Martin Zach

Structured Learning for Energy Based Models in Image Restoration

MASTER'S THESIS

to achieve the university degree of
Diplom-Ingenieur

Master's degree programme
Information and Computer Engineering

submitted to

Graz University of Technology

Supervisor

Prof. Dr. Thomas Pock
Institute for Computer Graphics and Vision

Dr. Erich Kobler
Institute for Computer Graphics and Vision

January 27, 2021

Abstract

Variational Methods have a very rich history of application for image restoration. This can be attributed to their profound mathematical underpinnings, interpretability and theoretical guarantees as well as their versatility of application. However, the rich theory and interpretability has in recent times been discarded in favor of better performance achieved by deep feed-forward convolutional neural networks. The success of these deep architectures can largely be attributed to an increase in model capacity and computational power, whilst theoretical advances have largely been neglected. This trend has been reversed somewhat very recently, with new work establishing connections between traditional variational methods and the feed-forward approaches, and transferring advances in one field to the other.

In this work we want to further unravel the connections between feed-forward architectures, variational methods, and the maximum-margin principle. We achieve this by casting the image restoration problem into the framework of energy-based variational models. Further, we consider a parametrized regularization term given by a multi-scale residual convolutional neural network. Motivated by the maximum-margin principle that is often employed in structured models, we propose a novel optimization scheme for learning the regularization term in a generative manner. We investigate the properties of the optimization scheme as well as the learned regularizer and show its broad applicability to a range of image restoration tasks.

Keywords. Image Restoration, Deep Learning, Variational Methods, Maximum Likelihood

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 master's thesis.

Date	Signature

Acknowledgments

First and foremost I want to thank my supervisor Prof. Pock, who was an exceptional guide in the journey of finishing this thesis. During this journey, we had lots of fruitful discussions where his profound understanding of the matter combined with his extraordinary ability to convey seemingly complicated theory in a simple manner helped me in gaining a strong foundation of knowledge in the fields of image processing, optimization, and others. He was ready to answer my many questions at any time, for which I am especially grateful.

I also want to express my gratitude to Erich Kobler, who was by my side along this journey and whose input was vital for defeating the devil in the details. I do not know if I would have won this battle alone. Further, I want to thank all other colleagues of mine at the Institute of Computer Graphics and Vision for making my time there not only interesting but also fun.

Finally, I want to use this opportunity to thank my family and especially my parents for enabling me to have such a carefree time of studying. Without their support over the many years I would not be anywhere near where I am today. There is no way for me to give back what you gave me, thank you!

Contents

List of Figures

List of Tables

1 Introduction

Contents

1.1 Image Restoration

Generally speaking, Image Restoration (IR) refers to the process of estimating a "clean" image from a "degraded" observation [49]. Historically, the considered degradations were mainly atmospheric blurring and photon noise, which was largely motivated by imaging systems used in astronomical applications [1]. Denoising and deconvolution are still extensively studied, along with demosaicking, Single-Image Super-Resolution (SISR), and image inpainting, which are also considered parts of the field [25, 64] today. We show a clean image along with some typical degraded instances in Fig. 1.1.

All of the above degradations are a direct consequence of the image acquisition process. For instance, blurring can be induced by motion of the camera with respect to (parts of) the scene during exposure. This can be remedied by decreasing the exposure time, however this would lead to less photons reaching the camera sensor and inevitably to higher noise levels. Demosaicking is an essential step in todays digital camera systems, where color information from the scene is recorded on a single sensor by means of a Color Filter Array (CFA). Typically, the Bayer *CFA* [3] is used. In this case, in a 2×2 patch of pixels, two

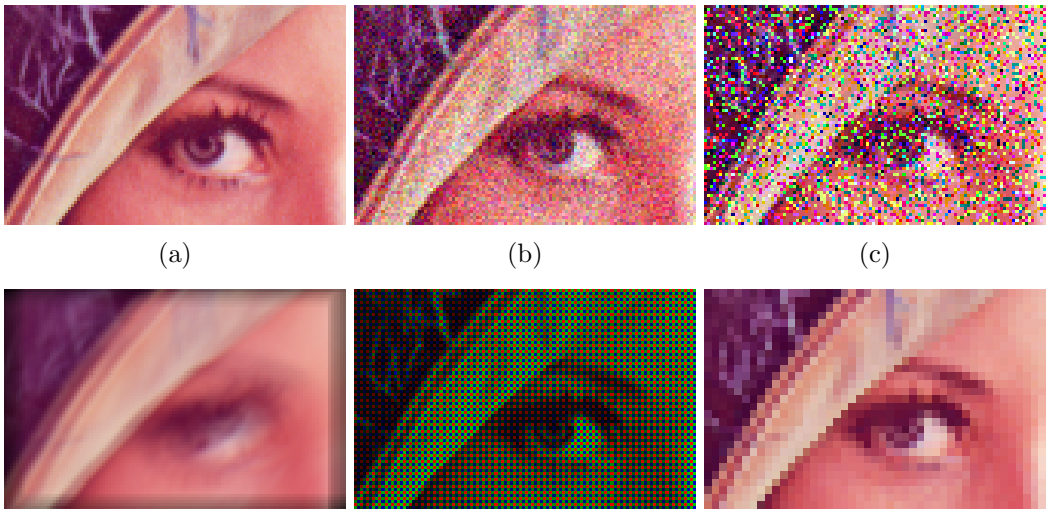


Figure 1.1: Examples of typical degradations considered in image restoration: The original, clean image y a, the same image corrupted by Gaussian b and Salt and Pepper noise c, and blurred d, mosaicked e, and downsampled f instances.

diagonal pixels are green, one is red and one blue. The process of determining the missing colors of the pixels is referred to as demosaicking.

Regardless of the origin of the degradations, *IR* is incredibly important since it often serves as a preprocessing stage for higher-level computer vision tasks. Therefore, at this stage it is important to employ algorithms with strong mathematical guarantees which allow for interpretability by humans. As such, the purpose of this thesis is not to break Peak Signal to Noise Ratio (PSNR) records for specific tasks, but to recast the *IR* problem into some well-known learning frameworks and incorporate recent advances from the deep learning community. Specifically, we revisit the framework of pure Energy Based Models (EBMs) [38], where the energy is given by a variational formulation [10]. As a regularization prior, we use a highly expressive deep Convolutional Neural Network (CNN), which we train in a generative manner inspired by Structured Support Vector Machines (SSVMs) such that it can be used as a generic prior for any *IR* task.

1.2 Mathematical Formulation

The purpose of *IR* is to recover the latent clean image $y \in \mathbb{R}^n$ of size $n = n_1 \times n_2$ from the degraded observation

$$x = Ay + \nu \in \mathbb{R}^m \,, \tag{1.1}$$

where $A \in \mathbb{R}^{m \times n}$ models the degradation and $\nu \in \mathbb{R}^m$ is an additive noise component drawn from some noise distribution. As an example, in the case of an Additive White

Gaussian Noise (AWGN) denoising problem, we have $n = m$ and $A = I_n$, where I_n is the $n \times n$ identity matrix. For *SISR*, A would model the composite operator of convolution and down-sampling.

In the most general sense, we seek a function

$$f \colon \mathbb{R}^m \to \mathbb{R}^n \tag{1.2}$$

such that

$$\hat{y} = f(x), \tag{1.3}$$

where $\hat{y} \in \mathbb{R}^n$ is the estimation of the latent clean image y. The goal of this estimation is to get as close as possible to y, according to some predefined loss function. In the next sections, we will outline the history of proposed solutions.

1.3 Analytical Solutions and Inverse Filtering

In this section we focus on a deblurring problem as an illustrative example. Typically, the blurring process is mathematically described by a convolution. In the continuous setting, we have

$$\mathrm{x}(\hat{\imath}, \hat{\jmath}) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mathrm{y}(\hat{\imath} - \gamma, \hat{\jmath} - \xi)\mathrm{a}(\gamma, \xi)\, \mathrm{d}\gamma\, \mathrm{d}\xi + \nu(\hat{\imath}, \hat{\jmath})\,, \tag{1.4}$$

where a is the blur function or Point Spread Function (PSF). In the discrete setting, the integrals are replaced by the respective sums such that

$$x(i, j) = \sum_{h=-H/2}^{H/2} \sum_{w=-W/2}^{W/2} y(i - h, j - w)a(h, w) + \nu(i, j)\,, \tag{1.5}$$

where $a \in \mathbb{R}^{H \times W}$ is the blur kernel. We see that Eq. (1.5) is of the form of Eq. (1.1), where the images x, y (and the noise ν) are vectorized and the convolution is described as a matrix-vector product i.e.

$$A\,\mathrm{vec}(y) + \mathrm{vec}(\nu) \equiv a * y + \nu\,. \tag{1.6}$$

where $\mathrm{vec}(\cdot)$ is the vectorization operator.

Note that here we are constructing the degraded image x by convolving two images defined on a finite domain. Assuming that we want x to be defined on the same domain as y, Eq. (1.5) requires to evaluate the image y on points outside of its domain. There exist many possibilities to extend y over its initial domain for the purpose of convolution. A simple approach is to extend the domain of y by padding it with zeros. Another possibility, often favored due to its theoretical properties, is the so-called circular boundary handling,

where the corresponding convolution is defined by

$$x(i,j) = \sum_{h=-H/2}^{H/2} \sum_{w=-W/2}^{W/2} y(i - h \bmod N_x, j - w \bmod N_y) a(h,w) + \nu(i,j) \tag{1.7}$$

for $y \in \mathbb{R}^{N_x \times N_y}$. We denote convolution with circular boundary conditions as $\mathring{*}$. For a (very brief) review of convolution arithmetic and details such as border handling we refer the reader to [18].

Now, let

$$\mathcal{F}\{y\}(\chi,\psi) = \sum_{i=0}^{N_x-1} \sum_{j=0}^{N_y-1} y(i,j) \exp\left(-2\pi \mathrm{i} \left(\frac{i\chi}{N_x} + \frac{j\psi}{N_y}\right)\right) \tag{1.8}$$

be the Discrete Fourier Transform (DFT) of the image $y \in \mathbb{R}^{N_x \times N_y}$, where $\mathrm{i} = \sqrt{-1}$ is the imaginary constant, and let

$$\mathcal{F}^{-1}\{\mathcal{F}\{y\}\}(i,j) = \sum_{\chi=0}^{N_x-1} \sum_{\psi=0}^{N_y-1} \mathcal{F}\{y\}(\chi,\psi) \exp\left(2\pi \mathrm{i} \left(\frac{i\chi}{N_x} + \frac{j\psi}{N_y}\right)\right) \tag{1.9}$$

be its inverse. A well known and widely used property of the *DFT* is the convolution theorem

$$\mathcal{F}\{a \mathring{*} y\} = \mathcal{F}\{a\}\mathcal{F}\{y\}\,, \tag{1.10}$$

with the circular convolution $\mathring{*}$ (Eq. (1.7)). Let us for the moment consider the case of a known blur kernel a and no noise ($\nu = 0$), i.e.

$$x = a \mathring{*} y\,. \tag{1.11}$$

Then, with Eq. (1.10) we see that $\mathcal{F}\{x\} = \mathcal{F}\{a\}\mathcal{F}\{y\}$, such that we can reconstruct the clean image y as

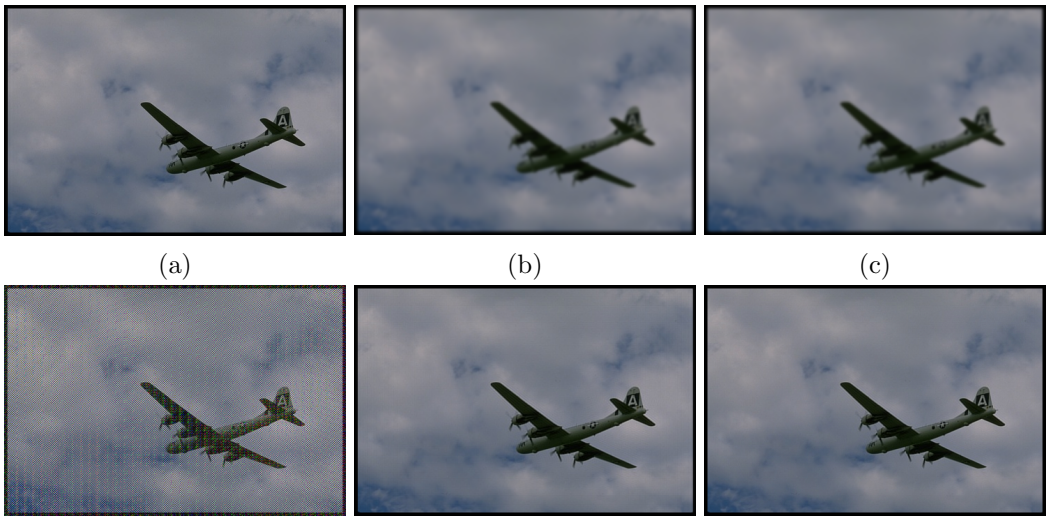
$$y = \mathcal{F}^{-1}\left\{\frac{\mathcal{F}\{x\}}{\mathcal{F}\{a\}}\right\}. \tag{1.12}$$

This is the inverse filtering approach [4], which is well studied in the field of discrete time digital signal processing. However, this approach breaks down as soon as Eq. (1.11) is not valid, i.e. if $\nu \neq 0$. In this case, due to the division in Eq. (1.12), the solution will look significantly worse even for $\nu \ll$. This can be somewhat remedied by introducing a constant that is added to the denominator, which leads to the Wiener deconvolution formulation [30, 62]

$$\hat{y} = \mathcal{F}^{-1}\left\{\frac{\mathcal{F}\{a\}^* \mathcal{F}\{x\}}{|\mathcal{F}\{a\}|^2 + \frac{1}{\mathrm{SNR}(x)}}\right\}, \tag{1.13}$$

with the complex conjugate $(\cdot)^*$, and the Signal to Noise Ratio (SNR) of x.

In Fig. 1.2 we show an example of noisy deconvolution using the inverse filtering



(a) (b) (c)

(d) (e) (f)

Figure 1.2: Filtering approaches for the deconvolution problem: The clean image a, the blurred b and noisy c observations are shown in the top row. The bottom row shows the reconstruction using inverse filtering d, the Wiener filter e and the Orieux approach [47] f. Even though there is almost no visible difference between b and c, the inverse filtering approach already introduces strong unwanted artifacts.

approach Eq. (1.12), the Wiener filter Eq. (1.13) and the approach of [47], which tries to estimate the proper additive term in the denominator of Eq. (1.13). Whilst the inverse filtering solution is perfect for $\nu = 0$, it is practically useless in the general case due to the amplification of the noise. On the other hand, the Wiener filter and the Orieux estimation still yield acceptable results. The degraded observation was produced by convolving the clean image with a Gaussian *PSF* of size 11×11 and standard deviation $\sigma_{\text{psf}} = 2$, and adding Gaussian noise with $\sigma = 1 \times 10^{-7}$.

1.4 Coefficient Shrinkage in Transformed Domain

Another particularly well studied approach is that of coefficient shrinkage or sparsity in a transformed domain. Starting with [16], which introduced the general framework of adaptive coefficient shrinkage (in this case in the wavelet domain), there is a large body of literature on the topic. Here, we want to highlight the principle by considering the denoising problem, i.e. $A = I_n$, such that

$$x = y + \nu\,. \tag{1.14}$$

The following description is very informal and should provide a general overview of the principle in the image restoration domain. For a more rigorous mathematical analysis we

refer to [15, 17, 55].

Let $\mathcal{D} = (\psi_\gamma)_{\gamma\in\Gamma}$ be a collection of waveforms, such that

$$x = \sum_{\gamma\in\Gamma} \alpha_\gamma \psi_\gamma \,. \tag{1.15}$$

Further, let $\mathcal{T}\{x\}$ be a transformation which maps $x \mapsto \alpha_\gamma\,, \gamma \in \Gamma$. Examples may be the Discrete Cosine Transform (DCT), the discrete wavelet [43] or shearlet transform [20]. Now, we assume that in the transformed space there exists some prior knowledge about the distribution of the coefficients. For instance, in the scale-time domain of the wavelet transform, in the world of natural images one might expect most of the "power" to exist at low scales (piecewise constant assumption), and high correlation of the coefficients across scales.

Therefore, a simple idea is to introduce a threshold and discard small coefficients. The reconstructed image $\hat{y}$ is then acquired by performing the inverse transformation on the altered coefficients, i.e.

$$\hat{y} = \mathcal{T}^{-1}\{\mathcal{P}\{\mathcal{T}\{x\}\}\} \tag{1.16}$$

where $\mathcal{P}\{\cdot\}$ is an operator that changes the coefficients by incorporating prior knowledge. One of the simplest approaches is soft-thresholding, where the operator acting on the coefficients is the soft-thresholding operator

$$S_t(\alpha) = \begin{cases} \frac{\alpha}{|\alpha|}(|\alpha| - t) & \text{if } |\alpha| > t, \\ 0 & \text{else.} \end{cases} \tag{1.17}$$

This idea can be extended by *learning* the distributions and interconnections of the coefficients. For instance, [15] uses this approach, where the statistical dependencies of wavelet coefficients are captured using a Hidden Markov Model (HMM). Chambolle et al. [8] cast the wavelet shrinkage algorithms in the theory of *variational problems*.

1.5 Variational Methods

A large set of image restoration problems (or more general linear inverse problems) can be cast into the framework of variational problems. In this framework, images are interpreted as a function defined on some finite domain Ω (e.g. $\Omega = [0,1]^2$), i.e. $y\colon \Omega \to \mathbb{R}$, and in the most general sense we seek a function $\hat{y}\colon \Omega \to \mathbb{R}$ that minimizes some functional $E(x, y)$. In the following, we let $(x, y) \in \mathcal{X} \times \mathcal{Y}$ where $\mathcal{X}$, $\mathcal{Y}$ are the sets of admissible functions for x and y respectively.

The energy functional $E\colon \mathcal{X}\times\mathcal{Y} \to \mathbb{R}$ is composed of a data fidelity term $D\colon \mathcal{X}\times\mathcal{Y} \to \mathbb{R}$ which measures the dissimilarity between x and Ay and a regularization term $R\colon \mathcal{Y} \to \mathbb{R}$, which encodes some prior belief on y. A proper statistical argument for this structure is given in Chapter 2, and we analyze variational methods as instances of *EBMs* in Chapter 3.

The estimation is then

$$\hat{y} = \arg\min_{y} \left\{ E(x, y) = \lambda D(x, y) + R(y) \right\}, \tag{1.18}$$

where $\lambda \in \mathbb{R}^+$ is a freely selectable parameter controlling the trade-off between the data term $D(x, y)$ and the regularization $R(y)$.

Modeling the data fidelity term $D(x, y)$ is often straight forward. For instance, in the case of *AWGN*, it is well known that the squared ℓ_2 norm

$$D(x, y) = \frac{1}{2} \|x - Ay\|_2^2 \tag{1.19}$$

is optimal (again, see Chapter 2). On the other hand, finding a suitable regularizer $R(y)$ for natural images has been subject to decades of research. We note that this formulation makes it exceptionally easy to apply a given regularizer $R(y)$ to many image restoration or other, more general, linear inverse problems. By simply substituting the appropriate degradation operator A in Eq. (1.19), the prior knowledge modeled by $R(y)$ can be used in a broad context.

1.5.1 Non-Parametric Regularizers

There exists a large body of literature on finding a suitable regularization term $R(y)$. In the influential Rudin Osher Fatemi (ROF) model [51], the authors, who the model is named after, designed a Total Variation (TV)-based regularizer, based on the assumption that natural images can be approximated reasonably well by piecewise constant images. The *TV* reads

$$\mathrm{TV}(y) = \sum_{i=1}^{n} \sqrt{(\nabla_1 y)_i^2 + (\nabla_2 y)_i^2}, \tag{1.20}$$

where ∇_1, ∇_2 are the first order finite difference operators in the principle directions. The *ROF* model combines the *TV* regularizer with a squared L_2 data term to read

$$E_{\mathrm{ROF}}(x, y) = \frac{\lambda}{2} \|x - Ay\|_2^2 + \mathrm{TV}(y). \tag{1.21}$$

TV regularization leads to piecewise constant images with sharp edges, which can yield surprisingly good results for natural images in the presence of *AWGN* or Salt and Pepper noise.

A well known drawback of the *ROF* model is the "staircasing" effect, which is especially apparent in regions with linearly changing intensity. This motivated many extensions of the *TV* regularizer [6, 9, 12] which incorporate higher order image statistics, e.g. the first order Total Generalized Variation (TGV) [6], which allows for linear image gradients and steep edges.

Another well-studied class of regularization functionals is based on penalizing the cur-

vature of level lines [11, 46, 54]. These regularizers are mathematically well understood and can be nicely motivated by the human visual system [32].

All of the above mentioned regularizers have a profound mathematical underpinning, are well understood and lead to convex problems when applied in the general framework of Eq. (1.18). However, these hand-crafted regularizers lack expressiveness to be applied to a wide variety of natural images, as they only model very specific and fundamental properties. Thus, the idea of *learning* a regularizer from a bank of natural images emerged.

1.5.2 Parametric Regularizers

To overcome the limited expressiveness of hand-crafted regularization functionals, many solutions have been proposed to learn a regularizer from data [29, 33, 35, 40, 50]. Specifically, we parametrize the regularization term in Eq. (1.18) by θ, such that

$$\hat{y} = \arg\min_{y} \left\{ E(x, y, \theta) = \lambda D(x, y) + R(y, \theta) \right\}, \tag{1.22}$$

where $\theta \in \Theta \subseteq \mathbb{R}^p$ are the parameters that are learned from data.

An early and very successful approach is the Fields of Experts (FoE) model [50] defined as

$$R(y, \theta) = -\sum_{i=1}^{n} \left(\sum_{k=1}^{K} \log \phi_k(J_k y, \alpha_k) \right)_i, \tag{1.23}$$

which consists of linear filters (convolution matrices) $J_k \in \mathbb{R}^{n \times n}$, $k = 1, \dots, K$, and potential functions

$$\phi_k(J_k y, \alpha_k) = \left(1 + \frac{1}{2} (J_k y)^2 \right)^{-\alpha_k} \in \mathbb{R}^n, \tag{1.24}$$

parametrized by $\{\alpha_k\}_{k=1}^K$, $\alpha_k \in \mathbb{R}$ such that $\theta = \{J_k, \alpha_k\}_{k=1}^K$. We note that the *FoE* model can be interpreted as a generalization of the anisotropic *TV* regularizer. In the *TV* regularizer, the potential function ϕ is fixed and we have $K = 2$, where $J_{\{1,2\}} \equiv \nabla_{\{1,2\}}$. The gradient operators $\nabla_{\{1,2\}}$ can be seen as convolutions with specific kernels, and $J_{\{1,2\}}$ are the corresponding convolution matrices.

Recently, this framework has been extended [33, 40, 41] by letting

$$R(y, \theta) = \mathcal{N}(y, \theta), \tag{1.25}$$

where $\mathcal{N}(y, \theta)$ is a generic neural network (of arbitrary complexity) parametrized by $\theta \in \Theta$. These highly expressive regularizers have lead to impressive results when trained in a discriminative fashion [33]. However, training and inference in such a framework usually involves sampling the model using Markov Chain Monte Carlo (MCMC) techniques [45, 61], which are computationally expensive. This motivated "truncated" gradient descent schemes, which are computationally much more efficient, and surprisingly yield better

results than their converged counterparts in a discriminative learning framework.

1.6 Iteratively Unrolled Gradient Descent Schemes

The idea of early stopping problems of the form of Eq. (1.18) was introduced in [2]. Given a training set $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^{N_S}$, they cast the learning problem as

$$\begin{cases} \min\limits_{\theta} \sum\limits_{i=1}^{N_S} \frac{1}{2} \left\| \hat{y}^i - y^i \right\|_2^2 \\ \text{s.t. } \hat{y}^i = \mathrm{AI}(x^i, \theta)\,, \end{cases} \tag{1.26}$$

where AI is an algorithm that performs approximate inference on the underlying Markov Random Field (MRF) model (e.g. *FoE*), given the current parameters θ. The joint training of a *MRF* given some inference algorithm AI was named Active Random Field (ARF) by the authors [2].

$\mathrm{AI}(x, \theta)$ sacrifices inference accuracy for speed, where for instance if $\mathrm{AI}(x, \theta)$ models a gradient descent optimization, they use 1 to 4 iterations, as opposed to tens of thousands needed for convergence. Even though (or, precisely *because* [21, 33]) the inference algorithm is truncated after very few steps and hence only approximates the energy minimization, the results show better performance [2] when compared to convergent algorithms.

The scheme in Eq. (1.26), although approximate, can still be interpreted in the framework of energy minimization. At the cost of abandoning the energy-based framework, the power of the scheme can be increased by parametrizing each step in the gradient descent scheme individually. The Cascade of Shrinkage Field (CSF) model [53] learns a cascade of Gaussian Conditional Random Fields (CRFs) in such a way, which relies on several assumptions on the considered problem. In [13] these assumptions are relaxed and the Trainable Nonlinear Reaction Diffusion (TNRD) model is proposed, which is interpreted as an unrolled nonlinear reaction-diffusion scheme. The convolution kernels J_k^t and parameters of the potential functions ϕ_k^t are free to vary at each step and are trained simultaneously. The whole training process is formulated as

$$\begin{cases} \min\limits_{\theta} \sum\limits_{i=1}^{N_S} \frac{1}{2} \left\| \hat{y}_T^i - y^i \right\|_2^2 \\ \text{s.t. } \begin{cases} \hat{y}_0^i = x^i \\ \hat{y}_t^i = \hat{y}_{t-1}^i - \left(\sum_{k=1}^{K} (J_k^t)^\top \phi_k^t (J_k^t \hat{y}_{t-1}^i) + \lambda^t A^\top (A \hat{y}_{t-1}^i - x) \right) \\ t = 1, \ldots, T \end{cases} \end{cases} \tag{1.27}$$

with the parameters $\theta = \{\lambda^t, J_k^t, \phi_k^t\}_{t=1}^{T}$. The update equation corresponds exactly to a gradient descent on the *FoE* model, assuming a quadratic data fidelity term of the form $D(x, y) = \frac{1}{2} \left\| x - Ay \right\|_2^2$. However, the parameters are free to change between steps, and

the depth T is usually kept below 10.

The authors of [13] already note that their approach is tightly linked to recurrent neural networks. The theory of Variational Networks (VNs) introduced in [34] makes this connection more clear, and relates such approaches to the popular residual neural networks [28].

