$	$84 \cdot 10^6$
EnsembleNet [124]	arXiv'19	80.5	$2.05 \cdot 10^6$	$84 \cdot 10^6$
DivLoss (Ours) [165]	ACCV'16	82.30	$0.59 \cdot 10^6$	$84 \cdot 10^6$
DivLoss on FC1+FC2 (Ours) [165]	ACCV'16	83.44	$0.59 \cdot 10^6$	$84 \cdot 10^6$
ActLoss (Ours) [167]	ICCV'17	83.10	$0.59 \cdot 10^6$	$84 \cdot 10^6$
AdvLoss (Ours) [168]	TPAMI'18	83.21	$0.59 \cdot 10^6$	$84 \cdot 10^6$
ONE [284][a]	NIPS'18	83.19	$0.59 \cdot 10^6$	$84 \cdot 10^6$
ONE [284][a] + ActLoss (Ours) [167]	NIPS'18 + ICCV'17	83.99	$0.59 \cdot 10^6$	$84 \cdot 10^6$

Table 4.4: Comparison to state-of-the-art ensembling methods on the CIFAR-10 dataset. We split the network at the last hidden layer into 4 learners.

[a] Parameters and FLOPS during training time are $2.05 \cdot 10^6$and $164 \cdot 10^6$respectively.

other by random initialization and a gating network. Explicitly introducing diversity further reduces the correlation between learners and therefore increases ensemble accuracy. A better ensemble accuracy yields a more accurate student network.

Method	Ens. Acc. ↑	Learner Acc. (Mean) ↑	Corr. ↓	Params.	FLOPS
EnsembleNet [124]	84.17	82.25 ± 0.001	**0.9509**	$2.05 \cdot 10^6$	$164 \cdot 10^6$
ActLoss (Ours) [167]	84.60	82.36 ± 0.14	**0.9264**	$2.05 \cdot 10^6$	$164 \cdot 10^6$
ONE-E [284]	83.65	83.10 ± 0.08	0.9892	$2.05 \cdot 10^6$	$164 \cdot 10^6$
ONE-E [284] + ActLoss (Ours) [167]	**85.26**	$\mathbf{83.77 \pm 0.14}$	0.9602	$2.05 \cdot 10^6$	$164 \cdot 10^6$

Table 4.5: Detailled comparison between our method, ONE [284], EnsembleNet [124] and the combination of both methods. We split the networks after the Pool2 layer into 4 learners.

4.6.5 Efficient ResNet Ensembles

In this section, we apply our method on the ResNet [75] architecture on the CIFAR-10, CIFAR-100, and SVHN datasets. We split a ResNet-32 after the second block into multiple classification heads. We apply our Activation Loss on the last hidden layer of each of these heads to encourage each head to be diverse from each other.

As such ensembles have a higher number of parameters and a higher computational complexity compared to standard ResNets, for a fair comparison, we compare our efficient ensemble method with Wider ResNets [263]. Wider ResNets enlarge the number of con-

volutional channels in ResNets by a factor $k \in \{1, 2, 4, 8\}$, where $k = 1$ denote standard ResNets. Consequently, they allow to trade-off accuracy vs. computational complexity and do not significantly change the underlying network architecture.

In our experiments on the CIFAR-10 and SVHN datasets, we train several ensembles with the number of learners of $\{2, 4, 8\}$. For each ensemble, we train Wide ResNets with an enlargement factor of $k \in \{1, 2, 4\}$. We compare the speed vs accuracy trade-off to standard Wide ResNets with an enlargement factor of $k \in \{1, 2, 4\}$. Similarly, for the CIFAR-100 dataset, we train several ensembles with the number of learners of $\{2, 4\}$ and an enlargement factor of $k \in \{1, 2, 4\}$. We train the standard Wide ResNets on this dataset with an enlargement factor of $k \in \{1, 2, 4\}$.

We compare the accuracy and the number of FLOPS of our models and summarize our results in Figure 4.8, 4.9, and 4.10 and Table 4.6, 4.7, and 4.8. We see that our efficient ensembles are typically computationally more efficient compared to standard ResNets, while achieving higher accuracy on the CIFAR-10, CIFAR-100, and SVHN datasets. Consequently, with a given computational budget it is typically possible to train a better performing efficient ensemble ResNet.

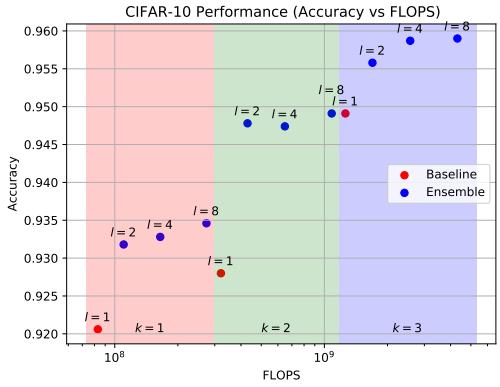


Figure 4.8: Speed vs. accuracy trade-off on the CIFAR-10 dataset. We highlight networks with different widening factor $k \in \{1, 2, 3\}$ in red, green and yellow, respectively.

k	l	FLOPS ↓	Accuracy ↑
1	1	$83 \cdot 10^6$	0.9206
1	2	$110 \cdot 10^6$	0.9318
1	4	$165 \cdot 10^6$	0.9328
1	8	$274 \cdot 10^6$	0.9346
2	1	$321 \cdot 10^6$	0.9280
2	2	$430 \cdot 10^6$	0.9478
2	4	$648 \cdot 10^6$	0.9474
2	8	$1,084 \cdot 10^6$	0.9491
4	1	$1,260 \cdot 10^6$	0.9491
4	2	$1,700 \cdot 10^6$	0.9558
4	4	$2,567 \cdot 10^6$	0.9587
4	8	$4,311 \cdot 10^6$	0.9590

Table 4.6: Speed vs. accuracy trade-off on the CIFAR-10 dataset with a ResNet widening factor of $k \in \{1, 2, 4\}$ and $l \in \{1, 2, 4, 8\}$ learners respectively.

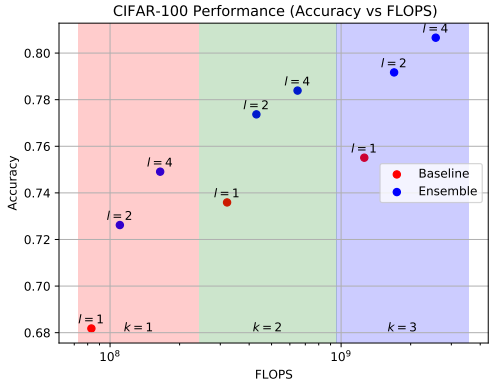


Figure 4.9: Speed vs. accuracy trade-off on the CIFAR-100 dataset. We highlight networks with different widening factor $k \in \{1, 2, 3\}$ in red, green and yellow, respectively.

k	l	FLOPS ↓	Accuracy ↑
1	1	$83 \cdot 10^6$	0.6818
1	2	$110 \cdot 10^6$	0.6834
1	4	$165 \cdot 10^6$	0.6984
2	1	$321 \cdot 10^6$	0.7359
2	2	$430 \cdot 10^6$	0.7462
2	4	$648 \cdot 10^6$	0.7375
4	1	$1,260 \cdot 10^6$	0.7551
4	2	$1,700 \cdot 10^6$	0.7754
4	4	$2,567 \cdot 10^6$	0.7797

Table 4.7: Speed vs. accuracy trade-off on the CIFAR-100 dataset with a ResNet widening factor of $k \in \{1, 2, 4\}$ and $l \in \{1, 2, 4\}$ learners respectively.

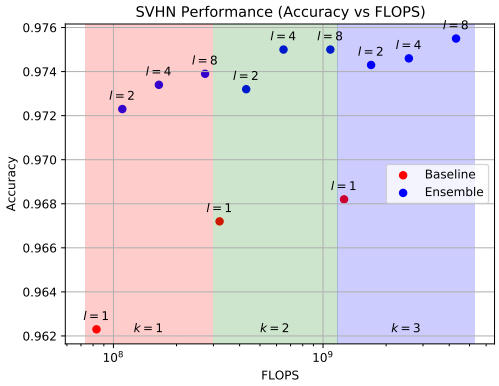


Figure 4.10: Speed vs accuracy trade-off on the SVHN dataset. We highlight networks with different widening factor $k \in \{1, 2, 3\}$ in red, green and yellow, respectively.

k	l	FLOPS ↓	Accuracy ↑
1	1	$83 \cdot 10^6$	0.9623
1	2	$110 \cdot 10^6$	0.9695
1	4	$165 \cdot 10^6$	0.9702
1	8	$274 \cdot 10^6$	0.9696
2	1	$321 \cdot 10^6$	0.9672
2	2	$430 \cdot 10^6$	0.9704
2	4	$648 \cdot 10^6$	0.9717
2	8	$1,084 \cdot 10^6$	0.9717
4	1	$1,260 \cdot 10^6$	0.9682
4	2	$1,700 \cdot 10^6$	0.9717
4	4	$2,567 \cdot 10^6$	0.9723
4	8	$4,311 \cdot 10^6$	0.9722

Table 4.8: Speed vs. accuracy trade-off on the SVHN dataset with a ResNet widening factor of $k \in \{1, 2, 4\}$ and $l \in \{1, 2, 4, 8\}$ learners respectively.

4.7 Metric Learning Experiments

In this section, we evaluate the performance of *BIER*, *i.e.* our method adapted to metric learning, on image retrieval problems and conduct our ablation studies on the CUB-200-2011 [229] dataset. To compare our method to the state-of-the-art, we evaluate the performance of our method on several other popular metric learning benchmarks, *i.e.* Cars-196 [109], Stanford Online Products [163], and In-Shop Clothes Retrieval [141].

We use the standard evaluation protocol on these datasets to measure our performance, *i.e.* the Recall@K metric. To measure Recall@K, for each query image we retrieve the K

nearest neighbors according to the learned similarity function. If the class label of at least one of these K images matches with the corresponding query image, the query is correctly retrieved. Recall@K averages all successful retrievals over all query images.

We implement our retrieval framework in TensorFlow [1]. For a fair comparison of our method to previous works, *e.g.* [163, 223], we use a standard pre-trained GoogLeNet [212] (without batch normalization) as feature extractor. As TensorFlow does not provide pre-trained ImageNet [186] weights for the original GoogLeNet, we dump the weights of the network from the publicly available Caffe [95] model.

In our ablation experiments, we show the benefits of adding either our adversarial auxiliary loss function or our activation auxiliary loss function to *BIER*. To optimize *BIER* without an auxiliary loss function, we use ADAM [104] with a learning rate of 10^{-6}. When we add an auxiliary loss function to our method, we can increase the learning rate by a factor of 10 to 10^{-5} (see Section 4.7.6), which significantly speeds up training.

Popular loss functions in metric learning operate on pairs, triplets, or n-tuples. They need pairs of similar images in a mini-batch to compute the loss. However, metric learning datasets typically have a large number of categories. Consequently, when we naïvely sample a small number of images from such datasets uniformly at random, the probability of having images with the same class label in the same mini-batch is small. Unfortunately, this makes it impossible to train metric learning *CNNs*. Therefore, we construct our batch by first uniformly sampling a fixed number of categories from the dataset. Then, we sample uniformly a fixed number of images for each of these categories. As a result, each of our mini-batches consists of approximately 5-10 images per category.

We follow previous works, *e.g.* [163, 223], and use standard preprocessing to train our networks. Specifically, we resize the longest axis of our images to 256 pixels and pad the shorter axis, so that the resulting image has a resolution of 256×256 pixels. We subtract the ImageNet mean channel-wise from the image. Before we construct our mini-batch during training, we randomly crop 224×224 patches from these 256×256 patches and randomly mirror them. At test-time, we use the 224×224 center crop of an image to predict the embedding.

In the following, we first do a detailed ablation study of *BIER* without any auxiliary loss function during training time (*i.e.* we introduce diversity only by reweighting samples and during initialization). We analyze the strength (accuracy) and correlation of *BIER* without auxiliary loss function (Section 4.7.1). We then show that several popular metric learning loss functions can benefit from *BIER* (Section 4.7.2). Further, we fix the size of the last embedding layer and show the influence of the number of learners (Section 4.7.3). Next, we vary the embedding size (*i.e.* the parameter budget) and show the impact on the performance of our method (Section 4.7.4). As standard *BIER* uses our activation loss to initialize the weights of the last embedding layer in our network, we also compare several initialization methods (Section 4.7.5).

Then, we do a detailed ablation study of combining *BIER* with two of our auxiliary loss function, *i.e.* the Activation Loss and the Adversarial Loss, as they can operate on

hidden and embedding layers of neural networks. We analyze the impact on learning rates and training time of this setup (Section 4.7.6) and evaluate the strength and correlation of such ensembles (Section 4.7.7). Finally, we compare our method to the state-of-the-art (Section 4.7.8).

4.7.1 Strength and Correlation

In this section, we analyze the impact of *BIER* on the ensemble performance. For these experiments, we analyze *BIER* as we originally proposed it in our original work [167]. Therefore, we only use our auxiliary loss function to initialize the weights of the last embedding layer and not during training time. We show how *BIER* impacts the strength (*i.e.* the accuracy of individual learners) and the correlation between the learners of the ensemble.

First, we train a baseline model with an embedding size of 512 (*Baseline*). Next, we train a parameter shared ensemble model without boosting. For this ensemble, we split the last embedding layer into three non-overlapping groups of sizes 170, 171, and 171, respectively. We initialize the embedding with our Activation Loss initialization method. To train our network, we optimize a discriminative metric loss function on each of these learners separately (*Init-170-171-171*). Finally, we re-weight training samples with boosting during training (*BIER-170-171-171*).

Successive learners in our ensemble tend to focus on harder examples due to our boosting-based training. Therefore, we propose to set the size of the embeddings of our learners proportional to their weighting in the online boosting algorithm (Section 4.3.2). Specifically, we set the embedding size of the first learner to 96, the second learner to 160 and the last learner to 256. We evaluate this choice for an ensemble without boosting (*Init-96-160-256*) and with boosting (*BIER-96-160-256*).

As we show in Table 4.9, we see that by splitting the embedding into several learners we already achieve a notable improvement in terms of accuracy. Specifically, as we initialize our learners independently to each other with our auxiliary loss function, learners tend to focus on different training examples during training. Consequently, we can reduce the correlation of the embedding and improve ensemble performance. Further, when we add online gradient boosting to our method, the correlation of our learners decreases compared to our learners trained without boosting. The individual weak learners trained without boosting achieve similar accuracy compared to our boosted learners (*e.g.* 51.94 vs 51.47 of *Learner-1-170*), but the combination of our boosted learners achieve a significant improvement. The main reason for this is the lower correlation of our boosting based ensemble.

4.7.2 Loss Functions

In this section, we show that many popular loss functions in metric learning, such as the triplet loss or the contrastive loss, benefit from *BIER*. We train our baseline, *i.e.* a

Method	Clf. Corr. ↓	Feature Corr. ↓	R@1 ↑
Baseline-512	-	0.1530	51.76
Init-170-171-171	0.8362	0.1005	53.73
Learner-1-170			51.94
Learner-2-171			51.99
Learner-3-171			52.26
Init-96-160-256	0.9008	0.1197	53.93
Learner-1-96			50.35
Learner-2-160			52.60
Learner-3-256			53.36
BIER-170-171-171	0.7882	0.0988	54.76
Learner-1-170			51.47
Learner-2-171			52.28
Learner-3-171			52.38
BIER-96-160-256	**0.7768**	**0.0934**	**55.33**
Learner-1-96			49.95
Learner-2-160			52.82
Learner-3-256			54.09

Table 4.9: Evaluation of classifier (Clf.) and feature correlation on CUB-200-2011 [229]. **Best** results are highlighted.

standard *CNN* without an ensemble with an embedding size of 512, with different loss functions, *i.e.* the triplet loss, the contrastive loss, and the binomial deviance loss. Next, for each of these three loss functions, we train a *CNN* with *BIER*. We split the embedding into learners of size 96, 160, and 256.

