MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Deep Gaussian Conditional Random Field Network: A Model-based Deep Network for Discriminative Denoising

Vemulapalli, R.; Tuzel, C.O.; Liu, M.-Y.

TR2016-079 June 2016

Abstract

We propose a novel end-to-end trainable deep network architecture for image denoising based on a Gaussian Conditional Random Field (GCRF) model. In contrast to the existing discriminative denoising methods that train a separate model for each individual noise level, the proposed deep network explicitly models the input noise variance and hence is capable of handling a range of noise levels. Our deep network, which we refer to as deep GCRF network, consists of two sub-networks: (i) a parameter generation network that generates the pairwise potential parameters based on the noisy input image, and (ii) an inference network whose layers perform the computations involved in an iterative GCRF inference procedure. We train two deep GCRF networks (each network operates over a range of noise levels: one for low input noise levels and one for high input noise levels) discriminatively by maximizing the peak signal-to-noise ratio measure. Experiments on Berkeley segmentation and PASCALVOC datasets show that the proposed approach produces results on par with the state of-the-art without training a separate network for each individual noise level.

IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

Deep Gaussian Conditional Random Field Network: A Model-based Deep Network for Discriminative Denoising

Raviteja Vemulapalli
Center for Automation Research, UMIACS
University of Maryland, College Park

Oncel Tuzel, Ming-Yu Liu
Mitsubishi Electric Research Laboratories
Cambridge, MA

Abstract

We propose a novel end-to-end trainable deep network architecture for image denoising based on a Gaussian Conditional Random Field (GCRF) model. In contrast to the existing discriminative denoising methods that train a separate model for each individual noise level, the proposed deep network explicitly models the input noise variance and hence is capable of handling a range of noise levels. Our deep network, which we refer to as deep GCRF network, consists of two sub-networks: (i) a parameter generation network that generates the pairwise potential parameters based on the noisy input image, and (ii) an inference network whose layers perform the computations involved in an iterative GCRF inference procedure. We train two deep GCRF networks (each network operates over a range of noise levels: one for low input noise levels and one for high input noise levels) discriminatively by maximizing the peak signal-to-noise ratio measure. Experiments on Berkeley segmentation and PASCALVOC datasets show that the proposed approach produces results on par with the state-of-the-art without training a separate network for each individual noise level.

1. Introduction

In the recent past, deep networks have been successfully used in various image processing and computer vision applications [3, 12, 34]. Their success can be attributed to several factors such as their ability to represent complex input-output relationships, feed-forward nature of their inference (no need to solve an optimization problem during run time), availability of large training datasets, etc. One of the positive aspects of deep networks is that fairly general architectures composed of fully-connected or convolutional layers have been shown to work reasonably well across a wide range of applications. However, these general architectures do not use problem domain knowledge which could be very helpful in some of the applications.

For example, in the case of image denoising, it has been recently shown that conventional multilayer perceptrons (MLP) are not very good at handling multiple levels of input noise [3]. When a single multilayer perceptron was trained to handle multiple input noise levels (by providing the noise variance as an additional input to the network), it produced inferior results compared to the state-of-the-art BM3D [6] approach. In contrast to this, the EPLL framework of [40], which is a model-based approach, has been shown to work well across a range of noise levels. These results suggest that we should work towards bringing deep networks and model-based approaches together. Motivated by this, in this work, we propose a new deep network architecture for image denoising based on a Gaussian conditional random field model. The proposed network explicitly models the input noise variance and hence is capable of handling a range of noise levels.

Gaussian Markov Random Fields (GMRFs) [29] are popular models for various structured inference tasks such as denoising, inpainting, super-resolution and depth estimation, as they model continuous quantities and can be efficiently solved using linear algebra routines. However, the performance of a GMRF model depends on the choice of pairwise potential functions. For example, in the case of image denoising, if the potential functions for neighboring pixels are homogeneous (i.e., identical everywhere), then the GMRF model can result in blurred edges and over-smoothed images. Therefore, to improve the performance of a GMRF model, the pairwise potential function parameters should be chosen according to the image being processed. A GMRF model that uses data-dependent potential function parameters is referred to as Gaussian Conditional Random Field (GCRF) [35].