Very recently, [19] showed that deep Neural Networks (NNs) can be interpreted as a discretization of a suitable nonlinear dynamical system. In this formulation, the training process is seen as finding the controls in the corresponding optimal control problem. Motivated by this insight and the observation that early stopping a scheme following Eq. (1.22) in general leads to better results, [21] proposed to learn the best stopping time in the optimal control framework. They extended the *FoE* model to use B-spline potential functions and 7×7 filters to achieve notably good results, especially considering the small inference time and low number of parameters. This work was extended in [33], where the regularizer is given by multi-scale residual *CNN* and the results are comparable to other State-of-the-Art (SotA) techniques, whilst the number of parameters is comparatively small.

1.7 Deep Feed-Forward Neural Networks

The *NN* community has received much attention in the recent years. In the *IR* domain, *CNNs* have achieved remarkable results [28, 65, 66], challenging or beating other *SotA* techniques. The success of these methods however can largely be attributed to the rapid increase in computational power in the recent years and the associated increase in model complexity, as well as the availability of large data sets. On the other hand, in general feed-forward *NNs* lack theoretical foundations as well as guarantees, and especially interpretability for *IR* problems.

The feed-forward *NN* approach to finding Eq. (1.2), is to suitably parametrize f, such that

$$\hat{y} = f(x, \theta)\,, \tag{1.28}$$

where $\theta \in \Theta$ are the parameters. Since for traditional *IR* tasks shift-equivariance is assumed [22, 37], weight sharing [52] is employed in the form of convolutions. This drastically reduces the number of parameters when compared to "dense" networks.

We can see that the formulation in Eq. (1.28) does not make use of any prior knowledge: The learned mapping is completely independent of the (a-priori known) degradation operator A, and it does not immediately incorporate any prior knowledge about the solution $\hat{y}$ (c.f. Eq. (1.18)). A direct consequence of this is that it is notoriously hard to transfer knowledge between tasks, e.g. to use a model trained for denoising in a *SISR* task. Further, such direct feed-forward architectures have no guarantee of consistency with respect to the measurement data, which is a concern especially for more general inverse problems such as Computed Tomography (CT) or similar medical imaging modalities.

2

Probabilistic Modeling and Bayesian Inference

In this chapter we want to show how the variational structure Eq. (1.18) naturally arises from a probabilistic interpretation of Image Restoration (IR). On our way, we will encounter other estimators, and show why these are in general not useful for *IR*. Let us consider Eq. (1.1), where we assume that the additive noise ν is drawn from zero-mean Gaussian distribution with variance σ^2, i.e.

$$\nu \sim \mathcal{N}(0, \sigma^2 I_m),\ p_\nu(\nu) \coloneqq \frac{1}{\sqrt{(2\pi)^m \sigma^{2m}}} \exp\left(-\frac{1}{2\sigma^2} \|\nu\|_2^2\right). \tag{2.1}$$

From this, it immediately follows that x is a random variable distributed according to

$$x \sim \mathcal{N}(Ay, \sigma^2 I_m),\ p_{x|y}(x \mid y) \coloneqq \frac{1}{\sqrt{(2\pi)^m \sigma^{2m}}} \exp\left(-\frac{1}{2\sigma^2} \|x - Ay\|_2^2\right), \tag{2.2}$$

with the conditional Probability Density Function (PDF) $p_{x|y}$ of the degraded observation x given a clean observation y. We can now consider the Maximum Likelihood (ML) estimate $\hat{y}_{\mathrm{ML}}$ for y as

$$\hat{y}_{\mathrm{ML}} = \arg\max_y p_{x|y}(x \mid y) = \arg\max_y \left\{ \frac{1}{\sqrt{(2\pi)^m \sigma^{2m}}} \exp\left(-\frac{1}{2\sigma^2} \|x - Ay\|_2^2\right) \right\}. \tag{2.3}$$

We can transform this problem to be more easily read by considering the negative logarithm and discarding constant terms, turning it into

$$\hat{y}_{\mathrm{ML}} = \arg\min_y \|x - Ay\|_2^2 . \tag{2.4}$$

The solution is easily computed as

$$\hat{y}_{\mathrm{ML}} = \left(A^\top A\right)^{-1} A^\top x. \tag{2.5}$$

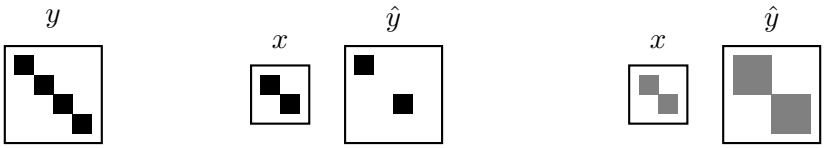


Figure 2.1: Least-Squares Solution of a *SISR* problem using the nearest-neighbor and box-averaging sub-sampling operators.

This is the famous least-squares solution of $Ay = x$, which has the probabilistic interpretation of being the *ML* estimate $\hat{y}_{\mathrm{ML}} = \arg\max_y p_{x|y}(x \mid y)$ under the assumption of Additive White Gaussian Noise (AWGN).

However, the *ML* solution is rarely useful. For instance, in the case of a denoising problem, the degradation operator A is the identity matrix I_m, such that $\hat{y}_{\mathrm{ML}} = x$, i.e. the *ML* solution is the degraded image. For Single-Image Super-Resolution (SISR), the least-squares estimation yields the trivial zero-filing solution in case of nearest-neighbor sub-sampling, and the nearest-neighbor interpolation scheme for box-averaged sub-sampling. We show a toy example of *SISR* in Fig. 2.1.

To guide the solution towards a more "physically plausible" one, we desire a way to incorporate prior knowledge about the clean observation y into the solution. Assuming we have some prior knowledge about the distribution of the clean image patches y, we can use Bayes Theorem to describe the posterior distribution as

$$p_{y|x}(y \mid x) = \frac{p_{x|y}(x \mid y)p_y(y)}{p_x(x)}. \tag{2.6}$$

where $p_y(y)$ models our prior belief on y. The *PDF* $p_{y|x}(y \mid x)$ describes the complete posterior distribution of y given x, and we can decide which point $\hat{y}$ is most compatible with a given degraded observation x.

However, the question which point $\hat{y}$ in $p_{y|x}(y \mid x)$ should be selected as the estimate is still unclear. The most general form of a Bayesian estimator is given as

$$\hat{y} = \arg\min_y \int_\nu \ell(y, \nu) p_{y|x}(\nu \mid x)\, \mathrm{d}\nu \tag{2.7}$$

where $\ell \colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ assigns a scalar loss to a given configuration (y, ν). A possible choice of ℓ is the euclidean distance, i.e. $\ell(y, \nu) = \|y - \nu\|_2^2$, which leads to

$$\hat{y}_{\mathrm{E}} = \arg\min_y \int_\nu p_{y|x}(\nu \mid x)\, \|y - \nu\|_2^2\, \mathrm{d}\nu, \tag{2.8}$$

for which the solution is given by

$$\hat{y}_{\mathrm{E}} = \frac{\int_\nu p_{y|x}(\nu \mid x)\nu \,\mathrm{d}\nu}{\int_\nu p_{y|x}(\nu \mid x)\,\mathrm{d}\nu} = \int_\nu p_{y|x}(\nu \mid x)\nu \,\mathrm{d}\nu, \tag{2.9}$$

which is the expectation of $p_{y|x}$ with respect to y.

In general, the expectation can not be evaluated because of the associated computational cost. Another very popular choice for ℓ is the zero-one loss $\ell(y,\nu) = 1 - \delta(y,\nu)$, where

$$\delta(y,\nu) = \begin{cases} 1 & \text{if } y = \nu, \\ 0 & \text{else.} \end{cases} \tag{2.10}$$

The corresponding estimate is

$$\hat{y}_{\mathrm{MAP}} = \arg\min_y \int_\nu p_{y|x}(\nu \mid x)(1 - \delta(y,\nu))\,\mathrm{d}\nu, \tag{2.11}$$

which selects $\hat{y}_{\mathrm{MAP}}$ for which $p_{y|x}(y \mid x)$ is maximized, i.e.

$$\hat{y}_{\mathrm{MAP}} = \arg\max_y p_{y|x}(y \mid x) = \arg\max_y p_{x|y}(x \mid y)p_y(y) \tag{2.12}$$

and hence it is called the Maximum A-Posteriori (MAP) estimate. The normalization by $p_x(x)$ is constant w.r.t. y and hence can be disregarded for optimization. In the negative-log domain this amounts to

$$\hat{y}_{\mathrm{MAP}} = \arg\min_y \left\{ -\log\left(p_{x|y}(x \mid y)\right) - \log\left(p_y(y)\right) \right\}, \tag{2.13}$$

which, assuming *AWGN* (Eq. (2.2)) leads to

$$\hat{y}_{\mathrm{MAP}} = \arg\min_y \left\{ \frac{1}{2\sigma^2} \underbrace{\|x - Ay\|_2^2}_{D(x,y)} \underbrace{- \log\left(p_y(y)\right)}_{R(y)} \right\}. \tag{2.14}$$

We see that this is of the form of Eq. (1.18), where $\lambda D(x,y)$ models the negative log-likelihood $-\log\left(p_{x|y}(x \mid y)\right)$ and $R(y)$ models the negative log-prior $-\log\left(p_y(y)\right)$.

3 Energy-Based Models

Contents

In the previous section, we already hinted at variational methods as an instance of Energy Based Models (EBMs). Here, we take a more general look at their principles and motivations, especially regarding the domain of Image Restoration (IR). *EBMs* are characterized by an energy function $E\colon \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ that measures the compatibility between $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. For *IR* problems, x might represent the degraded observation, and y would be the corresponding estimate of the clean scene.

3.1 Probabilistic Interpretation

The energy defines a *Gibbs-Boltzmann distribution* [23]

$$p(y \mid x) = \frac{\exp(-\beta E(x, y))}{Z} \tag{3.1}$$

where the partition function $Z = \int_{\mathcal{Y}} \exp(-\beta E(x, y))\, \mathrm{d}y$ normalizes the Probability Density Function (PDF). $\beta \in \mathbb{R}^+$ is a freely-chosen constant, the physical interpretation of which would be an inverse temperature. Computing the partition function Z is in general intractable for any interesting application. Much of this thesis will be concerned with different estimation strategies for Z, and how current learning strategies can be interpreted as estimators for Z.

3.2 Inference

Generally, energy-based models will be used in the setting where an input x is given, and the model should predict the most compatible output $\hat{y}$. In other words, the inference problem consists of fixing the value of the observed variables, and minimizing the energy with respect to the remaining variables. That is, the inference problem reads

$$\hat{y} = \arg\min_{y \in \mathcal{Y}} E(x, y)\,, \tag{3.2}$$

where we use the convention that low values of $E(x, y)$ correspond to favorable configurations of x and y.

Note that depending on the cardinality of $\mathcal{Y}$, finding $\hat{y}$ can be very challenging. If we consider an image labeling task in 2 labels (i.e. binary classification), we have $\mathcal{Y} = \{\mathbb{L}_1, \mathbb{L}_2\}$, where $\mathbb{L}_{\{1,2\}}$ are the two labels, and hence $|\mathcal{Y}| = 2$. Therefore, the most straight-forward approach to solving Eq. (3.2) is to evaluate it for both labels and simply choose $\hat{y} = \arg\min_{y \in \{\mathbb{L}_1, \mathbb{L}_2\}} E(x, y)$. However, in the case of *IR*, $|\mathcal{Y}|$ is infinite, since we view images as continuous functions. Even in the quantized case, e.g. $y \in [\![0, 255]\!]^n$, brute-force evaluation of Eq. (3.2) is infeasible due to the size of $\mathcal{Y}$. For instance, $|\mathcal{Y}| = 256^4 = 4\,294\,967\,296$ for an image of size $n = 2 \times 2 = 4$.

Thus, typically a problem-specific inference procedure is needed. Often this procedure will lead to an approximate result, possibly even an approximation of some local minimum of the energy function. In our case, for *IR* we will use gradient-based techniques, to land in a local minimum of our highly non-convex energy function.

3.3 Training

In the following, we parametrize our energy function by the parameters θ, such that $E\colon \mathcal{X} \times \mathcal{Y} \times \Theta \to \mathbb{R}$, $(x, y, \theta) \mapsto E(x, y, \theta)$. The, the goal of training is to find the optimal parameters $\hat{\theta}$ by considering the family of functions

$$\mathcal{E} = \{E(x, y, \theta)\colon \theta \in \Theta\}, \tag{3.3}$$

and choosing $\hat{\theta}$ such that $E(x, y, \hat{\theta})$ produces a good estimate $\hat{y}$ given any input x. In general, we assume access to a data set $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^{N_S}$, where x_i is the query for the i-th training point and y_i is the corresponding answer.

Based on the training set $\mathcal{S}$, we want to find

$$\hat{\theta} = \arg\min_{\theta \in \Theta} \frac{1}{N_S} \sum_{i=1}^{N_S} \mathcal{L}(y_i, \theta)\,, \tag{3.4}$$

where $\mathcal{L}$ is a loss functional. Intuitively, we require the loss functional $\mathcal{L}$ to assign a small

loss to the energy function E, if E acts as desired: It should assign the lowest energy to the correct answer and higher energy to all others. On the other hand, energy functions which assign high energy to the correct answer and lower energy to others should have a high loss.

In other words, training the *EBM* corresponds to shaping the energy landscape such that for any input x, the inference procedure will select the desired value y. The inference algorithm Eq. (3.2) selects y with the lowest energy, hence the learning algorithm should shape the landscape in a way that y has lower energy than all other answers. With this in mind, it is clear that a "good" loss function $\mathcal{L}$ should fulfill two requirements: The effect of an update on θ should 1. lower $E(x_i, y_i, \theta)$, and 2. increase the energy of incorrect answers $E(x_i, \hat{y}, \theta)$.

In what follows, we use the typical variational structure

$$E(x, y, \theta) = \lambda D(x, y) + R(y, \theta) \tag{3.5}$$

which is interpreted as defining the posterior distribution $p_{y|x}$, as derived in Chapter 2, through its Gibbs-Boltzmann distribution. As the data term is defined through our noise model and hence not parametrized in any way, we can ignore it during training. Therefor, in the following we are not dependent on any degraded examples x_i, but want to learn the prior $R(y, \theta)$ given a data set $\mathcal{S} = \{y_i\}_{i=1}^{N_S}$. This means that we are in the realm of generative models.

3.3.1 Energy Loss

The simplest meaningful loss function is the energy of the desired answer y_i, i.e.

$$\mathcal{L}_{\mathrm{E}}(y_i, \theta) = R(y_i, \theta)\,. \tag{3.6}$$

Training with this loss function certainly has the effect of decreasing the energy of the desired answer, however there exists no mechanism for increasing the energy of any other answer. As such, this loss can lead to the trivial solution $R(y, \theta) = \text{const}\ \forall\ y$, unless the architecture of $R(\cdot, \theta)$ itself is such that decreasing $R(y_i, \theta)$ necessarily leads to an increase of other energies.

3.3.2 Maximum Likelihood Estimation

As stated before, we assume access to a data set $\mathcal{S} = \{y_i\}_{i=1}^{N_S}$ of N_S independent and identically distributed (iid) data points $\mathbb{R}^d \ni y_i \sim p_D$, where p_D is the underlying data distribution. As discussed above, the regularizer defines a Gibbs-Boltzmann distribution

$$p_\theta(y) = \frac{\exp\left(-R(y, \theta)\right)}{\int_{\mathbb{R}^d} \exp\left(-R(\nu, \theta)\right) \mathrm{d}\nu}. \tag{3.7}$$

The goal of Maximum Likelihood (ML) is to find the parameters $\hat{\theta}_{\text{ML}} \in \Theta$ which maximize the likelihood of the observed data $\mathcal{S}$ under the model distribution p_θ. To this end, we introduce the *likelihood function*

$$\text{L}(\theta) = p_\theta(\mathcal{S}) \tag{3.8}$$

which measures the probability of observing $\mathcal{S}$ given a parameter estimate θ. Using the *iid* assumption, this decomposes into

$$\text{L}(\theta) = p_\theta(\mathcal{S}) = \prod_{i=1}^{N_S} p_\theta(y_i). \tag{3.9}$$

The *ML* estimate $\hat{\theta}_{\text{ML}}$ is then

$$\hat{\theta}_{\text{ML}} = \arg\max_{\theta \in \Theta} \text{L}(\theta). \tag{3.10}$$

In practice, it is often easier to minimize the negative logarithm of Eq. (3.8). We therefor define the negative log-likelihood $\text{NLL}(\theta)$ as

$$\text{NLL}(\theta) = -\log \text{L}(\theta) = \sum_{i=1}^{N_S} -\log p_\theta(y_i) \tag{3.11}$$

and choose

$$\hat{\theta}_{\text{ML}} = \arg\min_{\theta \in \Theta} \text{NLL}(\theta). \tag{3.12}$$

Taking the negative logarithm of Eq. (3.7) yields

$$-\log p_\theta(y) = R(y, \theta) + \log \int_{\mathbb{R}^d} \exp\left(-R(\nu, \theta)\right) \mathrm{d}\nu. \tag{3.13}$$

With the exception of some trivial cases, no closed form solution exists for Eq. (3.10) or Eq. (3.12). As such, the estimates are usually found using an iterative technique. Thus, we require the gradient of the negative log-likelihood:

$$\nabla_\theta \text{NLL}(\theta) = \sum_{i=1}^{N_S} \nabla_\theta R(y_i, \theta) - \frac{N_S}{Z(\theta)} \int_{\mathbb{R}^d} \exp\left(-R(\nu, \theta)\right) \nabla_\theta R(\nu, \theta) \, \mathrm{d}\nu, \tag{3.14}$$

where $Z(\theta) = \int_{\mathbb{R}^d} \exp\left(-R(\nu, \theta)\right) \mathrm{d}\nu$ is the normalizing constant or *partition function*. The second term in Eq. (3.14) is exactly the expected gradient of $R(y, \theta)$ with respect to $y \sim p_\theta$. Thus, rewriting in terms of expected values yields

$$\nabla_\theta \text{NLL}(\theta) = \Big\langle \nabla_\theta R(y, \theta) \Big\rangle_{y \sim p_D} - \Big\langle \nabla_\theta R(y, \theta) \Big\rangle_{y \sim p_\theta} \tag{3.15}$$

where $\langle \cdot \rangle_{x \sim p}$ denotes the expected value with respect to $x \sim p$. Consequently, *ML* esti-

mation of θ is equivalent to training the *EBM* with the Maximum-Likelihood loss

$$\mathcal{L}_{\mathrm{ML}}(y_i, \theta) = R(y_i, \theta) - \Big\langle R(y, \theta) \Big\rangle_{y \sim p_\theta}. \tag{3.16}$$

Clearly, Eq. (3.16) differs from Eq. (3.6) only by the inclusion of the contrastive term $-\langle R(y,\theta)\rangle_{y\sim p_\theta}$. However, it is this term that keeps θ from collapsing to a trivial solution, and as such it is vital. While the first term is easily calculated given a data set $\mathcal{S}$, the difficulty for *ML* in this case lies in calculating the expected gradient of $R(y, \theta)$ under $y \sim p_\theta$. This requires evaluating the partition function $Z(\theta)$, which in general is intractable for any $d \gg$.

Note that the *ML* objective also naturally arises when considering the Kullback-Leibler Divergence (KLD) between the data distribution p_D and the model distribution p_θ:

$$\begin{aligned} \mathrm{KL}(p_D \parallel p_\theta) &= \int_{\mathbb{R}^d} p_D(\nu) \log \frac{p_D(\nu)}{p_\theta(\nu)} \,\mathrm{d}\nu \\ &= \int_{\mathbb{R}^d} p_D(\nu) \log p_D(\nu) - p_D(\nu) \log p_\theta(\nu) \,\mathrm{d}\nu \\ &= -H(p_D) - \big\langle \log p_\theta(y) \big\rangle_{y \sim p_D} \end{aligned} \tag{3.17}$$

where $\mathrm{KL}(p \parallel q)$ is the *KLD* between p and q and $H(p)$ denotes the entropy of p. Since the entropy of the data distribution does not depend on θ, we have

$$\arg\min_{\theta \in \Theta} \mathrm{KL}(p_D \parallel p_\theta) = \arg\min_{\theta \in \Theta} \mathrm{NLL}(\theta). \tag{3.18}$$

Since we can not compute the second term in Eq. (3.15), it is usually approximated through sampling. This is done by Markov Chain Monte Carlo (MCMC) methods such as Gibbs or Metropolis sampling or simulated annealing. These methods do not make use of the local landscape of p_θ and hence require many iterations to converge to the steady-state distribution. Other methods such as Hamiltonian (Hybrid) Monte Carlo (HMC) or Stochastic Gradient Langevin Dynamics (SGLD) utilize the gradient $\nabla_y R(y, \theta)$ to explore the space more efficiently.

Regardless of the strategy, sampling from the current distribution p_θ is a computationally intensive task. It has to be performed after every update to the parameters θ, and many burn-in steps may be required before the samples approximate the steady-state distribution. Therefore, it is interesting to look at the properties of non-convergent *ML*-like algorithms.

3.3.2.1 Contrastive Divergence

Contrastive Divergence (CD) was introduced in [29] to train a Producs of Experts (PoE) model. In the previous section, we showed how the *ML* estimate of p_θ is equivalent to minimizing $\mathrm{KL}(p_D \parallel p_\theta)$ between the data distribution p_D and the model distribution p_θ.

CD aims to almost completely eliminate the computational cost associated with the *ML* method by considering a slightly different objective function.

In order to convey the idea, we slightly change the notation: Let $p^0 = p_D$ and let $p_\theta^\infty = p_\theta$ to emphasize that the equilibrium distribution (i.e. after infinitely many steps) of the *MCMC* process is the model distribution. $p^0 = p_D$ is a reasonable choice to emphasize that the Markov chain is initialized with the data distribution p_D.

The idea of *CD* is to compute

$$\min_\theta \mathrm{CD}(\theta) = \min_\theta \mathrm{KL}(p^0 \parallel p_\theta^\infty) - \mathrm{KL}(p_\theta^1 \parallel p_\theta^\infty) \tag{3.19}$$

where p_θ^1 is the distribution after one step of the *MCMC* process. Note that the first term is the *ML* objective.

Intuitively, the goal is that $p_\theta^\infty \approx p^0$. In this case, the *MCMC* process, initialized at the data distribution, should leave the distribution unaltered. Instead of letting the chain run to equilibrium, the idea is to let the chain run for one step and to compare the resulting distribution with the data distribution. Updating θ according to Eq. (3.19) makes the *MCMC* process less likely to wander away from the data distribution.

The objective Eq. (3.19) can also be motivated as follows: It is clear that p_θ^1 is closer to the equilibrium distribution p_θ^∞ than p^0, and hence $\mathrm{KL}(p^0 \parallel p_\theta^\infty) \geq \mathrm{KL}(p_\theta^1 \parallel p_\theta^\infty)$, i.e. the *CD* is always non-negative. Equality would imply $p^0 = p_\theta^1$, which in turn implies $p^0 = p_\theta^\infty$, since the distribution necessarily has to already be in equilibrium if it does not change on the first step. In other words, *CD* is always non-negative and only zero if the model is perfect.

Mathematically, we find

$$-\nabla_\theta \mathrm{CD}(\theta) = \Big\langle \nabla_\theta R(y,\theta) \Big\rangle_{y\sim p^0} - \Big\langle \nabla_\theta R(y,\theta) \Big\rangle_{y\sim p_\theta^1} + \nabla_\theta H(p_\theta^1), \tag{3.20}$$

where the third term has been empirically shown to be comparatively small to and in accordance with the first two terms, such that it can be ignored. Therefore, the usual update rule for the *CD* objective is given by

$$\Delta\theta \propto \Big\langle \nabla_\theta R(y,\theta) \Big\rangle_{y\sim p^0} - \Big\langle \nabla_\theta R(y,\theta) \Big\rangle_{y\sim p_\theta^1}. \tag{3.21}$$

The *CD* algorithm can be generalized to CD_n [7] where

$$\mathrm{CD}_n(\theta) = \mathrm{KL}(p^0 \parallel p_\theta^\infty) - \mathrm{KL}(p_\theta^n \parallel p_\theta^\infty), \tag{3.22}$$

such that

$$\Delta\theta \propto \Big\langle \nabla_\theta R_\theta(x) \Big\rangle_{x\sim p^0} - \Big\langle \nabla_\theta R_\theta(x) \Big\rangle_{x\sim p_\theta^n}. \tag{3.23}$$

Since $\mathrm{KL}(p_\theta^n \parallel p_\theta^\infty) \xrightarrow{n\to\infty} 0$, it is clear that

$$\arg\min_{\theta\in\Theta} \mathrm{CD}_n(\theta) \xrightarrow{n\to\infty} \arg\min_{\theta\in\Theta} \mathrm{NLL}(\theta). \tag{3.24}$$

3.3.2.2 Persistent Contrastive Divergence

The idea of CD_n, in essence, is to use distribution after the n-step reconstruction as an approximation for the equilibrium distribution p_θ^∞ defined by the model. This becomes especially clear when we consider the parameter update equations Eq. (3.15) versus Eq. (3.23), where in the second term the expectation is computed with respect to p_θ^n instead of p_θ^∞. If we assume that the parameter updates are small, i.e. $\Delta\theta \ll$, the model distribution p_θ^∞ is not expected to significantly change between iterations. Motivated by this insight, one might choose not to reset the Markov chains after the parameter updates. This is the idea of Persistent Contrastive Divergence (PCD) [58]: The Markov chain is initialized at the state in which it ended for the previous model.

3.4 Model Sampling

In the discussion of the derivation of the *ML* objective, we noted that except for trivial cases we can not compute the contrastive term $-\langle \nabla_\theta R(y,\theta)\rangle_{y\sim p_\theta}$ analytically, as it requires to evaluate the partition function $\int_{\mathbb{R}^d} \exp\left(-R(\nu,\theta)\right) \mathrm{d}\nu$. As such, it is often approximated by sampling the model distribution p_θ using *MCMC*. We also reasoned why we may consider non-convergent Markov chains, by showing intuitively how a non-convergent objective leads to a solution that is close to the true *ML* solution. Here, we want to discuss the specific sampling strategy that we will use throughout this work.

3.4.1 Stochastic Gradient Langevin Dynamics

SGLD [61] makes use of the gradient of the *PDF* to draw samples from it. This improves mixing time when compared to traditional *MCMC* methods such as Gibbs sampling, *HMC* or Metropolis-Hastings (MH) sampling. Let $p_\theta(y)$ be the Gibbs-Boltzmann distribution of the regularizer $R(y,\theta)$. We use the gradient $\nabla_y R(y,\theta)$ to perform

$$y^k = y^{k-1} - \frac{\epsilon^k}{2}\nabla_y R(y^{k-1},\theta) + \nu^k, \tag{3.25}$$

where $\nu^k \sim \mathcal{N}(0,\epsilon^k)$ and ϵ^k are step sizes satisfying

$$\sum_{k=1}^{\infty} \epsilon^k = \infty, \quad \text{and} \quad \sum_{k=1}^{\infty} \left(\epsilon^k\right)^2 < \infty\,. \tag{3.26}$$

The above walk defines the distribution $q_\theta(y)$, such that $y^k \sim q_\theta(y)$. *SGLD* relies on the fact that $q_\theta(y) \to p_\theta(y)$ as $k \to \infty$, i.e. the procedure generates samples from $p_\theta(y)$.