In Table 4.10, we see that our method significantly improves performance for all three loss functions. Interestingly, our method performs best with loss functions with a smooth (*i.e.* continuous) gradient. We hypothesize that this is because loss functions with a smooth gradient also assign smooth weights to successive learners. In contrast, loss functions with non-smooth gradients (*i.e.* the triplet loss or the contrastive loss) assign only 1 or 0 as a weight to training samples for successive learners. Consequently, smooth loss functions can convey more information to successive learners.

Method	Feature Corr. ↓	R@1 ↑
Triplet-512	0.2122	50.12
Triplet-96-160-256	**0.1158**	**53.31**
Contrastive-512	0.1639	50.62
Contrastive-96-160-256	**0.1246**	**53.8**
Binomial-Deviance-512	0.1530	51.76
Binomial-Deviance-96-160-256	**0.0934**	**55.33**

Table 4.10: Evaluation of loss functions on CUB-200-2011 [229].

4.7.3 Number of Learners

In this section, we fix the parameter and computational budget for our embedding layer and analyze the impact on the accuracy of the number of learners. Therefore, we set the total embedding size to 512 and split it into $M = \{2, 3, 4, 5\}$ learners and train them with *BIER*.

We summarize the results of this experiment in Table 4.11. With a parameter and computational budget of 512 neurons in the embedding layer, *BIER* works best with $3-4$ learners. If we increase the number of learners in the ensemble, the strength (*i.e.* accuracy) of individual learners is too low as the embedding dimensionality is too small to learn a meaningful similarity function. In contrast, if we decrease the number of learners to *e.g.* 2, the strength of the individual learners increases. However, the learners are more correlated with each other, since they benefit less from our gradient boosting algorithm.

Group Sizes	Clf. Corr. ↓	Avg R@1 ↑	R@1 ↑
Baseline	-	-	51.76
170-342	0.8252	**53.06**	54.66
96-160-256	0.7768	52.29	55.33
52-102-152-204	0.7091	50.67	**55.62**
34-68-102-138-170	**0.6250**	48.5	54.9

Table 4.11: Evaluation of group sizes on CUB-200-2011 [229].

4.7.4 Embedding Sizes

In contrast to the previous section, we vary the parameter and computational budget of the last embedding layer in this experiments. We set the embedding layer size of our baseline *CNN* to $\{384, 512, 1024\}$. We compare the accuracy of this baseline *CNN* to a *BIER* ensemble. We split the 384 sized embedding into three learners of size 64, 128, and 192. For the 512 sized embedding, we follow our previous experiments and divide it into three learners of size 96, 160 and 256. For the large embedding of size 1024, we train 5 learners of size 50, 96, 148, 196, 242, and 292. For all our experiments, we use the binomial deviance loss, as it typically outperforms the triplet loss and contrastive loss in our experiments in Section 4.7.2.

In our experiments in Table 4.12 *BIER* achieves consistent improvements over a baseline *CNN* with a parameter and computational budget of $\{384, 512, 1024\}$ for the last embedding layer. For an embedding size of 1024, a large number of learners is more beneficial for performance. The main reason for this is that learners with a size significantly larger than 256 tend to over-fit on this dataset. Therefore, it is more beneficial to split the embedding into a larger number of smaller learners.

In Figure 4.11 we also summarize the performance, when we vary the number of learners in an ensemble with a parameter budget of 512 and 1024. We set the embedding sizes of the

Method	Feature Corr. ↓	R@1 ↑
Baseline-384	0.1453	51.57
BIER-64-128-192	**0.0939**	**54.66**
Baseline-512	0.1530	51.76
BIER-96-160-256	**0.0934**	**55.33**
Baseline-1024	0.1480	52.89
BIER-50-96-148-196-242-292	**0.0951**	**55.99**

Table 4.12: Evaluation of embedding size on CUB-200-2011 [229].

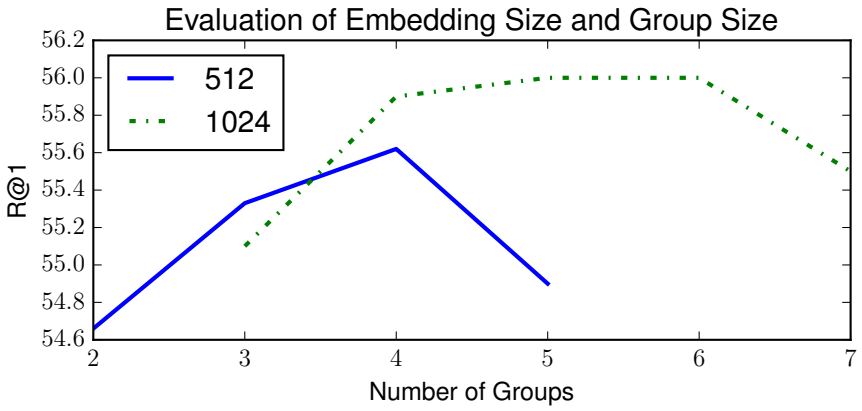


Figure 4.11: Evaluation of different embedding sizes and group sizes on the CUB-200-2011 [229] dataset.

individual learners proportional to the weight of our boosting algorithm (Section 4.3.2). Here we also see that a larger number of learners is beneficial for an embedding of size 1024. The main reason for this is that the accuracy of individual learners tends to saturate at an embedding size of approximately 256 for the CUB-200-2011 [229] dataset. Therefore, to better exploit the large dimensionality of the embedding, it is more beneficial for large embedding sizes to divide them into a larger number of learners.

4.7.5 Impact of Initialization

In our original version of *BIER* [167], we use our auxiliary loss function, *i.e.* the Activation Loss, only for weight initialization for the last embedding layer. Therefore, in this section, we evaluate the impact of different initialization methods, *i.e.* our Activation Loss and our Adversarial Loss [168], for the embedding layer. We compare the performance to other popular initialization methods, *i.e.* random initialization as proposed by Glorot *et al.* [64] and an orthogonal initialization method [190].

In all these experiments we use *BIER* with the binomial deviance loss with 3 learners and an embedding capacity of 512 for training. We report the R@1 on the CUB-200-2011 [229] dataset for all four initialization methods. In Table 4.13 we see that *BIER* works better with our two auxiliary loss functions as initialization method compared to

random initialization and orthogonal initialization. The reason for this is that our auxiliary loss functions already initialize all our learners so that they are weakly correlated with each other. Consequently, our boosting method only needs to maintain the diversity of our learners during training.

Method	R@1
Glorot	54.41
Orthogonal	54.58
Activation Loss	**55.33**
Adversarial Loss	55.04

Table 4.13: Evaluation of Glorot, orthogonal and our Activation Loss and Adversarial Loss initialization method on CUB-200-2011 [229].

4.7.6 Impact of Auxiliary Loss Functions

In this section, we evaluate the benefits of adding our auxiliary loss functions during training time. We originally presented this as an extension to *BIER* together with our adversarial loss function in [168]. For our evaluation, we run several experiments on the CUB-200-2011 [229] dataset. We train a network without auxiliary loss function during training (*i.e.* standard *BIER*), a network with our Activation Loss, and a network with our Adversarial Loss during training. We follow our previous experimental setup and use a capacity of 512 for the ensemble embedding. We split this 512 sized embedding into 3 learners (*i.e.* 96, 160, and 256 sized learners). Further, we observe that we can train our networks with auxiliary loss function with an order of magnitude higher learning rate (*i.e.* 10^{-5} instead of 10^{-6}), which results in significantly faster convergence times. We report the R@1 accuracy of all our methods.

In Table 4.14 we see that training our original *BIER* [167] with an auxiliary loss function during training time significantly improves accuracy (*i.e.* 57.5 vs 55.3 R@1). Notably, with our auxiliary loss functions, we can train our network with an magnitude higher learning rates, as they stabilize the diversity of the training. Consequently, we can achieve faster convergence times with auxiliary loss functions. When we use the same learning rates with standard *BIER* without auxiliary loss, training becomes unstable and the performance drops (*i.e.* 52.3 vs 55.3 R@1).

Further, our Adversarial Loss function outperforms our Activation Loss function (*i.e.* 57.5 vs 56.5 R@1). We hypothesize that this is because our Adversarial Loss imposes a weaker constraint on the diversity of the individual learners compared to our Activation Loss. Our Activation Loss encourages that for a given training sample, only a single learner is active (*i.e.* has non-zero activations). The remaining learners in the ensemble should make predictions close to 0. However, this constraint might be too strong. In contrast, our Adversarial Loss function uses an adversarial regressor to make learners diverse from each other. The regressor tries to make two vector spaces of pairs of learners as similar as

Method	R@1	Learning Rate	Iterations
No Auxiliary Loss	55.3	$1e^{-6}$	50K
No Auxiliary Loss	52.3	$1e^{-5}$	15K
Activation Loss	56.5	$1e^{-5}$	15K
Adversarial Loss	**57.5**	$1e^{-5}$	15K

Table 4.14: Comparison of auxiliary loss functions on CUB [229]. Our adversarial loss function significantly improves accuracy over our baseline (*BIER* [167]) and enables higher learning rates and faster convergence.

possible under a non-linear mapping. Consequently, as we insert a gradient reversal layer between the embedding layer and the regressor, we encourage our learners to be diverse from each other w.r.t. to this non-linear mapping. According to our results, this is more effective for reducing correlation than trying to set the entire embeddings of learners to 0 by our Activation Loss.

We also evaluate the impact of the Adversarial and Activation Loss on the strength and correlation of the learners in the ensemble in Table 4.15 and compare it to standard *BIER*. We see that by including these loss functions, we can further reduce the correlation of the feature vectors as well as the correlation between classifiers. Interestingly, the individual learners of our Activation Loss and our Adversarial Loss achieve similar accuracy (*i.e.* 51.1% vs 51.3%, 53.8% vs 53.5%, and 55.3% vs 55.2%). However, our Adversarial Loss is more effective in reducing the correlation between learners (*i.e.* 0.6031 vs 0.7310) and features (*i.e.* 0.0731 vs 0.0882) and therefore achieves higher ensemble accuracy in terms of R@1 (*i.e.* 57.5% vs 56.5%).

Our Adversarial auxiliary loss function uses a gradient reversal layer [54] to make learners dissimilar from each other. In contrast to the original work of Ganin *et al.* [54], which introduces the gradient reversal layer for domain adaptation, we do not use a dynamic schedule for the regularization parameter $\lambda_{\texttt{div}}$ (see Section 4.4.3). Instead of dynamically increasing $\lambda_{\texttt{div}}$, we keep the hyper-parameter fixed. Further, rather than scaling the gradients with $\lambda_{\texttt{div}}$, we scale the loss function of our adversarial network, as we describe in Section 4.4.3. Consequently, as $\lambda_{\texttt{div}}$ is typically smaller than 1, our adversarial auxiliary regression network trains slower compared to our metric learning network. We found that this turns out to be beneficial for the training process, as optimizing the regression network with higher learning rates results in a too strong adversarial network. Subsequently, this degrades the performance of the base network.

4.7.7 Evaluation of the Regularization Parameter

In this section, we evaluate the impact of our regularization parameter $\lambda_{\texttt{div}}$ (recall Section 4.4) of our Activation and Adversarial Loss on the ensemble performance. We vary $\lambda_{\texttt{div}}$ and train several models with a learning rate of 10^{-5} on the CUB-200-2011 dataset [229].

Method	Clf. Corr. ↓	Feature Corr. ↓	R@1 ↑
BIER-96-160-256	0.7768	0.0934	55.3
Learner-1-96			50.0
Learner-2-160			52.8
Learner-3-256			54.1
Activation BIER-96-160-256	0.7130	0.0882	56.5
Learner-1-96			51.3
Learner-2-160			53.5
Learner-3-256			55.2
Adversarial BIER-96-160-256	**0.6031**	**0.0731**	**57.5**
Learner-1-96			51.1
Learner-2-160			53.8
Learner-3-256			55.3

Table 4.15: Impact of auxiliary loss functions on strength and correlation.

We summarize our results in Figure 4.12. The performance of our Adversarial Loss peaks around $\lambda_{\texttt{div}} = 10^{-3}$, whereas the performance of our Activation Loss peaks at $\lambda_{\texttt{div}} = 10^{-2}$. Our Adversarial Loss outperforms our Activation Loss by about 1% R@1. However, our Activation Loss tends to perform more stable over a larger parameter range. We hypothesize that this is because training adversarial networks typically requires to carefully balance the strength of the generator, *i.e.* our metric learning *CNN*, and the discriminator, *i.e.* our adversarial regressors. We also illustrate our baseline, *i.e.* standard *BIER* without auxiliary loss function trained with a learning rate of 10^{-5}, as a dotted line. Applying any of our two auxiliary loss functions during training significantly outperforms our baseline and improves the training stability of *BIER* at higher learning rates.

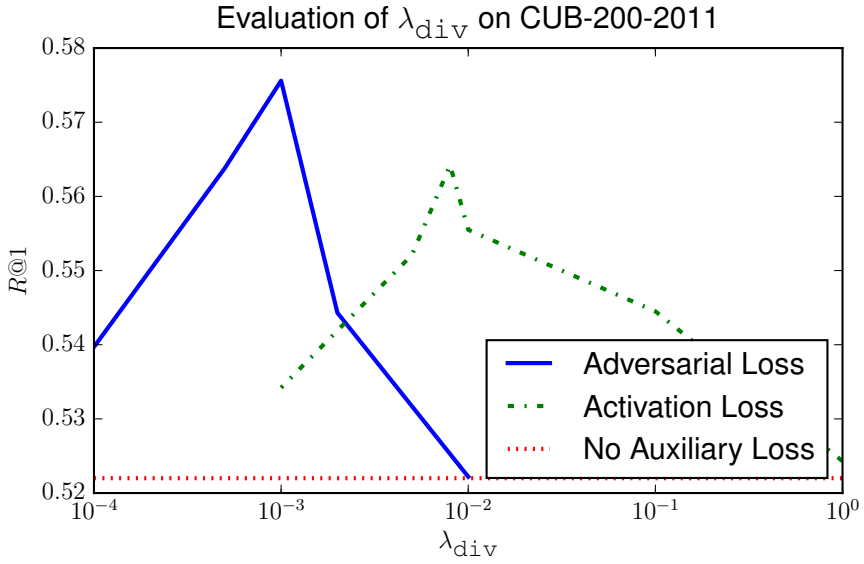


Figure 4.12: Evaluation of $\lambda_{\texttt{div}}$ on CUB-200-2011 [229].

4.7.8 Comparison with the State-of-the-Art

Dataset	Images	Classes	Comment
CUB-200-2011 [229]	11,788	200	fine-grained birds
Cars-196 [109]	16,185	196	car types
Stanford Online Products [163]	120,053	22,634	fine-grained products from 12 coarse categories (*e.g.* cup, bicycle, *etc.*)
In-Shop Clothes Retrieval [141]	54,642	11,735	clothes
VehicleID [136]	221,763	26,267	vehicle instances (*e.g.* same car type but different car, different car type, *etc.*)

Table 4.16: Summary of dataset statistics in our experiments.

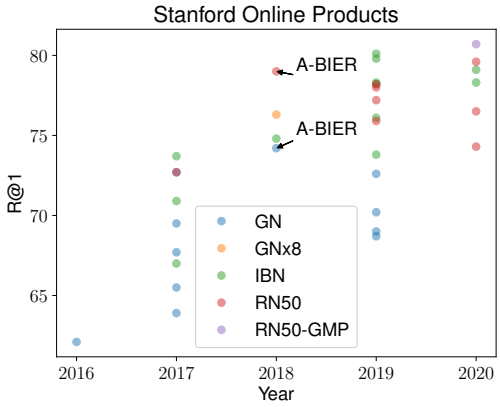


Figure 4.13: Visualization of the progress in retrieval accuracy over time. We color code the backbone network architecture.