Image denoising using a GCRF model consists of two steps: a *parameter selection step* in which the potential function parameters are chosen based on the input image, and an *inference step* in which energy minimization is performed for the chosen parameters. In this work, we propose a novel model-based deep network architecture, which we refer to as *deep GCRF network*, by converting both the

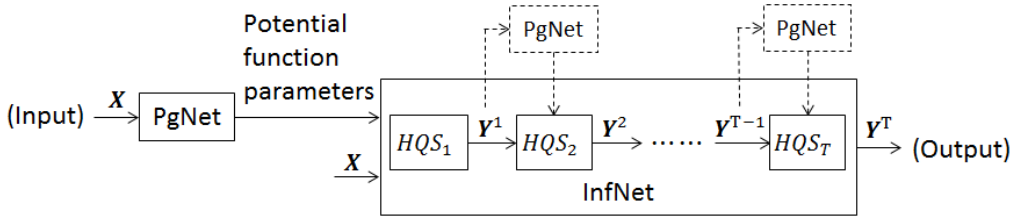

Figure 1: The proposed deep GCRF network: Parameter generation network (PgNet) followed by inference network (InfNet). The PgNets in dotted boxes are the additional parameter generation networks introduced after each HQS iteration.

parameter selection and inference steps into feed-forward networks.

The proposed deep GCRF network consists of two sub-networks: a *parameter generation network (PgNet)* that generates appropriate potential function parameters based on the input image, and an *inference network (InfNet)* that performs energy minimization using the potential function parameters generated by PgNet. Since directly generating the potential function parameters for an entire image is very difficult (as the number of pixels could be very large), we construct a full-image pairwise potential function indirectly by combining potential functions defined on image patches. If we use $d \times d$ patches, then our construction defines a graphical model in which each pixel is connected to its $(2d-1) \times (2d-1)$ spatial neighbors. This construction is motivated by the recent EPLL framework of [40]. Our PgNet directly operates on each $d \times d$ input image patch and chooses appropriate parameters for the corresponding potential function.

Though the energy minimizer can be obtained in closed form for GCRF, it involves solving a linear system with number of variables equal to the number of image pixels (usually of the order of 10^6). Solving such a large linear system could be computationally prohibitive, especially for dense graphs (each pixel is connected to 224 neighbors when 8×8 image patches are used). Hence, in this work, we use an iterative optimization approach based on *Half Quadratic Splitting* (HQS) [11, 19, 36, 40] for designing our inference network. Recently, this approach has been shown to work very well for image restoration tasks even with very few (5-6) iterations [40]. Our inference network consists of a new type of layer, which we refer to as HQS layer, that performs the computations involved in a HQS iteration.

Combining the parameter generation and inference networks, we get our deep GCRF network shown in Figure 1. Note that using appropriate pairwise potential functions is crucial for the success of GCRF. Since PgNet operates on the noisy input image, it becomes increasingly difficult to generate good potential function parameters as the image noise increases. To address this issue, we introduce an additional PgNet after each HQS iteration as shown in dotted boxes in Figure 1. Since we train this deep GCRF network discriminatively in an end-to-end fashion, even if the first PgNet fails to generate good potential function parameters, the later PgNets can learn to generate appropriate parameters based on partially restored images.

Contributions:

- We propose a new end-to-end trainable deep network architecture for image denoising based on a GCRF model. In contrast to the existing discriminative denoising methods that train a separate model for each individual noise level, the proposed network explicitly models the input noise variance and hence is capable of handling a range of noise levels.
- We propose a differentiable parameter generation network that generates the GCRF pairwise potential parameters based on the noisy input image.
- We unroll a half quadratic splitting-based iterative GCRF inference procedure into a deep network and train it jointly with our parameter generation network.
- We show that the proposed approach produces results on par with the state-of-the-art without training a separate network for each individual noise level

2. Related Work

Gaussian CRF: GCRFs were first introduced in [35] by modeling the parameters of the conditional distribution of output given input as a function of the input image. The precision matrix associated with each image patch was modeled as a linear combination of twelve derivative filter-based matrices. The combination weights were chosen as a parametric function of the responses of the input image to a set of oriented edge and bar filters, and the parameters were learned using discriminative training. This GCRF model was extended to Regression Tree Fields (RTFs) in [18], where regression trees were used for selecting the parameters of Gaussians defined over image patches. These regression trees used responses of the input image to various hand-chosen filters for selecting an appropriate leaf node for each image patch. This RTF-based model was trained by iteratively growing the regression trees and optimizing the

Gaussian parameters at leaf nodes. Recently, a cascade of RTFs [30] has also been used for image restoration tasks. In contrast to the RTF-based approaches, all the components of our network are differentiable, and hence it can be trained end-to-end using standard gradient-based techniques.