3.5 Is the *ML* Solution the Best in General?

ML is a nice mathematical framework for parameter estimation. The objective is derived by fitting the model to the data such that the likelihood of the data under the model is maximized. As such, one would intuitively find no reason to deviate from the *ML* objective. However, here we want to illustrate that the *ML* objective is not necessarily always optimal for fitting a model to data. We do this with a simple example taken from [27]:

Let $Z \sim \mathcal{U}[0,1]$, and let p_D be the distribution of $(0, Z) \in [0,1]^2$. Now, let $p_\theta(\theta, z)$ be the distribution of $g_\theta(\theta, z) = (\theta, z)$, $z \sim \mathcal{U}[0,1]$. Recall that the *ML* objective also arises when minimizing the *KLD* and recall that the *KLD* between the data distribution p_D and the distribution p_θ is

$$\mathrm{KL}(p_D \parallel p_\theta) = \int_{[0,1]^2} p_D(\nu) \log \frac{p_D(\nu)}{p_\theta(\nu)} \, \mathrm{d}\nu = \begin{cases} \text{undefined} & \text{if } \theta \neq 0, \\ 0 & \text{else.} \end{cases} \tag{3.27}$$

Clearly, the *KLD* is undefined if there exist points γ where $p_\theta(\gamma) = 0$ while $p_D(\gamma) > 0$ or vice versa, and as such does not provide a gradient for training. Hence, the *KLD* and the *ML* objective are not necessarily optimal for all circumstances and we may explore other objectives.

We have seen how *ML* introduced the contrastive term $-\langle \nabla_\theta R(y, \theta) \rangle_{y \sim p_\theta}$ into the energy loss function Eq. (3.6) to keep the model from collapsing to a trivial solution $R(y, \theta) = \text{const}\ \forall y$. This is the expected gradient of the regularizer $R(\cdot, \theta)$ with respect to samples drawn from its corresponding Gibbs-Boltzmann distribution p_θ. In this work, we want to explore other contrastive terms, where we do not compute the expected gradient of $R(\cdot, \theta)$ under its own distribution, but under a *loss-augmented* distribution $\bar{p}_\theta$. Specifically, we motivate the construction of $\bar{p}_\theta$ with the concept of loss augmented inference used in Structured Support Vector Machine (SSVM) training.

4 Maximum Margin Classifiers

Contents

4.1 Support Vector Machines

In this section, we will give a very brief overview of the principles of the classical Support Vector Machine (SVM) [5], to outline the idea behind maximum margin classifiers. The extension of the *SVM* to structured output spaces is discussed in more detail in Section 4.2.

The *SVM* is an instance of a linear two-class classifier of the form

$$\hat{y}(x) = \theta^\top \Psi(x) \tag{4.1}$$

where $\theta \in \Theta$ are the parameters of the linear model and $\Psi\colon \mathbb{R}^m \to \Theta$ is a fixed feature transform. For training the model, we assume access to a data set $\mathcal{S} = \{(x_i, t_i)\}_{i=1}^{N_S}$ where $x_i \in \mathbb{R}^m$ are the input vectors and $t_i \in \{-1, 1\}$ code the corresponding target classes. New data points x are then classified according to the sign of $\hat{y}$, i.e.

$$\hat{t} = \operatorname{sign}(\hat{y}(x)) = \operatorname{sign}(\theta^\top \Psi(x))\,. \tag{4.2}$$

If we assume that the probability density functions of the two classes do not overlap, there are (infinitely) many choices for the parameter vector θ such that all training points $\{x_i\}_{i=1}^{N_S}$ are correctly classified. In this family of possible choices, we should seek the solution which yields the smallest generalization error. A particularly fruitful idea which identifies a unique optimal solution is that of the *maximum margin* [56, 57]. The margin is defined as the distance between the decision boundary and the closest data point. For

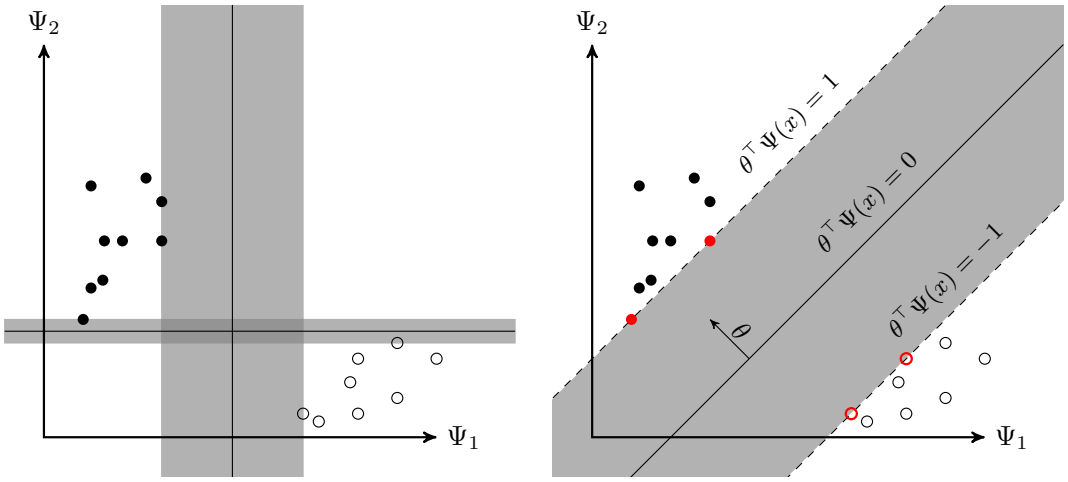


Figure 4.1: Illustration of the *SVM* solution of a classification task of two classes with non-overlapping distributions. Of the infinite possible θ such that all points are correctly classified, we show two examples on the left. The right plot shows the maximum margin solution of the *SVM*. The solution is defined by a small subset of the initial training set $\{x_i\}_{i=1}^{N_S}$, the *support vectors*, shown in red. The shaded areas indicate the margins of the corresponding solutions.

any data point x_i the distance to the separation hyperplane is given by

$$d_i = \frac{t_i(\theta^\top \Psi(x_i))}{\|\theta\|_2} \,. \tag{4.3}$$

Using this, we can write the optimization problem as

$$\arg\max_{\theta} \left\{ \frac{1}{\|\theta\|_2} \min_{(t_i,x_i)\in\mathcal{S}} \left\{ t_i(\theta^\top \Psi(x_i)) \right\} \right\} \tag{4.4}$$

which maximizes the minimum distance, i.e. the margin between the separation hyperplane and the closest data point. The *SVM* objective is usually written in a different form as

$$\begin{cases} \arg\min_{\theta} \frac{1}{2} \|\theta\|_2^2 \\ \text{s.t. } t_i(\theta^\top \Psi(x_i)) \geq 1 \quad \forall\ i = 1, \ldots, N_S \end{cases} \tag{4.5}$$

where we use one degree of freedom to rescale the margin to unity and note the equivalence $\arg\max_\theta \frac{1}{\|\theta\|_2} = \arg\min_\theta \|\theta\|_2^2$. An example of a 2-class classification problem is shown in Fig. 4.1 where the concept of margin maximization is nicely seen.

The formulation above does not allow for overlapping class distributions. This approach can be modified such that points are allowed to be inside the margin, where the distance to the margin is penalized linearly [14]. This is often referred to as a *soft-margin*

SVM. To this end, we introduce slack variables

$$\xi_i = \begin{cases} 0 & \text{if } x_i \text{ not inside the margin,} \\ |t_i - \hat{y}(x_i)| & \text{else.} \end{cases} \tag{4.6}$$

The corresponding optimization problem reads

$$\begin{cases} \arg\min\limits_{\theta,\xi_i} \dfrac{1}{2}\|\theta\|_2^2 + C\sum\limits_{i=1}^{N_S}\xi_i \\ \text{s.t.} \left\{ \begin{matrix} t_i(\theta^\top \Psi(x_i)) \geq 1 - \xi_i \\ \xi_i \geq 0 \end{matrix} \right\} \forall\ i = 1, \ldots, N_S, \end{cases} \tag{4.7}$$

where $C \geq 0$ controls the trade-off between the margin violation penalty and the margin maximization objective. Note that the *SVM* can be extended to a multi-class classifier by employing a one-vs-one or one-vs-all scheme [42, 48, 60].

4.2 Structured Support Vector Machines

Structured Support Vector Machines (SSVMs) [59] are an extension of the popular *SVMs* to structured output spaces. The goal is to learn a map from inputs $x \in \mathcal{X}$ to discrete outputs $y \in \mathcal{Y}$ based on a training data set $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^{N_S}$ drawn from a fixed but unknown probability distribution. $\mathcal{Y}$ is a structured output space, with interdependent variables, such as trees, graphs, or sequences.

Note that the naive extension of multi-class *SVMs* to structured output spaces would be to treat each possible element in $\mathcal{Y}$ as a class. This is in general intractable, as $\mathcal{Y}$ grows exponentially with the length of its members, and as such the number of classes can easily become too large or infinite.

4.2.1 Loss and Discriminant Function

In SSVM training one is interested in learning a discriminant function $F\colon \mathcal{X} \times \mathcal{Y} \times \Theta \to \mathbb{R}$, $(x, y, \theta) \mapsto F(x, y, \theta)$ from input/target pairs. Inference is then performed by choosing the $\hat{y} \in \mathcal{Y}$ which maximizes F, i.e.

$$\hat{y} = f(x, \theta) = \arg\max_{y \in \mathcal{Y}} F(x, y, \theta)\,, \tag{4.8}$$

where θ is a parameter vector. Here, F is a θ-parametrized family of cost functions, where θ is the weight vector of the *combined feature space* $\Psi(x, y)$, such that

$$F(x, y, \theta) = \theta^\top \Psi(x, y). \tag{4.9}$$

Clearly, an *SSVM* is an instance of an Energy Based Model (EBM), where $F(x, y, \theta)$ is akin to a negative energy.

Since elements in the structured space $\mathcal{Y}$ are interdependent, we require the loss function to mirror this interdependence. To this end, we introduce the loss function $\Delta\colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, such that $\Delta(y, \hat{y})$ measures the dissimilarity between the prediction $\hat{y}$ and the target y. We require the loss function to be non-negative and zero only if the prediction is the target, i.e. $\Delta(y, y') > 0 \; \forall\, y' \in \mathcal{Y}, y \neq y'$ and $\Delta(y, y) = 0$. If our data is distributed according to the data distribution $p_{\text{data}}(x, y)$, the goal is to minimize the risk

$$\mathcal{R}^{\Delta}_{p_{\text{data}}}(f) = \int_{\mathcal{X}\times\mathcal{Y}} \Delta(y, f(x, \theta)) p_{\text{data}}(x, y)\, \mathrm{d}x\, \mathrm{d}y. \tag{4.10}$$

Assuming p_{data} is unknown, and a data set $\mathcal{S} = \{(x_i, y_i) \sim p_{\text{data}}\}_{i=1}^{N_S}$ is given, the performance of $f(x, \theta)$ is given by

$$\mathcal{R}^{\Delta}_{\mathcal{S}}(f) = \frac{1}{N_S} \sum_{i=1}^{N_S} \Delta(y_i, f(x_i, \theta)). \tag{4.11}$$

4.2.2 Margins and Margin Maximization

Assuming that there exists a function $f(x, \theta)$ such that there is zero risk over the training set $\mathcal{S}$, we can enumerate the constraints as

$$\max_{y \in \mathcal{Y}^{-i}} \left\{ \theta^\top \Psi(x_i, y) \right\} < \theta^\top \Psi(x_i, y_i) \quad \forall\; i = 1, \ldots, N_S\,, \tag{4.12}$$

where $\mathcal{Y}^{-i} \coloneqq \mathcal{Y} \setminus \{y_i\}$. We observe that this can be rewritten as linear constraints by introducing separate constraints for each y_i as

$$\theta^\top \delta\Psi_i(y) > 0 \quad \forall\; i = 1, \ldots, N_S,\; \forall\; y \in \mathcal{Y}^{-i}\,, \tag{4.13}$$

where $\delta\Psi_i(y) = \Psi(x_i, y_i) - \Psi(x_i, y)$. In general, there are multiple θ such that Eq. (4.13) is satisfied. To define a unique solution, analogous to classification in an *SVM* setting, we require $\|\theta\|_2 \leq 1$ and choose θ such that the score difference from the correct label y_i to the *most offending* label $\tilde{y} = \arg\max_{y \in \mathcal{Y}^{-i}} F(x_i, y, \theta)$ is maximized. Therefore, we arrive at the following optimization problem:

$$\begin{cases} \arg\min\limits_{\theta} \dfrac{1}{2} \|\theta\|_2^2 \\ \text{s.t. } \theta^\top \delta\Psi_i(y) \geq 1 \quad \forall\; i = 1, \ldots, N_S,\; \forall\; y \in \mathcal{Y}^{-i}\,. \end{cases} \tag{4.14}$$

To allow for margin violation, similar to the soft-margin *SVM*, a slack variable ξ_i is

introduced for every non-linear constraint. The corresponding optimization problem reads

$$\begin{cases} \underset{\theta,\xi_i}{\arg\min} \dfrac{1}{2}\left\|\theta\right\|_2^2 + \dfrac{C}{N_S}\displaystyle\sum_{i=1}^{N_S}\xi_i \\ \text{s.t.} \begin{cases} \theta^\top \delta\Psi_i(y) \geq 1 - \xi_i & \forall\ y \in \mathcal{Y}^{-i} \\ \xi_i \geq 0 \end{cases} \forall\ i = 1, \ldots, N_S, \end{cases} \tag{4.15}$$

which implicitly uses the zero-one classification loss

$$\Delta_{01}(y, \hat{y}) = \begin{cases} 0 & \text{if } y = \hat{y}, \\ 1 & \text{else.} \end{cases} \tag{4.16}$$

This is undesirable, since margin constraints should be weighted according to their severity as defined by the loss Δ. In the literature [59], there are two approaches to incorporate the user-defined loss function Δ in the optimization problem, which we define in the following sections.

4.2.2.1 Slack Rescaling

In Slack Rescaling, the margin is fixed at unity, and the slack variables are scaled with the inverse of their losses. This leads to the *slack rescaling* formulation

$$\begin{cases} \underset{\theta,\xi_i}{\arg\min} \dfrac{1}{2}\left\|\theta\right\|_2^2 + \dfrac{C}{N_S}\displaystyle\sum_{i=1}^{N_S}\xi_i\,, \\ \text{s.t.} \begin{cases} \theta^\top \delta\Psi_i(y) \geq 1 - \dfrac{\xi_i}{\Delta(y_i, y)}\,, \forall\ y \in \mathcal{Y}^{-i} \\ \xi_i \geq 0 \end{cases} \forall\ i = 1, \ldots, N_S. \end{cases} \tag{4.17}$$

4.2.2.2 Margin Rescaling

Here, the margin is no longer fixed at unity but we require the score difference between y_i and the most offending runner-up $\tilde{y} = \arg\max_{y \in \mathcal{Y}^{-1}} F(x, y, \theta)$ to be at least $\Delta(y_i, \tilde{y})$. This modifies Eq. (4.15) to

$$\begin{cases} \underset{\theta,\xi}{\arg\min} \dfrac{1}{2}\left\|\theta\right\|_2^2 + \dfrac{C}{N_S}\displaystyle\sum_{i=1}^{N_S}\xi_i\,, \\ \text{s.t.} \begin{cases} \theta^\top \delta\Psi_i(y) \geq \Delta(y_i, y) - \xi_i\,, \forall\ y \in \mathcal{Y}^{-i} \\ \xi_i \geq 0 \end{cases} \forall\ i = 1, \ldots, N_S. \end{cases} \tag{4.18}$$

4.2.3 Training

In general, solving Eq. (4.17) or Eq. (4.18) is infeasible, since there exists a constraint for every $y \in \mathcal{Y}^{-i}$, for every $i = 1, \ldots, N_S$. However, by iteratively considering only the most

Algorithm 1: Cutting Plane Algorithm for *SSVM* training.

Input : $\mathcal{S},\ C,\ \epsilon$
choose: SR: $H_i(y) = \Delta(y_i, y)(1 - \theta^\top \delta\Psi_i(y))$
MR: $H_i(y) = \Delta(y_i, y) - \theta^\top \delta\Psi_i(y)$
1 $\mathcal{S}_i \leftarrow \emptyset \quad \forall i = 1, \ldots, N$
2 **while** $\mathcal{S}_i$ *changing* **do**
3 **for** $i = 1, \ldots, N$ **do**
4 $\bar{y} = \arg\max_{y \in \mathcal{Y}^{-i}} H_i(y)$
5 $\xi_i = \max\{0, \max_{y \in S_i} H_i(y)\}$
6 **if** $H_i(\bar{y}) > \xi_i + \epsilon$ **then**
7 $\mathcal{S}_i \leftarrow \mathcal{S}_i \cup \{\bar{y}\}$
8 $\theta \leftarrow$ optimize Eq. (4.17) (SR) or Eq. (4.18) (MR) over $\bigcup_i \mathcal{S}_i$
9 **end**
10 **end**
11 **end**

violating constraints, the problem can be solved in polynomial time. The Cutting Plane algorithm [31] to find solutions for Eq. (4.17) and Eq. (4.18) is shown in Algorithm 1. The algorithms only differ in the cost function $H(y)$.

We note that when using the margin rescaling formulation, Line 4 can be expanded to

$$\bar{y} = \arg\max_{y \in \mathcal{Y}^{-i}} H_i(y) = \arg\max_{y \in \mathcal{Y}^{-i}} \{\Delta(y_i, y) + F(x_i, y, \theta)\}, \tag{4.19}$$

which adds the loss term $\Delta(y_i, y)$ to the inference equation (Eq. (4.8)). As such, this step is referred to as *loss-augmented inference*. In the next section we will combine the concept of loss-augmented inference with Maximum Likelihood (ML) to derive a generative training algorithm.

5 Loss Augmented Inference for Image Restoration

Contents

We now introduce the concept of loss augmented inference into the Maximum Likelihood (ML) framework. Recall that the *ML* loss given a point $y_i \sim \mathcal{S}$ is

$$\mathcal{L}_{\mathrm{ML}}(y_i, \theta) = R(y_i, \theta) - \Big\langle R(y, \theta) \Big\rangle_{y \sim p_\theta} \tag{5.1}$$

where p_θ is the Gibbs-Boltzmann distribution of the regularizer $R(\cdot, \theta)$. Inspired by Structured Support Vector Machine (SSVM) training, we define the loss-augmented loss as

$$\mathcal{L}_{\mathrm{LA}}(y_i, \theta) = R(y_i, \theta) - \arg\min_y \bar{R}(y, y_i, \theta) \tag{5.2}$$

with the loss-augmented regularizer $\bar{R}(y, y_i, \theta)$. In what follows, we consider the euclidean distance loss-augmentation

$$\Delta(y_i, y) = \frac{1}{2} \|y_i - y\|_2^2 , \tag{5.3}$$

such that the loss augmented energy regularizer is given as

$$\bar{R}(y, y_i, \theta) = R(y, \theta) - \frac{\mu}{2} \|y_i - y\|_2^2 \tag{5.4}$$

where $\mu \in \mathbb{R}^+$ is a freely selectable parameter controlling the influence of $\Delta(y_i, y)$. The

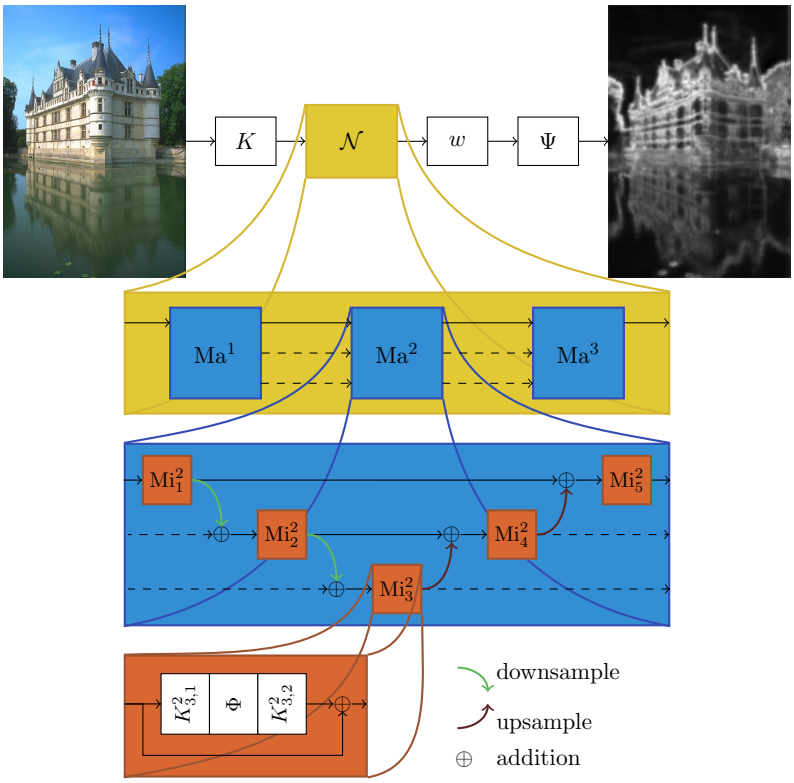


Figure 5.1: Schematic of the multi-scale *TDV* [33] regularizer $R(y, \theta)$ with 3 macro blocks Ma^i which each operate on three scales.

gradient of the loss augmented regularizer is then

$$\nabla_y \bar{R}(y, y_i, \theta) = \nabla_y R(y, \theta) + \mu(y - y_i). \tag{5.5}$$

Note that in practice, we also sample our loss-augmented regularizer, i.e. we modify the loss to be

$$\mathcal{L}_{\mathrm{LA}}(y_i, \theta) = R(y_i, \theta) - \Big\langle R(y, \theta) \Big\rangle_{y \sim \bar{p}_\theta} \tag{5.6}$$

where $\bar{p}_\theta$ is the Gibbs-Boltzmann distribution of the loss-augmented regularizer.

We consider the regularizer $R(y, \theta) \equiv \mathit{TDV}_{N_{\mathrm{Ma}}}^{N_{\mathrm{Sc}}}$ [33], where N_{Ma} is the number of macro blocks Ma^i and N_{Sc} is the number of scales the regularizer operates on. Fig. 5.1 shows a sketch of the TDV_3^3 network, where we schematically show the macro blocks and scales.

5.1 Training and Inference

Let $\mathcal{S} = \{y_i\}_{i=1}^{N_S}$, $y_i \sim p_D$ be a set of N_S ground truth images $y_i \in \mathbb{R}^n$. We empirically estimate $\langle R(y,\theta)\rangle_{y\sim p_D}$ as

$$\big\langle R(y_i,\theta)\big\rangle_{y_i\sim p_D} \approx \frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}} R(y_i,\theta) \tag{5.7}$$

and similarly

$$\big\langle \|y_i - y\|_2^2\big\rangle_{y_i\sim p_D} \approx \frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}} \|y_i - y\|_2^2 \tag{5.8}$$

where $\mathcal{I}$ is a set of indices drawn from $\{1,\dots,N_S\}$ of cardinality $|\mathcal{I}| = b$. Given the batch indices $\mathcal{I}$, we estimate $\langle R(y,\theta)\rangle_{y\sim\bar{p}_\theta}$ as

$$\big\langle R(y,\theta)\big\rangle_{y\sim\bar{p}_\theta} \approx \frac{1}{N_{\mathrm{LA}}|\mathcal{I}|}\sum_{j=1}^{N_{\mathrm{LA}}}\sum_{i\in\mathcal{I}} R(\bar{y}_i^j,\theta)\,, \tag{5.9}$$

where each $\bar{y}_i^j$ is drawn from $\bar{p}_\theta$ using Stochastic Gradient Langevin Dynamics (SGLD), i.e. we update

$$\bar{y}_i^{j,k+1} = \bar{y}_i^{j,k} - \frac{\epsilon_t}{2}\nabla_{\bar{y}}\bar{R}(\bar{y}_i^{j,k}, y_i, \theta) + \nu^k\,, \tag{5.10}$$

with $\bar{y}_i^{j,0} = y_i$, $\nu^k \sim \mathcal{N}(0,\epsilon^k I_n)$, and ϵ^k is the step size at step k. Conceptually, for each $y^i\,, i \in \mathcal{I}$ we draw N_{LA} random samples $\{\bar{y}_i^j\}_{j=1}^{N_{\mathrm{LA}}}$ from $\bar{p}_\theta$ using Eq. (5.10), each initialized with y^i. The training algorithm is summarized in Algorithm 2.

For all the following experiments, we chose a learning rate of 4×10^{-4} for the Adam optimizer and set the momentum terms to $\beta_0 = 0.9$ and $\beta_1 = 0.999$. If not stated otherwise, $N_{\mathrm{LA}} = 1$. To ensure boundedness from below of the regularizer, the potential function $\Psi\colon \mathbb{R}^n \to \mathbb{R}^n, (x_1,\dots,x_n) \mapsto (\psi(x_1),\dots,\psi(x_n))$ of the *TDV* network is the softplus

$$\psi(x) = \frac{1}{\beta}\log(1+\exp(\beta x)), \tag{5.11}$$

where we chose $\beta = 1$. The micro blocks are activated by $\Phi\colon \mathbb{R}^{nN_f} \to \mathbb{R}^{nN_f}, (x_1,\dots,x_{nN_f}) \mapsto (\phi(x_1),\dots,\phi(x_{nN_f}))$, where the component-wise activation is the log-student-t function

$$\phi(x) = \frac{1}{2}\log(1+x^2), \tag{5.12}$$

with $N_f = 32$ features per kernel. The *SGLD* procedure is performed for $K = 120$ steps and we set $M = 20$, i.e. we calculate the mean over the last 20 samples to get the final sample.

Once the parameters θ of the regularizer are learned by the procedure described above,

Algorithm 2: Generative training of a data driven regularizer using the proposed learning framework, motivated by *ML* and *SSVM* training.