In this section, we compare our method to the state-of-the-art on the CUB-200-2011 [229], Cars-196 [109], Stanford Online Product [163], In-Shop Clothes Retrieval [141] and VehicleID [136] datasets. We summarize all dataset statistics, *i.e.* the number of images and categories of each dataset, in Table 4.16. As the overall accuracy largely depends on implementation details, such as backbone network, embedding size, data augmentation, *etc.* [184], we also try to summarize these details in our comparison. Specifically, we report publication year and conference, backbone network (GN denotes GoogLeNet [212] without batch normalization, IBN denotes Inception with batch normalization [89], RN50 denotes ResNet-50 [75], RN50-GMP denotes ResNet-50 with global max pooling, GNx8 denotes 8 GoogLeNet [212], and RN18x48 denotes 48 ResNet-18 [75]) and embedding size.

In our experiments on CUB-200-2011, Cars-196, and Stanford Online Products, we follow the evaluation protocol proposed in [163]. When we run our experiments on the CUB-200-2011 dataset, we use the first 100 classes (5,864 images) for training and the

Method	Year	Net	Size	R@1	R@2	R@4	R@8
Contrastive [163]	CVPR'16	GN	128	26.4	37.7	49.8	62.
Triplet [163]	CVPR'16	GN	128	36.1	48.6	59.3	70.0
LiftedStruct [163]	CVPR'16	GN	128	47.2	58.9	70.2	80.2
Binomial Deviance [223]	NIPS'16	GN	512	52.8	64.4	74.7	83.9
Histogram Loss [223]	NIPS'16	GN	512	50.3	61.9	72.6	82.4
Our Baseline [167]	ICCV'17	GN	512	51.8	63.8	74.1	83.1
N-Pair-Loss [206]	NIPS'16	GN	64	51.0	63.3	74.3	83.2
Clustering [207]	CVPR'17	IBN	64	48.2	61.4	71.8	81.9
ProxyNCA [153]	ICCV'17	IBN	64	49.2	61.9	67.9	72.4
Smart Mining [73]	ICCV'17	GN	64	49.8	62.3	74.1	83.3
Angular Loss [238]	ICCV'17	GN	512	54.7	66.3	76.0	83.9
HDML [280]	CVPR'19	GN	512	53.7	65.7	75.7	85.7
NormSoftmax [264]	BMVC'19	GN	512	55.3	67.0	77.6	85.4
HTG [278]	ECCV'18	RN18	512	59.5	71.8	81.3	88.2
HTL [58]	ECCV'18	IBN	512	57.1	68.8	78.7	86.5
RLL-H [242]	CVPR'19	IBN	512	57.4	69.7	79.2	68.9
NormSoftmax [264]	BMVC'19	IBN	512	59.6	72.0	81.2	88.6
NormSoftmax [264]	BMVC'19	RN50	512	61.3	73.9	83.5	90.0
Margin [280]	ICCV'17	RN50	128	63.6	74.7	83.1	90.0
SoftTriple [173]	ICCV'19	IBN	512	65.4	76.4	84.5	90.4
MS [243]	CVPR'19	IBN	512	65.7	77.0	86.3	91.2
MS+DIR [151]	CVPR'20	IBN	512	66.1	77.0	85.1	91.1
CircleLoss [211]	CVPR'20	IBN	512	66.7	77.4	86.2	91.2
RankMI [99]	CVPR'20	RN50	128	66.7	77.2	85.1	91.0
PADS [183]	CVPR'20	RN50	128	67.3	78.0	85.9	-
ProxyAnchor [101]	CVPR'20	IBN	512	68.4	79.2	86.8	91.6
ProxyNCA++[217][a]	ECCV'20	RN50-GMP	512	69.0	79.8	87.3	92.7
Ensemble approaches:							
HDC [262]	ICCV'17	GN	384	53.6	65.7	77.0	85.6
BIER Learner-3 [167]	ICCV'17	GN	256	54.1	66.1	76.5	84.7
BIER [167]	ICCV'17	GN	512	55.3	67.2	76.9	85.1
A-BIER Learner-3 [168]	TPAMI'18	GN	256	55.3	67.0	76.8	86.0
A-BIER [168]	TPAMI'18	GN	512	57.5	68.7	78.3	86.2
HORDE [91][a]	CVPR'19	GN	512	59.4	71.0	81.0	88.0
ABE-8 [103]	ECCV'18	GNx8	512	60.6	71.5	79.8	87.4
DREML [248]	ECCV'18	RN18x48	9216	63.9	75.0	83.1	89.7
MIC [182]	CVPR'19	RN50	128	66.1	76.8	85.6	-
RLL-(L,M,H) [242]	CVPR'19	IBN	1536	61.3	72.7	82.4	89.4
D&C [188]	CVPR'19	RN50	128	65.9	76.6	84.4	90.6
HORDE [91][a]	CVPR'19	IBN	512	66.8	77.4	85.1	91.0
DiVA [150]	ECCV'20	RN50	512	69.2	79.3	-	-

Table 4.17: Comparison with the state-of-the-art on the CUB-200-2011 [229] dataset. **Our** results are highlighted.

[a] Uses non-standard data augmentation (multi-scale training, *etc.*).

Method	Year	Net	Size	R@1	R@2	R@4	R@8
Contrastive [163]	CVPR'16	GN	128	21.7	32.3	46.1	58.9
Triplet [163]	CVPR'16	GN	128	39.1	50.4	63.3	74.5
LiftedStruct [163]	CVPR'16	GN	128	49.0	60.3	72.1	81.5
Our Baseline [167]	ICCV'17	GN	512	73.6	82.6	89.0	93.5
N-Pair-Loss [206]	NIPS'16	GN	64	71.1	79.7	86.5	91.6
Clustering [207]	CVPR'17	IBN	64	58.1	70.6	80.3	87.8
Proxy NCA [153]	ICCV'17	IBN	64	73.2	82.4	86.4	87.8
Smart Mining [73]	ICCV'17	GN	64	64.7	76.2	84.2	90.2
Angular Loss [238]	ICCV'17	GN	512	71.4	81.4	87.5	92.1
RLL-H [242]	CVPR'19	IBN	512	74.0	83.6	90.1	94.1
NormSoftmax [264]	BMVC'19	GN	512	75.2	84.7	90.4	94.2
HDML [280]	CVPR'19	GN	512	79.1	87.1	92.1	95.5
HTG [278]	ECCV'18	RN18	512	76.5	84.7	90.4	94.0
Margin [280]	ICCV'17	RN50	128	79.6	86.5	91.9	95.1
HTL [58]	ECCV'18	IBN	512	81.4	88.0	92.7	95.7
NormSoftmax [264]	BMVC'19	IBN	512	81.7	88.9	93.4	96.0
MS [243]	CVPR'19	IBN	512	84.1	90.4	94.0	96.5
SoftTriple [173]	ICCV'19	IBN	512	84.5	90.7	94.5	96.9
MS+DIR [151]	CVPR'20	IBN	512	85.0	90.5	94.1	96.4
CircleLoss [211]	CVPR'20	IBN	512	83.4	89.8	94.1	96.5
RankMI [99]	CVPR'20	RN50	128	83.3	89.8	93.8	96.5
PADS [183]	CVPR'20	RN50	128	83.5	89.7	93.8	-
NormSoftmax [264]	BMVC'19	RN50	512	84.2	90.4	94.4	96.9
ProxyAnchor [101]	CVPR'20	IBN	512	86.1	91.7	95.0	97.3
ProxyNCA++[217][a]	ECCV'20	RN50-GMP	512	86.5	92.5	95.7	97.7
Ensemble approaches:							
HDC [262]	ICCV'17	GN	384	73.7	83.2	89.5	93.8
BIER Learner-3 [167]	ICCV'17	GN	256	76.5	84.9	90.9	94.9
BIER [167]	ICCV'17	GN	512	78.0	85.8	91.1	95.1
A-BIER Learner-3 [168]	TPAMI'18	GN	256	80.6	88.2	92.3	95.8
A-BIER [168]	TPAMI'18	GN	512	82.0	89.0	93.2	96.1
HORDE [91][a]	CVPR'19	GN	512	83.2	89.6	93.6	96.3
RLL-(L,M,H) [242]	CVPR'19	IBN	1536	82.1	89.3	93.7	96.7
MIC [182]	CVPR'19	RN50	128	82.6	89.1	93.2	-
D&C [188]	CVPR'19	RN50	128	84.6	90.7	94.1	96.5
ABE-8 [103]	ECCV'18	GNx8	512	85.2	90.5	94.0	96.1
DREML [248]	ECCV'18	RN18x48	9216	86.0	91.7	95.0	97.2
HORDE [91][a]	CVPR'19	IBN	512	86.2	92.9	95.1	97.2
DiVA [150]	ECCV'20	RN50	512	87.6	92.9	-	-

Table 4.18: Comparison with the state-of-the-art on the Cars-196 [109] dataset.

[a] Uses non-standard data augmentation (multi-scale training, *etc.*).

Method	Year	Net	Size	1	10	100	1000
Contrastive [163]	CVPR'16	GN	128	42.0	58.2	73.8	89.1
Triplet [163]	CVPR'16	GN	128	42.1	63.5	82.5	94.8
LiftedStruct [163]	CVPR'16	GN	128	62.1	79.8	91.3	97.4
Binomial Deviance [223]	NIPS'16	GN	512	65.5	82.3	92.3	97.6
Histogram Loss [223]	NIPS'16	GN	512	63.9	81.7	92.2	97.7
N-Pair-Loss [206]	NIPS'17	GN	64	67.7	83.8	93.0	97.8
Our Baseline	ICCV'17	GN	512	66.2	82.3	91.9	97.4
Clustering [207]	CVPR'17	IBN	64	67.0	83.7	93.2	-
NormSoftmax [264]	BMVC'19	GN	512	69.0	84.5	93.1	-
HDML [280]	CVPR'19	GN	512	68.7	83.2	92.4	-
DVML [135]	ECCV'18	GN	512	70.2	85.2	93.8	-
Angular Loss [238]	ICCV'17	GN	512	70.9	85.0	93.5	98.0
Margin [280]	ICCV'17	RN50	128	72.7	86.2	93.8	98.0
ProxyNCA [153]	ICCV'17	IBN	64	73.7	-	-	-
NormSoftmax [264]	BMVC'19	IBN	512	73.8	88.1	95.0	-
RankMI [99]	CVPR'20	RN50	128	74.3	89.7	94.9	98.3
HTL [58]	ECCV'18	IBN	512	74.8	88.3	94.8	98.4
RLL-H [242]	CVPR'19	IBN	512	76.1	89.1	95.4	89.7
PADS [183]	CVPR'20	RN50	128	76.5	89.0	95.4	-
Our Baseline (RN)	TPAMI'18	RN50	512	77.7	-	-	-
TML [260]	ICCV'19	RN50	512	78.0	91.2	96.7	99.0
MS [243]	CVPR'19	IBN	512	78.2	90.5	96.0	98.7
NormSoftmax [264]	BMVC'19	RN50	512	78.2	90.6	96.2	-
DiVA Baseline (Margin) [150]	ECCV'20	RN50	512	78.3	90.0	-	-
SoftTriple [173]	ICCV'19	IBN	512	78.3	90.3	95.9	-
CircleLoss [211]	CVPR'20	IBN	512	78.3	90.5	96.1	98.6
ProxyAnchor [101]	CVPR'20	IBN	512	79.1	90.8	96.2	98.7
ProxyNCA++[217][a]	ECCV'20	RN50-GMP	512	80.7	92.0	96.7	98.9
Ensemble approaches:							
HDC [262]	ICCV'17	GN	384	69.5	84.4	92.8	97.7
BIER Learner-3 [167]	ICCV'17	GN	256	72.5	86.3	93.9	97.9
HORDE [91][a]	ICCV'19	GN	512	72.6	85.9	93.7	97.9
BIER [167]	ICCV'17	GN	512	72.7	86.5	94.0	98.0
A-BIER Learner-3	TPAMI'18	GN	256	74.0	86.8	93.9	97.8
A-BIER [168]	TPAMI'18	GN	512	74.2	86.9	94.0	97.8
D&C [188]	CVPR'19	RN50	128	75.9	88.4	94.9	-
ABE-8 [103]	ECCV'18	GNx8	512	76.3	88.4	94.8	-
MIC [182]	CVPR'19	RN50	128	77.2	89.4	95.6	-
A-BIER Learner	TPAMI'18	RN50	128	77.2	-	-	-
A-BIER	TPAMI'18	RN50	512	79.0	-	-	-
DiVA [150]	ECCV'20	RN50	512	79.6	91.2	-	-
RLL-(L,M,H) [242]	CVPR'19	IBN	1536	79.8	91.3	96.3	90.4
HORDE [91][a]	ICCV'19	IBN	512	80.1	91.3	96.2	98.7

Table 4.19: Comparison with the state-of-the-art on the Stanford Online Products [163] dataset.

[a] Uses non-standard data augmentation (multi-scale training, *etc.*).

Method	Year	Net	Size	R@1	R@10	R@20	R@30
FashionNet + Joints [141]	CVPR'16	VGG	4096	41.0	64.0	68.0	71.0
FashionNet + Poselets [141]	CVPR'16	VGG	4096	42.0	65.0	70.0	72.0
FashionNet [141]	CVPR'16	VGG	4096	53.0	73.0	76.0	77.0
HTL [58]	ECCV'18	IBN	512	80.9	94.3	95.8	97.2
MS [243]	CVPR'19	IBN	512	89.7	97.9	98.5	98.8
ProxyNCA++[217][a]	ECCV'20	RN50 GMP	512	90.4	98.1	98.8	
ProxyAnchor [101]	CVPR'20	IBN	512	91.5	98.1	98.8	-
MS+DIR [151]	CVPR'20	IBN	512	91.7	98.1	98.7	98.9
Ensemble approaches:							
HDC [262]	ICCV'17	GN	384	62.1	84.9	89.0	91.2
Ours Baseline	ICCV'17	GN	512	70.6	90.5	93.4	94.7
BIER Learner-3 [167]	ICCV'17	GN	256	76.4	92.7	95.0	96.1
BIER [167]	ICCV'17	GN	512	76.9	92.8	95.2	96.2
A-BIER Learner-3	TPAMI'18	GN	256	82.8	95.0	96.8	97.4
A-BIER	TPAMI'18	GN	512	83.1	95.1	96.9	97.5
HORDE [91][a]	ICCV'19	GN	512	84.4	95.4	96.8	97.4
ABE-8 [103]	ECCV'18	GNx8	512	87.3	96.7	97.9	98.2
HORDE [91][a]	ICCV'19	IBN	512	90.4	97.8	98.4	98.7

Table 4.20: Comparison with the state-of-the-art on the In-Shop Clothes Retrieval [141] dataset.

[a] Uses non-standard data augmentation (multi-scale training, *etc.*).

remaining 100 classes (5,924 images) for testing. Similarly, we split the Cars-196 dataset by using the first 98 classes (8,054 images) for training and the remaining 98 classes for testing (8,131 images). The Stanford Online Products consists of 59,551 training images of 11,318 classes and 60,502 test images of 11,316 classes. After training on these dataset, for each test image, we retrieve the nearest neighbors from the remaining test images. We then compute the Recall@K metric.

The In-Shop Clothes Retrieval dataset has predefined 25,882 images of 3,997 classes for training. For evaluation the dataset defines a query set consisting of 14,218 images of 3,985 classes and a gallery set consisting of 12,612 images of 3,985 classes. During evaluation, for each image of the query set we retrieve the nearest neighbors from the gallery set and compute the Recall@K metric.

VehicleID also has a predefined training set consisting of 25,882 images of 3,997 classes. The dataset defines three increasingly large datasets for testing (Small, Medium, Large) on which we evaluate our trained network with the Recall@K metric.