Recently, [31] proposed a cascade of shrinkage fields for image restoration tasks. They learned a separate filter bank and shrinkage function for each stage of their cascade using discriminative training. Though this model can also be seen as a cascade of GCRFs, the filter banks and shrinkage functions used in the cascade do not depend on the noisy input image during test time. In contrast to this, the pairwise potential functions used in our GCRF model are generated by our PgNets based on the noisy input image.

Our work is also related to the EPLL framework of [40], which decomposed the full-image Gaussian model into patch-based Gaussians, and used HQS iterations for GCRF inference. Following are the main differences between EPLL and this work: (i) We propose a new deep network architecture which combines HQS iterations with a differentiable parameter generation network. (ii) While EPLL chooses the potential parameters for each image patch as one of the K possible matrices, we construct each potential parameter matrix as a convex combination of K base matrices. (iii) While EPLL learns the K possible potential parameter matrices in a generative fashion by fitting a Gaussian Mixture Model (GMM) to clean image patches, we learn the K base matrices in a discriminative fashion by end-to-end training of our deep network. As shown later in the experiments section, our discriminative model clearly outperforms the generatively trained EPLL.

Denoising: Image denoising is one of the oldest problems in image processing and various denoising algorithms have been proposed over the past several years. Some of the most popular algorithms include wavelet shrinkage [33], fields of experts [28], Gaussian scale mixtures [26], BM3D [6], non-linear diffusion process-based approaches [5, 15, 25], sparse coding-based approaches [7, 8, 9, 22], weighted nuclear norm minimization (WNNM) [14], and non-local Bayesian denoising [20]. Among these, BM3D is currently the most widely-used state-of-the-art denoising approach. It is a well-engineered algorithm that combines non-local patch statistics with collaborative filtering.

Denoising with neural networks: Recently, various deep neural network-based approaches have also been proposed for image denoising [1, 3, 17, 37, 38]. While [17] used a convolutional neural network (CNN), [3, 38] used multilayer perceptrons, and [1, 37] used stacked sparse denoising autoencoders (SSDA). Among these MLP [3] has been shown to work very well outperforming the BM3D approach. However, none of these deep networks explicitly model the input noise variance, and hence are not good at handling multiple noise levels. In all these works, a different network was trained for each noise level.

Unfolding inference as a deep network: The proposed approach is also related to a class of algorithms that learn model parameters discriminatively by back-propagating the gradient through a fixed number of inference steps. In [2], the fields of experts [28] MRF model was discriminatively trained for image denoising by unfolding a fixed number of gradient descent inference steps. In [27], message-passing inference machines were trained for structured prediction tasks by considering the belief propagation-based inference of a discrete graphical model as a sequence of predictors. In [13], a feed-forward sparse code predictor was trained by unfolding a coordinate descent based sparse coding inference algorithm. In [32, 39], deep CNNs and discrete graphical models were jointly trained by unfolding the discrete mean-field inference. In [16], a new kind of non-negative deep network was introduced by deep unfolding of non-negative factorization model. Recently, [5] revisited the classical non-linear diffusion process [24] by modeling it using several parameterized linear filters and influential functions. The parameters of this diffusion process were learned discriminatively by back-propagating the gradient through a fixed number of diffusion process iterations. Though this diffusion process-based approach has been shown to work well for the task of image denoising, it uses a separate model for each noise level.

In this work, we design our inference network using HQS-based inference of a Gaussian CRF model, resulting in a different network architecture compared to the above unfolding works. In addition to this inference network, our deep GCRF network also consists of other sub-networks used for modeling the GCRF pairwise potentials.

Notations: We use bold face capital letters to denote matrices and bold face small letters to denote vectors. We use $\text{vec}(\mathbf{A})$, $\mathbf{A}^\top$ and $\mathbf{A}^{-1}$ to denote the column vector representation, transpose and inverse of a matrix $\mathbf{A}$, respectively. $\mathbf{A} \succeq 0$ means $\mathbf{A}$ is symmetric and positive semidefinite.