Input : Data set $\mathcal{S} = \{y_i\}_{i=1}^{N_s}$, batch size b, initial parameters θ, number of loss augmented samples N_{LA}, sampling steps K and number of averaging steps M

while θ *not converged* **do**
  $\mathcal{I} = \{I_1, I_2, \ldots, I_b\} \sim \{1, \ldots, N_S\}$
  for $i \in \mathcal{I}$ **do**
    for $j = 1, \ldots, N_{LA}$ **do**
      $\bar{y}_i^{j,0} = y_i$
      for $k = 0, \ldots, K-1$ **do**
        $\bar{y}_i^{j,k+1} = \bar{y}_i^{j,k} - \frac{\epsilon_t}{2} \nabla_{\bar{y}} \bar{R}(\bar{y}_i^{j,k}, y_i, \theta) + \nu^k$
      end
    end
    $\bar{y}_i^{j} = \text{mean}(\{\bar{y}_i^{j,K-M}, \ldots, \bar{y}_i^{j,K}\})$
    $L_i(\theta) = R(y_i, \theta) - \frac{1}{N_{\text{LA}}} \sum_{j=1}^{N_{\text{LA}}} R(\bar{y}_i^{j}, \theta)$
  end
  $\theta \leftarrow \text{Adam}\left(\frac{1}{|\mathcal{I}|} \nabla_\theta \sum_{i \in \mathcal{I}} L_i(\theta)\right)$
end

inference can be performed by solving the variational problem

$$\hat{y} = \arg\min_y \left\{ E(x, y, \theta) = \lambda D(x, y) + R(y, \theta) \right\}. \tag{5.13}$$

We solve the above variational problem using a backtracked proximal gradient descent scheme. The inference procedure is summarized in Algorithm 3.

In the following inference tasks, we manually pick λ for all the considered problems individually. We note however that the optimal λ for any given task can be learned in a discriminative manner by considering the bilevel optimization problem

$$\begin{cases} \arg\min_\lambda \sum_i^{N_s} \|\hat{y}_i - y_i\|_2^2, \\ \text{s.t. } \hat{y}_i = \arg\min_y \lambda D(x, y) + R(y, \theta). \end{cases} \tag{5.14}$$

Algorithm 3: Lipschitz-backtracked proximal gradient inference algorithm given learned parameters θ, with the inner product $\langle \cdot, \cdot \rangle$.

Input : Learned parameters θ, degraded observation x, data weight λ, proximal map $\operatorname{prox}_{\lambda D(x,\cdot)}$, initial guess $\hat{y}^0$, suitable initial L

$\hat{y} \leftarrow \hat{y}^0$
while $\hat{y}$ *not converged* **do**
  $g = \nabla_{\hat{y}} R(\hat{y}, \theta)$
  while *True* **do**
    $\bar{y} = \operatorname{prox}_{\lambda D(x,\cdot)}(\hat{y} - g/L)$
    $d = \bar{y} - \hat{y}$
    if $R(\bar{y}, \theta) \leq R(\hat{y}, \theta) + \langle d, g \rangle + \frac{L}{2} \|d\|_2^2$ **then**
      $\hat{y} \leftarrow \bar{y}$
      $L \leftarrow \frac{L}{2}$
      break
    end
    else
      $L \leftarrow 2L$
    end
  end
end

5.2 MNIST

To assess the applicability of our learning scheme, we trained a TDV_3^3 regularizer using 3 macro blocks operating on 3 scales, which amounts to 332 096 trainable parameters, on the MNIST [39] data set. The relative simplicity of the MNIST data set allows to quickly see whether the regularizer can be learned at all with the proposed learning scheme.

5.2.1 Samples of Loss Augmented Inference

To understand the influence of the choice of μ during training, we consider the samples $\bar{y}_i^j$ of $\bar{p}_\theta$ as a function of training epoch. We show the samples from $\bar{p}_\theta$ for $\mu \in \{0, 0.01, 0.1, 1, 10\}$ in Fig. 5.2. The *ML* learning scheme seems to converge faster, however this observation has to be taken with care. Since the samples $\bar{y}_i^j$ are initialized with the ground truth samples y_i, one can interpret Fig. 5.2 to show that after relatively few parameter updates, there is no longer any incentive for the model to change, since $\bar{y}_i^j$ are almost indistinguishable from y_i. On the other hand, when the samples are drawn using loss augmented inference, there is much more variety in the samples for a longer period of time.

In any case, the samples indicate successful learning of certain characteristics of the data set. During the initial training phases, the regularizer does not represent any meaningful energy landscape and hence there is little to no structure in the samples. As training progresses, we can see the emergence of connected lines, and finally the handwritten digits

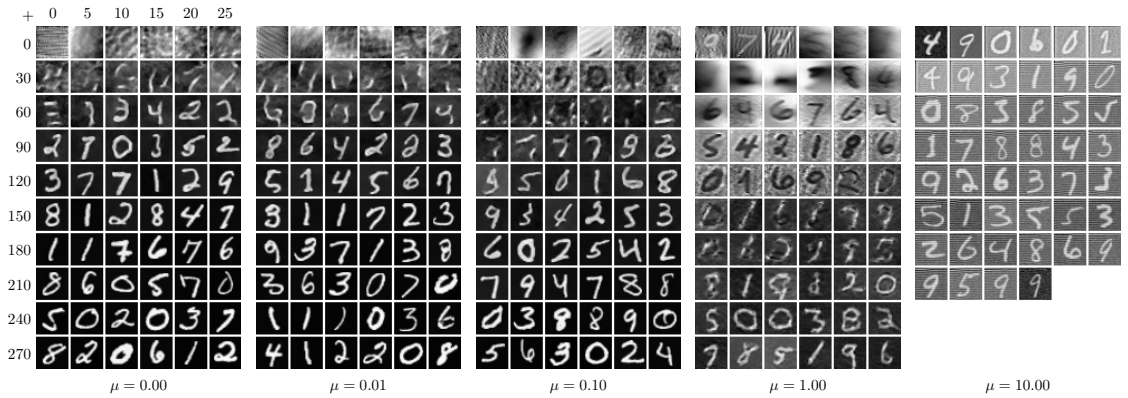


Figure 5.2: Samples from $\bar{p}_\theta$ using *SGLD* (Eq. (5.10)) for different $\mu \in \{0, 0.01, 0.1, 1, 10\}$. The axis tick labels indicate the number of parameter updates, and are shared across the different μ. For $\mu = 10$, the training was not stable and the model collapsed after approximately 200 parameter updates.

can be clearly seen. Even for $\mu = 1$, it seems as thought the model capacity of regularizer was large enough to counter the influence of $\mu(y - y_i)$ in Eq. (5.5), such that after sufficient parameter updates, the samples strongly resemble the ground truth images. On the other hand, for $\mu = 10$, we see a vertical oscillatory pattern in all the samples of the loss augmented regularizer, indicating that the model is not strong enough to counter the pull-away term $\mu(y - y_i)$. This results in unstable training, such that after approximately 200 parameter updates the training stopped.

5.2.2 Modes of the Learned Regularizer

The proposed training scheme, being generative in nature, leads to a regularizer $R(y, \theta)$ which can be interpreted as a Probability Density Function (PDF). Specifically the associated Gibbs-Boltzmann distribution is given by

$$p_\theta(y) = \frac{\exp\left(-\beta R(y, \theta)\right)}{\int_y \exp\left(-\beta R(\nu, \theta)\right) \, \mathrm{d}\nu}, \tag{5.15}$$

where $\beta \in \mathbb{R}^+$ determines the "variance" of p_θ. Explicitly calculating $p_\theta(x)$ for any given x is problematic because we can not calculate the integral in the denominator. However, we can sample $p_\theta(y)$, since the partition $\int_y \exp\left(-\beta R(\nu, \theta)\right) \, \mathrm{d}\nu$ is constant w.r.t. y. Therefor, the modes of $R(\cdot, \theta)$ coincide with those of p_θ.

We analyze the modes of the trained regularizers by applying Algorithm 3 with $\lambda = 0$ and $y_0 \sim \mathcal{U}[0, 1]^{64 \times 64}$ and show some results in Fig. 5.3. The figure shows that the nearest modes to y_0 are given by connected structures, which resembles the handwritten digits of the MNIST data set. Although this characteristic of the modes is shared across all the regularizers trained for $\mu \in \{0, 0.01, 0.1, 1\}$, there are differences between them. The *ML*

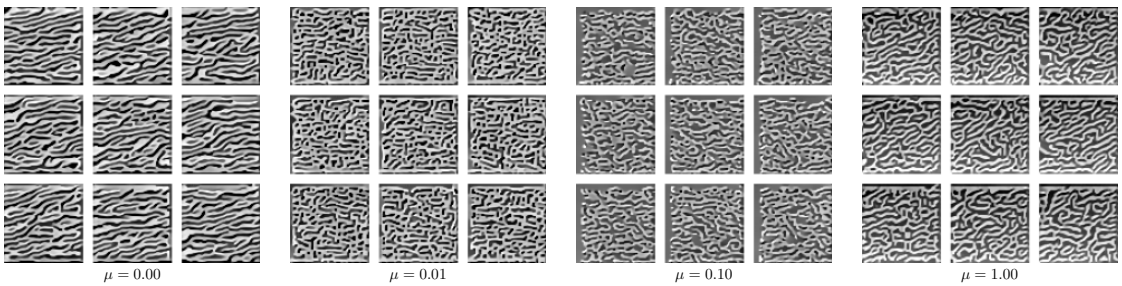


Figure 5.3: Modes of the trained regularizers for different $\mu \in \{0, 0.01, 0.1, 1\}$ for the MNIST data set. All of the regularizers prefer connected, curvy white structures, which is characteristic of the MNIST data set of handwritten digits.

($\mu = 0$) solution shows a tendency for horizontally connected lines, whereas this is not necessarily the case for the other regularizers. Furthermore, the structures are significantly coarser for $\mu = 0$.

In Fig. 5.4 we show the evolution of the modes starting from y_0. We can see that Algorithm 3 has converged at $k = 50$, such that the resulting images can really be interpreted as modes of p_θ.

5.2.3 Eigenimage Analysis

To gain more insights into the trained regularizers, we perform a nonlinear eigenimage analysis [26]. We compute the nonlinear eigenimages by considering

$$y_{\text{eig}} = \operatorname*{arg\,min}_{y \in [0,1]^n} \frac{1}{2} \left\| \nabla_y R(y, \theta) - \Lambda(y) y \right\|_2^2, \tag{5.16}$$

where $\Lambda(y)$ is the generalized Rayleigh quotient

$$\Lambda(y) = \frac{\langle \nabla_y R(y, \theta), y \rangle}{\|y\|_2^2}, \tag{5.17}$$

with the inner product $\langle \cdot, \cdot \rangle$.

We show results for $\mu \in \{0, 1\}$ in Fig. 5.5. The eigenimages indicate that the trained regularizers prefer piecewise constant regions, delineated with stark contrast. Especially in the top row it can be seen that the regularizer trained using *ML* retains smaller structures, like the hook or the letting on the body plane. On the other hand, when using $\mu = 1$ features that are retained seem to remain very detailed, like the lettering on the wings of the plane. Further, when using $\mu = 1$ edges seem to be accentuated, to the point where for instance the creases in the ground are clearly surrounded by a white border.

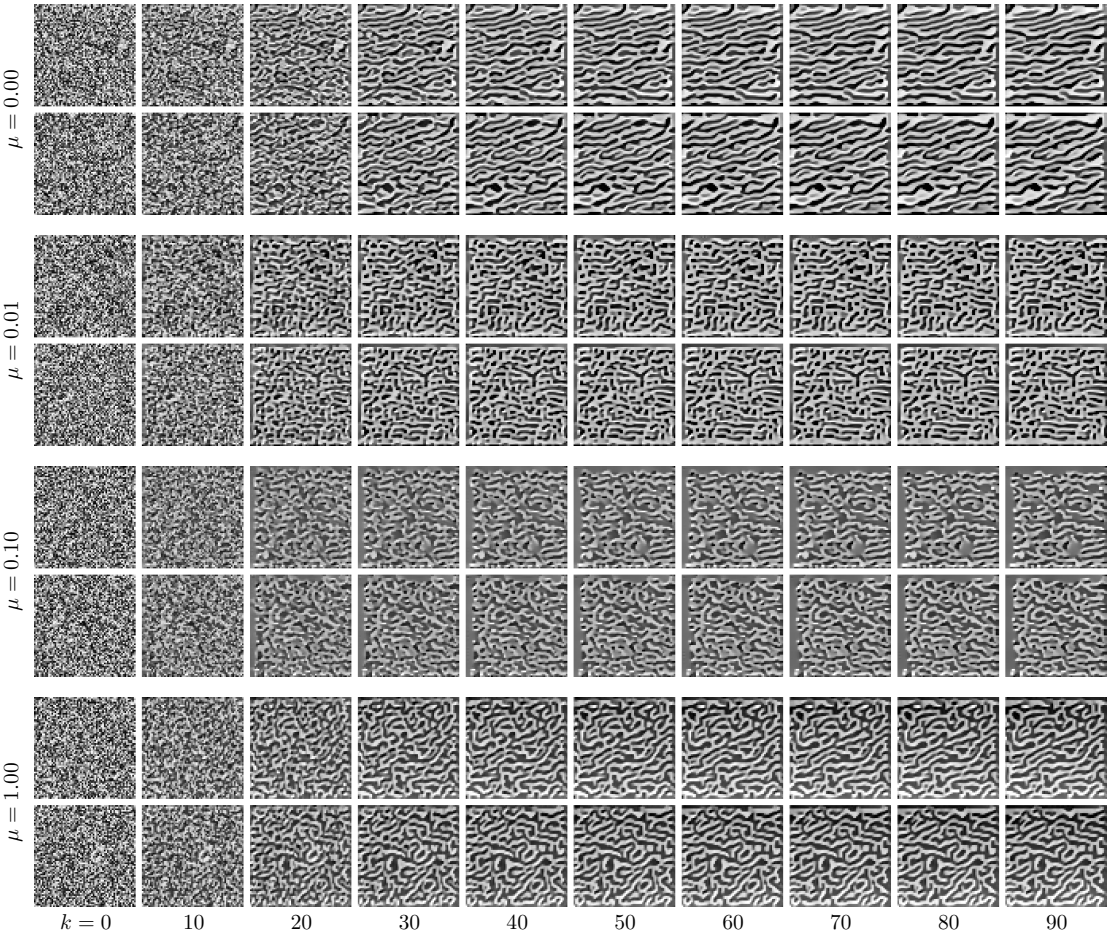


Figure 5.4: Visualization of the gradient descent on $R(\cdot, \theta)$ starting from uniform noise. k is the number of steps in Algorithm 3.

5.2.4 Regularization Strength

In the above, we have shown that the proposed learning scheme leads to reasonable solutions for a range of $\mu \in [0, 1]$. Here, we want to explore in more detail the influence of μ on the regularizer by considering the neighborhood of a training example. Specifically, let

$$y_{\epsilon,x} = y + \epsilon(x - y), \tag{5.18}$$

where $y \sim$ test set, $\epsilon \in [0, 1]$ and $x \in \mathbb{R}^n$ is drawn from either $\mathcal{N}(0.5, I_n)$ or $\mathcal{U}[0, 1]^n$. In Fig. 5.6a and Fig. 5.6b we show $R(y_{\epsilon,x}, \theta)$ trained with $\mu \in \{0, 1\}$ respectively. In Fig. 5.7 we show some examples of $y_{\epsilon,x}$ for a range of ϵ to ease interpreting Fig. 5.6.

We observe that when $\epsilon \ll$, i.e. near y, the gradient norm $\|\nabla_y R(y_{\epsilon,x}, \theta)\|_2^2$ is significantly larger. The model trained with loss augmented inference seems to have steeper slopes around the samples from the data distribution, which is in accordance with the

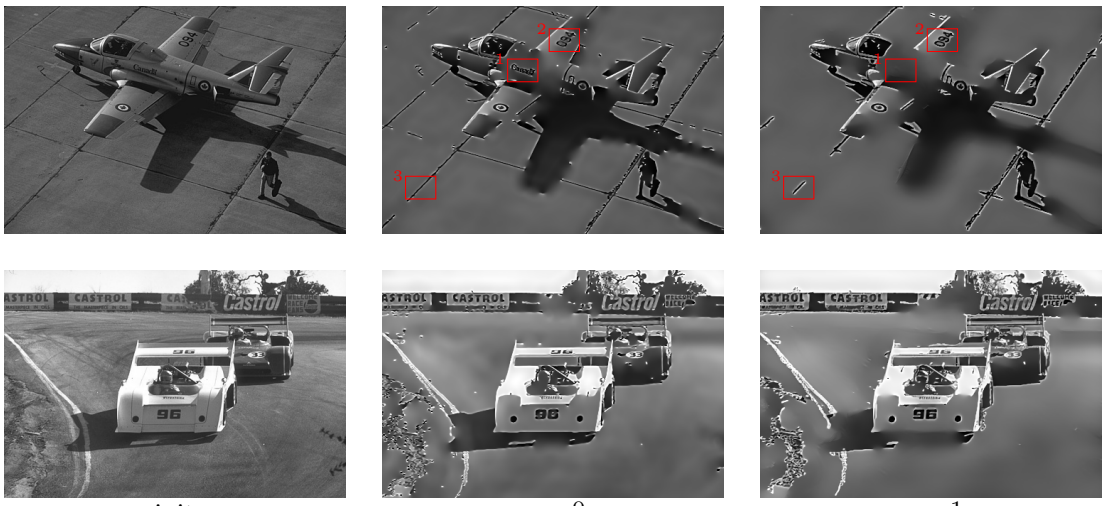
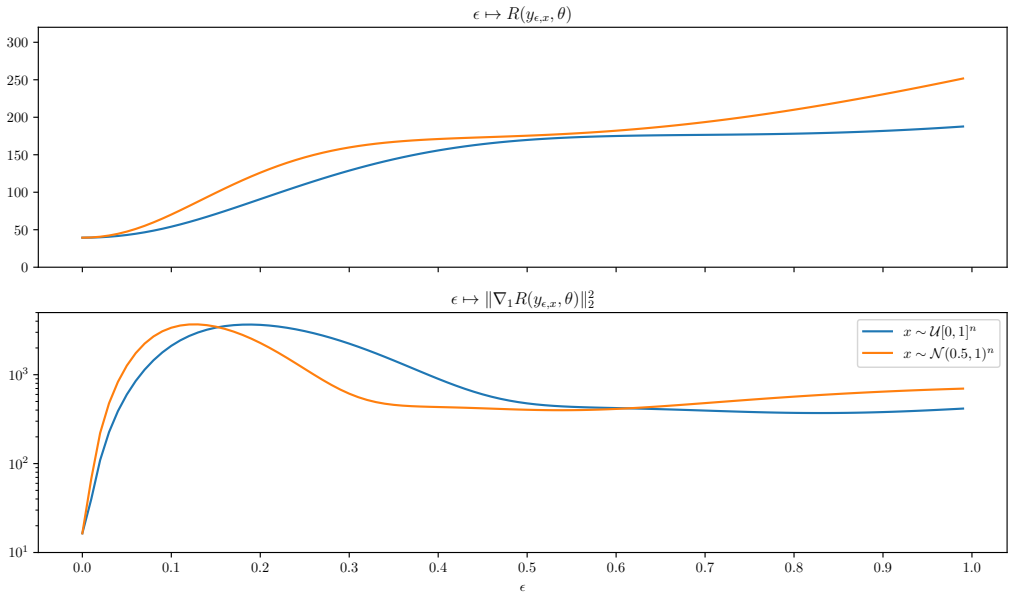


Figure 5.5: Nonlinear eigenfunction analysis of the TDV_3^3 regularizer trained on the MNIST data set using the proposed training algorithm for $\mu \in \{0, 1\}$. Highlighted are regions where there are strong differences between the regularizers: At 1 $\mu = 0$ retains details that get lost in $\mu = 1$. On the other hand, there is more detail at 2 for $\mu = 1$ and finally we highlight the stark accentuation of edges in 3.

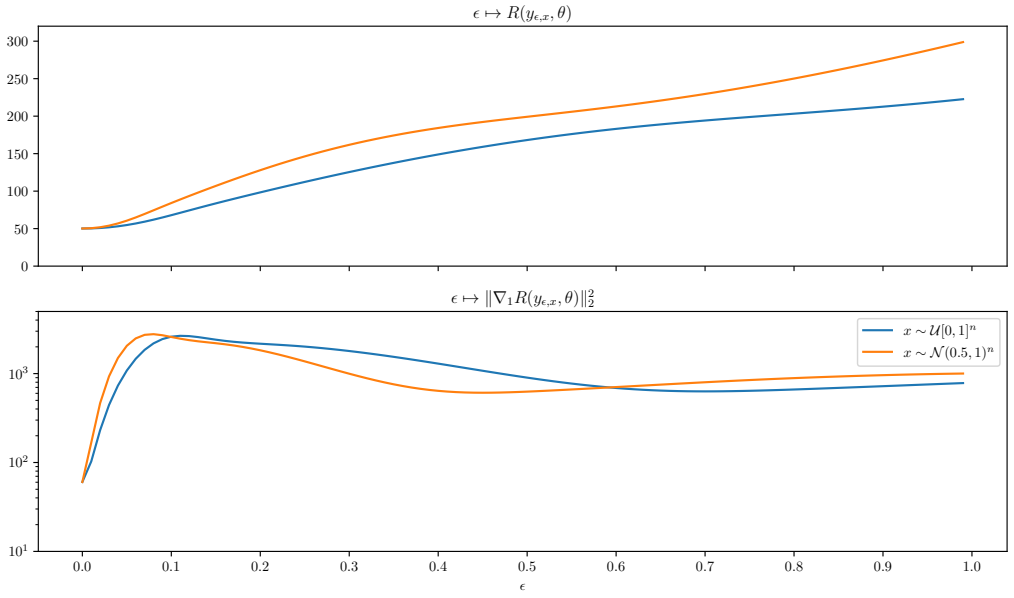
expectations, as it has to counter the influence of the loss Eq. (5.5). As such, we believe that loss augmented inference is a valuable tool in controlling the shape and slopes of the trained regularizer and may be combined with Lipschitz-penalization [27] to tune the expressiveness of a model.

5.2.5 Shape Completion

In the previous sections we have demonstrated that learning with loss augmented inference yields meaningful regularizers. Here, we want to put these regularizers to the test on concrete restoration examples. Due to the simplicity of the MNIST data set, traditional restoration tasks such as Gaussian denoising or deconvolution are not of much interest. Therefor, we consider an inpainting task with the forward degradation operator $\{0, 1\}^{n \times n} \ni A = \operatorname{diag}(\{a_1, \dots, a_n\})$, $a_i \sim \mathcal{B}(0.7), \forall i = 1, \dots, n$ with the Bernoulli distribution $\mathcal{B}(p)$ of probability p. To perform the shape completion, we perform a gradient descent on the regularizer, where after each step we clamp the known variables. In Fig. 5.8 we can see some examples of shape completion on MNIST digits. It can be seen that the trained regularizers are clearly capable of restoring the digits.



(a) $\mu = 0$



(b) $\mu = 1$

Figure 5.6: Regularization energy $R(\cdot, \theta)$ and gradient norm $\|\nabla_y R(y_{\epsilon,x}, \theta)\|_2^2$ around an image in the test set for $\mu \in \{0, 1\}$. Especially around $\epsilon = 0$, i.e. near y, $\|\nabla_y R(y_{\epsilon,x}, \theta)\|_2^2$ is significantly larger for $\mu = 1$.

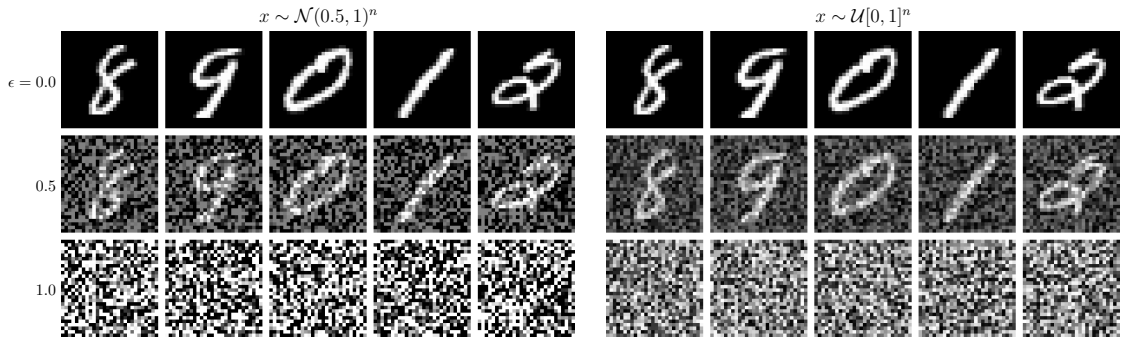


Figure 5.7: Examples of the trajectory of the images used for analyzing the regularizer in Fig. 5.6.

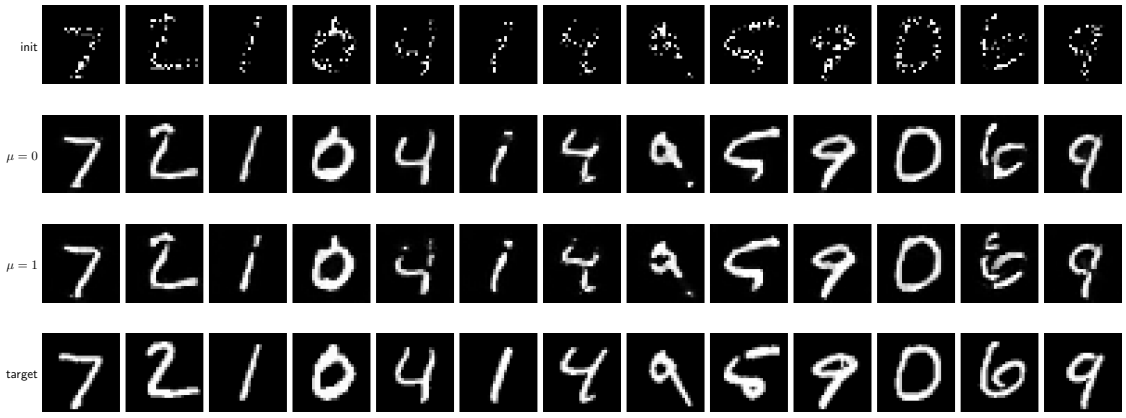


Figure 5.8: Shape completion of MNIST digits: The starting images are constructed by $x = Ay$, $A = \text{diag}(\{a_1, \ldots, a_n\})$, $a_i \sim \mathcal{B}(0.7)$. The trained regularizers are clearly capable of restoring the digits, however the *ML* regularizer does better visually.

5.3 FashionMNIST

In the previous section we showed that applying the proposed learning scheme to the *TDV* regularizer leads to meaningful solutions. The simplicity of the MNIST data set allowed us to nicely show the properties of the trained regularizers and how they relate to the training set. However, clearly the model is oversized for the task, and the data set is not interesting for traditional Image Restoration (IR) tasks.

In this section, we will scale the model capacity down and consider a more complicated data set. Specifically, we consider is the TDV_2^2 regularizer, which has 147 776 trainable parameters, which we train on the FashionMNIST [63] data set. The FashionMNIST data set is more complex than the MNIST data set, as it exhibits more structure than simple connected streaks.