In our experiments, we train *BIER* with the binomial deviance loss and an embedding size of 512. We set the number of learners to 3 with embedding sizes proportional to their

boosting weights, *i.e.* 96, 160 and 256. When we evaluate on the CUB-200-2011 and Cars-196 dataset, we follow previous works, *e.g.* [163], and report Recall@K, $K \in \{1, 2, 4, 8\}$. As Stanford Online Products is larger than the Cars-196 and CUB-200-2011 dataset, similar to previous works (*e.g.* [163]), we report Recall@K, $K \in \{1, 10, 100, 1000\}$. For the In-Shop Clothes Retrieval and VehicleID datasets, we report Recall@K with $K \in \{1, 10, 20, 30\}$ and $K \in \{1, 5\}$, repectively.

In our experiments we also report the results of the last learner in our ensemble, *i.e.* *BIER Learner-3*, as it achieves very competitive results an its own and was trained to focus on the most difficult examples. Further, we report the accuracy of *BIER* with our auxiliary adversarial loss function during training time (*A-BIER*) on all datasets, as it outperforms our activation loss function in our ablation study experiments. Similar to standard *BIER* we also report the results of our last learner in this ensemble (*A-BIER Learner-3*).

Finally, as retrieval accuracy largely depends on the feature extractor [184], we make *BIER* great again by using a more recent ResNet-50 [75] feature extractor. We evaluate *BIER* with a ResNet-50 feature extractor on the Stanford Online Products [163] dataset. For these experiments, we also follow the standard training protocols, *e.g.* [150, 182, 184]. Specifically, as our GoogLeNet version, we use input sizes of 224×224 pixels, do standard data augmentation and use an embedding size of 512. Similar to our GoogLeNet version, we also use the binomial deviance loss function.

Results and baselines are shown in Tables 4.17, 4.18, 4.19, 4.20 and 4.21. At time of publication, our method in combination with a simple loss function was able to achieve competitive results or outperform state-of-the-art methods, which typically use more complex loss functions, *e.g.* [153, 163, 206, 238], or sampling strategies, *e.g.* [73, 246, 262] compared our method. When we apply our adversarial loss function during training (*A-BIER* [168]), we further improve the performance of *BIER* [167]. Both our methods improve our baseline, *i.e.* a network trained with binomial deviance loss, by a large margin. Notably, most published methods, which adopt the GoogLeNet typically achieve lower accuracy in terms of R@1 compared to our method.

Further, similar to the findings of Roth *et al.* [184], we find that by switching to a more modern backbone network, we significantly can improve our method on the Stanford Online Products dataset [163]. *A-BIER* trained with a ResNet-50 backbone and a simple loss function, can still achieve very competitive results compared to modern state-of-the-art approaches, which use a similar embedding size and feature extractor, *e.g.* [150, 217, 260, 264]. A single learner in our ensemble with a dimensionality of 128 achieves also comparable results to state-of-the-art works using the same feature extractor and dimensionality, *e.g.* [99, 182, 183, 188]. We also illustrate the progress in retrieval accuracy on the Stanford Online Products in Figure 4.13. We see that the choice of feature extractor has a large impact on retrieval accuracy.

Several successive works adopted and improved efficient ensembles for metric learning over the years, *e.g.* [91, 103, 150, 182, 188, 242, 248, 262]. This confirms their success in

Method	Year	Net	Size	Small		Medium		Large	
				1	5	1	5	1	5
Mixed Diff+CCL [136]	CVPR'16	VGG	1024	49.0	73.5	42.8	66.8	38.2	61.6
GS-TRS loss [5]	arxiv'17	VGG	1024	75.0	83.0	74.1	82.6	73.2	81.9
Ensemble approaches:									
Ours Baseline	ICCV'17	GN	512	78.0	87.5	73.0	84.7	67.9	82.4
BIER Learner-3 [167]	ICCV'17	GN	512	82.6	90.5	79.3	88.0	75.5	86.0
BIER [167]	ICCV'17	GN	512	82.6	90.6	79.3	88.3	76.0	86.4
A-BIER Learner-3	TPAMI'18	GN	512	86.0	92.7	83.2	88.6	81.5	88.6
A-BIER	TPAMI'18	GN	512	86.3	92.7	83.3	88.7	81.9	88.7

Table 4.21: Comparison with the state-of-the-art on VehicleID [136].

metric learning.

Specifically, Attention-Based Ensemble (ABE)-8 [103] combines diversity loss functions with attention to make learners diverse from each other. While this method typically achieves better results compared to our method, it also has significantly higher computational cost, as the ensemble needs to be split at an early layer in the network. Deep Randomized Ensembles for Metric Learning (DREML) [248] uses a different bagging of the label space per learner to make the ensemble diverse. However, in their experiments, they use a large ensemble of several ResNets. Consequently, there is a large computational overhead. Higher Order Regularizer for Deep Embeddings (HORDE) [91] uses "higher-order" regularization to train their ensemble. While this method achieves state-of-the-art results, they employ non-standard data augmentation (*e.g.* multi-scale training, higher input resolution, *etc.*). Ranked List Loss (RLL) [242] and Hardness Aware Deeply Cascaded Embedding (HDC) [262] use the benefits of deeply-supervised networks [118] to train a multi-scale ensemble. This ensembling approach is orthogonal to ours. Divide and Conquer (D&C) [188] and Mining Interclass Characteristics (MIC) [182] use clusterings [182, 188] and a diversity loss function [182] to make learners diverse from each other. In our comparison they perform worse than our ResNet implementation, because their embedding size is too small to exploit the benefits of ensembles. Their successive work, *i.e.* Diverse Visual Feature Aggregation for Deep Metric Learning (DiVA) [150], combines our auxiliary loss function with different self-supervised auxiliary tasks for different learners. They employ a state-of-the-art loss function [184, 246], and consequently outperform our ensemble approach.

4.8 Summary

In this chapter, we analyzed parameter shared *CNN* ensembles. These ensembles share a common feature representation but have multiple classification heads. All heads try to solve the same classification problem. The main problem of these efficient ensembles

is that especially when they share a large number of parameters, the learner predictions are highly correlated. To address this issue, we introduced several auxiliary loss functions, *i.e.* the DivLoss, Activation Loss, and Adversarial Loss. We showed that auxiliary loss functions benefit the diversity of such ensembles and consequently improve classification performance. We showed the effectiveness of our method on several small object categorization datasets.

In our extensive experiments, our method achieves competitive results to more recently published ensembling methods. Compared to dropout, a traditional ensembling method, our learners achieve higher accuracy and are more diverse from each other. However, unlike dropout, our ensemble consists of a comparably smaller number of learners (*i.e.* 2-8). In contrast, dropout ensembles have an exponential number of learners.

Further, we showed that our method also benefits modern ResNet architectures. Efficient ResNet ensembles can typically achieve a better speed vs accuracy trade-off on these datasets.

Finally, we adapted our method to real-world image retrieval tasks. As we showed in our evaluation, efficient parameter-shared ensembles trained with gradient online boosting and with diversity auxiliary loss functions significantly benefit image retrieval performance. In this specific field, efficient ensembles then gained a lot of research attention. A large number of successive works, *e.g.* [91, 103, 150, 182, 188, 242, 248, 262], improved over our diversity encouraging methods, and show that state-of-the-art methods in this field, can still significantly benefit from efficient ensembles.

5 Conclusion

"Wisdom comes from experience. Experience is often a result of lack of wisdom."

— Terry Pratchett

Contents

5.1 Summary 105

5.2 Future Work 107

5.1 Summary

In this work, we gave an overview of several popular ensemble methods in the field of deep learning. We categorized ensemble methods into implicit methods (*e.g.* dropout variants) and explicit methods (*e.g.* standard ensembles). We analyzed explicit ensembles for deep learning. The main disadvantage of these ensembles is that they are computationally too expensive for applications with real-time requirements. Therefore, we proposed to share most parameters in these ensembles. Consequently, such ensembles are computationally cheaper and have less memory demand compared to standard ensembles. However, one limitation of such ensembles is, that by sharing most parameters with each other, the individual learners become increasingly correlated with each other. Consequently, such ensembles show no benefits, according to the bias-variance-covariance decomposition (Chapter 2).

This observation lead us to the following research question: "How can we increase the diversity in explicit parameter shared ensembles?". To address this question, we proposed

several methods to increase the diversity in such ensembles. Specifically, we made three contributions:

- First, in Chapter 3 we leveraged spatial independence by training discriminative part-detectors for face detection. These part-detectors share a common feature representation. Subsequently, we could map them back to a standard Convolutional Neural Network (CNN) during test-time. Consequently, it does not impose any additional runtime-overhead. However, one limitation of this approach is, that our method has specific requirements on the network architecture. Specifically, the receptive fields in the layer, where we split our shared network into multiple learners, should just cover the prototypical object parts (such as eye, mouth, nose, *etc.*) and not the full object. In addition, our method needs pose-annotations during training time.

- Second, in Chapter 4.2, we proposed auxiliary loss functions to increase diversity in a parameter-shared *CNN* for an object categorization task. The main benefit of this contribution is, that it addresses the limitation of our first one. It does not impose any requirements on the network architecture and does not need additional annotations. In our experiments, we showed that our ensembles are also very competitive to recent state-of-the-art methods, which propose similar ideas to ours. Further, we showed that our ensembles are typically more parameter-efficient (*i.e.* achieve higher accuracy with less parameters and FLOPS) compared to (wide) ResNet based networks.

- Third, in Chapter 4.3, we proposed a gradient-boosting based re-weighting scheme and combined it with our second contribution for a real-world image-retrieval problem. In this setting, we showed that our approach can benefit standard *CNNs* in terms of accuracy, without introducing any additional parameters and only negligible additional runtime and memory overhead. Since time of publication, several successive works in the area of metric learning for image retrieval, *e.g.* [103, 150, 170, 182, 188] (see Chapter 2), adopted this multi-head ensemble strategy and improved it by using different means to encourage diversity between learners.

Most current research in visual recognition typically focuses on improving a single model by *e.g.* improving architectural choices, *e.g.* [75, 212], improving loss functions *e.g.* [58, 242], increasing the spatial resolutions of a single network, *e.g.* [132], *etc.* Our contributions are complementary to most other research efforts. Specifically, our contributions in Chapter 4 could benefit a wide variety of different computer vision problems, as they do not impose any requirements on the network architecture or require any additional supervision.

One limitation of our work is that our ensemble has a small number of learners compared to traditional ensemble approaches such as Random Forest or Boosting. The main

reason for this is that during training, we need to keep all learners simultaneously in memory, as they share parameters with each other. Further, we encourage diversity between learners with an auxiliary loss function. These functions need to evaluate the activations of pairs of learners during training for each sample. In contrast, standard ensembles can be arbitrarily large, as we can train and evaluate them successively.

Another disadvantage of our work is that compared to standard architectural choices in *CNNs*, our method requires changes to the last layer of a *CNN*. Consequently, we cannot just pre-train a large network with our method on a large dataset (*e.g.* ImageNet) and reuse it in "downstream" applications such as object detection and semantic segmentation. We need to adapt these "downstream" tasks to properly use the ensemble, which requires additional engineering effort.

5.2 Future Work

Due to the rapid advances in Graphics Processing Unit (GPU) hardware and special hardware accelerators for deep learning, such as the Tensor Processing Unit (TPU), more computational resources and more memory capacity will be available for training neural networks. Consequently, this will allow to scale parameter-shared ensembles to a larger number of learners and to reduce the amount of shared parameters among learners. Further, more computational resources will make efficient ensembles more applicable to memory intensive tasks such as object detection, or semantic segmentation. Further, we showed in our experiments that parameter-shared ensembles typically achieve better pareto-optimal results compared to standard *CNNs*. Consequently, with the advances in compute, it will be possible to include these types of networks into the search space of Neural Architecture Search (NAS) frameworks, *e.g.* [286].

Successive works, *e.g.* [103, 150, 170, 182, 188], in the area of metric learning already explored the benefits of parameter-shared ensembles and proposed novel methods to make learners diverse from each other. These approaches could be adopted for generic object classification tasks, making them available to a wider variety of computer vision applications. Specifically, attention-based ensembles [103] might be beneficial for building part-based object detectors. However, as these ensembles typically do not share most parameters, this is currently computationally expensive. Another interesting line of work are methods which introduce diversity by self-supervised learning methods [150]. These methods could be adopted by *e.g.* biasing different learners with different auxiliary self-supervised functions or using different unlabeled datasets for different learners. Further, the diversity encouraging loss functions we proposed in this thesis could also be applied to non-labelled datasets.

Another interesting future research direction for our work is trying to combine efficient ensembles with other types of neural networks, such as Transformers [225], PointNets [172], *etc.* Specifically, recent research suggests that Transformers can achieve very competitive or better performance compared to *CNNs* on image classification, *e.g.* [221]. It might be

beneficial for these architectures to exploit the benefits of ensembles by *e.g.* integrating multiple classification tokens and bias their attention mask towards different channels, *etc.*

Another research direction is applying parameter-shared ensembles in applications, where ensembles are successful. For example, uncertainty estimation, *e.g.* [2, 87], for deep learning is an increasingly important problem. Standard *CNNs* tend to make high confidence predictions on samples, which are very different from the training distribution, *e.g.* [131]. To address this problem, several works use MC-Dropout to measure the variation in predictions of the individual dropout learners, *e.g.* [2, 87]. Uncertainty estimates from ensembles are also useful for active learning [9]. Active learning tries to minimize the labeling effort by carefully selecting non-annotated images for labeling. To minimize labeling effort, such methods try to select samples for labeling, where the current network is uncertain. By re-training the network on uncertain samples, the accuracy improves more compared to re-training on random samples. Parameter-shared ensembles could allow to obtain cheaper uncertainty estimations compared to full ensembles or MC-Dropout approaches.

Another emerging line of work is continual learning [115]. These approaches try to incrementally train a learner (*i.e.* a neural network) over time. In these setting the data distribution is not independent and identically distributed, but changes over time. One of the challenges in this area is so-called "catastrophic forgetting". For example, consider a continual learning system for self-driving cars. Such a learning system should be able to incrementally learn to navigate through changing weather conditions. In this case, the data distribution might be sunny in the beginning, but then shift towards rainy. Neural network tend to forget in this case the sunny data distribution, while driving in the rain. However, if the weather gets sunny again, the network should still be able to perform well in these conditions. Continual learning approaches address this problem. One way parameter-shared ensembles could be useful in these settings is to allocate certain learners to specific sub-parts of the data distribution (*i.e.* one learner to sunny, another to rainy). Another way to apply ensembles is to let several learners change slower over time, so that they have a better chance at learning the full data distribution. Another part of the learners changes faster over time. Subsequently, they have a chance to become "experts" for certain weather conditions.

To conclude this thesis, while compared to traditional ensembles, our method cannot scale easily to arbitrarily large ensembles, future increase in *GPU* memory and compute will alleviate this problem. Further, as we showed in our experiments, even small ensemble sizes can benefit current *CNNs* and achieve a better speed-vs-accuracy trade-off. We think that such ensembles can benefit a wide variety of computer vision problems in the field of visual recognition. Further, they might be beneficial for emerging research works such as uncertainty estimation or continual learning.

A

List of Acronyms

"If only we had laboratories to produce self-replicating scientists, to explore all the worlds. Ah, but we do! They're called university campuses."