3. Gaussian Conditional Random Field

Let $\mathbf{X}$ be the given (noisy) input image and $\mathbf{Y}$ be the (clean) output image that needs to be inferred. Let $\mathbf{X}(i,j)$ and $\mathbf{Y}(i,j)$ represent the pixel (i,j) in images $\mathbf{X}$ and $\mathbf{Y}$, respectively. In this work, we model the conditional probability density $p(\mathbf{Y}|\mathbf{X})$ as a Gaussian distribution given by $p\left(\mathbf{Y}|\mathbf{X}\right) \propto \exp\left\{-E\left(\mathbf{Y}|\mathbf{X}\right)\right\}$, where

$$\left.\begin{aligned} E\left(\mathbf{Y}|\mathbf{X}\right) = \frac{1}{2\sigma^2}\sum_{ij}\left[\mathbf{Y}(i,j)-\mathbf{X}(i,j)\right]^2 \end{aligned}\right\} \coloneqq E_d\left(\mathbf{Y}|\mathbf{X}\right)$$
$$\left.\begin{aligned} + \frac{1}{2}\text{vec}(\mathbf{Y})^\top \mathbf{Q}(\mathbf{X})\text{vec}(\mathbf{Y}) \end{aligned}\right\} \coloneqq E_p\left(\mathbf{Y}|\mathbf{X}\right). \tag{1}$$

Here, σ^2 is the input noise variance and $\mathbf{Q}(\mathbf{X}) \succeq 0$ are the input-dependent parameters of the quadratic pairwise potential function $E_p(\mathbf{Y}|\mathbf{X})$ defined over the image $\mathbf{Y}$. Note that if the pairwise potential parameters Q are constant, then this model can be interpreted as a generative model with E_d as the data term, E_p as the prior term and $p(\mathbf{Y}|\mathbf{X})$ as the posterior. Hence, our GCRF is a discriminative model inspired by a generative Gaussian model.

3.1. Patch-based pairwise potential functions

Directly choosing the (positive semi-definite) pairwise potential parameters $\mathbf{Q}(\mathbf{X})$ for an entire image $\mathbf{Y}$ is very challenging since the number of pixels in an image could be of the order of 10^6. Hence, motivated by [40], we construct the (full-image) pairwise potential function E_p by combining patch-based pairwise potential functions.

Let $\mathbf{x}_{ij}$ and $\mathbf{y}_{ij}$ be $d^2 \times 1$ column vectors representing the $d \times d$ patches centered on pixel (i, j) in images $\mathbf{X}$ and $\mathbf{Y}$, respectively. Let $\bar{\mathbf{x}}_{ij} = \mathbf{G}\mathbf{x}_{ij}$ and $\bar{\mathbf{y}}_{ij} = \mathbf{G}\mathbf{y}_{ij}$ be the mean-subtracted versions of vectors $\mathbf{x}_{ij}$ and $\mathbf{y}_{ij}$, respectively, where $\mathbf{G} = \mathbf{I} - \frac{1}{d^2}\mathbf{1}\mathbf{1}^\top$ is the mean subtraction matrix. Here, $\mathbf{1}$ is the $d^2 \times 1$ vector of ones and $\mathbf{I}$ is the $d^2 \times d^2$ identity matrix. Let

$$V(\bar{\mathbf{y}}_{ij}|\bar{\mathbf{x}}_{ij}) = \frac{1}{2}\bar{\mathbf{y}}_{ij}^\top \left(\mathbf{\Sigma}_{ij}(\bar{\mathbf{x}}_{ij})\right)^{-1} \bar{\mathbf{y}}_{ij},\ \mathbf{\Sigma}_{ij}(\bar{\mathbf{x}}_{ij}) \succeq 0, \quad (2)$$

be a quadratic pairwise potential function defined on patch $\bar{\mathbf{y}}_{ij}$, with $\mathbf{\Sigma}_{ij}(\bar{\mathbf{x}}_{ij})$ being the corresponding (input) data-dependent parameters. Combining the patch-based potential functions at all the pixels, we get the following full-image pairwise potential function:

$$\begin{aligned} E_p(\mathbf{Y}|\mathbf{X}) &= \frac{1}{d^2}\sum_{ij} V(\bar{\mathbf{y}}_{ij}|\bar{\mathbf{x}}_{ij}) \\ &= \frac{1}{2d^2}\sum_{ij} \mathbf{y}_{ij}^\top \mathbf{G}^\top \left(\mathbf{\Sigma}_{ij}(\bar{\mathbf{x}}_{ij})\right)^{-1} \mathbf{G}\mathbf{y}_{ij}. \end{aligned} \quad (3)$$

Note that since we are using all $d \times d$ image patches, each pixel appears in d^2 patches that are centered on its $d \times d$ neighbor pixels. In every patch, each pixel interacts with all the d^2 pixels in that patch. This effectively defines a graphical model of neighborhood size $(2d-1) \times (2d-1)$ on image $\mathbf{Y}$.