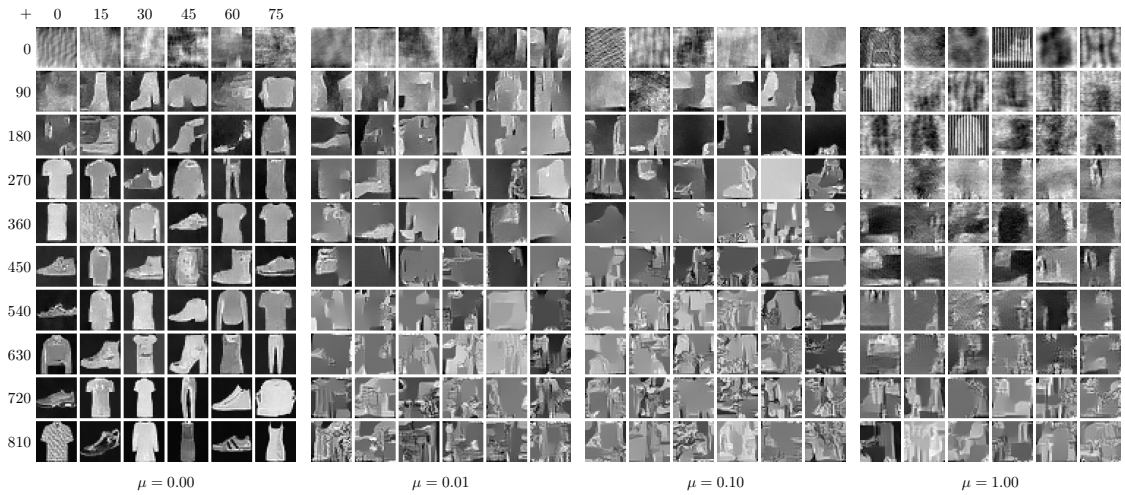


Figure 5.9: Samples from $\bar{p}_\theta$ for different $\mu \in \{0, 0.01, 0.1, 1\}$ during training. The ticks indicate the number of optimizer steps. Clearly the results show that for any $\mu > 0$, the samples are far from the initial data set. We interpret this as the model not being able to overfit to the training data, such that the pull-away term $\mu(y - y_i)$ leads to non-trivial loss augmented inference solutions.

5.3.1 Samples of Loss Augmented Inference

As in the previous section, to understand which samples the regularizer learns to discriminate, we look at the loss augmented samples $\bar{y}_i^j$. In Fig. 5.9 we see the evolution of the samples of $\bar{p}_\theta$ during training. Clearly, for $\mu > 0$, samples from $\bar{p}_\theta$ are far from any y_i during training. This seems to be because the smaller model is not able to overfit the more complex training data, such that loss augmented inference always leads to non-trivial solutions. Again we see that there is much more variety in the loss augmented samples when compared to the *ML* case.

5.3.2 Modes of the Learned Regularizers

Again we want to investigate the properties of the trained regularizers by considering the nearest modes of p_θ to uniform noise. We show $\arg\min_y R(y, \theta)$, which is solved using Algorithm 3 where we set $\lambda = 0$ and let $\hat{y}^0 \sim \mathcal{U}[0, 1]^{64 \times 64}$ for $\mu \in \{0, 0.01, 0.1, 1\}$ in Fig. 5.10.

We note that the *ML* regularizer leads to constant images, while all other regularizers clearly show characteristic features of the FashionMNIST data set, which are piecewise constant regions of certain shapes. We can also observe some differences between the regularizers trained with $\mu \in \{0.01, 0.1, 1\}$: Clearly, the contrast between the piecewise constant regions decreases with μ, to the point where the images are constant for $\mu = 0$. Similarly, the structures themselves seem to be more pronounced and resemble the features

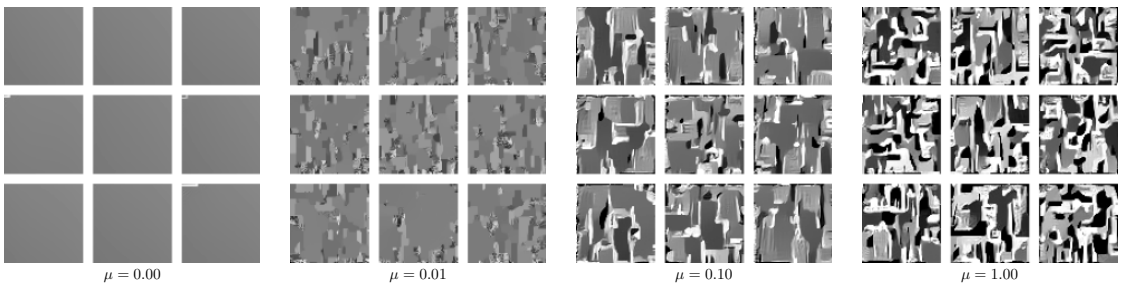


Figure 5.10: Modes of trained regularizers for different $\mu \in \{0, 0.01, 0.1, 1\}$ for the FashionMNIST data set. Clearly, the regularizer trained with $\mu = 0$ does not preserve any structure and we arrive at a constant image. The modes of the other regularizers trained with loss augmented inference show characteristics of the FashionMNIST data set, which are piecewise constant regions of a certain structure.

in the data set better. Fig. 5.11 shows how the models finally arrive at the modes. We see that we have converged to a mode, indicating that for all the models trained for $\mu > 0$ have indeed reached convergence.

5.3.3 Eigenimage Analysis

As in the previous section, we compute the eigenimages of the regularizers with Eq. (5.16), which we show in Fig. 5.12. We see that $\mu = 0$ preserves smaller details and regions of similar intensity are delineated more cleanly. On the other hand, with $\mu = 1$ the regularizer prefers coarser structures and increases contrast in the image.

5.3.4 Shape Completion

Similar to the previous section, we want to investigate the capabilities of the trained regularizers on the shape completion task. Here, we set $\{0, 1\}^{n \times n} \ni A = \operatorname{diag}(\{a_1, \ldots, a_n\})$, $a_i \sim \mathcal{B}(0.8)$, i.e. the degraded observation x is the clean observation y, where each pixel is turned off with a probability of 0.8. Clearly, both regularizers can reconstruct the images reasonably well, although the *ML* regularizer leads to results that are visually slightly better.

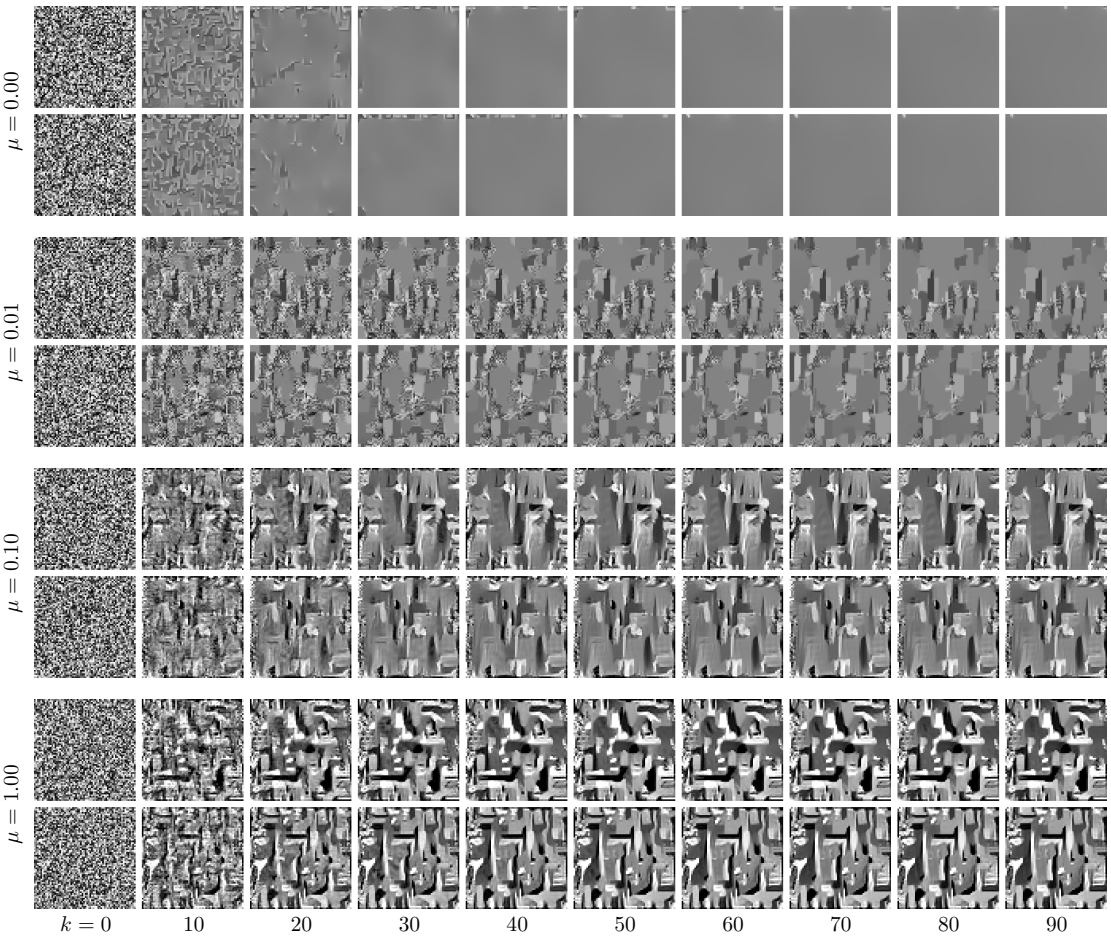


Figure 5.11: Evolution of the images starting from $\hat{y}^0 \sim \mathcal{U}[0,1]^{64\times64}$ during gradient descent on $R(\cdot,\theta)$. Clearly, we see that the gradient descent has converged at $k = 90$.

Figure 5.12: Nonlinear eigenfunction analysis of the TDV_2^2 regularizer trained on the FashionMNIST data set using the proposed training algorithm for $\mu \in \{0, 1\}$. We see that both regularizers tend to piecewise constant images, although we note that for $\mu = 1$, the regularizer introduces structures in regions where there is no stark intensity change.

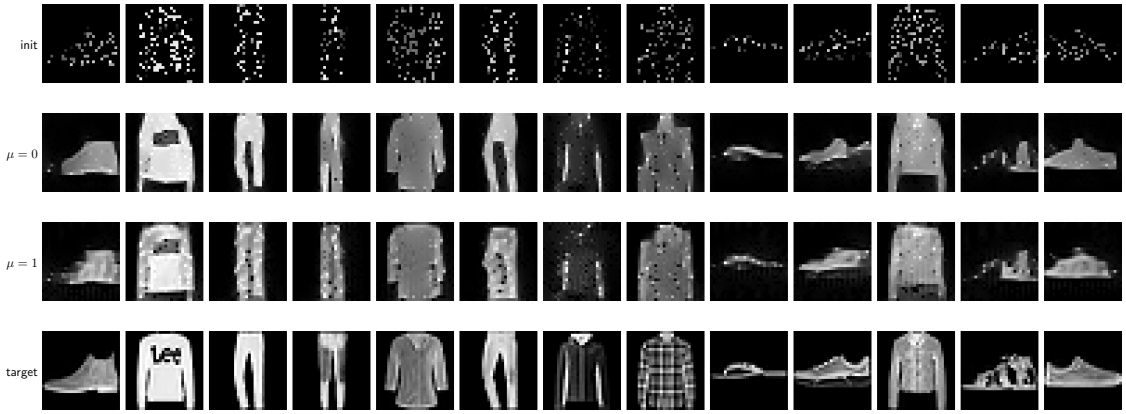


Figure 5.13: Shape completion of FashionMNIST samples: The degraded observation is $x = Ay$, $A = \operatorname{diag}(\{a_1, \dots, a_n\})$, $a_i \sim \mathcal{B}(0.8)$. Both regularizers lead to reasonable solutions, although the ML regularizers reconstruction is sometimes visually more pleasing.

5.4 Image Restoration on the BSDS500 Data Set

Although the experiments on the MNIST and FashionMNIST data set showed that the proposed learning scheme is suitable for the *TDV* regularizer, we have yet to investigate the potential in image restoration. To this end, we will learn a TDV_3^3 regularizer with the proposed learning scheme on the BSDS500 data set and show how the trained regularizers can be applied to Gaussian and Salt and Pepper denoising, as well as image inpainting, and non-blind deconvolution. We use a patch size of 100, a batch size of $\tilde{n} = 93 \times 93$, such that $n = \tilde{n}C$, $C \in \{1, 3\}$ for gray-scale and color images respectively. We augment the training data using random horizontal and vertical flipping, and by rotating the images by $\rho \in \{0, \frac{\pi}{2}, \pi, \frac{3\pi}{3}\}$rad. Since the input layers of the model differ slightly between $C = 1$ and $C = 3$, we summarize that the number of trainable parameters in the TDV_3^3 regularizer is 332 096 and 332 672 for $C \in \{1, 3\}$ respectively.

5.4.1 Samples of Loss Augmented Inference

In this section we want to put the trained regularizers to the test in practical *IR* tasks. Despite the previous results, it is still interesting to consider the samples of $\bar{p}_\theta$ during training. We show samples of the loss augmented inference procedure for $C \in \{1, 3\}$ and $\mu \in \{0, 1\}$ in Fig. 5.14. In accordance with the analysis in the previous sections, we see that for $\mu > 1$ it takes many more parameter updates until $\bar{y}_i^j$ resemble y_i, which clearly is caused by the pull-away term $\mu(y - y_i)$. This forces the regularizer to have steeper flanks around the training data points y_i.

In the next sections, we will test the trained regularizers on typical *IR* tasks and compare them to their discriminatively trained counterparts as well as the prolific Total Variation (TV) regularizer. We expect the generatively trained regularizer to perform reasonably well on all tasks, as it should have learned the distribution of the data set, and as such can be used as a generic prior. Therefor, it should outperform the simple *TV* model and discriminative models on tasks different from their training task. However, we expect the discriminatively trained regularizers to outperform our models on the tasks they were trained on.

5.4.2 Additive Gaussian Denoising

Here, we consider an additive Gaussian denoising task with variance σ^2. Specifically, the degraded observation is

$$x = y + \nu \in \mathbb{R}^n \tag{5.19}$$

with the latent clean image $y \in \mathbb{R}^n$, the additive noise $\nu \sim \mathcal{N}(0, \sigma^2 I_n)$, for $n = \tilde{n}C$, where $C = 1$ for gray-scale and $C = 3$ for color images.

For the additive Gaussian denoising task, we consider the

1. *TV* regularizer, the

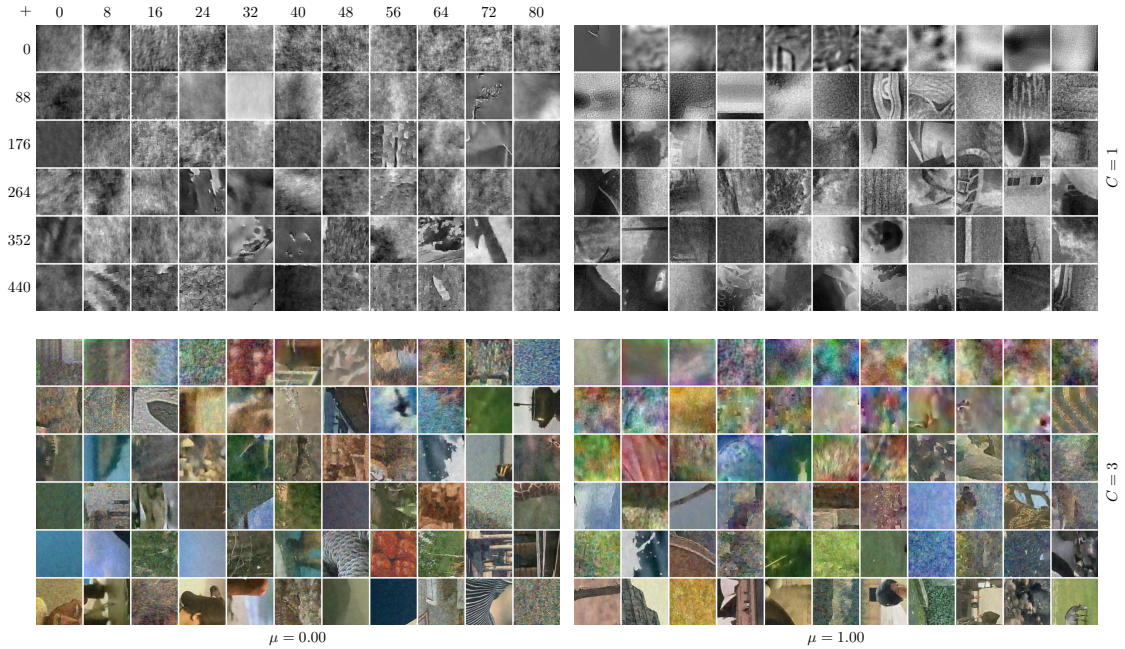


Figure 5.14: Samples from $\bar{p}_\theta$ for $\mu \in \{0, 1\}$ for regularizers trained on the BSDS500 data set. The inserts show the number of parameter updates. We see that, similarly to the analysis in the previous sections, when $\mu > 0$, it takes considerably more updates until $\bar{y}_i$ is close to y_i. Clearly, this is caused by the pull-away term $\mu(y - y_i)$, which forces the regularizer to have steeper flanks around y.

2. TDV^3_3 regularizer trained with the early-stopping framework of [33] on a Gaussian denoising task with $\sigma = 25$, and the

3. TDV^3_3 trained with Algorithm 2 for $\mu \in \{0, 1\}$.

For any of the above, we use a data term $\lambda D(x, y) = \frac{\lambda}{2} \|x - y\|_2^2$. Note that combining the TV regulrizer with a Gaussian data term correspond to the Rudin Osher Fatemi (ROF) model. The proximal map for the squared L_2 norm is computed as

$$\operatorname{prox}_{\frac{\lambda}{2}\|x-\cdot\|_2^2}(y) = \frac{y + \lambda x}{1 + \lambda}. \tag{5.20}$$

In what follows, we will denote the TDV regularizer trained in the framework of [33] as $TDV^{N_{\text{Sc}},d}_{N_{\text{Ma}}}$. For $C = 3$, we apply the ROF model channel-wise and we hand-tune the trade-off between the data term and the regularization. We expect the trained models to largely outperform the ROF model, and between the trained models we expect the dicriminatively trained model to perform better than the generative model.

We show the Peak Signal to Noise Ratio (PSNR) on the 76 landscape images of the BSDS500 validation data set for $\sigma \in \{15, 25, 50\}$ in Table 5.1. In line with our expectation, we see that $TDV^{3,d}_3$ performs best on the given task. Note that to apply $TDV^{3,d}_3$, which

Table 5.1: Comparison of expected *PSNR* values for additive Gaussian denoising for $\sigma \in \{15, 25, 50\}$ on the BSDS500 validation data set.

	σ	*ROF*	TDV_3^3		$TDV_3^{3,d}$
			$\mu = 0$	$\mu = 1$	
$C = 1$	15	29.19	30.62	30.14	31.33
	25	27.21	27.57	27.38	28.74
	50	24.09	24.20	23.04	25.11
$C = 3$	15	29.56	32.81	32.53	33.66
	25	27.09	29.54	29.27	30.67
	50	23.91	23.87	23.51	26.25

was trained on a additive Gaussian denoising task with $\sigma = 25$, we had to scale the input and output of the algorithm by $\frac{25}{\sigma}$ and $\frac{\sigma}{25}$ respectively. As such, to apply this model to an image corrupted by Gaussian noise of unknown variance, one would first have to estimate the noise variance, as without this rescaling the results are considerably worse. On the other hand, we do not need to rescale the images for the generative model, as we only have to adjust λ to the new task.

The model trained with our proposed training algorithm performs significantly better than the *ROF* model for $\sigma \in \{15, 25\}$, however interestingly the performance deteriorates for larger σ such that for $\sigma = 50$ the *ROF* model performs better. To investigate this, we show some qualitative results in Fig. 5.15. We see that although the *PSNR* values are worse for our model, there is more detail in the final solution and it lacks the typical artifacts found in the *ROF* model. However, as expected the $TDV_3^{3,d}$ model delivers the best results over the whole range of $\sigma \in \{15, 25, 50\}$. We show more qualitative examples of our regularizer applied to the additive Gaussian denoising task for $\sigma \in \{15, 25, 50\}$ in Figs. 5.16 to 5.18 respectively.

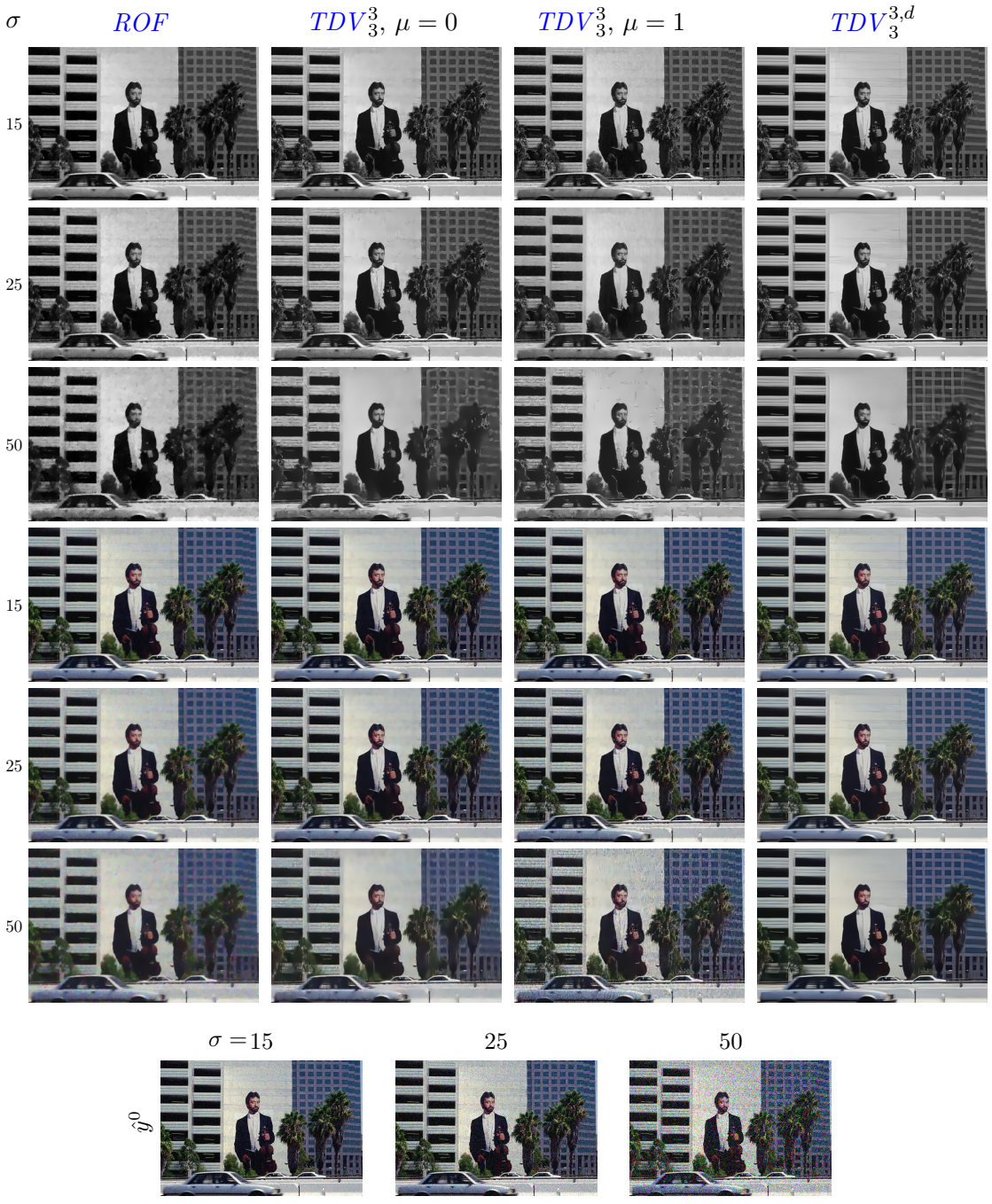


Figure 5.15: Qualitative comparison between the *ROF* model, the TDV^3_3 model trained with the proposed learning scheme for $\mu \in \{0, 1\}$ and the $TDV^{3,d}_3$ regularizer trained on an additive Gaussian denoising task with $\sigma = 25$. On the bottom we show the initial noisy images for $\sigma \in \{15, 25, 50\}$. Although for $\sigma = 50$ the proposed learning scheme is outperformed by the *ROF* model on the *PSNR* metric, we find that the solutions are able to preserve more detail and are visually superior, as they do not exhibit the typical staricasing artifacts found in the *ROF* model.