— Terry Pratchett

ABE	Attention-Based Ensemble
ACF	Aggregate Channel Features
AFLW	Annotated Facial Landmarks in the Wild
AFW	Annotated Faces in the Wild
AP	Average Precision
BIER	Boosting Independent Embeddings Robustly
CED	Coalition-based Ensemble Design
CNN	Convolutional Neural Network
COFW	Caltech Occluded Faces in the Wild
CPU	Central Processing Unit
D&C	Divide and Conquer
DiVA	Diverse Visual Feature Aggregation for Deep Metric Learning
DPM	Deformable Parts Model
DREML	Deep Randomized Ensembles for Metric Learning
FDDB	Face Detection Data Set and Benchmark
FPN	Feature Pyramid Network
FPPI	False Positives Per Image
GAN	Generative Adversarial Network

GPU	Graphics Processing Unit
HDC	Hardness Aware Deeply Cascaded Embedding
HOG	Histogram of Oriented Gradients
HORDE	Higher Order Regularizer for Deep Embeddings
IoU	Intersection over Union
LCN	Local Contrast Normalization
LDA	Linear Discriminant Analysis
mAP	mean Average Precision
MIC	Mining Interclass Characteristics
MSE	Mean Square Error
NAS	Neural Architecture Search
NCA	Neighborhood Component Analysis
NCL	Negative Correlation Learning
NMS	Non Maxima Suppression
OHEM	Online Hard Example Mining
ReLU	Rectified Linear Unit
RLL	Ranked List Loss
RoI	Region of Interest
RPN	Region Proposal Network
SGD	Stochastic Gradient Descent
SNR	Signal to Noise Ratio
SSE	Sum of Squares Error
SVM	Support Vector Machine
TPU	Tensor Processing Unit

List of Publications

"How many roads must a man walk down?"

— Douglas Noel Adams

My work at the Institute of Computer Graphics and Vision led to the following peer-reviewed publications.

2016

Grid Loss: Detecting Occluded Faces

M. Opitz, G. Waltner, G. Poier, H. Possegger, and H. Bischof.
In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016

Abstract: Detection of partially occluded objects is a challenging computer vision problem. Standard Convolutional Neural Network (CNN) detectors fail if parts of the detection window are occluded, since not every sub-part of the window is discriminative on its own. To address this issue, we propose a novel loss layer for CNNs, named grid loss, which minimizes the error rate on sub-blocks of a convolution layer independently rather than over the whole feature map. This results in parts being more discriminative on their own, enabling the detector to recover if the detection window is partially occluded. By mapping our loss layer back to a regular fully connected layer, no additional computational cost is incurred at runtime compared to standard CNNs. We demonstrate our method for face detection on several public face detection benchmarks and show that our method outperforms regular CNNs, is suitable for realtime applications and achieves state-of-the-art performance.

Chapters: 3

Efficient Model Averaging for Deep Neural Networks

M. Opitz, H. Possegger, and H. Bischof.
In *Proceedings of the Asian Conference on Computer Vision (ACCV)*,
2016

Abstract: Large neural networks trained on small datasets are increasingly prone to overfitting. Traditional machine learning methods can reduce overfitting by employing bagging or boosting to train several diverse models. For large neural networks, however, this is prohibitively expensive. To address this issue, we propose a method to leverage the benefits of ensembles without explicitely training several expensive neural network models. In contrast to Dropout, to encourage diversity of our sub-networks, we propose to maximize diversity of individual networks with a loss function: DivLoss. We demonstrate the effectiveness of DivLoss on the challenging CIFAR datasets.

Chapters: 4

BaCoN: Building a Classifier from only N Samples

G. Waltner, **M. Opitz**, and H. Bischof.
In *Proceedings of the Computer Vision Winter Workshop (CVWW)*,
2016

Abstract: We propose a model able to learn new object classes with a very limited amount of training samples (*i.e.* 1 to 5), while requiring near zero runtime cost for learning new object classes. After extracting Convolutional Neural Network (CNN) features, we discriminatively learn embeddings to separate the classes in feature space. The proposed method is especially useful for applications such as dish or logo recognition, where users typically add object classes comprising a wide variety of representations. Another benefit of our method is the low demand for computing power and memory, making it applicable for object classification on embedded devices. We demonstrate on the Food-101 dataset that even one single training example is sufficient to recognize new object classes and considerably improve results over the probabilistic Nearest Class Means (NCM) formulation.

2017

BIER: Boosting Independent Embeddings Robustly

M. Opitz, G. Waltner, H. Possegger, and H. Bischof.
In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*,
2017.
(accepted for oral presentation)

Abstract: Learning similarity functions between image pairs with deep neural networks yields highly correlated activations of large embeddings. In this work, we show how to improve the robustness of embeddings by exploiting independence in ensembles. We divide the last embedding layer of a deep network into an embedding ensemble and formulate training

this ensemble as an online gradient boosting problem. Each learner receives a reweighted training sample from the previous learners. This leverages large embedding sizes more effectively by significantly reducing correlation of the embedding and consequently increases retrieval accuracy of the embedding. Our method does not introduce any additional parameters and works with any differentiable loss function. We evaluate our metric learning method on image retrieval tasks and show that it improves over state-of-the-art methods on the CUB-200-2011, Cars-196, Stanford Online Products, In-Shop Clothes Retrieval and VehicleID datasets by a significant margin.

Chapters: 4

Loss-Specific Training of Random Forests for Super-Resolution

A. Grabner, G. Poier, **M. Opitz**, S. Schulter, and P. Roth.
In *Proceedings of the Computer Vision Winter Workshop (CVWW)*,
2017

Abstract: Super-resolution addresses the problem of image upscaling by reconstructing high-resolution output images from low-resolution input images. One successful approach for this problem is based on random forests. However, this approach has a large memory footprint, since complex models are required to achieve high accuracy. To overcome this drawback, we present a novel method for constructing random forests under a global training objective. In this way, we improve the fitting power and reduce the model size. In particular, we combine and extend recent approaches on loss-specific training of random forests. However, in contrast to previous works, we train random forests with globally optimized structure and globally optimized prediction models. We evaluate our proposed method on benchmarks for single image super-resolution. Our method shows significantly reduced model size while achieving competitive accuracy compared to state-of-the art approaches.

Pedestrian Detection in RGB-D Images from an Elevated Viewpoint

C. Ertler, H. Possegger, **M. Opitz**, and H. Bischof
In *Proceedings of the Computer Vision Winter Workshop (CVWW)*,
2017

Abstract: We propose an extension to the state-of-the-art Faster R-CNN detection model for multi-modal pedestrian detection from RGB-D images. The proposed architectures address this problem by fusing convolutional neural network (CNN) representations. We elaborate two architectures, which primarily differ in the position of the fusion inside the model, and further compare several static and parametrized fusion layers. Moreover, we show how recent advances in the area of non-maximum suppression (NMS) can improve the detection results of our models and make them more robust in applications with varying

pedestrian densities. Our models are trained and evaluated on a custom dataset com-prising images of crosswalk scenes taken from an elevated viewpoint. This viewpoint results in uncommon and highly variable poses of pedestrians, demanding powerful detection models.

2018

Deep Metric Learning with BIER: Boosting Independent Embeddings Robustly

M. Opitz, G. Waltner, H. Possegger, and H. Bischof.
Transactions on Pattern Analysis and Machine Intelligence (TPAMI),
2018

Abstract: Learning similarity functions between image pairs with deep neural networks yields highly correlated activations of embeddings. In this work, we show how to improve the robustness of such embeddings by exploiting the independence within ensembles. To this end, we divide the last embedding layer of a deep network into an embedding ensemble and formulate training this ensemble as an online gradient boosting problem. Each learner receives a reweighted training sample from the previous learners. Further, we propose two loss functions which increase the diversity in our ensemble. These loss functions can be applied either for weight initialization or during training. Together, our contributions leverage large embedding sizes more effectively by significantly reducing correlation of the embedding and consequently increase retrieval accuracy of the embedding. Our method works with any differentiable loss function and does not introduce any additional parameters during test time. We evaluate our metric learning method on image retrieval tasks and show that it improves over state-of-the-art methods on the CUB 200-2011, Cars-196, Stanford Online Products, In-Shop Clothes Retrieval and VehicleID datasets.

Chapters: 4

An Intent-Based Automated Traffic Light for Pedestrians

C. Ertler, H. Possegger, **M. Opitz**, and H. Bischof.
In *Proceedings of the IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*,
2018

Abstract: We propose a fully automated, vision-based traffic light for pedestrians. Traditional industrial solutions only report people standing in a constrained waiting zone near the crosswalk. However, reporting only people below the traffic light does not allow for efficient traffic scheduling. For example, some pedestrians do not want to cross the street and walk past the traffic light, or just wait for another person to arrive. In contrast, our system leverages intent prediction to estimate which pedestrians are actually going to cross the road by analyzing both short-term and long-term trajectory cues. In this way, we can

decrease the waiting times and pave the road for optimal and adaptive traffic light scheduling. We conduct a long-term evaluation in a European capital that proves the applicability and reliability of our system and demonstrates that it is not only able to replace existing push-button solutions but also yields additional information that can be used to further optimize traffic light scheduling.

Deep 2.5D Vehicle Classification with Sparse SfM Depth Prior for Automated Toll Systems

G. Waltner, M. Maurer, T. Holzmann, P. Ruprecht, **M. Opitz**, H. Possegger, F. Fraundorfer, and H. Bischof.
In *Proceedings of the IEEE Intelligent Transportation Sys tems Conference (ITSC)*, 2018

Abstract: Automated toll systems rely on proper classification of the passing vehicles. This is especially difficult when the images used for classification only cover parts of the vehicle. To obtain information about the whole vehicle. We reconstruct the vehicle as 3D object and exploit this additional information within a Convolutional Neural Network (CNN). However, when using deep networks for 3D object classification, large amounts of dense 3D models are required for good accuracy, which are often neither available nor feasible to process due to memory requirements. Therefore, in our method we reproject the 3D object onto the image plane using the reconstructed points, lines or both. We utilize this sparse depth prior within an auxiliary network branch that acts as a regularizer during training. We show that this auxiliary regularizer helps to improve accuracy compared to 2D classification on a real-world dataset. Furthermore due to the design of the network, at test time only the 2D camera images are required for classification which enables the usage in portable computer vision systems.

2019

HiBsteR: Hierarchical Boosted Deep Metric Learning for Image Retrieval

G. Waltner, **M. Opitz**, H. Possegger, and H. Bischof.
In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*,
2019

Abstract:
When the number of categories is growing into thou-sands, large-scale image retrieval becomes an increasingly hard task. Retrieval accuracy can be improved by learning distance metric methods that separate categories in a trans-formed embedding space. Unlike most methods that utilize a single embedding to learn a distance metric, we build on the idea of boosted metric learning, where an embed-ding is split into a boosted ensemble of embeddings. While in general metric learning is directly applied on fine labels to learn embeddings,

we take this one step further and incorporate hierarchical label information into the boosting framework and show how to properly adapt loss functions for this purpose. We show that by introducing several sub-embeddings which focus on specific hierarchical classes, the retrieval accuracy can be improved compared to standard flat label embeddings. The proposed method is especially suitable for exploiting hierarchical datasets or when additional labels can be retrieved without much effort.Our approach improves R@1 over state-of-the-art methods on the biggest available retrieval dataset (Stanford Online Products) and sets new reference baselines for hierarchical metric learning on several other datasets (CUB-200-2011,VegFru, FruitVeg-81). We show that the clustering quality in terms of NMI score is superior to previous works.

Semi-supervised Detector Training with Prototypes for Vehicle Detection

G. Waltner, **M. Opitz**, G. Krispel, H. Possegger, and H. Bischof

In *Proceedings of the IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019

Abstract:
Adapting detectors to new datasets is needed in scenarios where a user has a specific dataset that contains novel classes or is recorded in a setting where a pretrained detector fails. While detectors based on Convolutional Neural Networks(CNNs) are state-of-the-art and nowadays publicly available,they suffer from bad generalization capabilities when applied on datasets that notably differ from the one they were trained on. Finetuning the detector is only possible if the dataset is large enough to not destroy the underlying feature representation.We propose a method where only a few prototypes are labeled for training in a semi-supervised manner. In particular, we separate the detection from the classification step to avoid impairing the bounding box proposal generation. Our trained prototype classification network provides labels to automatically source a large dataset containing 20 to 30 times more samples without further supervision, which we then use to train a more powerful network. We evaluate our method on a private vehicle dataset with six classes and show that evaluating on a previously unseen recording site we can gain an accuracy increase of 9% at same precision and recall levels. We further show that finetuning with as few as 25 labeled samples per class doubles accuracy compared to directly using pretrained features for nearest neighbor classification

MURAUER: Mapping Unlabeled Real Data for Label AUstERity

G. Poier, **M. Opitz**, D. Schinagl, and H. Bischof.

In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*,

2019

Abstract: Data labeling for learning 3D hand pose estimation models is a huge effort. Readily available, accurately labeled synthetic data has the potential to reduce the effort. However, to successfully exploit synthetic data, current state-of-the-art methods still require a large amount of labeled real data. In this work, we remove this requirement by learning

to map from the features of real data to the features of synthetic data mainly using a large amount of synthetic and unlabeled real data. We exploit unlabeled data using two auxiliary objectives, which enforce that (i) the mapped representation is pose specific and (ii) at the same time, the distributions of real and synthetic data are aligned. While pose specifity is enforced by a self-supervisory signal requiring that the representation is predictive for the appearance from different views, distributions are aligned by an adversarial term. In this way, we can significantly improve the results of the baseline system, which does not use unlabeled data and outperform many recent approaches already with about 1% of the labeled real data. This presents a step towards faster deployment of learning based hand pose estimation, making it accessible for a larger range of applications.

Detecting Out-of-Distribution Traffic Signs

M. Iyengar, **M. Opitz**, and H. Bischof.

In *Proceedings of the Austrian Association for Pattern Recognition Workshop (AAPR)*,

2019

Abstract:
This work addresses the problem of novel traffic sign detection, i.e. detecting new traffic sign classes during test-time, which were not seen by the classifier during training. This problem is especially relevant for the development of autonomous vehicles, as these vehicles operate in an open-ended environment. Due to which, the vehicle will always come across a traffic sign that it has never seen before. These new traffic signs need to be immediately identified so that they can be used later for re-training the vehicle. However, detecting these novel traffic signs becomes an extremely difficult task,as there is no mechanism to identify from the output of the classifier whether it has seen a given test sample before or not.To address this issue, we pose the novel traffic-sign detection problem as an out-of-distribution (OOD) detection problem.We apply several state-of-the-art OOD detection methods and novelty detection methods on the novel traffic-sign detection problem and also establish a benchmark using the German Traffic Sign Recognition Benchmark dataset(GTSRB). In our evaluation, we show that both out-of-distribution approaches and novelty detection approaches are suitable for OOD traffic sign detection.

2020

FuseSeg: LiDAR Point Cloud Segmentation Fusing Multi-Modal Data

G. Krispel, **M. Opitz**, G. Waltner, H. Possegger, and H. Bischof.

In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*,

2020

Abstract: We introduce a simple yet effective fusion method of LiDAR and RGB data to segment LiDAR point clouds. Utilizing the dense native range representation of a LiDAR

sensor and the setup calibration, we establish point correspondences between the two input modalities. Subsequently, we are able to warp and fuse the features from one domain into the other. Therefore, we can jointly exploit information from both data sources within one single network. To show the merit of our method, we extend SqueezeSeg, a point cloud segmentation network, with an RGB feature branch and fuse it into the original structure. Our extension called FuseSeg leads to an improvement of up to 18% IoU on the KITTI benchmark. In addition to the improved accuracy, we also achieve real-time performance at 50 fps, five times as fast as the KITTI LiDAR data recording speed.

Bibliography

"Science and technology are not advanced by people who believe, but by people who don't know but are doing their best to find out."