3.2. Inference

Given the (input) data-dependent parameters $\{\mathbf{\Sigma}_{ij}(\bar{\mathbf{x}}_{ij})\}$ of the pairwise potential function $E_p(\mathbf{Y}|\mathbf{X})$, the Gaussian CRF inference solves the following optimization problem:

$$\mathbf{Y}^* = \underset{\mathbf{Y}}{\operatorname{argmin}} \sum_{ij} \left\{ \begin{array}{l} \frac{d^2}{\sigma^2}\left[\mathbf{Y}(i,j) - \mathbf{X}(i,j)\right]^2 \\ + \mathbf{y}_{ij}^\top \mathbf{G}^\top \left(\mathbf{\Sigma}_{ij}(\bar{\mathbf{x}}_{ij})\right)^{-1} \mathbf{G}\mathbf{y}_{ij} \end{array} \right\}. \quad (4)$$

Note that the optimization problem (4) is an unconstrained quadratic program and hence can be solved in closed form. However, the closed form solution for $\mathbf{Y}$ requires solving a linear system of equations with number of variables equal to the number of image pixels. Since solving such linear systems could be computationally prohibitive for large images, in this work, we use a half quadratic splitting-based iterative optimization method, that has been recently used in [40] for solving the above optimization problem. This approach allows for efficient optimization by introducing auxiliary variables.

Let $\mathbf{z}_{ij}$ be an auxiliary variable corresponding to the patch $\mathbf{y}_{ij}$. In half quadratic splitting method, the cost function in (4) is modified to

$$J(\mathbf{Y}, \{\mathbf{z}_{ij}\}, \beta) = \sum_{ij} \left\{ \begin{array}{l} \frac{d^2}{\sigma^2}\left[\mathbf{Y}(i,j) - \mathbf{X}(i,j)\right]^2 \\ + \beta\|\mathbf{y}_{ij} - \mathbf{z}_{ij}\|_2^2 \\ + \mathbf{z}_{ij}^\top \mathbf{G}^\top \left(\mathbf{\Sigma}_{ij}(\bar{\mathbf{x}}_{ij})\right)^{-1} \mathbf{G}\mathbf{z}_{ij} \end{array} \right\}. \quad (5)$$

Note that as $\beta \to \infty$, the patches $\{\mathbf{y}_{ij}\}$ are restricted to be equal to the auxiliary variables $\{\mathbf{z}_{ij}\}$, and the solutions of (4) and (5) converge. For a fixed value of β, the cost function J can be minimized by alternatively optimizing for $\mathbf{Y}$ and $\{\mathbf{z}_{ij}\}$. If we fix $\mathbf{Y}$, then the optimal $\mathbf{z}_{ij}$ is given by

$$\begin{aligned} f(\mathbf{y}_{ij}) &= \underset{\mathbf{z}_{ij}}{\operatorname{argmin}} \left\{ \begin{array}{l} \mathbf{z}_{ij}^\top \mathbf{G}^\top \left(\mathbf{\Sigma}_{ij}(\bar{\mathbf{x}}_{ij})\right)^{-1} \mathbf{G}\mathbf{z}_{ij} \\ + \beta\|\mathbf{y}_{ij} - \mathbf{z}_{ij}\|_2^2 \end{array} \right\} \\ &= \left(\mathbf{G}^\top \left(\mathbf{\Sigma}_{ij}(\bar{\mathbf{x}}_{ij})\right)^{-1} \mathbf{G} + \beta\mathbf{I}\right)^{-1} \beta\mathbf{y}_{ij} \\ &= \left(\mathbf{I} - \mathbf{G}^\top \left(\beta\mathbf{\Sigma}_{ij}(\bar{\mathbf{x}}_{ij}) + \mathbf{G}\mathbf{G}^\top\right)^{-1} \mathbf{G}\right) \mathbf{y}_{ij}. \end{aligned} \quad (6)$$