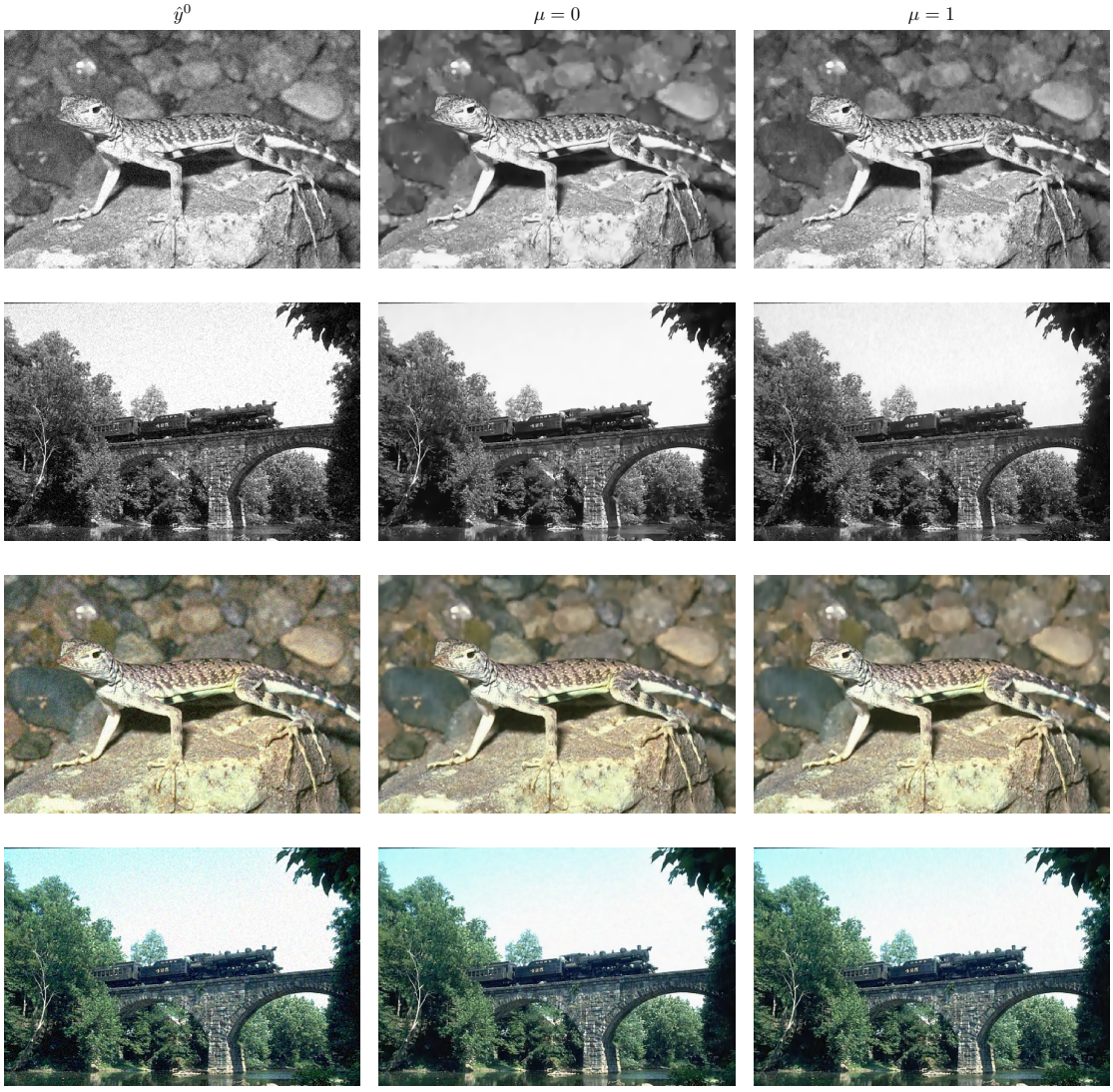


Figure 5.16: Results of additive Gaussian denoising with $\sigma = 15$.

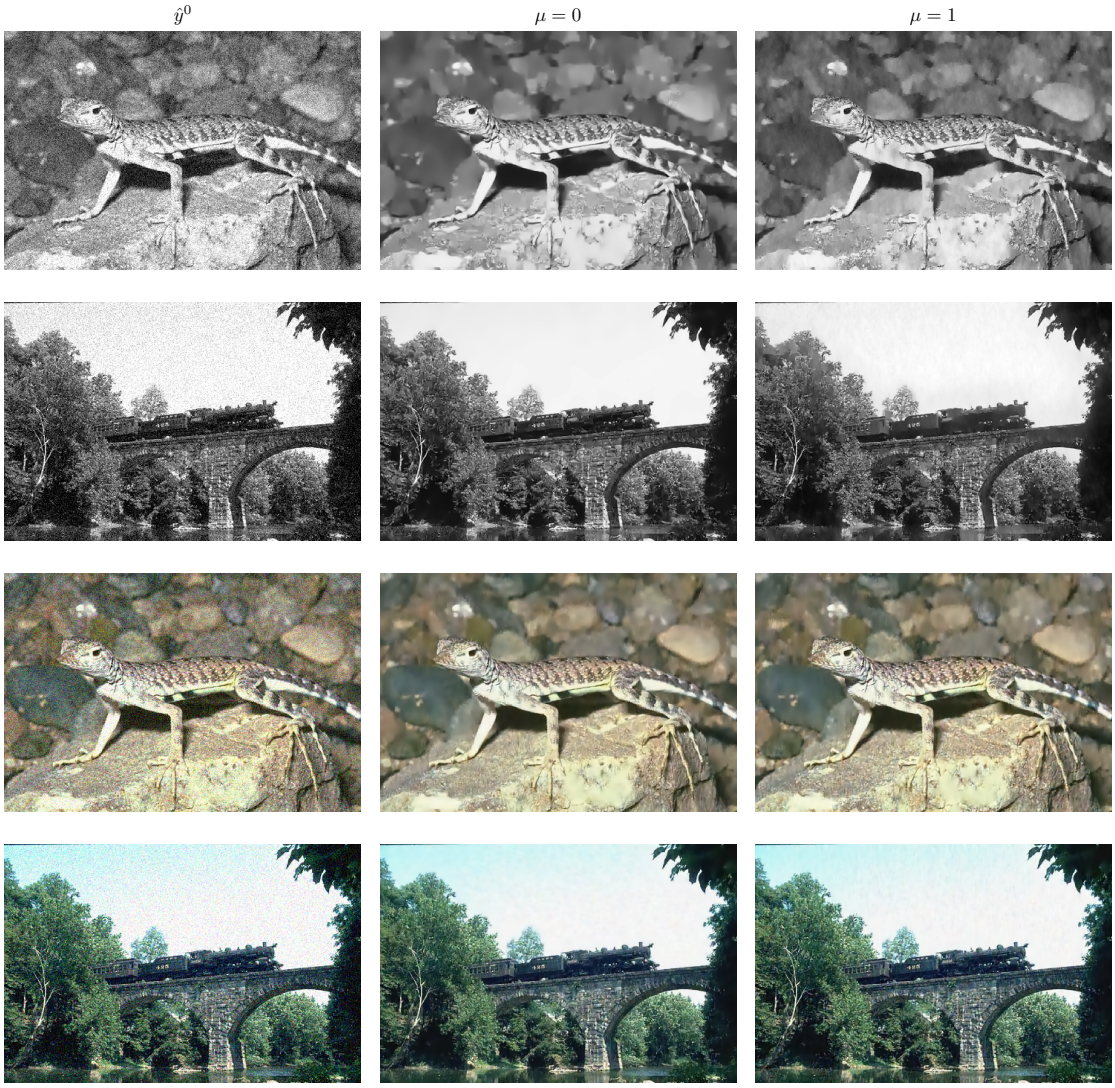


Figure 5.17: Results of additive Gaussian denoising with $\sigma = 25$.

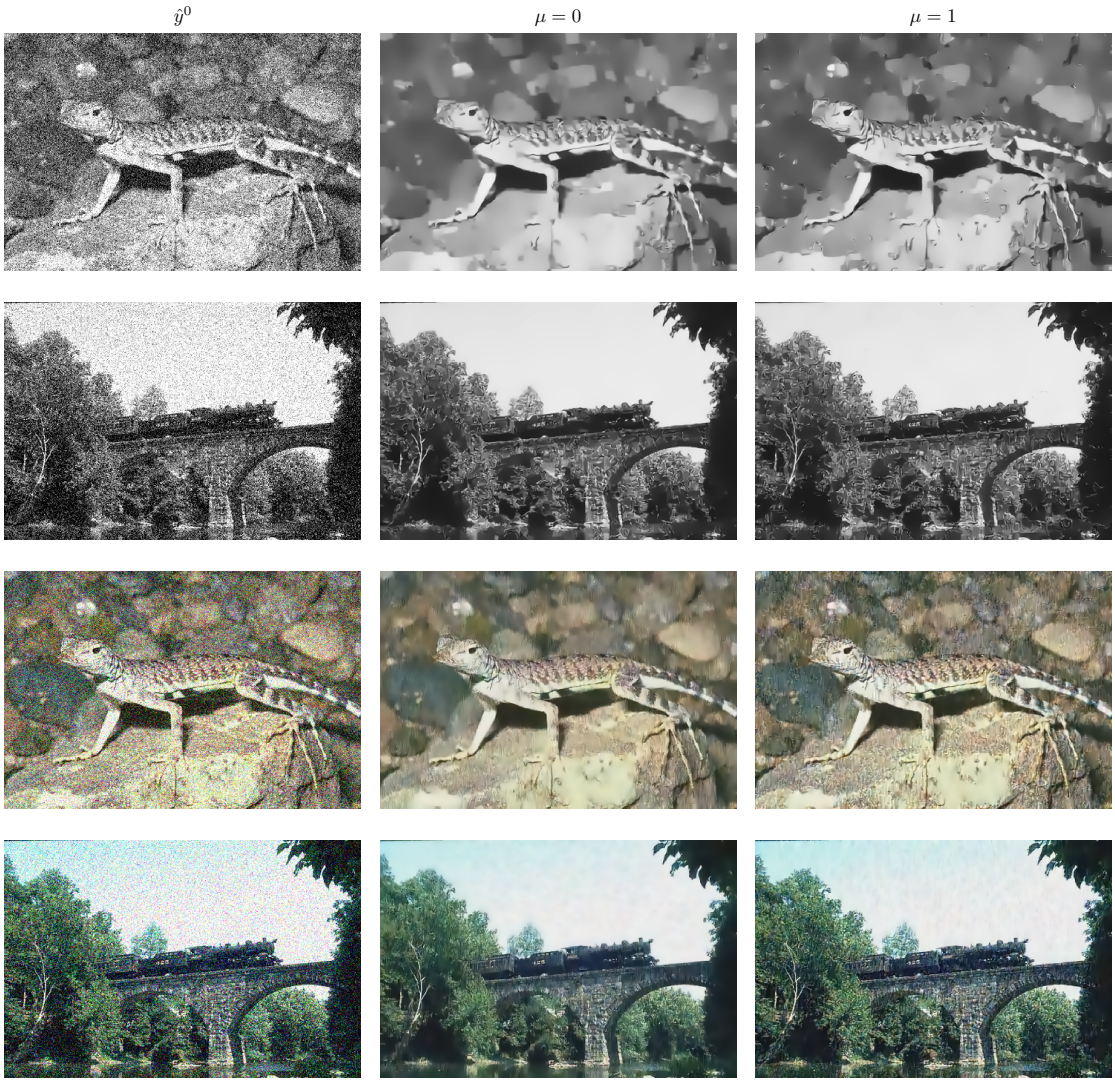


Figure 5.18: Results of additive Gaussian denoising with $\sigma = 50$.

5.4.3 Salt and Pepper Denoising

In this section, we focus on the task of Salt and Pepper denoising on color images ($C = 3$). Being a traditional denoising task, the degradation model is similar to that of additive Gaussian denoising, but with $\nu \sim \mathcal{B}(p) \times \{-1, 1\}^n$ where $\mathcal{B}(p)$ is the Bernoulli distribution. Notice that, due to the images being constrained to $[0, 1]^n$, any given pixel has a chance p to be set to either 0 or 1. Clearly, this results in a much heavier tailed distribution as compared to the Gaussian case. As the data term should mirror the statistics the noise model, we chose the robust L_1 norm for all the experiments, such that our energy is given by

$$E(x, y, \theta) = \lambda \|x - y\|_1 + R(y, \theta), \tag{5.21}$$

where we again consider $R(y, \theta) \in \{TV(y), TDV_3^3(y, \theta), TDV_3^{3,d}(y, \theta)\}$. For the L_1 norm data term $\lambda D(x, y) = \lambda \|x - y\|_1$ the proximal operator is the soft-shrinkage

$$\text{prox}_{\lambda\|x-\cdot\|_1}(y) = x + \text{sign}(x - y) \max(|x - y| - \lambda, 0), \tag{5.22}$$

where all operations are understood element-wise.

Note that $TDV_3^{3,d}$ is trained on additive Gaussian denoising. As such, for the Salt and Pepper denoising task, we expect the proposed model to perform best. However, since the there are strong similarities between these tasks, the $TDV_3^{3,d}$ model should still perform reasonably well, especially for small p. For all models, we hand-tune the data fidelity parameter λ on the inference task using the images in the BSDS500 training set.

We evaluate the models trained on color images ($C = 3$) on a subset of 8 images of the 76 landscape scenes in the BSDS500 validation data set. These images were picked prior to evaluation based on diversity. We show the *PSNR* on the considered test set in Table 5.2. In line with our expectations, we see that the proposed learning scheme leads to the best results. We note that the performance is significantly better for $\mu = 1$ as compared to the *ML*. Further, in contrast to the Gaussian task, the results are consistently better for our models when compared to the *TV*-L_1 or the $TDV_3^{3,d}$.

We compare the models qualitatively for $p \in \{0.1, 0.2, 0.5\}$ in Fig. 5.19, where again we see that our TDV_3^3 performs best visually across the full range of $p \in \{0.1, 0.2, 0.5\}$. We observe that $TDV_3^{3,d}$ leaves high-contrast speckles in the solution, even for $p = 0.1$.

Table 5.2: Comparison of the *PSNR* values for Salt and Pepper denoising for $p \in \{0.1, 0.2, 0.5\}$ on our test set.

p	*TV*-L_1	TDV_3^3		$TDV_3^{3,d}$
		$\mu = 0$	$\mu = 1$	
0.1	28.71	33.61	35.80	30.93
0.2	27.07	30.82	32.58	22.52
0.5	23.90	25.50	26.16	17.88

To ensure that this is not due to an improperly chosen data weight λ, we show inference of TDV^3_3 for a range of λ in Fig. 5.20. Clearly, even for $\lambda \ll$ where the regularizer already greatly simplified the underlying structure in the image, it does not remove the high-contrast speckles. We also note the regularizer for $\mu = 1$ has some interesting properties. In particular, it is best at preserving the details in the image, where for instance for $p = 0.2$ and even $p = 0.5$ some features in the image are still clearly visible that vanish in all other cases. It was also noticeably harder to optimize, which leads us to believe that these resulting images do not fully represent what the regularizer is capable of. We show more qualitative examples in Figs. 5.21 to 5.23.

Figure 5.19: Qualitative comparison between the $TV\text{-}L_1$ model, the proposed TDV_3^3 scheme for $\mu \in \{0, 1\}$ and the $TDV_3^{3,d}$ trained on an additive Gaussian denoising task with $\sigma = 25$. We see that, especially for $p \in \{0.1, 0.2\}$, the proposed learning scheme leads to the best solution, where it can reconstruct the scene up to small details. Note that even for $p = 0.1$, $TDV_3^{3,d}$ leaves high-contrast speckles in the image, and for $p = 0.5$ it hallucinates structure into the image, similar to that seen in its eigenimage analysis [33].

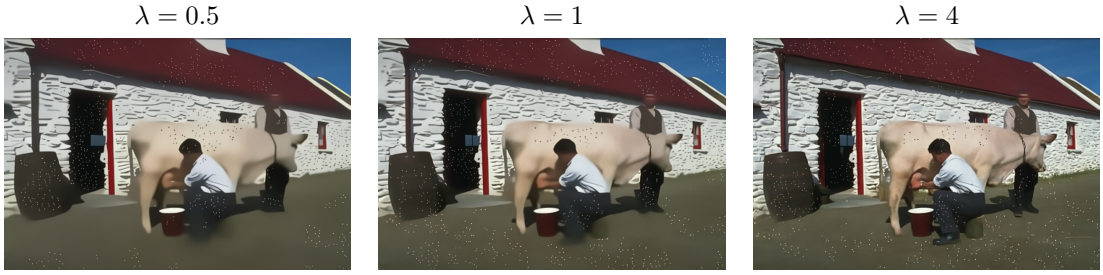


Figure 5.20: Inference of the *TDV*$^{3,d}_{3}$ regularizer trained on additive Gaussian denoising of $\sigma = 25$ on a Salt and Pepper denoising task ($p = 0.1$): Clearly, high contrast speckles are not removed even for $\lambda \ll$, where the image has already been greatly simplified by the regularizer.

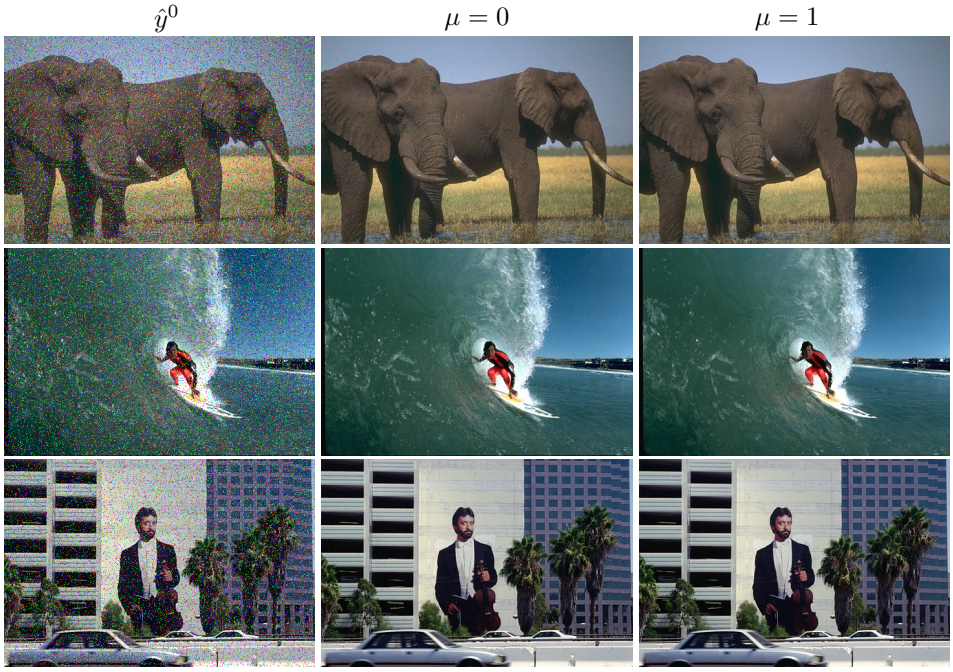


Figure 5.21: Results of Salt and Pepper denoising with $p = 0.1$.

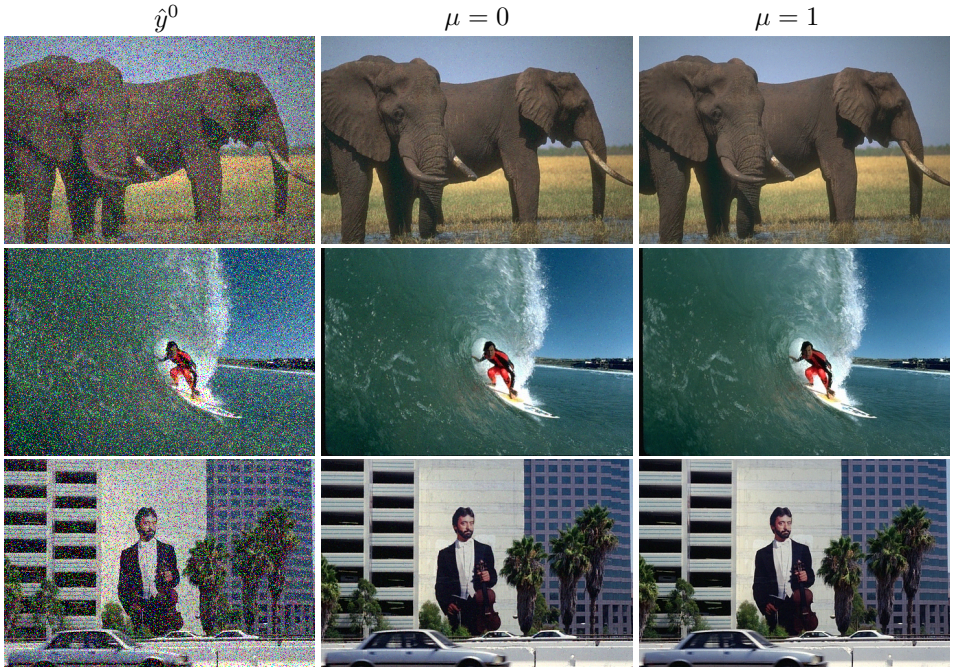


Figure 5.22: Results of Salt and Pepper denoising with $p = 0.2$.

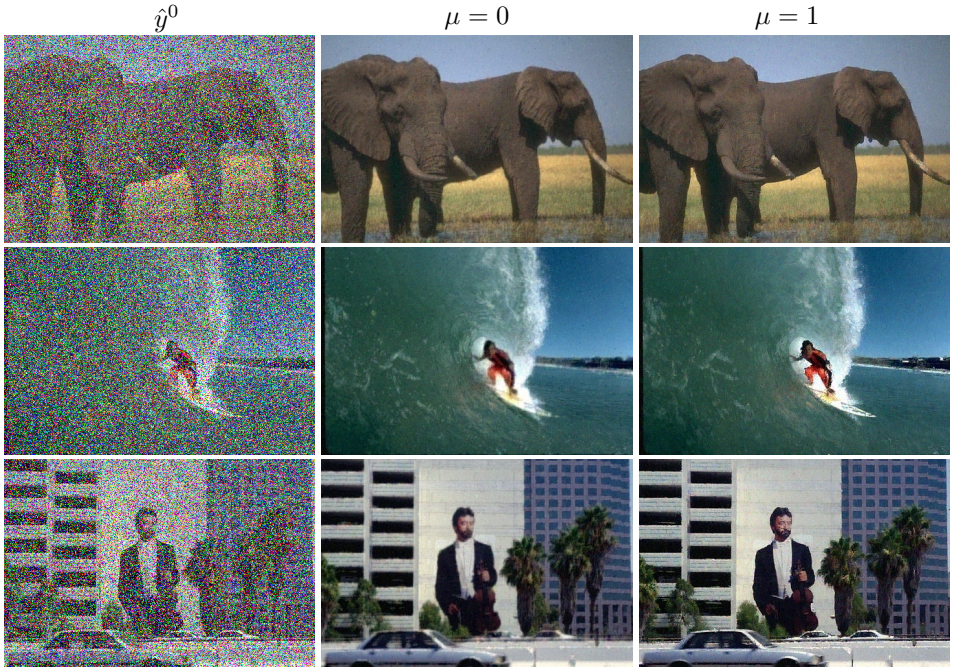


Figure 5.23: Results of Salt and Pepper denoising with $p = 0.5$.

5.4.4 Image Inpainting

In Image Inpainting, we assume that a subset $\mathcal{I}$ of the image domain Ω is known, whereas there is no information in $\tilde{\mathcal{I}} = \Omega \setminus \mathcal{I}$. Specifically, the degradation operator is $A = \operatorname{diag}(\{\mathbb{1}_{\mathcal{I}}(i), i = 1, \dots, n\})$, where $\mathbb{1}_{\mathcal{I}} \colon \Omega \to \{0, 1\}$ is the indicator function over the set $\mathcal{I}$. This motivates us to formulate the energy as

$$E(x, y, \theta) = \delta_{\mathcal{I}}(x, Ay) + R(y, \theta) \tag{5.23}$$

with the Dirac data term

$$D(x, y) = \delta_{\mathcal{I}}(x, Ay) = \begin{cases} \infty & \text{if } y \neq x \text{ anywhere in } \mathcal{I}, \\ 0 & \text{else,} \end{cases} \tag{5.24}$$

where the corresponding proximal map is easily computed as

$$\operatorname{prox}_{\delta_{\mathcal{I}}(x,\cdot)}(Ay) = \begin{cases} x & \text{in } \mathcal{I}, \\ y & \text{else.} \end{cases} \tag{5.25}$$

We construct the inpainting domain in two different ways:

1. Line Inpainting: For each horizontal line, there is a chance p that all of its pixels are in $\tilde{\mathcal{I}}$.

2. Pixel Inpainting: Each pixel has a chance p to be in $\tilde{\mathcal{I}}$.

For both of these tasks, we set

$$\hat{y}_i^0 = \begin{cases} \gamma_i \sim \mathcal{U}[0, 1] & i \in \tilde{\mathcal{I}} \\ x_i & i \notin \tilde{\mathcal{I}} \end{cases}, \quad \forall i \in \{1, \dots, n\}. \tag{5.26}$$

Clearly, there is a strong difference in "difficulty" between these two problems. Line inpainting can result in the loss of information in a large, convex area, such that a large receptive field is needed to faithfully reconstruct the image. There is also much more ambiguity in terms of possible solutions, as they can only be guided by information that is potentially far away from some parts of the inpainting domain. This ambiguity also makes evaluation delicate, as the inpainted regions may gain or lose certain features compared to the target.

It is obvious that *TV* is not a particularly strong prior for such tasks, since it can only consider $\partial\mathcal{I}$ for inpainting $\tilde{\mathcal{I}}$. On the other hand, if $\tilde{\mathcal{I}}$ does not have large coherent subsets, as in the case of pixel inpainting, it can lead to reasonable solutions. All in all, we expect the proposed regularizer to perform best on the inpainting tasks when compared to *TV* regularization and the discriminatively trained $TDV_3^{3,d}$.

Table 5.3: Comparison of *PSNR* values for pixel- and line-wise image inpainting for $p \in \{0.1, 0.3, 0.7\}$ on our test set.

	p	*TV*	TDV_3^3		$TDV_3^{3,d}$
			$\mu = 0$	$\mu = 1$	
pixel	0.1	38.08	45.28	45.10	45.75
	0.3	32.37	41.82	41.87	35.83
	0.7	26.25	30.87	29.87	24.20
line	0.1	36.28	34.31	34.47	18.25
	0.3	29.26	28.44	26.93	12.92
	0.7	23.22	20.85	19.62	8.52

We show the *PSNR* values of the different models in Table 5.3 and show qualitative results for pixel inpainting in Fig. 5.24, where we chose $p \in \{0.1, 0.3, 0.7\}$. The quantitative results show that the proposed regularizer is the most robust, i.e. it performs reasonably well across both tasks and across a range of p. We see that our proposed model performs best visually, as it achieves satisfactory results even for $p = 0.7$. Observe that the $TDV_3^{3,d}$ model seems to be a strong inpainting prior for this specific task, were it not for the high-contrast speckles that we already observed in the Salt and Pepper denoising task.

More results of the proposed regularizers for pixel-wise inpainting for $p \in \{0.1, 0.3, 0.7, 0.9\}$ are shown in Figs. 5.25 to 5.28. We see that even for $p = 0.9$, the reconstructed image looks pleasing, although we observe the tendency to introduce vertical structure into the image. Here, we see that the *ML* regularizer outperforms the model trained with loss augmented inference of $\mu = 1$, although again this model was harder to optimize, such that our inference algorithm may not be converged.

We qualitatively compare the different models on the line inpainting task in Fig. 5.29. We see that our TDV_3^3 leads to the most natural looking results, as it is able to reconstruct the pattern and texture of the fur best. A more detailed view is seen in Fig. 5.30, where this can be nicely seen. This is in contrast to the quantitative analysis, which suggests that the *TV* model is better at line inpainting. We think that the *PSNR* values are highly ambiguous when considering line inpainting, as the reconstructed regions may look much more natural, while being far from the original solution. More results for line inpainting are show in Figs. 5.31 to 5.33 for $p \in \{0.1, 0.3, 0.7\}$ respectively.

Figure 5.24: Qualitative comparison between the TV regularizer, the proposed TDV_3^3 scheme for $\mu \in \{0, 1\}$ and the $TDV_3^{3,d}$ regularizer on a pixel-wise inpainting task for $p \in \{0.1, 0.3, 0.7\}$. In line with our expectation, we see that the proposed learning scheme leads to the best results. Note the tendency of the $TDV_3^{3,d}$ model to prefer high-contrast speckles in the image.