— Terry Pratchett

Bibliography

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. (page 81, 89)

[2] V. B. Alex Kendall and R. Cipolla. Bayesian SegNet: Model Uncertainty in Deep Convolutional Encoder-Decoder Architectures for Scene Understanding. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2017. (page 108)

[3] J. Alzubi. Optimal Classifier Ensemble Design Based on Cooperative Game Theory. *Research Journal of Applied Sciences, Engineering and Technology*, 11:1336–1346, 12 2015. (page 16)

[4] T. An and M. Kim. A New Diverse AdaBoost Classifier. In *International Conference on Artificial Intelligence and Computational Intelligence (AICI)*, 2010. (page 16)

[5] Y. Bai, F. Gao, Y. Lou, S. Wang, T. Huang, and L.-Y. Duan. Incorporating Intra-Class Variance to Fine-Grained Visual Recognition. *arXiv*, abs/1703.00196, 2017. (page 103)

[6] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio. Theano: New Features and Speed Improvements. In *Proc. NIPS Deep Learning Workshop*, 2012. (page 60)

[7] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding (CVIU)*, 110(3):346–359, 2008. (page 3)

[8] A. Bellet, A. Habrard, and M. Sebban. A survey on metric learning for feature vectors and structured data. *arXiv*, abs/1306.6709, 2013. (page 32)

[9] W. H. Beluch, T. Genewein, A. Nürnberger, and J. M. Köhler. The power of ensembles for active learning in image classification. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. (page 19, 21, 108)

[10] R. Benenson, M. Mathias, T. Tuytelaars, and L. Van Gool. Seeking the Strongest Rigid Detector. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. (page 41)

[11] A. Beygelzimer, E. Hazan, S. Kale, and H. Luo. Online Gradient Boosting. In *Advances of Neural Information Processing (NIPS)*, 2015. (page 17, 72, 74)

[12] A. Beygelzimer, S. Kale, and H. Luo. Optimal and Adaptive Algorithms for Online Boosting. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2015. (page 17, 72)

[13] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, Berlin, Heidelberg, 2006. (page 3)

[14] L. Breiman. Bagging Predictors. *Machine Learning (ML)*, 24(2):123–140, 1996. (page 14)

[15] L. Breiman. Random Forests. *Machine Learning (ML)*, 45(1):5–32, 2001. (page 5, 14, 20, 22)

[16] A. Brown, W. Xie, V. Kalogeiton, and A. Zisserman. Smooth-AP: Smoothing the Path Towards Large-Scale Image Retrieval. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. (page 37)

[17] G. Brown, J. L. Wyatt, and P. Tiňo. Managing diversity in regression ensembles. *Journal of Machine Learning Research (JMLR)*, 6(9):1621–1650, 2005. (page 12)

[18] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language Models are Few-Shot Learners. *arXiv*, abs/2005.14165, 2020. (page 1, 3)

[19] S. R. Bulò, L. Porzi, and P. Kontschieder. Dropout Distillation. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2016. (page 27)

[20] X. Burgos-Artizzu, P. Perona, and P. Dollár. Robust face landmark estimation under occlusion. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2013. (page 53)

[21] Z. Cai and N. Vasconcelos. Cascade R-CNN: Delving into High Quality Object Detection. In *CVPR*, 2018. (page 23)

[22] F. Cakir, K. He, X. Xia, B. Kulis, and S. Sclaroff. Deep Metric Learning to Rank. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. (page 37)

[23] W. Chan, N. Jaitly, Q. Le, and O. Vinyals. Listen, Attend and Spell: A Neural Network for Large Vocabulary Conversational Speech Recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016. (page 1, 3)

[24] R. Chatpatanasiri, T. Korsrilabutr, P. Tangchanachaianan, and B. Kijsirikul. A new kernelization framework for mahalanobis distance learning algorithms. *Neurocomputing*, 73(10-12):1570–1579, 2010. (page 32)

[25] B. Chen, W. Deng, and H. Shen. Virtual Class Enhanced Discriminative Embedding Learning. In *Advances of Neural Information Processing (NIPS)*, 2018. (page 38)

[26] D. Chen, S. Ren, Y. Wei, X. Cao, and J. Sun. Joint Cascade Face Detection and Alignment. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014. (page 30, 60)

[27] G. Chen, T. Zhang, J. Lu, and J. Zhou. Deep Meta Metric Learning. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. (page 37)

[28] S.-T. Chen, H.-T. Lin, and C.-J. Lu. An Online Boosting Algorithm with Theoretical Justifications. *Proceedings of the International Conference on Machine Learning (ICML)*, 2012. (page 32, 72)

[29] S. Chen, C. Gong, J. Yang, X. Li, Y. Wei, and J. Li. Adversarial Metric Learning. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, 2018. (page 38)

[30] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-view 3D Object Detection Network for Autonomous Driving. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. (page 22)

[31] C. Chi, S. Zhang, J. Xing, Z. Lei, S. Z. Li, and X. Zou. Selective Refinement Network for High Performance Face Detection. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, 2019. (page 31, 32)

[32] S. Chopra, R. Hadsell, and Y. LeCun. Learning a Similarity Metric Discriminatively, with Application to Face Verification. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005. (page 33, 34, 69, 71)

[33] X. Chu, Y. Lin, Y. Wang, X. Wang, H. Yu, X. Gao, and Q. Tong. Distance Metric Learning with Joint Representation Diversification. In *International Conference on Machine Learning (ICML)*, 2020. (page 36)

[34] M. Cogswell, F. Ahmed, R. Girshick, L. Zitnick, and D. Batra. Reducing Overfitting in Deep Networks by Decorrelating Representations. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016. (page 82)

[35] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu. Boosting for Transfer Learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 193–200. ACM, 2007. (page 16)

[36] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005. (page 3, 30)

[37] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon. Information-theoretic metric learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2007. (page 32)

[38] J. Deng, J. Guo, E. Ververas, I. Kotsia, and S. Zafeiriou. RetinaFace: Single-Shot Multi-Level Face Localisation in the Wild. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. (page 31, 32)

[39] J. Deng, J. Guo, N. Xue, and S. Zafeiriou. Arcface: Additive Angular Margin Loss for Deep Face Recognition. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. (page 36, 69)

[40] T. DeVries and G. W. Taylor. Improved Regularization of Convolutional Neural Networks with CutOut. *arXiv*, abs/1708.04552, 2017. (page 27)

[41] P. Dollár, R. Appel, S. Belongie, and P. Perona. Fast Feature Pyramids for Object Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 36(8):1532–1545, 2014. (page xviii, 3, 11, 15, 41, 43)

[42] Y. Duan, W. Zheng, X. Lin, J. Lu, and J. Zhou. Deep Adversarial Metric Learning. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. (page 38)

[43] C. Dubout and F. Fleuret. Exact Acceleration of Linear Object Detectors. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2012. (page 60)

[44] I. Elezi, S. Vascon, A. Torcinovich, M. Pelillo, and L. Leal-Taixe. The Group Loss for Deep Metric Learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. (page 37)

[45] C. Ertler, H. Possegger, M. Opitz, and H. Bischof. An Intent-Based Automated Traffic Light for Pedestrians. In *Proceedings of the IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2018. (page)

[46] C. Ertler, H. Possegger, M. Opitz, and H. Bischof. Pedestrian Detection in RGB-D Images from an Elevated Viewpoint. In *Proceedings of the Computer Vision Winter Workshop (CVWW)*, 2017. (page)

[47] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes Challenge: A Retrospective. *International Journal of Computer Vision (IJCV)*, 111(1):98–136, 2015. (page 31, 50)

[48] S. S. Farfade, M. Saberian, and L.-J. Li. Multi-view Face Detection Using Deep Convolutional Neural Networks. In *Proceedings of the International Conference on Multimedia Retrieval (ICMR)*, 2015. (page 41, 61, 62)

[49] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object Detection with Discriminatively Trained Part-Based Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 32(9):1627–1645, 2010. (page 11, 22, 30, 41)

[50] Y. Freund and R. E. Schapire. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, 55(1):119 – 139, 1997. (page 5, 14, 15, 20)

[51] J. H. Friedman. Greedy Function Approximation: a Gradient Boosting Machine. *Annals of Statistics (AOS)*, 29(5):1189–1232, 2001. (page 16, 17)

[52] Y. Gal and Z. Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2016. (page 19, 27, 28)

[53] Y. Gal, J. Hron, and A. Kendall. Concrete Dropout. In *Advances of Neural Information Processing (NIPS)*, 2017. (page 28)

[54] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-Adversarial Training of Neural Networks. *Journal of Machine Learning Research (JMLR)*, 17(59):1–35, 2016. (page 79, 95)

[55] C. Garcia and M. Delakis. Convolutional Face Finder: A Neural Architecture for Fast and Robust Face Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 26(11):1408–1423, 2004. (page 31)

[56] T. Garipov, P. Izmailov, D. Podoprikhin, D. P. Vetrov, and A. G. Wilson. Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs. In *Advances of Neural Information Processing (NIPS)*, 2018. (page 24)

[57] X. Gastaldi. Shake-Shake Regularization. In *International Conference on Learning Representation Workshops (ICLRW)*, 2017. (page 27, 28)

[58] W. Ge. Deep Metric Learning with Hierarchical Triplet Loss. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. (page 34, 98, 99, 100, 101, 106)

[59] G. Ghiasi and C. C. Fowlkes. Occlusion Coherence: Localizing Occluded Faces with a Hierarchical Deformable Part Model. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. (page 22, 30, 60)

[60] G. Ghiasi, T.-Y. Lin, and Q. V. Le. DropBlock: A Regularization Method for Convolutional Networks. In *Advances of Neural Information Processing (NIPS)*, 2018. (page 27)

[61] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. (page 30)

[62] R. Girshick, F. Iandola, T. Darrell, and J. Malik. Deformable Part Models are Convolutional Neural Networks. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. (page 60)

[63] A. Globerson and S. T. Roweis. Metric Learning by Collapsing Classes. In *Advances of Neural Information Processing (NIPS)*, 2006. (page 32)

[64] X. Glorot and Y. Bengio. Understanding the Difficulty of Training Deep FeedForward Neural Networks. In *Proceedings of the International Conference of Artificial Intelligence and Statistics (AISTATS)*, 2010. (page 93)

[65] J. Goldberger, G. E. Hinton, S. T. Roweis, and R. R. Salakhutdinov. Neighbourhood Components Analysis. In *Advances of Neural Information Processing (NIPS)*, 2005. (page 32)

[66] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Nets. In *Advances of Neural Information Processing (NIPS)*, 2014. (page 79)

[67] A. Grabner, G. Poier, M. Opitz, S. Schulter, and P. Roth. Loss-Specific Training of Random Forests for Super-Resolution. In *Proceedings of the Computer Vision Winter Workshop (CVWW)*, 2017. (page)

[68] H. Grabner and H. Bischof. On-line Boosting and Vision. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006. (page 15, 16)

[69] G. Gu and B. Ko. Symmetrical Synthesis for Deep Metric Learning. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, 2020. (page 38)

[70] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik. Learning Rich Features from RGB-D Images for Object Detection and Segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014. (page 22)

[71] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality Reduction by Learning an Invariant Mapping. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006. (page 34, 69, 71)

[72] Z. Hao, Y. Liu, H. Qin, J. Yan, X. Li, and X. Hu. Scale-Aware Face Detection. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. (page 31)

[73] B. Harwood, V. Kumar B G, G. Carneiro, I. Reid, and T. Drummond. Smart Mining for Deep Metric Learning. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. (page 38, 98, 99, 102)

[74] T. Hastie, S. Rosset, J. Zhu, and H. Zou. Multi-Class AdaBoost. *Statistics and its Interface (SII)*, 2(3):349–360, 2009. (page 15)

[75] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. (page 1, 3, 7, 21, 27, 65, 82, 86, 97, 102, 106)

[76] D. P. Helmbold and P. M. Long. Surprising Properties of Dropout in Deep Networks. *Journal of Machine Learning Research (JMLR)*, 18(1):7284–7311, 2017. (page 28)

[77] A. Hermans, L. Beyer, and B. Leibe. In Defense of the Triplet Loss for Person Re-Identification. *arXiv*, abs/1703.07737, 2017. (page 34, 38)

[78] G. Hinton, O. Vinyals, and J. Dean. Distilling the Knowledge in a Neural Network. *arXiv:1503.02531*, 2015. (page 25, 27)

[79] M. Hirzer, P. M. Roth, M. Köstinger, and H. Bischof. Relaxed Pairwise Learned Metric for Person Re-Identification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2012. (page 32)

[80] J. Hosang, M. Omran, R. Benenson, and B. Schiele. Taking a Deeper Look at Pedestrians. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. (page 41)

[81] J. Hron, D. G. Matthews, Z. Ghahramani, et al. Variational Bayesian Dropout: Pitfalls and Fixes. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018. (page 28)

[82] P. Hu and D. Ramanan. Finding Tiny Faces. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. (page 31)

[83] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger. Snapshot Ensembles: Train 1, get M for free. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017. (page 24)

[84] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely Connected Convolutional Networks. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. (page 3)

[85] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger. Deep Networks with Stochastic Depth. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016. (page 27)

[86] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007. (page 32)

[87] P.-Y. Huang, W.-T. Hsu, C.-Y. Chiu, T.-F. Wu, and M. Sun. Efficient Uncertainty Estimation for Semantic Segmentation in Videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. (page 108)

[88] G. B. Hughes and M. Chraibi. Calculating Ellipse Overlap Areas. *Computing and Visualizations in Science*, 15(5):291–301, 2012. (page 49)

[89] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2015. (page 28, 97)

[90] M. Iyengar, M. Opitz, and H. Bischof. Detecting Out-of-Distribution Traffic Signs. In *Proceedings of the Austrian Association for Pattern Recognition Workshop (AAPR)*, 2019. (page)

[91] P. Jacob, D. Picard, A. Histace, and E. Klein. Metric Learning With HORDE: High-Order Regularizer for Deep Embeddings. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019. (page 38, 39, 98, 99, 100, 101, 102, 103, 104)

[92] M. Jaderberg, A. Vedaldi, and A. Zisserman. Speeding up Convolutional Neural Networks with Low Rank Expansions. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2014. (page 60)

[93] V. Jain and E. Learned-Miller. FDDB: A Benchmark for Face Detection in Unconstrained Settings. Technical Report UM-CS-2010-009, University of Massachusetts, Amherst, 2010. (page xviii, 50, 51, 52, 53, 55, 57, 60)

[94] A. Jaiswal, A. R. Babu, M. Z. Zadeh, D. Banerjee, and F. Makedon. A Survey on Contrastive Self-supervised Learning. *arXiv*, abs/2011.00362, 2020. (page 36)

[95] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional Architecture for Fast Feature Embedding. *arXiv*, abs/1408.5093, 2014. (page 81, 89)

[96] H. Jiang and E. Learned-Miller. Face Detection with the Faster R-CNN. In *IEEE International Conference on Automatic Face & Gesture Recognition (FG)*. IEEE, 2017. (page 31)

[97] V. Kazemi and J. Sullivan. One Millisecond Face Alignment with an Ensemble of Regression Trees. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. (page 15)

[98] D. Kedem, S. Tyree, F. Sha, G. R. Lanckriet, and K. Q. Weinberger. Non-linear Metric Learning. In *Advances of Neural Information Processing (NIPS)*, 2012. (page 33)

[99] M. Kemertas, L. Pishdad, K. G. Derpanis, and A. Fazly. RankMI: A Mutual Information Maximizing Ranking Loss. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. (page 98, 99, 100, 102)

[100] D. H. Kim and B. C. Song. Virtual Sample-Based Deep Metric Learning using Discriminant Analysis. *Pattern Recognition (PR)*, 2021. (page 38)

[101] S. Kim, D. Kim, M. Cho, and S. Kwak. Proxy Anchor Loss for Deep Metric Learning. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. (page 36, 98, 99, 100, 101)

[102] S. Kim, M. Seo, I. Laptev, M. Cho, and S. Kwak. Deep Metric Learning Beyond Binary Supervision. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. (page 34)

[103] W. Kim, B. Goyal, K. Chawla, J. Lee, and K. Kwon. Attention-based Ensemble for Deep Metric Learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. (page 22, 24, 38, 39, 98, 99, 100, 101, 102, 103, 104, 106, 107)

[104] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015. (page 89)

[105] D. P. Kingma, T. Salimans, and M. Welling. Variational Dropout and the Local Reparameterization Trick. In *Advances of Neural Information Processing (NIPS)*, 2015. (page 28)