The last equality in (6) follows from Woodbury matrix identity. If we fix $\{\mathbf{z}_{ij}\}$, then the optimal $\mathbf{Y}(i,j)$ is given by

$$\begin{aligned} g(\{\mathbf{z}_{ij}\}) &= \underset{\mathbf{Y}(i,j)}{\operatorname{argmin}} \left\{ \begin{array}{l} \frac{d^2}{\sigma^2}\left[\mathbf{Y}(i,j) - \mathbf{X}(i,j)\right]^2 \\ + \beta \sum_{p,q=-\lfloor\frac{d-1}{2}\rfloor}^{\lceil\frac{d-1}{2}\rceil} \left[\mathbf{Y}(i,j) - \mathbf{z}_{pq}(i,j)\right]^2 \end{array} \right\} \\ &= \frac{\mathbf{X}(i,j)}{1+\beta\sigma^2} + \frac{\beta\sigma^2}{(1+\beta\sigma^2)d^2} \sum_{p,q=-\lfloor\frac{d-1}{2}\rfloor}^{\lceil\frac{d-1}{2}\rceil} \mathbf{z}_{pq}(i,j), \end{aligned} \quad (7)$$

where $\lfloor\ \rfloor, \lceil\ \rceil$ are the floor and ceil operators, respectively, and $\mathbf{z}_{pq}(i,j)$ is the intensity value of pixel (i,j) according to the auxiliary patch $\mathbf{z}_{pq}$.

In half quadratic splitting approach, the optimization steps (6) and (7) are repeated while increasing the value of β in each iteration. This iterative approach has been shown to work well in [40] for image restorations tasks even with few (5-6) iterations.

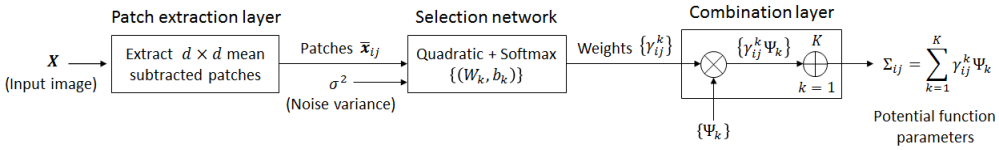

Figure 2: Parameter generation network: Mean subtracted patches $\bar{\mathbf{x}}_{ij}$ extracted from the input image $\mathbf{X}$ are used to compute the combination weights $\{\gamma_{ij}^k\}$, which are used for generating the pairwise potential parameters $\{\mathbf{\Sigma}_{ij}\}$.

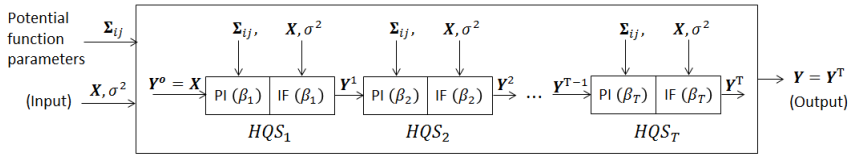

Figure 3: Inference network uses the pairwise potential parameters $\{\mathbf{\Sigma}_{ij}(\bar{\mathbf{x}}_{ij})\}$ generated by the PgNet and performs T HQS iterations.

4. Deep Gaussian CRF network

As mentioned earlier, the proposed deep GCRF network consists of the following two components:

- **Parameter generation network:** This network takes the noisy image $\mathbf{X}$ as input and generates the parameters $\{\mathbf{\Sigma}_{ij}(\bar{\mathbf{x}}_{ij})\}$ of pairwise potential function $E_p(\mathbf{Y}|\mathbf{X})$.
- **Inference network:** This network performs Gaussian CRF inference using the pairwise potential parameters $\{\mathbf{\Sigma}_{ij}(\bar{\mathbf{x}}_{ij})\}\}$ given by the parameter generation network.