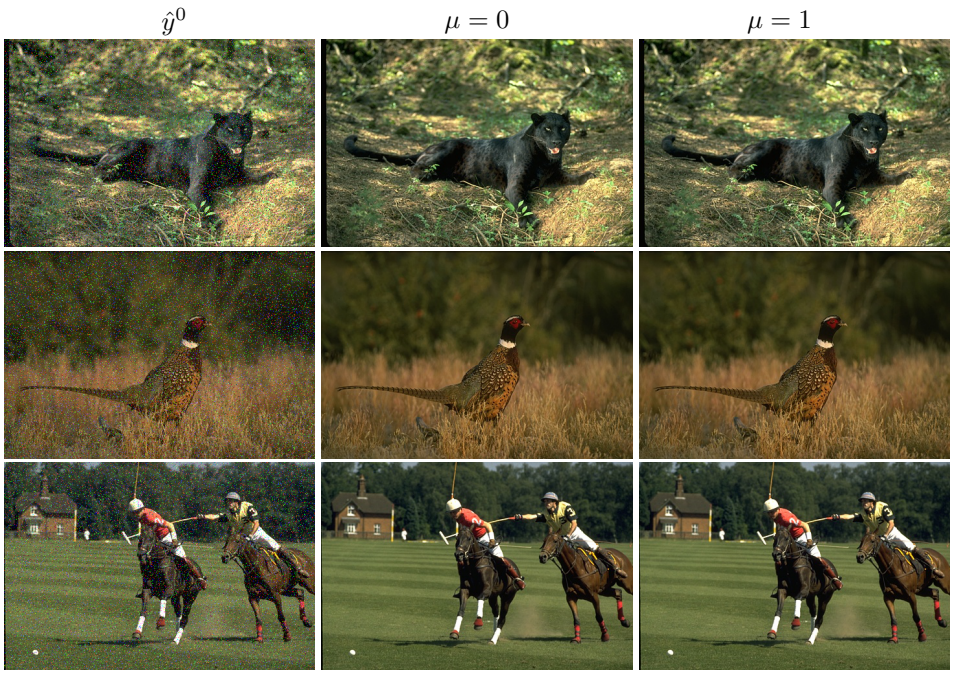


Figure 5.25: Results of pixel-wise inpainting with $p = 0.1$.

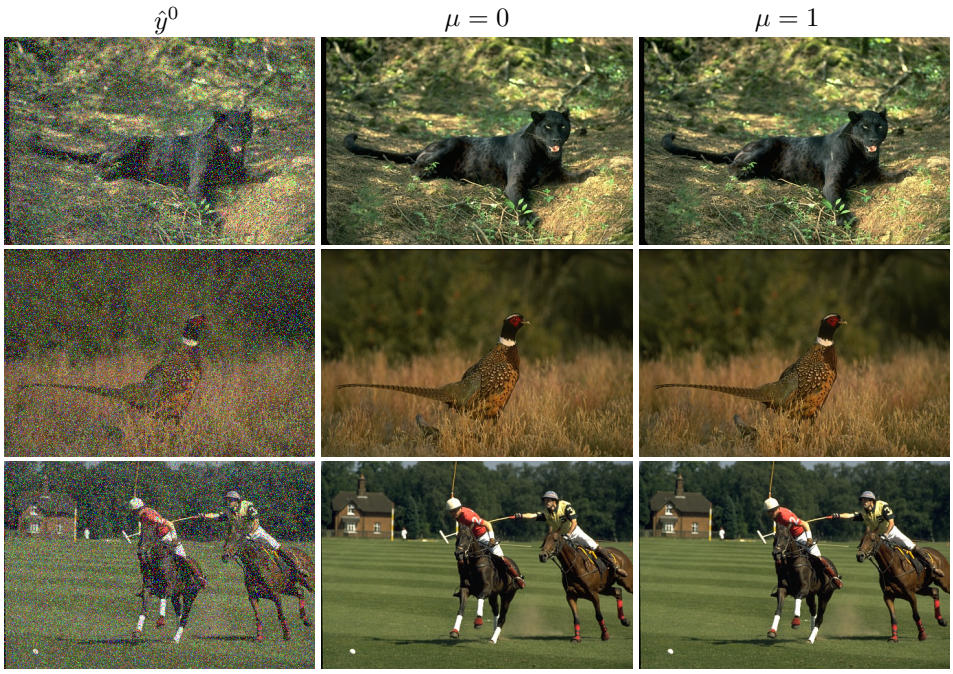


Figure 5.26: Results of pixel-wise inpainting with $p = 0.3$.

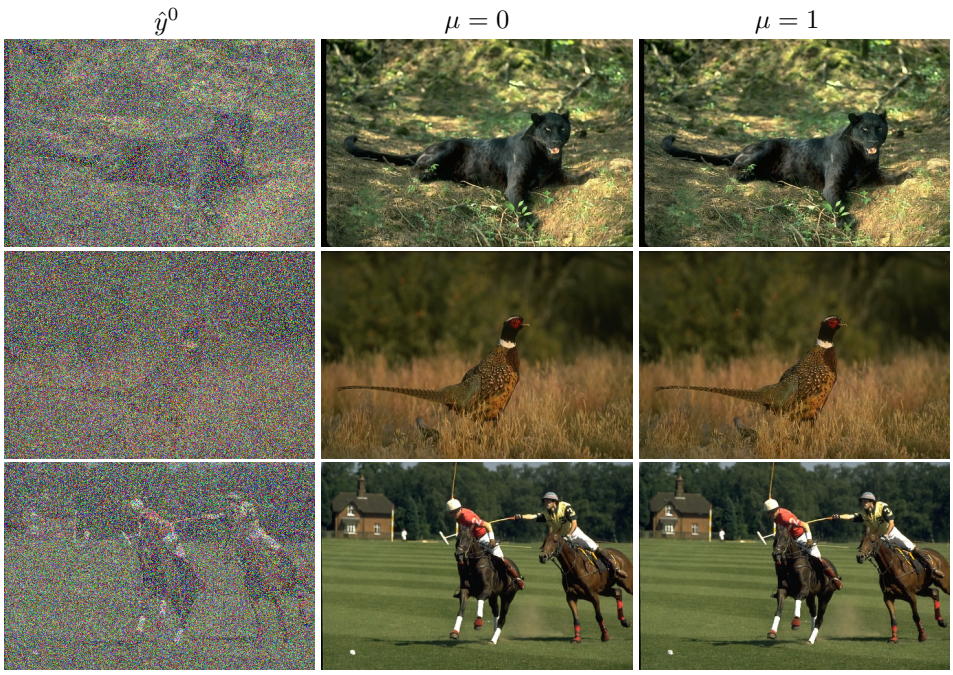


Figure 5.27: Results of pixel-wise inpainting with $p = 0.7$.

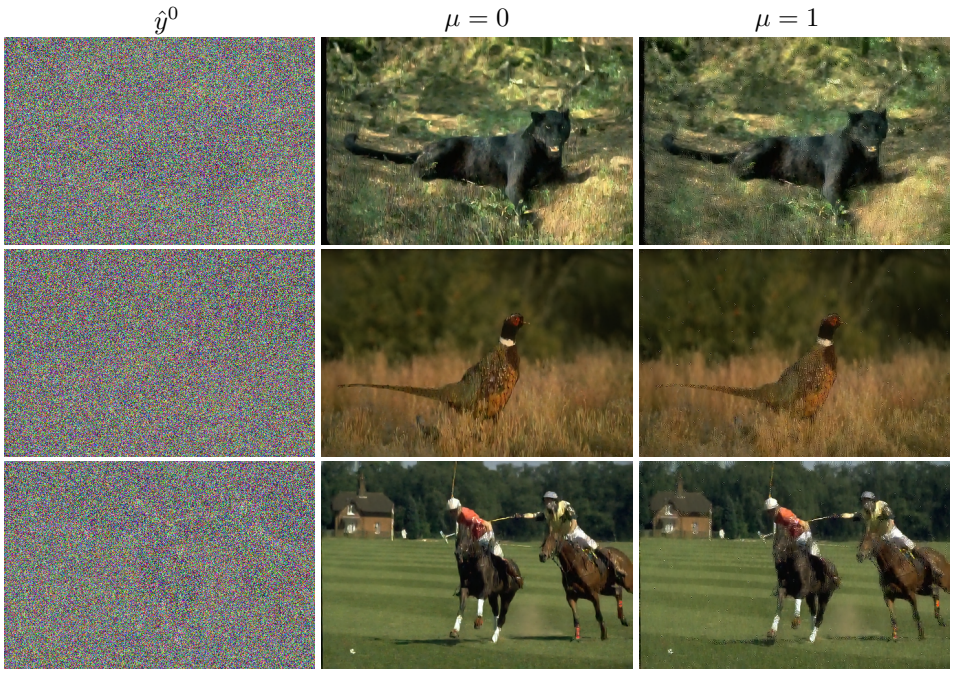


Figure 5.28: Results of pixel-wise inpainting with $p = 0.9$.

Figure 5.29: Qualitative comparison between the TV regularizer, the proposed TDV^3_3 scheme for $\mu \in \{0, 1\}$ and the $TDV^{3,d}_3$ regularizer on a line-wise inpainting task for $p \in \{0.1, 0.3, 0.7\}$. The proposed regularizer performs best, as it reconstructs the texture of the fur nicely, even for larger inpainting regions. Note how the preference of high contrast, piecewise constant regions of the $TDV^{3,d}_3$ leads to vary unnatural looking images in this task.

Figure 5.30: Detailed view of the reconstruction in a line-inpainting task of the trained TDV^3_3 model versus TV reconstruction. Clearly, we see that TDV^3_3 is able to connect contours better, although in large inpainting regions we observe very little detail also for the TDV^3_3 model.

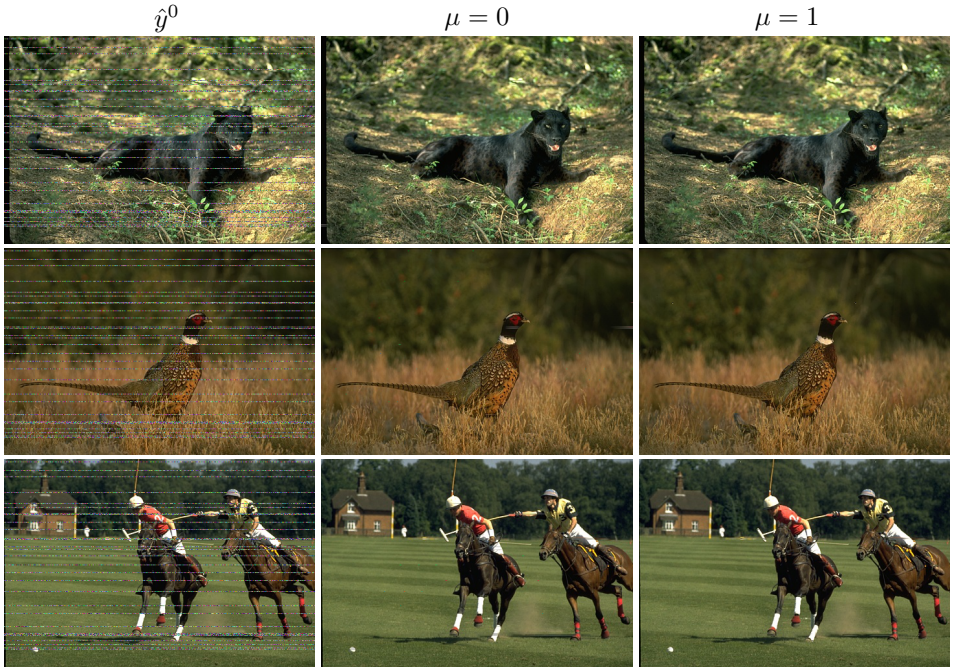


Figure 5.31: Results for line-wise inpainting with $p = 0.1$.

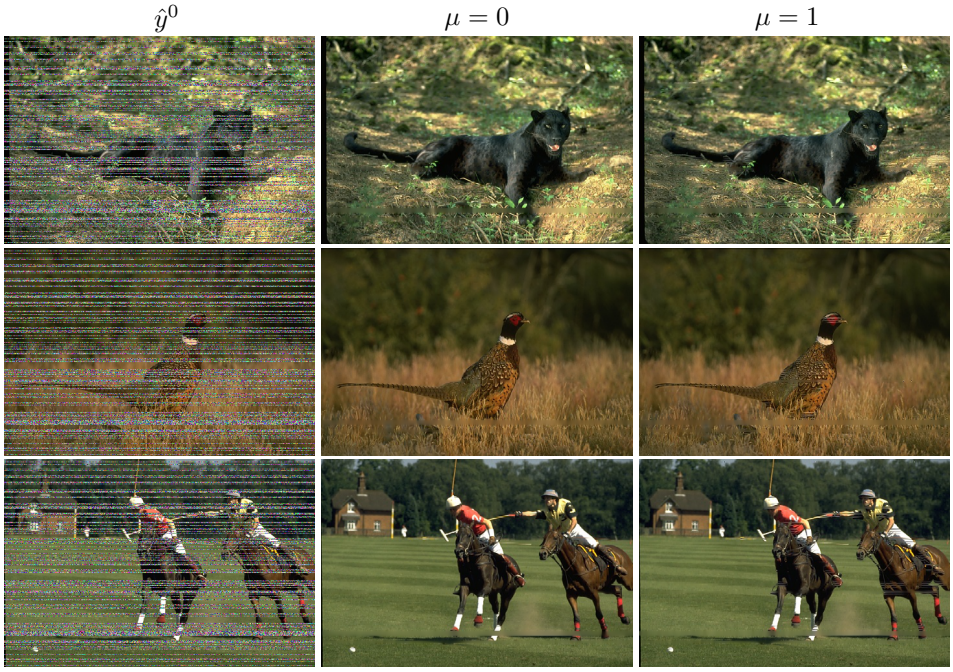


Figure 5.32: Results for line-wise inpainting with $p = 0.3$.

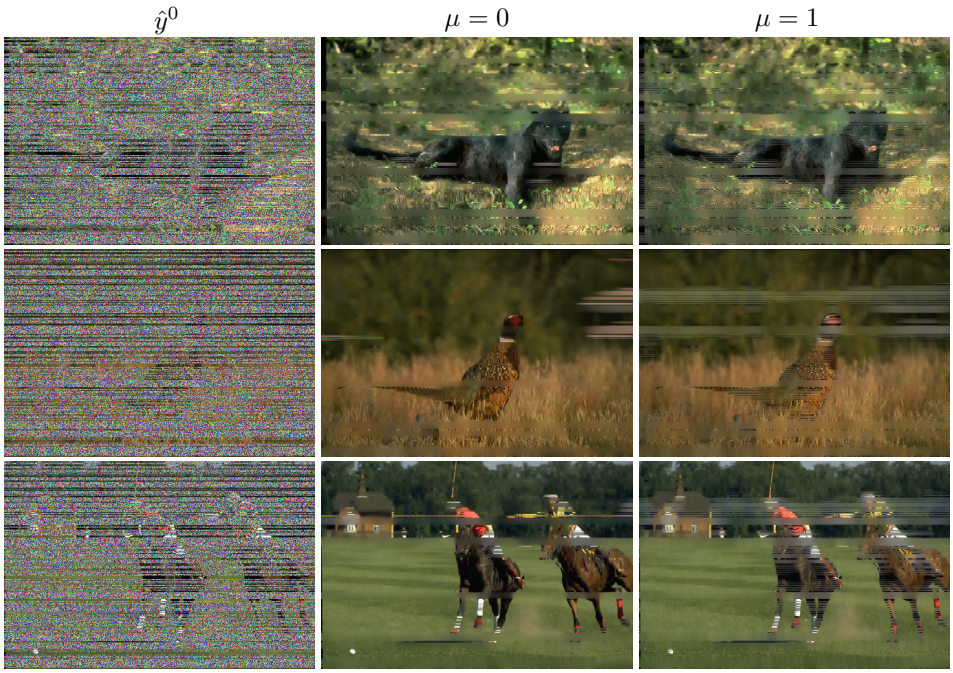


Figure 5.33: Results for line-wise inpainting with $p = 0.7$.

5.4.5 Image Deconvolution

Image deconvolution is a traditional image restoration task, with its roots in applications in astronomy. The degradation model reads

$$x = Ay + \nu, \tag{5.27}$$

where the A models the convolution, i.e. it is a Toeplitz matrix containing the kernel elements, and the noise ν may be drawn from some arbitrary distribution. In our experiments, we let $\nu \sim \mathcal{N}(0, \sigma^2 I_n,)$, hence we chose a quadratic L_2 data term $\lambda D(x, y) = \frac{\lambda}{2} \|x - Ay\|_2^2$. Note that the proximal map $\operatorname{prox}_{\frac{\lambda}{2}\|x-A\cdot\|_2^2}(y)$ can be computed in the Fourier domain as

$$\operatorname{prox}_{\frac{\lambda}{2}\|x-A\cdot\|_2^2}(y) = \mathcal{F}^{-1}\left\{\frac{\lambda\mathcal{F}\{x\}\mathcal{F}\{a\}^* + \mathcal{F}\{y\}}{\lambda\mathcal{F}\{a\}^2 + 1}\right\} \tag{5.28}$$

where $Ay \equiv a * y$ and the operations are understood element-wise.

We differentiate between two different classes of problems in image deconvolution:

1. Non-blind deconvolution: Knowledge about the blur process is assumed, i.e. the Point Spread Function (PSF) is known or estimated, and subsequently given the degraded observation x the sharp image y should be estimated.

2. Blind deconvolution: For the blind deconvolution problem, we are only given the degraded observation x and the operator A corresponding to the *PSF* as well as the clean image y should be estimated.

In what follows, we only consider non-blind deconvolution problems where we use the data set of [36], which contains both images and blur kernels. The models are evaluated on a subset of 8 images and 2 blur kernels of the provided data set, which were chosen a-priori, and we hand-tuned λ on a different set of 8 images with the same 2 kernels. For all experiments, we choose $\sigma = 0.025$ and compare our model to *TV* regularization using the method of [24] and the channel-wise Wiener filter

$$\hat{y}_{\mathrm{WF}} = \mathcal{F}^{-1}\left\{\frac{\mathcal{F}\{a\}^*\mathcal{F}\{x\}}{|\mathcal{F}\{a\}|^2 + \frac{1}{\mathrm{SNR}(x)}}\right\}, \tag{5.29}$$

with the denominator estimate of [47].

We compare the different methods in Fig. 5.34, where we see that the trained regularizers are capable of restoring the image truthfully. The Wiener filter solution of [47] has the drawback of strongly amplifying the noise in the input image and, although sharp, exhibits clear ringing artefacts. The *TV* regularization of [24] can nicely suppress the noise, however we see that small details are lost. On the other hand, our TDV_3^3 regularizer restores the detail in the image whilst there is no perceptible noise in the estimate. We show further results in Fig. 5.35 and Fig. 5.36. We show the quantitative evaluation

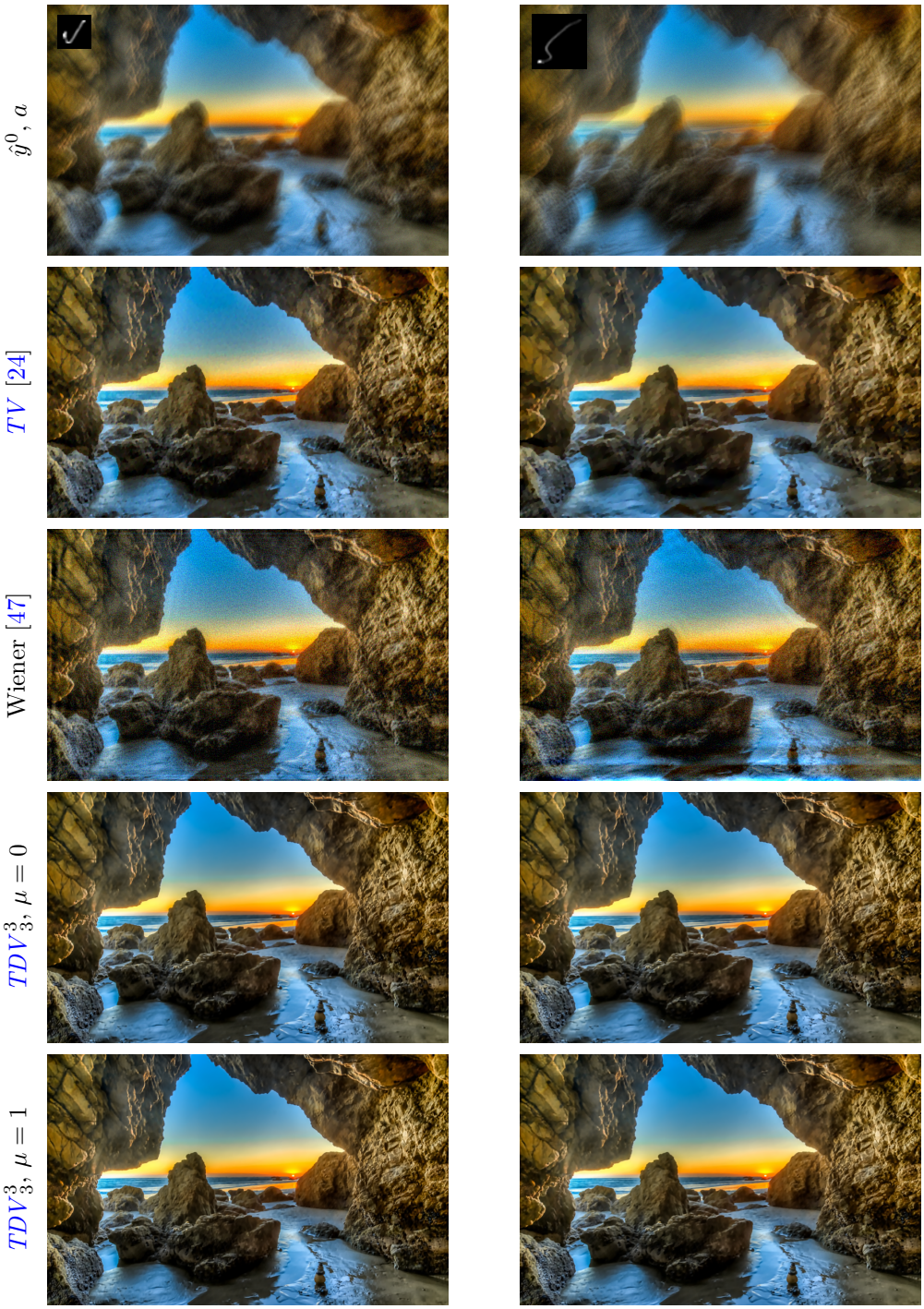


Figure 5.34: Qualitative comparison between the TV model of [24], the unsupervised Wiener filter [47] and the proposed models on a deconvolution task. The inlay shows the two different blur kernels from the data set of [36]. Note that the scale between the kernels is accurate, but the scale of the kernels w.r.t. the images is not.

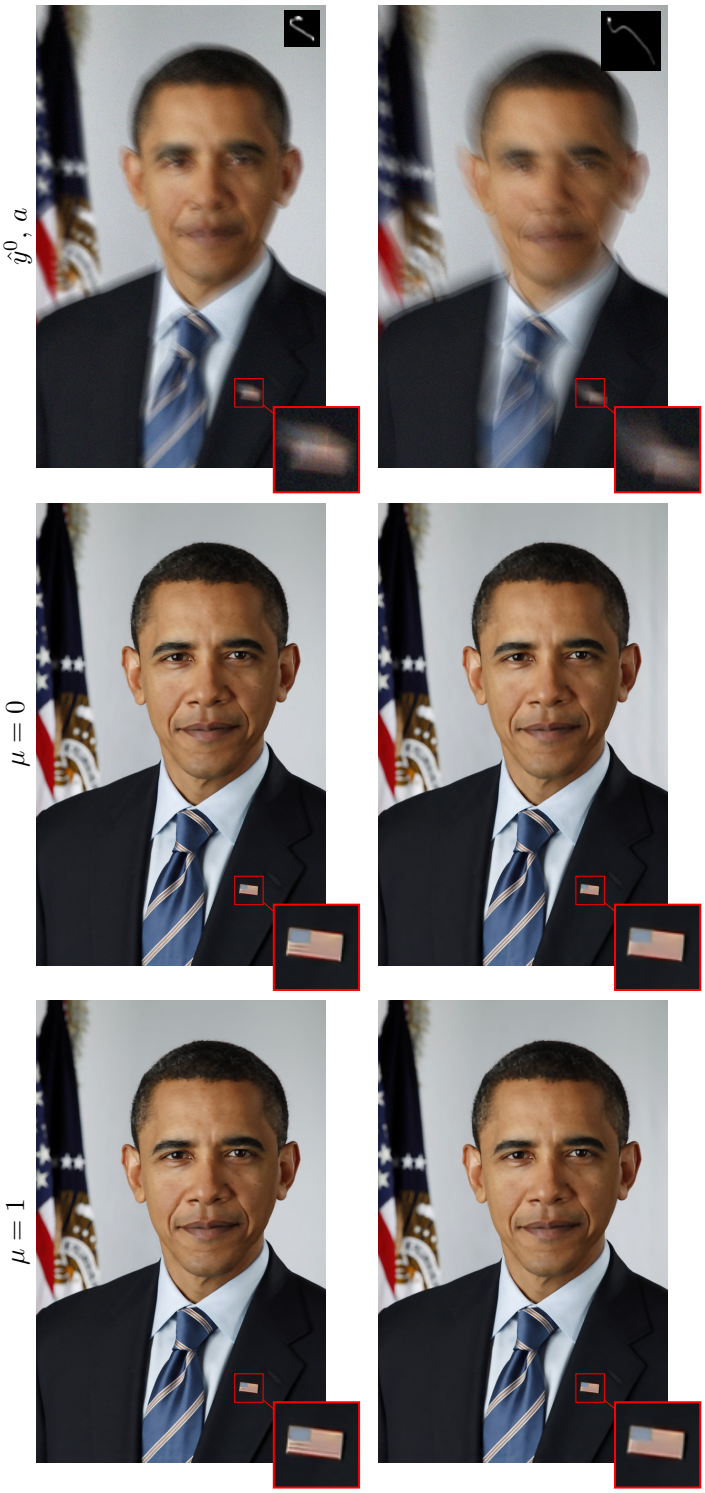


Figure 5.35: Examples of deconvolution with the trained models for $\mu \in \{0, 1\}$. While both models restore the images satisfactorily, we do not observe a significant difference in the models. Note the kernels are rotated by $\frac{\pi}{2}$ rad, as the images was processed in landscape.