[106] B. Ko and G. Gu. Embedding Expansion: Augmentation in Embedding Space for Deep Metric Learning. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. (page 38)

[107] M. Koestinger, M. Hirzer, P. Wohlhart, P. M. Roth, and H. Bischof. Large Scale Metric Learning from Equivalence Constraints. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. (page 32)

[108] M. Köstinger, P. Wohlhart, P. M. Roth, and H. Bischof. Annotated Facial Landmarks in the Wild: A Large-scale, Real-world Database for Facial Landmark Localization. In *IEEE International Workshop on Benchmarking Facial Image Analysis Technologies (BeFIT)*, 2011. (page 44, 50)

[109] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3D Object Representations for Fine-Grained Categorization. In *Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCVW)*, 2013. (page xxii, 88, 97, 99)

[110] G. Krispel, M. Opitz, G. Waltner, H. Possegger, and H. Bischof. FuseSeg: LiDAR Point Cloud Segmentation Fusing Multi-Modal Data. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2020. (page)

[111] A. Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Technical report, University of Toronto, 2009. (page 67, 81)

[112] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet Classification with Deep Convolutional Neural Networks. In *Advances of Neural Information Processing (NIPS)*, 2012. (page 1, 3, 30, 41)

[113] V. Kumar, A. M. Namboodiri, and C. V. Jawahar. Visual Phrases for Exemplar Face Detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015. (page 30, 60)

[114] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In *Advances of Neural Information Processing (NIPS)*, 2017. (page 19)

[115] M. D. Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. G. Slabaugh, and T. Tuytelaars. Continual learning: A comparative study on how to defy forgetting in classification tasks. *arXiv*, abs/1909.08383, 2019. (page 108)

[116] M. T. Law, N. Thome, and M. Cord. Quadruplet-wise Image Similarity Learning. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2013. (page 34, 69)

[117] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply-Supervised Nets. In *Proceedings of the International Conference of Artificial Intelligence and Statistics (AISTATS)*, 2015. (page 80)

[118] C.-Y. Lee, S. Xie, P. W. Gallagher, Z. Zhang, and Z. Tu. Deeply-Supervised Nets. In *Proceedings of the International Conference of Artificial Intelligence and Statistics (AISTATS)*, 2015. (page 39, 103)

[119] S. Lee, S. Purushwalkam, M. Cogswell, D. Crandall, and D. Batra. Why M Heads are Better than One: Training a Diverse Ensemble of Deep Networks. *arXiv*, abs/1511.06314, 2015. (page xxi, 20, 21, 24, 66, 82, 83, 84, 86)

[120] C. Leistner, A. Saffari, P. M. Roth, and H. Bischof. On Robustness of On-line Boosting - a Competitive Study. In *Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCVW)*, 2009. (page 15, 17, 18, 72)

[121] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research (IJRR)*, 37(4-5):421–436, 2018. (page 1, 3)

[122] A. H. Li and J. Bradic. Boosting in the Presence of Outliers: Adaptive Classification with Non-Convex Loss Functions. *Journal of the American Statistical Association (ASA)*, 113(522):660–674, 2018. (page 15)

[123] F. Li, N. Neverova, C. Wolf, and G. Taylor. Modout: Learning Multi-Modal Architectures by Stochastic Regularization. In *IEEE International Conference on Automatic Face & Gesture Recognition (FG)*, 2017. (page 27)

[124] H. Li, J. Y.-H. Ng, and P. Natsev. EnsembleNet: End-to-End Optimization of Multi-headed Models. *arXiv*, abs/1905.09979, 2019. (page xxii, 24, 85, 86)

[125] H. Li, G. Hua, Z. Lin, J. Brandt, and J. Yang. Probabilistic Elastic Part Model for Unsupervised Face Detector Adaptation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2013. (page 30)

[126] H. Li, Z. Lin, J. Brandt, X. Shen, and G. Hua. Efficient Boosted Exemplar-Based Face Detection. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. (page 30, 47, 52, 56, 60)

[127] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua. A Convolutional Neural Network Cascade for Face Detection. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. (page 31, 41, 60, 62)

[128] J. Li, Y. Wang, C. Wang, Y. Tai, J. Qian, J. Yang, C. Wang, J. Li, and F. Huang. DSFD: Dual Shot Face Detector. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. (page 31, 32)

[129] J. Li and Y. Zhang. Learning Surf Cascade for Fast and Accurate Object Detection. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. (page 30)

[130] Z. Li, X. Tang, J. Han, J. Liu, and R. He. PyramidBox++: High Performance Detector for Finding Tiny Face. *arXiv:1904.00386*, 2019. (page 31, 32)

[131] S. Liang, Y. Li, and R. Srikant. Enhancing The Reliability of Out-of-distribution Image Detection in Neural Networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018. (page 108)

[132] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature Pyramid Networks for Object Detection. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. (page 3, 22, 31, 106)

[133] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal Loss For Dense Object Detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. (page 22, 31)

[134] 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 *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014. (page xvii, 3, 4, 19, 31)

[135] X. Lin, Y. Duan, Q. Dong, J. Lu, and J. Zhou. Deep Variational Metric Learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. (page 100)

[136] H. Liu, Y. Tian, Y. Wang, L. Pang, and T. Huang. Deep Relative Distance Learning: Tell the Difference Between Similar Vehicles. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. (page xxii, 97, 103)

[137] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single Shot Multibox Detector. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016. (page 31, 41)

[138] Y. Liu, X. Tang, J. Han, J. Liu, D. Rui, and X. Wu. HAMBox: Delving Into Mining High-Quality Anchors on Face Detection. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. (page 31, 32)

[139] Y. Liu and X. Yao. Ensemble Learning via Negative Correlation. *Neural Networks (NN)*, 12(10):1399–1404, 1999. (page 14, 16, 18, 23, 82)

[140] Y. Liu, W. Dong, L. Zhang, D. Gong, and Q. Shi. Variational Bayesian Dropout. *arXiv*, abs/1811.07533, 2018. (page 28)

[141] Z. Liu, P. Luo, S. Qiu, X. Wang, and X. Tang. DeepFashion: Powering Robust Clothes Recognition and Retrieval with Rich Annotations. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. (page xxii, 88, 97, 101)

[142] J. Long, E. Shelhamer, and T. Darrell. Fully Convolutional Networks for Semantic Segmentation. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. (page 22)

[143] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110, 2004. (page 3)

[144] J. Lu, C. Xu, W. Zhang, L.-Y. Duan, and T. Mei. Sampling Wisely: Deep Image Embedding by Top-k Precision Optimization. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. (page 37)

[145] X. Ma, Y. Gao, Z. Hu, Y. Yu, Y. Deng, and E. Hovy. Dropout with Expectation-Linear Regularization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017. (page 27)

[146] P. K. Mallapragada, R. Jin, A. K. Jain, and Y. Liu. SemiBoost: Boosting for Semi-Supervised Learning. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 31(11):2000–2014, 2008. (page 15)

[147] H. Masnadi-Shirazi and N. Vasconcelos. On the Design of Loss Functions for Classification: Theory, Robustness to Outliers, and SavageBoost. In *Advances of Neural Information Processing (NIPS)*, 2008. (page 15)

[148] M. Mathias, R. Benenson, M. Pedersoli, and L. Van Gool. Face Detection without Bells and Whistles. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014. (page 3, 30, 41, 44, 50, 57, 60, 61, 62)

[149] M. Mathias, R. Benenson, R. Timofte, and L. Van Gool. Handling Occlusions with Franken-Classifiers. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2013. (page 3, 22)

[150] T. Milbich, K. Roth, H. Bharadhwaj, S. Sinha, Y. Bengio, B. Ommer, and J. P. Cohen. DiVA: Diverse Visual Feature Aggregation for Deep Metric Learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. (page 23, 24, 38, 39, 98, 99, 100, 102, 103, 104, 106, 107)

[151] D. D. Mohan, N. Sankaran, D. Fedorishin, S. Setlur, and V. Govindaraju. Moving in the Right Direction: A Regularization for Deep Metric Learning. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. (page 98, 99, 101)

[152] W. Mou, Y. Zhou, J. Gao, and L. Wang. Dropout Training, Data-dependent Regularization, and Generalization Bounds. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018. (page 28, 29)

[153] Y. Movshovitz-Attias, A. Toshev, T. K. Leung, S. Ioffe, and S. Singh. No Fuss Distance Metric Learning Using Proxies. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. (page 36, 98, 99, 100, 102)

[154] M. Muja and D. G. Lowe. Scalable Nearest Neighbor Algorithms for High Dimensional Data. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 36(11):2227–2240, 2014. (page 75)

[155] M. Najibi, P. Samangouei, R. Chellappa, and L. S. Davis. SSH: Single Stage Headless Face Detector. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. (page 31, 32)

[156] M. Najibi, B. Singh, and L. S. Davis. Fa-rpn: Floating region proposals for face detection. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. (page 31)

[157] W. Nam, P. Dollár, and J. H. Han. Local Decorrelation for Improved Pedestrian Detection. In *Advances of Neural Information Processing (NIPS)*, 2014. (page 30)

[158] B. Neal, S. Mittal, A. Baratin, V. Tantia, M. Scicluna, S. Lacoste-Julien, and I. Mitliagkas. A Modern Take on the Bias-Variance Tradeoff in Neural Networks. *arXiv*, abs/1810.08591, 2018. (page 21)

[159] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading Digits in Natural Images with Unsupervised Feature Learning. In *Advances of Neural Information Processing (NIPS) Workshops*, 2011. (page 67, 81)

[160] N. Neverova, C. Wolf, G. Taylor, and F. Nebout. Moddrop: Adaptive multi-modal gesture recognition. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 38(8):1692–1706, 2015. (page 27)

[161] Nitish Srivastava and Geoffrey Hinton and Alex Krizhevsky and Ilya Sutskever and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research (JMLR)*, 15:1929–1958, 2014, `http://jmlr.org/papers/v15/srivastava14a.html`. (page 44, 66, 82, 86)

[162] H. Oh Song, S. Jegelka, V. Rathod, and K. Murphy. Deep Metric Learning via Facility Location. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. (page 37)

[163] H. Oh Song, Y. Xiang, S. Jegelka, and S. Savarese. Deep Metric Learning via Lifted Structured Feature Embedding. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. (page xxii, 32, 34, 70, 88, 89, 97, 98, 99, 100, 102)

[164] D. Opitz and R. Maclin. Popular Ensemble Methods: An Empirical Study. *Journal of Artificial Intelligence Research*, 11:169–198, 1999. (page 12)

[165] M. Opitz, H. Possegger, and H. Bischof. Efficient Model Averaging for Deep Neural Networks. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, 2016. (page 19, 20, 23, 24, 31, 65, 66, 76, 83, 86)

[166] M. Opitz, G. Waltner, G. Poier, H. Possegger, and H. Bischof. Grid Loss: Detecting Occluded Faces. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016. (page 20, 22)

[167] M. Opitz, G. Waltner, H. Possegger, and H. Bischof. BIER: Boosting Independent Embeddings Robustly. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. (page xxii, 17, 20, 21, 23, 24, 38, 39, 65, 66, 69, 70, 75, 76, 83, 84, 85, 86, 90, 93, 94, 95, 98, 99, 100, 101, 102, 103)

[168] M. Opitz, G. Waltner, H. Possegger, and H. Bischof. Deep Metric Learning with BIER: Boosting Independent Embeddings Robustly. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2018. (page 17, 19, 20, 21, 23, 24, 38, 39, 65, 66, 69, 70, 75, 76, 83, 84, 85, 86, 93, 94, 98, 99, 100, 102)

[169] N. C. Oza. Online Bagging and Boosting. In *IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 2005. (page 16)

[170] W. Park, W. Kim, K. You, and M. Cho. Diversified Mutual Learning for Deep Metric Learning. In *Proceedings of the European Conference on Computer Vision Workshops (ECCVW)*, 2020. (page 106, 107)

[171] G. Poier, M. Opitz, D. Schinagl, and H. Bischof. MURAUER: Mapping Unlabeled Real Data for Label AUstERity. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019. (page)

[172] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Advances of Neural Information Processing (NIPS)*, 2017. (page 107)

[173] Q. Qian, L. Shang, B. Sun, J. Hu, H. Li, and R. Jin. SoftTriple Loss: Deep Metric Learning without Triplet Sampling. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019. (page 36, 98, 99, 100)

[174] H. Qin, J. Yan, X. Li, and X. Hu. Joint training of cascaded cnn for face detection. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. (page 31)

[175] C.-X. Ren, J.-Z. Li, P. Ge, and X.-L. Xu. Deep Metric Learning via Subtype Fuzzy Clustering. *Pattern Recognition (PR)*, 90:210–219, 2019. (page 34, 38)

[176] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards Real-time Object Detection with Region Proposal Networks. In *Advances of Neural Information Processing (NIPS)*, 2015. (page 3, 22, 31, 41)

[177] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese. Generalized Intersection over Union: A Metric and a Loss for Bounding Box Regression. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. (page 49)

[178] K. Ridgeway and M. C. Mozer. Learning Deep Disentangled Embeddings with the F-Statistic Loss. In *Advances of Neural Information Processing (NIPS)*, 2018. (page 37)

[179] O. Rippel, M. Paluri, P. Dollar, and L. Bourdev. Metric Learning with Adaptive Density Discrimination. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016. (page 36)

[180] M. Rolínek, V. Musil, A. Paulus, M. Vlastelica, C. Michaelis, and G. Martius. Optimizing Rank-based Metrics with Blackbox Differentiation. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. (page 37)

[181] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio. FitNets: Hints for Thin Deep Nets. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015. (page 25)

[182] K. Roth, B. Brattoli, and B. Ommer. MIC: Mining Interclass Characteristics for Improved Metric Learning. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. (page 23, 24, 38, 39, 98, 99, 100, 102, 103, 104, 106, 107)

[183] K. Roth, T. Milbich, and B. Ommer. PADS: Policy-Adapted Sampling for Visual Similarity Learning. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. (page 38, 98, 99, 100, 102)

[184] K. Roth, T. Milbich, S. Sinha, P. Gupta, B. Ommer, and J. P. Cohen. Revisiting Training Strategies and Generalization Performance in Deep Metric Learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2020. (page 38, 97, 102, 103)

[185] H. Rowley, S. Baluja, T. Kanade, et al. Neural Network-Based Face Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 20(1):23–38, 1998. (page 31)

[186] 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. *International Journal of Computer Vision (IJCV)*, 115(3):1–42, 2015. (page xvii, 3, 4, 19, 61, 89)

[187] R. Salakhutdinov and G. Hinton. Learning a Nonlinear Embedding by Preserving Class Neighbourhood Structure. In *Proceedings of the International Conference of Artificial Intelligence and Statistics (AISTATS)*, 2007. (page 33)

[188] A. Sanakoyeu, V. Tschernezki, U. Büchler, and B. Ommer. Divide and Conquer the Embedding Space for Metric Learning. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. (page 20, 21, 23, 39, 98, 99, 100, 102, 103, 104, 106, 107)

[189] A. Sanakoyeu, V. Tschernezki, U. Buchler, and B. Ommer. Divide and Conquer the Embedding Space for Metric Learning. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. (page 24, 38)

[190] A. M. Saxe, J. L. McClelland, and S. Ganguli. Exact Solutions to the Nonlinear Dynamics of Learning in Deep Linear Neural Networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014. (page 93)

[191] F. Schroff, D. Kalenichenko, and J. Philbin. FaceNet: A Unified Embedding for Face Recognition and Clustering. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. (page 34, 38, 69, 70)

[192] S. Schulter, C. Leistner, P. Wohlhart, P. M. Roth, and H. Bischof. Accurate Object Detection with Joint Classification-Regression Random Forests. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. (page 41)

[193] S. Schulter, P. Wohlhart, C. Leistner, A. Saffari, P. M. Roth, and H. Bischof. Alternating Decision Forests. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. (page 11, 17)