4.1. Parameter generation network

We model the pairwise potential parameters $\{\mathbf{\Sigma}_{ij}\}$ as convex combinations of K symmetric positive semidefinite matrices $\mathbf{\Psi}_1, \ldots, \mathbf{\Psi}_K$:

$$\mathbf{\Sigma}_{ij} = \sum_k \gamma_{ij}^k \mathbf{\Psi}_k, \; \gamma_{ij}^k \geq 0, \; \sum_k \gamma_{ij}^k = 1. \tag{8}$$

The combination weights $\{\gamma_{ij}^k\}$ are computed from the mean-subtracted input image patches $\{\bar{\mathbf{x}}_{ij}\}$ using the following two layer selection network:

$$\begin{aligned} &\text{Layer 1 - Quadratic layer : For } k = 1, 2, \ldots, K \\ &\qquad s_{ij}^k = -\frac{1}{2} \bar{\mathbf{x}}_{ij}^\top \left(\mathbf{W}_k + \sigma^2 \mathbf{I}\right)^{-1} \bar{\mathbf{x}}_{ij} + b_k. \end{aligned} \tag{9}$$

$$\begin{aligned} &\text{Layer 2 - Softmax layer : For } k = 1, 2, \ldots, K \\ &\qquad \gamma_{ij}^k = e^{s_{ij}^k} / \sum_{p=1}^{K} e^{s_{ij}^p}. \end{aligned} \tag{10}$$

Figure 2 shows the overall parameter generation network which includes a patch extraction layer, a selection network and a combination layer. Here, $\{(\mathbf{W}_k \succeq 0, \mathbf{\Psi}_k \succeq 0, b_k)\}$ are the network parameters, and σ^2 is the noise variance.

Our choice of the above quadratic selection function is motivated by the following two reasons: (i) Since the selection network operates on mean-subtracted patches, it should be symmetric, i.e., both $\bar{\mathbf{x}}$ and $-\bar{\mathbf{x}}$ should have the same combination weights $\{\gamma^k\}$. To achieve this, we compute each s^k as a quadratic function of $\bar{\mathbf{x}}$. (ii) Since we are computing the combination weights using the noisy image patches, the selection network should be robust to input noise. To achieve this, we include the input noise variance σ^2 in the computation of $\{s^k\}$. We choose the particular form $\left(\mathbf{W}_k + \sigma^2 \mathbf{I}\right)^{-1}$ because in this case, we can (roughly) interpret the computation of $\{s^k\}$ as evaluating Gaussian log likelihoods. If we interpret $\{\mathbf{W}_k\}$ as covariance matrices associated with clean image patches, then $\{\mathbf{W}_k + \sigma^2 \mathbf{I}\}$ can be interpreted as covariance matrices associated with noisy image patches.

4.2. Inference network

We use the half quadratic splitting method described in Section 3.2 to create our inference network. Each layer of the inference network, also referred to as a HQS layer, implements one half quadratic splitting iteration. Each HQS layer consists of the following two sub-layers:

- **Patch inference layer (PI):** This layer uses the current image estimate $\mathbf{Y}^t$ and computes the auxiliary patches $\{\mathbf{z}_{ij}\}$ using $f(\mathbf{y}_{ij})$ given in (6).
- **Image formation layer (IF):** This layer uses the auxiliary patches $\{\mathbf{z}_{ij}\}$ given by the PI layer and computes the next image estimate $\mathbf{Y}^{t+1}$ using $g(\{\mathbf{z}_{ij}\})$ given in (7).

Let $\{\beta_1, \beta_2, \ldots, \beta_T\}$ be the β schedule for half quadratic splitting. Then, our inference network consists of T HQS layers as shown in Figure 3. Here, $\mathbf{X}$ is the input image with noise variance σ^2, and $\{\mathbf{\Sigma}_{ij}(\bar{\mathbf{x}}_{ij})\}$ are the (data-dependent) pairwise potential parameters generated by the PgNet.

Remark: Since our inference network implements a fixed number of HQS iterations, its output may not be optimal for (4). However, since we train our parameter generation and inference networks jointly in a discriminative fashion, the PgNet will learn to generate appropriate pairwise potential parameters such that the output after a fixed number of HQS iterations would be close to the desired output.

4.3. GCRF network

Combining the above parameter generation and inference networks, we get our full Gaussian CRF network with parameters $\{(\mathbf{W}_k \succeq 0, \mathbf{\Psi}_k \succeq 0, b_k)\}$. Note that this GCRF network has various new types of layers that use quadratic functions, matrix inversions and multiplicative interactions, which are quite different from the computations used in standard deep networks.