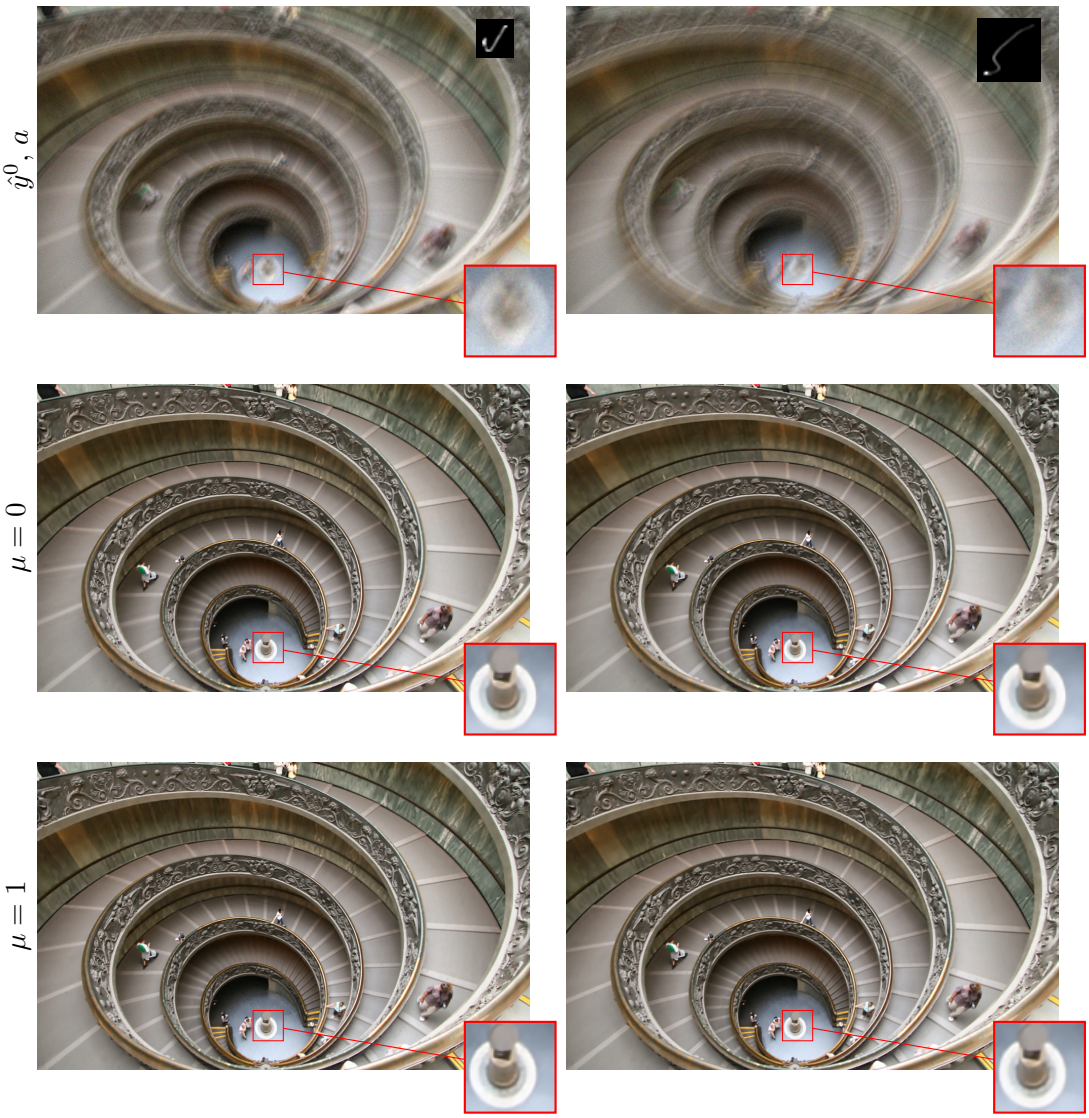


Figure 5.36: Examples of deconvolution with the trained models for $\mu \in \{0, 1\}$.

in Table 5.4, where the *PSNR* values confirm that our models are able to restore the image best.

Table 5.4: Comparison of *PSNR* values for non-blind deconvolution for different kernels a_1, a_2 on our test set.

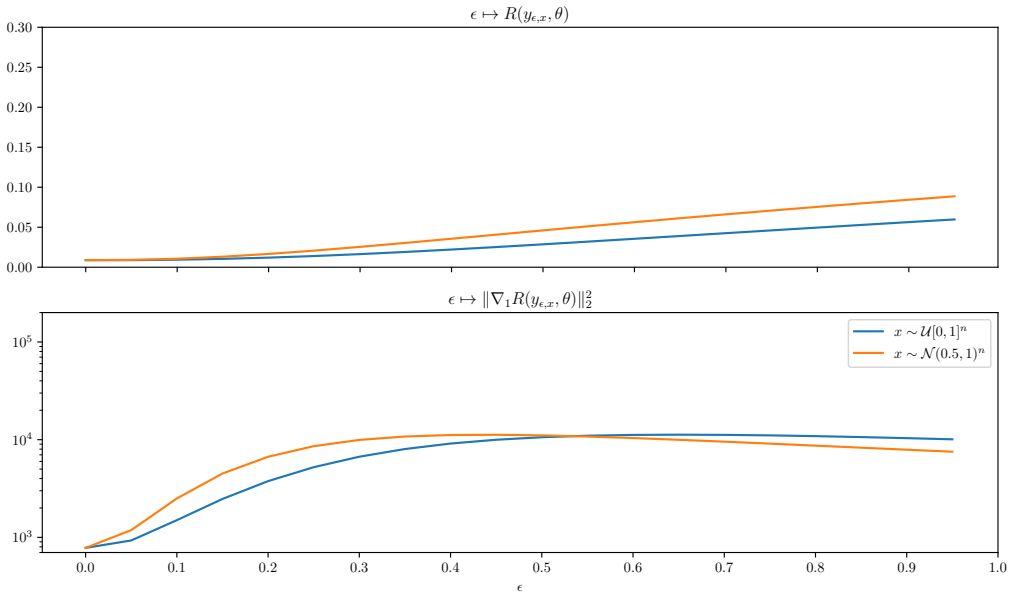
Kernel	*TV* [24]	*TDV*$^3_3$ $\mu = 0$	*TDV*$^3_3$ $\mu = 1$	Wiener [47]
$a_1 \in \mathbb{R}^{31\times31}$	23.00	27.60	28.18	19.15
$a_2 \in \mathbb{R}^{51\times51}$	21.00	26.22	26.31	17.75

5.4.6 Regularization Strength

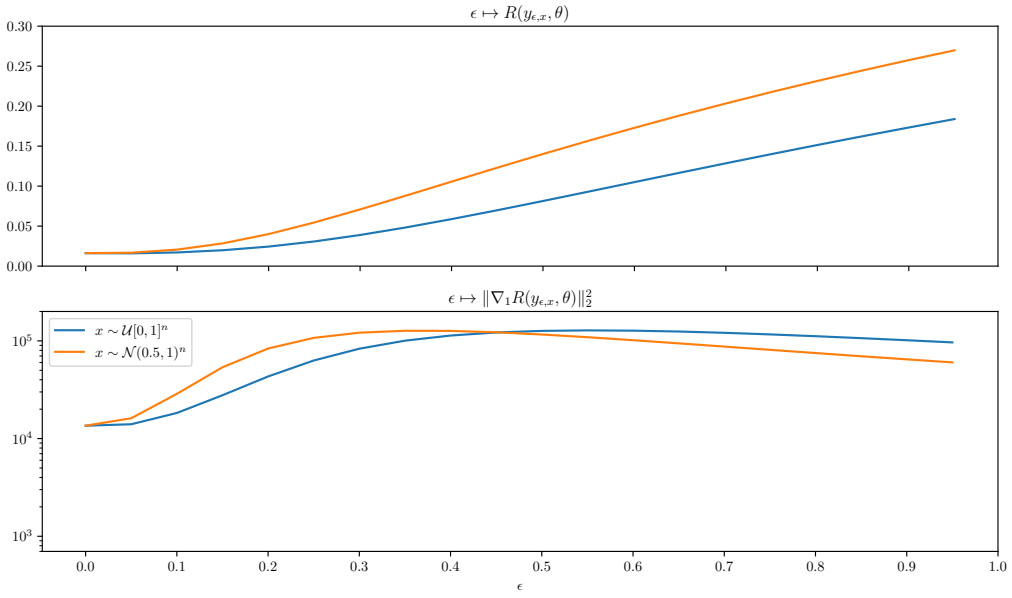
Similar to the analysis in Section 5.2.4, we examine the local landscape of the regularizer around samples from the test set to study the influence of μ. Specifically, we let

$$y_{\epsilon,x} = y + \epsilon(x - y), \tag{5.30}$$

where $y \sim$ test set, $\epsilon \in [0, 1]$ and $x \in \mathbb{R}^n$ is drawn from either $\mathcal{N}(0.5, I_n)$ or $\mathcal{U}[0, 1]^n$. We show $R(y_{\epsilon,x}, \theta)$ and $\|\nabla_y R(y_{\epsilon,x}, \theta)\|_2^2$ for $\mu \in \{0, 1\}$ in Fig. 5.37a and Fig. 5.37b respectively. Similarly to Section 5.2.4, we see that $\|\nabla_y R(y_{\epsilon,x}, \theta)\|_2^2$ is larger for $\mu = 1$, especially for $\epsilon \ll$.



(a) $\mu = 0$



(b) $\mu = 1$

Figure 5.37: Regularization energy $R(\cdot, \theta)$ and gradient norm $\|\nabla_y R(y_{\epsilon,x}, \theta)\|_2^2$ around test samples y for $\mu \in \{0, 1\}$. Especially around $\epsilon = 0$, i.e. near y, $\|\nabla_y R(y_{\epsilon,x}, \theta)\|_2^2$ is significantly larger for $\mu = 1$.

6 Conclusion and Outlook

6.1 Conclusion

In this work, we first introduced a general mathematical formulation for Image Restoration (IR) and showed some concrete examples of typical problems in the field. We outlined historical solutions for *IR* problems, going from inverse filtering over variational methods to deep feed-forward neural networks. We put particular focus on the profound mathematical framework of *variational methods*, draw its relation to *energy based models* and show its probabilistic interpretation. Within this probabilistic interpretation, we show how the regularizer should encode prior knowledge of natural images. We then derive the Maximum Likelihood (ML) solution of fitting a parametrized regularizer to a data set of natural images, and outline the idea behind some non-convergent *ML*-like algorithms. Using a simple example, we show that *ML* is not generally a desirable objective and introduce a learning algorithm which takes the idea of loss augmented inference from Structured Support Vector Machine (SSVM) training to sample from the Gibbs-Boltzmann distribution of a regularizer.

We then use our proposed algorithm to train the multi-scale $TDV^{N_{\text{Sc}}}_{N_{\text{Ma}}}$ regularizer of [33] on the MNIST ($(N_{\text{Ma}}, N_{\text{Sc}}) = (3, 3)$) and FashionMNIST ($(N_{\text{Ma}}, N_{\text{Sc}}) = (2, 2)$) data sets, and analyze the learned regularizer by considering its modes, eigenimages and local landscape around samples from the data distribution. The experiments show that we can meaningfully train the $TDV^{N_{\text{Sc}}}_{N_{\text{Ma}}}$ regularizer with the proposed algorithm and indicate that using loss augmented inference forces the models to have larger gradients near samples in the data set.

Having established the applicability of the algorithm, we train a TDV^3_3 regularizer on the BSDS500 data set and show results for traditional *IR* tasks, namely Gaussian and Salt and Pepper denoising, image inpainting, and image deconvolution. We compared our approach to classical models in the literature as well as the discriminatively learned $TDV^{3,d}_3$ model [33]. Our model performs well across all tasks, beating archetypical non-parametric models in terms of Peak Signal to Noise Ratio (PSNR), often by a large margin.

Comparing it to the $TDV_3^{3,d}$, we observed that our model is a stronger prior for a range of tasks, as it outperforms $TDV_3^{3,d}$ on all tasks except the very task the $TDV_3^{3,d}$ model was trained on. This is in line with our expectations, as we did not expect a generative model of the same size to be on-par with a discriminative model on a given task.

Although we did not observe any major quantitative benefit from loss augmented inference in the *PSNR* values, the models were at least on-par with the *ML* models. Further, models trained with loss augmented inference exhibited again a steeper landscape around samples from the data distribution. Hence we believe that loss augmented inference is a valuable tool for controlling model capacity and expressiveness.

6.2 Outlook

Our training was inspired by *SSVM* training, where we used loss augmented inference for sampling of the model. However, we did not consider another integral part of *SSVM* training, which is the concept of the margin for parameter optimization. In *SSVM* training, one only optimizes the parameters over the set of samples which violate the margin, after the samples have been drawn with loss augmented inference. While we used the notion of loss augmented inference for sampling, we did not discard any of the samples based on margin violation, but used all samples in our loss. It may be interesting to introduce the concept of the margin into training, where we use loss augmented inference for sampling, but discard samples that have high probability under our model for optimization. Doing this, there is less of a disconnect between sampling and optimization, as we do not consider samples which already were improbably under our model to further tune the parameters.

It is also interesting to consider different loss functions for loss augmented inference. We used the conceptually simplest loss during loss augmented inference, which is the euclidean distance. The idea is to find samples which are far away from the data distribution, but have high probability under our model, where "far away" is measured by the euclidean distance. However, although simple it is hard to interpret for natural images. One may consider other, more easily interpretable losses to guide the loss augmented inference. For instance, one could augment sampling with the Total Variation (TV) loss, such that the loss augmented inference finds samples with high *TV* and high probability under the model. Combining this with the idea of the margin, we believe that it would be possible to learn a highly expressive and largely unbiased regularizer.

In Chapter 5, we showed how we can tune the local landscape of our regularizer with loss augmented inference, where stronger pull-away terms forced the regularizer to have steeper flanks around the training data. In the context of highly expressive energy based models, it is still largely unexplored how to control the "variance" of a model. Although concepts such as spectral normalization [44] and Lipschitz-penalties [27] exist in the Generative Adversarial Network (GAN) community and may be used in probabilistic energy based models as well, we believe that loss augmented inference is another tool that can be used.

A

List of Acronyms

ARF Active Random Field. 9

AWGN Additive White Gaussian Noise. 2, 7, 12, 13

CD Contrastive Divergence. 19, 20

CFA Color Filter Array. 1

CNN Convolutional Neural Network. 2, 10

CRF Conditional Random Field. 9

CSF Cascade of Shrinkage Field. 9

CT Computed Tomography. 10

DCT Discrete Cosine Transform. 6

DFT Discrete Fourier Transform. 4

EBM Energy Based Model. 2, 6, 15, 17, 19, 26

FoE Fields of Experts. 8–10

GAN Generative Adversarial Network. 72

HMC Hamiltonian (Hybrid) Monte Carlo. 19, 21

HMM Hidden Markov Model. 6

iid independent and identically distributed. 17, 18

IR Image Restoration. 1, 2, 10, 11, 15, 16, 39, 44, 71

KLD Kullback-Leibler Divergence. 19, 22

MAP Maximum A-Posteriori. 13

MCMC Markov Chain Monte Carlo. 8, 19–21

MH Metropolis-Hastings. 21

ML Maximum Likelihood. 11, 12, 18–22, 28, 29, 32–35, 39–41, 43, 51, 57, 71, 72

MRF Markov Random Field. 9

NN Neural Network. 10

PCD Persistent Contrastive Divergence. 21

PDF Probability Density Function. 11, 12, 15, 21, 34

PoE Producs of Experts. 19

PSF Point Spread Function. 3, 5, 64

PSNR Peak Signal to Noise Ratio. 2, 45–47, 51, 57, 67, 68, 71, 72

ROF Rudin Osher Fatemi. 7, 45–47

SGLD Stochastic Gradient Langevin Dynamics. 19, 21, 31, 34

SISR Single-Image Super-Resolution. 1, 3, 10, 12

SNR Signal to Noise Ratio. 4

SotA State-of-the-Art. 10

SSVM Structured Support Vector Machine. 2, 22, 25, 26, 28, 29, 32, 71, 72

SVM Support Vector Machine. 23–26

TDV Total Deep Variation. 30, 31, 33, 37, 39, 43–47, 51–54, 56–58, 61, 62, 64, 65, 68, 71, 72

TGV Total Generalized Variation. 7

TNRD Trainable Nonlinear Reaction Diffusion. 9

TV Total Variation. 7, 8, 44, 45, 51, 53, 56–58, 61, 62, 64, 65, 68, 72

VN Variational Network. 10

Bibliography

[1] Banham, M. and Katsaggelos, A. (1997). Digital image restoration. *IEEE Signal Processing Magazine*, 14(2):24–41. (page 1)

[2] Barbu, A. (2009). Training an active random field for real-time image denoising. *IEEE Transactions on Image Processing*, 18(11):2451–2462. (page 9)

[3] Bayer, B. E. (1976). Color imaging array. (page 1)

[4] Blackledge, J. M. (2005). Chapter 12 - image restoration and reconstruction. In *Digital Image Processing*, Woodhead Publishing Series in Electronic and Optical Materials, pages 404 – 438. Woodhead Publishing. (page 4)

[5] Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, COLT '92, pages 144–152. Association for Computing Machinery. (page 23)

[6] Bredies, K., Kunisch, K., and Pock, T. (2010). Total generalized variation. *SIAM Journal on Imaging Sciences*, 3(3):492–526. (page 7)

[7] Carreira-Perpiñ, M. A. and Hinton, G. (2005). On contrastive divergence learning. In *AISTATS05*, pages 33–40. Society for Artificial Intelligence and Statistics. (page 20)

[8] Chambolle, A., De Vore, R., Lee, N.-Y., and Lucier, B. (1998). Nonlinear wavelet image processing: variational problems, compression, and noise removal through wavelet shrinkage. *IEEE Transactions on Image Processing*, 7(3):319–335. (page 6)

[9] Chambolle, A. and Lions, P.-L. (1997). Image recovery via total variation minimization and related problems. *Numerische Mathematik*, 76(2):167–188. (page 7)

[10] Chambolle, A. and Pock, T. (2016). An introduction to continuous optimization for imaging. *Acta Numerica*, 25:161–319. (page 2)

[11] Chambolle, A. and Pock, T. (2019). Total roto-translational variation. *Numerische Mathematik*. (page 8)

[12] Chan, T. F., Esedoglu, S., and Park, F. (2010). A fourth order dual method for staircase reduction in texture extraction and image restoration problems. In *IEEE International Conference on Image Processing*, pages 4137–4140. (page 7)

[13] Chen, Y. and Pock, T. (2017). Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1256–1272. (page 9, 10)

[14] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297. (page 24)

[15] Crouse, M., Nowak, R., and Baraniuk, R. (1998). Wavelet-based statistical signal processing using hidden markov models. *IEEE Transactions on Signal Processing*, 46(4):886–902. (page 6)

[16] Donoho, D. L. and Johnstone, I. M. (1994). Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81(3):425–455. (page 5)

[17] Donoho, D. L. and Johnstone, I. M. (1995). Adapting to unknown smoothness via wavelet shrinkage. *Journal of the American Statistical Association*, 90(432):1200–1224. (page 6)

[18] Dumoulin, V. and Visin, F. (2018). A guide to convolution arithmetic for deep learning. (page 4)

[19] E, W. (2017). A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 5(1):1–11. (page 10)

[20] Easley, G. R., Labate, D., and Colonna, F. (2009). Shearlet-based total variation diffusion for denoising. *IEEE Transactions on Image Processing*, 18(2):260–268. (page 6)

[21] Effland, A., Kobler, E., Kunisch, K., and Pock, T. (2019). An optimal control approach to early stopping variational methods for image restoration. *Journal of Mathematical Imaging and Vision*. (page 9, 10)

[22] Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202. (page 10)

[23] Geman, S. and Geman, D. (1984). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741. (page 15)

[24] Getreuer, P. (2012). Total Variation Deconvolution using Split Bregman. *Image Processing On Line*, 2:158–174. (page 64, 65, 68)

[25] Gharbi, M., Chaurasia, G., Paris, S., and Durand, F. (2016). Deep joint demosaicking and denoising. *ACM Transactions on Graphics*, 35(6):1–12. (page 1)

[26] Gilboa, G. (2018). *Nonlinear Eigenproblems in Image Processing and Computer Vision*. Springer. (page 35)

[27] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. (2017). Improved training of wasserstein GANs. In *Advances in Neural Information Processing Systems*, volume 30, pages 5767–5777. Curran Associates, Inc. (page 22, 37, 72)

[28] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *CVPR*, pages 770–778. IEEE. (page 10)

[29] Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800. (page 8, 19)

[30] Hunt, B. (1972). Deconvolution of linear systems by constrained regression and its relationship to the wiener theory. *IEEE Transactions on Automatic Control*, 17(5):703–705. (page 4)

[31] Joachims, T., Finley, T., and Yu, C.-N. J. (2009). Cutting-plane training of structural svms. *Machine Learning*, 77(1):27–59. (page 28)

[32] Kanizsa, G. (1979). *Organization in Vision: Essays on Gestalt Perception*. Praeger. (page 8)

[33] Kobler, E., Effland, A., Kunisch, K., and Pock, T. (2020). Total deep variation for linear inverse problems. In *CVPR*, pages 7546–7555. (page 8, 9, 10, 30, 45, 53, 71)

[34] Kobler, E., Klatzer, T., Hammernik, K., and Pock, T. (2017). Variational networks: Connecting variational methods and deep learning. In *Pattern Recognition*, volume 10496, pages 281–293. Springer International Publishing. (page 10)

[35] Kunisch, K. and Pock, T. (2013). A bilevel optimization approach for parameter learning in variational models. *SIAM Journal on Imaging Sciences*, 6(2):938–983. (page 8)

[36] Lai, W.-S., Huang, J.-B., Hu, Z., Ahuja, N., and Yang, M.-H. (2016). A comparative study for single image blind deblurring. In *IEEE Conferene on Computer Vision and Pattern Recognition*. (page 64, 65)

[37] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551. (page 10)

[38] Lecun, Y., Chopra, S., Hadsell, R., Ranzato, M., and Huang, F. (2006). *A tutorial on energy-based learning*. MIT Press. (page 2)

[39] LeCun, Y., Cortes, C., and Burges, C. (2010). Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2. (page 33)

[40] Li, H., Schwab, J., Antholzer, S., and Haltmeier, M. (2020). NETT: solving inverse problems with deep neural networks. *Inverse Problems*, 36(6):065005. (page 8)

[41] Lunz, S., Öktem, O., and Schönlieb, C.-B. (2018). Adversarial regularizers in inverse problems. In *Advances in Neural Information Processing Systems 31*, pages 8507–8516. Curran Associates, Inc. (page 8)

[42] Mayoraz, E. and Alpaydin, E. (1999). Support vector machines for multi-class classification. In *Engineering Applications of Bio-Inspired Artificial Neural Networks*, pages 833–842. Springer. (page 25)

[43] Meyer, Y. (1993). *Wavelets and Operators*, volume 1 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press. (page 6)

[44] Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. (2018). Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*. (page 72)

[45] Neal, R. (2011). MCMC using hamiltonian dynamics. In *Handbook of Markov Chain Monte Carlo*, volume 20116022. Chapman and Hall/CRC. (page 8)

[46] Nitzberg, M., Mumford, D., and Shiota, T. (1993). Finding contours and junctions. In *Filtering, Segmentation and Depth*, pages 33–49. Springer. (page 8)

[47] Orieux, F., Giovannelli, J.-F., and Rodet, T. (2010). Bayesian estimation of regularization and point spread function parameters for wiener–hunt deconvolution. *Journal of the Optical Society of America A*, 27(7):1593. (page 5, 64, 65, 68)

[48] Peng Xu and Chan, A. K. (2003). Support vector machines for multi-class signal classification with unbalanced samples. In *Proceedings of the International Joint Conference on Neural Networks, 2003.*, volume 2, pages 1116–1119 vol.2. (page 25)

[49] Reeves, S. J. (2014). Chapter 6 - image restoration: Fundamentals of image restoration. In *Academic Press Library in Signal Processing*, volume 4, pages 165–192. Elsevier. (page 1)

[50] Roth, S. and Black, M. (2005). Fields of experts: A framework for learning image priors. In *CVPR*, volume 2, pages 860–867. IEEE. (page 8)

[51] Rudin, L. I., Osher, S., and Fatemi, E. (1992). Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259–268. (page 7)

[52] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536. (page 10)

[53] Schmidt, U. and Roth, S. (2014). Shrinkage fields for effective image restoration. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2774–2781. IEEE. (page 9)

[54] Shen, J., Kang, S. H., and Chan, T. F. (2003). Euler's elastica and curvature-based inpainting. *SIAM Journal on Applied Mathematics*, 63(2):564–592. (page 8)

[55] Simoncelli, E. and Adelson, E. (1996). Noise removal via bayesian wavelet coring. In *Proceedings of 3rd IEEE International Conference on Image Processing*, volume 1, pages 379–382. IEEE. (page 6)

[56] Taskar, B., Chatalbashev, V., Koller, D., and Guestrin, C. (2005). Learning structured prediction models: A large margin approach. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, page 896–903. Association for Computing Machinery. (page 23)

[57] Taskar, B., Guestrin, C., and Koller, D. (2003). Max-margin markov networks. In *Proceedings of the 16th International Conference on Neural Information Processing Systems*, NIPS'03, page 25–32. MIT Press. (page 23)

[58] Tieleman, T. (2008). Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 1064–1071. Association for Computing Machinery. (page 21)

[59] Tsochantaridis, I., Joachims, T., Hofmann, T., and Altun, Y. (2005). Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6(50):1453–1484. (page 25, 27)

[60] Vapnik, V., Golowich, S., and Smola, A. (1997). Support vector method for function approximation, regression estimation and signal processing. In *Advances in Neural Information Processing Systems*, volume 9, pages 281–287. MIT Press. (page 25)

[61] Welling, M. and Teh, Y. W. (2011). Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 681–688. Omnipress. (page 8, 21)

[62] Wiener, N. (1964). *The Wiener RMS (Root Mean Square) Error Criterion in Filter Design and Prediction*, pages 129–148. MIT Press. (page 4)

[63] Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. (page 39)

[64] Yang, W., Zhang, X., Tian, Y., Wang, W., and Xue, J.-H. (2019). Deep learning for single image super-resolution: A brief review. *IEEE Transactions on Multimedia*, 21(12):3106–3121. (page 1)

[65] Zhang, K., Zuo, W., Gu, S., and Zhang, L. (2017). Learning deep cnn denoiser prior for image restoration. In *CVPR*, pages 2808–2817. (page 10)

[66] Zhang, K., Zuo, W., and Zhang, L. (2018). FFDNet: Toward a fast and flexible solution for CNN based image denoising. *IEEE Transactions on Image Processing*, 27(9):4608–4622. (page 10)