[194] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014, `http://openreview.net/document/d332e77d-459a-4af8-b3ed-55ba`. (page 43)

[195] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. LeCun. Pedestrian Detection with Unsupervised Multi-Stage Feature Learning. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. (page 41)

[196] X. Shen, Z. Lin, J. Brandt, and Y. Wu. Detecting and Aligning Faces by Image Retrieval. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. (page 30, 62)

[197] Z. Shen, Z. He, and X. Xue. MEAL: Multi-Model Ensemble via Adversarial Learning. In *AAAI*, 2019. (page 25)

[198] H. Shi, Y. Yang, X. Zhu, S. Liao, Z. Lei, W. Zheng, and S. Z. Li. Embedding Deep Metric for Person Re-identification: A Study Against Large Variations. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016. (page 32)

[199] Z. Shi, L. Zhang, Y. Liu, X. Cao, Y. Ye, M.-M. Cheng, and G. Zheng. Crowd Counting with Deep Negative Correlation Learning. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. (page 23)

[200] C. A. Shipp and L. I. Kuncheva. An Investigation into How ADABOOST Affects Classifier Diversity. In *Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU)*, 2002. (page 15)

[201] A. Shrivastava, A. Gupta, and R. Girshick. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. (page 32)

[202] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature*, 529(7587):484–489, 2016. (page 1, 3)

[203] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014. (page 65)

[204] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014. (page 59)

[205] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015. (page 44)

[206] K. Sohn. Improved Deep Metric Learning with Multi-Class n-Pair Loss Objective. In *Advances of Neural Information Processing (NIPS)*, 2016. (page 34, 98, 99, 100, 102)

[207] H. O. Song, S. Jegelka, V. Rathod, and K. Murphy. Deep Metric Learning via Facility Location. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. (page 98, 99, 100)

[208] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research (JMLR)*, 15:1929–1958, 2014. (page 9, 22, 23, 25, 26, 28)

[209] Y. Suh, B. Han, W. Kim, and K. M. Lee. Stochastic Class-Based Hard Example Mining for Deep Metric Learning. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. (page 38)

[210] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, S. Zhao, S. Cheng, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov. Scalability in Perception for Autonomous Driving: Waymo Open Dataset. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. (page xvii, 3, 4)

[211] Y. Sun, C. Cheng, Y. Zhang, C. Zhang, L. Zheng, Z. Wang, and Y. Wei. Circle Loss: A Unified Perspective of Pair Similarity Optimization. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. (page 98, 99, 100)

[212] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going Deeper with Convolutions. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. (page 39, 47, 52, 56, 89, 97, 106)

[213] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going Deeper with Convolutions. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. (page 80)

[214] M. Tan and Q. Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2019. (page 3)

[215] X. Tang, D. K. Du, Z. He, and J. Liu. Pyramidbox: A Context-Assisted Single Shot Face Detector. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. (page 31, 32)

[216] R. Tao, E. Gavves, and A. W. Smeulders. Siamese Instance Search for Tracking. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. (page 32)

[217] E. W. Teh, T. DeVries, and G. W. Taylor. ProxyNCA++: Revisiting and Revitalizing Proxy Neighborhood Component Analysis. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. (page 36, 98, 99, 100, 101, 102)

[218] Y. Tian, P. Luo, X. Wang, and X. Tang. Deep Learning Strong Parts for Pedestrian Detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015. (page 22)

[219] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler. Efficient Object Localization using Convolutional Networks. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. (page 27)

[220] L. Torresani and K.-c. Lee. Large Margin Component Analysis. In *Advances of Neural Information Processing (NIPS)*, 2007. (page 32)

[221] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. JÃ©gou. Training Data-Efficient Image Transformers & Distillation through Attention. *arXiv*, abs/2012.12877, 2020. (page 107)

[222] N. Ueda and R. Nakano. Generalization Error of Ensemble Estimators. In *Proceedings of the International Conference on Neural Networks (ICNN)*, 1996. (page 12, 13)

[223] E. Ustinova and V. Lempitsky. Learning Deep Embeddings with Histogram Loss. In *Advances of Neural Information Processing (NIPS)*, 2016. (page 37, 70, 71, 89, 98, 100)

[224] R. Vaillant, C. Monrocq, and Y. LeCun. Original Approach for the Localisation of Objects in Images. *IEEE Proceedings of Vision, Image and Signal Processing*, 141(4):245–250, 1994. (page 31)

[225] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is All you Need. In *Advances of Neural Information Processing (NIPS)*, 2017. (page 62, 107)

[226] V. Vielzeuf, A. Lechervy, S. Pateux, and F. Jurie. CentralNet: a Multilayer Approach for Multimodal Fusion. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. (page 22)

[227] P. Viola and M. J. Jones. Robust Real-Time Face Detection. *International Journal of Computer Vision (IJCV)*, 57(2):137–154, 2004. (page 3, 11, 15, 30, 41, 73)

[228] S. Wager, S. Wang, and P. S. Liang. Dropout Training as Adaptive Regularization. In *Advances of Neural Information Processing (NIPS)*, 2013. (page 28)

[229] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. (page xix, xxii, 88, 91, 92, 93, 94, 95, 96, 97, 98)

[230] G. Waltner, M. Opitz, G. Krispel, H. Possegger, and H. Bischof. Semi-supervised Detector Training with Prototypes for Vehicle Detection. In *Proceedings of the IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019. (page)

[231] G. Waltner, M. Opitz, H. Possegger, and H. Bischof. HiBsteR: Hierarchical Boosted Deep Metric Learning for Image Retrieval. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019. (page)

[232] G. Waltner, M. Maurer, T. Holzmann, P. Ruprecht, M. Opitz, H. Possegger, F. Fraundorfer, and H. Bischof. Deep 2.5 D Vehicle Classification with Sparse SfM Depth Prior for Automated Toll Systems. In *Proceedings of the IEEE Intelligent Transportation Systems Conference (ITSC)*, 2018. (page)

[233] G. Waltner, M. Opitz, and H. Bischof. Bacon: Building a Classifier from only N Samples. In *Proceedings of the Computer Vision Winter Workshop (CVWW)*, 2016. (page)

[234] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus. Regularization of Neural Networks using DropConnect. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2013. (page 27)

[235] L. Wan, K. Tang, and R. Wang. Gradient Boosting-Based Negative Correlation Learning. In H. Yin, K. Tang, Y. Gao, F. Klawonn, M. Lee, T. Weise, B. Li, and X. Yao, editors, *Intelligent Data Engineering and Automated Learning (IDEAL)*, 2013. (page 23)

[236] H. Wang, Z. Li, X. Ji, and Y. Wang. Face R-CNN. *arXiv:1706.01061*, 2017. (page 31)

[237] J. Wang, F. Zhou, S. Wen, X. Liu, and Y. Lin. Deep Metric Learning with Angular Loss. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. (page 34)

[238] J. Wang, F. Zhou, S. Wen, X. Liu, and Y. Lin. Deep Metric Learning With Angular Loss. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. (page 98, 99, 100, 102)

[239] S. Wang, H. Chen, and X. Yao. Negative Correlation Learning for Classification Ensembles. In *International Joint Conference on Neural Networks (IJCNN)*, 2010. (page 16, 23)

[240] S. Wang and C. Manning. Fast Dropout Training. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2013. (page 27, 28)

[241] X. Wang, S. Zhang, Z. Lei, S. Liu, X. Guo, and S. Z. Li. Ensemble Soft-margin Softmax Loss for Image Classification. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, 2018. (page 23, 86)

[242] X. Wang, Y. Hua, E. Kodirov, G. Hu, R. Garnier, and N. M. Robertson. Ranked List Loss for Deep Metric Learning. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. (page 37, 98, 99, 100, 102, 103, 104, 106)

[243] X. Wang, X. Han, W. Huang, D. Dong, and M. R. Scott. Multi-Similarity Loss with General Pair Weighting for Deep Metric Learning. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. (page 34, 98, 99, 100, 101)

[244] A. Wasay, Y. Liao, and S. Idreos. Rapid Training of Very Large Ensembles of Diverse Neural Networks. *arXiv*, abs/1809.04270, 2018. (page 20, 24)

[245] K. Q. Weinberger and L. K. Saul. Distance Metric Learning for Large Margin Nearest Neighbor Classification. *Journal of Machine Learning Research (JMLR)*, 10(2):207–244, 2009. (page 32, 34, 69)

[246] C.-Y. Wu, R. Manmatha, A. J. Smola, and P. Krähenbühl. Sampling Matters in Deep Embedding Learning. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. (page 34, 38, 102, 103)

[247] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated Residual Transformations for Deep Neural Networks. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. (page 27)

[248] H. Xuan, R. Souvenir, and R. Pless. Deep Randomized Ensembles for Metric Learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. (page 21, 38, 39, 98, 99, 102, 103, 104)

[249] Y. Yamada, M. Iwamura, T. Akiba, and K. Kise. ShakeDrop Regularization for Deep Residual Learning. In *International Conference on Learning Representation Workshops (ICLRW)*, 2018. (page 27)

[250] J. Yan, Z. Lei, L. Wen, and S. Li. The Fastest Deformable Part Model for Object Detection. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. (page 22, 30, 60)

[251] J. Yan, X. Zhang, Z. Lei, and S. Z. Li. Face Detection by Structural Models. *Image and Vision Computing (IVC)*, 32(10):790 – 799, 2014, `http://www.sciencedirect.com/science/article/pii/S0262885613001765`. (page xviii, 50, 51, 57, 61, 62)

[252] B. Yang, J. Yan, Z. Lei, and S. Z. Li. Aggregate Channel Features for Multi-View Face Detection. In *IEEE International Joint Conference on Biometrics (IJCB)*, 2014. (page 30, 60)

[253] B. Yang, J. Yan, Z. Lei, and S. Z. Li. Convolutional Channel Features. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015. (page 30, 41, 60)

[254] H. Yang, C. Yuan, J. Xing, and W. Hu. Diversity Encouraging Ensemble of Convolutional Networks for High Performance Action Recognition. In *Proceedings of the International Conference on Image Processing (ICIP)*, 2017. (page 24)

[255] S. Yang, P. Luo, C.-C. Loy, and X. Tang. From Facial Parts Responses to Face Detection: A Deep Learning Approach. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015. (page 22, 31, 60, 61, 62)

[256] S. Yang, P. Luo, C. C. Loy, and X. Tang. WIDER FACE: A Face Detection Benchmark. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. (page 31)

[257] S. Yang, Y. Xiong, C. C. Loy, and X. Tang. Face Detection through Scale-Friendly Deep Convolutional Networks. *arXiv:1706.02863*, 2017. (page 31)

[258] D. Yashunin, T. Baydasov, and R. Vlasov. MaskFace: multi-task face and landmark detector. *arXiv*, abs/2005.09412, 2020. (page 31, 32)

[259] D. Yi, Z. Lei, S. Liao, and S. Z. Li. Deep Metric Learning for Person Re-Identification. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, 2014. (page 34)

[260] B. Yu and D. Tao. Deep Metric Learning with Tuplet Margin Loss. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. (page 34, 100, 102)

[261] T. Yuan, W. Deng, J. Tang, Y. Tang, and B. Chen. Signal-to-Noise Ratio: A Robust Distance Metric for Deep Metric Learning. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. (page 38)

[262] Y. Yuan, K. Yang, and C. Zhang. Hard-Aware Deeply Cascaded Embedding. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. (page 38, 98, 99, 100, 101, 102, 103, 104)

[263] S. Zagoruyko and N. Komodakis. Wide Residual Networks. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2016. (page 82, 86)

[264] A. Zhai, H.-Y. Wu, and U. San Francisco. Classification is a Strong Baseline for Deep Metric Learning. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2019. (page 36, 69, 98, 99, 100, 102)

[265] J. Zhang, X. Wu, S. C. Hoi, and J. Zhu. Feature Agglomeration Networks for Single Stage Face Detection. *Neurocomputing*, 380:180–189, 2020. (page 31)

[266] J. Zhang, S. Shan, M. Kan, and X. Chen. Coarse-to-Fine Auto-Encoder Networks (CFAN) for Real-Time Face Alignment. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014. (page 23)

[267] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao. Joint Face Detection and Alignment using Multitask Cascaded Convolutional Networks. *IEEE Signal Processing Letters (SPL)*, 23(10):1499–1503, 2016. (page 31)

[268] K. Zhang, Z. Zhang, H. Wang, Z. Li, Y. Qiao, and W. Liu. Detecting faces using inside cascaded contextual cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. (page 31)

[269] S. Zhang, R. Benenson, and B. Schiele. Filtered Channel Features for Pedestrian Detection. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. (page 30, 41)

[270] S. Zhang, J. Yang, and B. Schiele. Occluded Pedestrian Detection through Guided Attention in CNNs. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. (page 22)

[271] S. Zhang, L. Wen, X. Bian, Z. Lei, and S. Z. Li. Occlusion-Aware R-CNN: Detecting Pedestrians in a Crowd. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. (page 22)

[272] S. Zhang, R. Zhu, X. Wang, H. Shi, T. Fu, S. Wang, T. Mei, and S. Z. Li. Improved Selective Refinement Network for Face Detection. *arXiv*, abs/1901.06651, 2019. (page 31, 32)

[273] S. Zhang, X. Zhu, Z. Lei, H. Shi, X. Wang, and S. Z. Li. FaceBoxes: A CPU Real-Time Face Detector with High Accuracy. In *IEEE International Joint Conference on Biometrics (IJCB)*, pages 1–9, 2017. (page 31)

[274] S. Zhang, X. Zhu, Z. Lei, H. Shi, X. Wang, and S. Z. Li. S3FD: Single Shot Scale-Invariant Face Detector. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. (page 31, 32)

[275] X. Zhang, J. Zou, K. He, and J. Sun. Accelerating Very Deep Convolutional Networks for Classification and Detection. *arXiv*, abs/1505.06798, 2015. (page 60)

[276] X. Zhang, F. X. Yu, S. Kumar, and S.-F. Chang. Learning Spread-Out Local Feature Descriptors. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. (page 38)

[277] Y. Zhang, X. Xu, and X. Liu. Robust and High Performance Face Detector. *arXiv:1901.02350*, 2019. (page 31, 32)

[278] Y. Zhao, Z. Jin, G.-j. Qi, H. Lu, and X.-s. Hua. An Adversarial Approach to Hard Triplet Generation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. (page 38, 98, 99)

[279] W. S. Zheng, S. Gong, and T. Xiang. Reidentification by Relative Distance Comparison. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 35(3):653–668, 2013. (page 34, 69)

[280] W. Zheng, Z. Chen, J. Lu, and J. Zhou. Hardness-Aware Deep Metric Learning. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. (page 38, 98, 99, 100)

[281] T. Zhou, S. Wang, and J. A. Bilmes. Diverse Ensemble Evolution: Curriculum Data-Model Marriage. In *Advances of Neural Information Processing (NIPS)*, 2018. (page 20)

[282] C. Zhu, R. Tao, K. Luu, and M. Savvides. Seeing Small Faces from Robust Anchor's Perspective. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. (page 31, 32)

[283] X. Zhu and D. Ramanan. Face Detection, Pose Estimation and Landmark Estimation in the Wild. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. (page xviii, 22, 30, 50, 51, 57, 61, 62)

[284] X. Zhu, S. Gong, et al. Knowledge Distillation by On-the-Fly Native Ensemble. In *Advances of Neural Information Processing (NIPS)*, 2018. (page xxii, 11, 24, 25, 85, 86)

[285] Y. Zhu, M. Yang, C. Deng, and W. Liu. Fewer is More: A Deep Graph Metric Learning Perspective Using Fewer Proxies. In *Advances of Neural Information Processing (NIPS)*, 2020. (page 36, 37)

[286] B. Zoph and Q. V. Le. Neural Architecture Search with Reinforcement Learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017. (page 107)