Additional PgNets: Note that using appropriate pairwise potential functions is crucial for the success of GCRF. Since the parameter generation network operates on the noisy input image $\mathbf{X}$, it is very difficult to generate good parameters at high noise levels (even after incorporating the noise variance σ^2 into the selection network). To overcome this issue, we introduce an additional PgNet after each HQS iteration (shown with dotted boxes in Figure 1). The rationale behind adding these additional PgNets is that even if the first PgNet fails to generate good parameters, the later PgNets could generate appropriate parameters using the partially restored images. Our final deep GCRF network consists of T PgNets and T HQS layers as shown in Figure 1.

Training: We train the proposed deep GCRF network end-to-end in a discriminative fashion by maximizing the average PSNR measure. We use standard back-propagation to compute the gradient of the network parameters. Please refer to the appendix for relevant derivative formulas. Note that we have a constrained optimization problem here because of the symmetry and positive semi-definiteness constraints on the network parameters $\{\mathbf{W}_k\}$ and $\{\mathbf{\Psi}_k\}$. We convert this constrained problem into an unconstrained one by parametrizing $\mathbf{W}_k$ and $\mathbf{\Psi}_k$ as $\mathbf{W}_k = \mathbf{P}_k\mathbf{P}_k^\top, \mathbf{\Psi}_k = \mathbf{R}_k\mathbf{R}_k^\top$, where $\mathbf{P}_k$ and $\mathbf{R}_k$ are lower triangular matrices, and use limited memory BFGS [21] for optimization.

5. Experiments

In this section, we use the proposed deep GCRF network for image denoising. We trained our network using a dataset of 400 images (200 images from BSD300 [23] training set and 200 images from PASCALVOC 2012 [10] dataset), and evaluated it using a dataset of 300 images (100 images from BSD300 [23] test set and 200 images from PASCALVOC 2012 [10] dataset). For our experiments, we used white Gaussian noise of various standard deviations. For realistic evaluation, all the images were quantized to [0-255] range after adding the noise. The noisy and clean images used for training and testing can be downloaded from http://ravitejav.weebly.com/gcrfdenosing.html. We use the standard PSNR measure for quantitative evaluation.

Though we use Gaussian noise, due to quantization (clipping to 0-255 range), the noise characteristics deviate from being a Gaussian as the noise variance increases. To cope up with this variation in noise characteristics, we trained two different networks, one for low input noise levels ($\sigma \leq 25$, noise reasonably close to a Gaussian after quantization) and one for high input noise levels ($25 < \sigma < 60$, noise far from being a Gaussian after quantization) [1]. For training the low noise network, we used $\sigma = [8, 13, 18, 25]$ and for training the high noise network, we used $\sigma = [30, 35, 40, 50]$. Note that both the networks were trained to handle a range of input noise levels. For testing, we varied the σ from 10 to 60 in intervals of 5.

We performed experiments with two patch sizes (5×5 and 8×8) [2], and the number of matrices $\mathbf{\Psi}_k$ was chosen as 200. Following [40], we used six HQS iterations with β values given by $\frac{1}{\sigma^2}[1, 4, 8, 16, 32, 64]$ [3]. To avoid overfitting, we regularized the network, by sharing the parameters $\{\mathbf{W}_k, \mathbf{\Psi}_k\}$ across all PgNets. We initialized the network parameters using the parameters of a GMM learned on clean image patches.

Table 1 compares the proposed deep GCRF network with various image denoising approaches on 300 test images. Here, DGCRF_5 and DGCRF_8 refer to the deep GCRF networks that use 5×5 and 8×8 patches, respectively. For each noise level, the top two PSNR values are shown in boldface style. Note that the CSF [31] and MLP [3] approaches train a different model for each noise level. Hence, for these approaches, we report the results only for those noise levels for which the corresponding authors have provided their trained models. As we can see, the proposed deep GCRF network clearly outperforms the ClusteringSR [7], EPLL [40], BM3D [6], NL-Bayes [20], NCSR [8] and CSF approaches on all noise levels, and the WNNM [14] approach on all noise levels except $\sigma = 10$ (where it performs equally well). Specifically, it produces significant improvement in the PSNR compared to the ClusteringSR (0.29 - 1.24 dB), EPLL (0.24 - 1.18 dB), BM3D (0.18 - 0.83 dB), NL-Bayes (0.10 - 1.27 dB), NCSR (0.11 - 1.07 dB) and WNNM (upto 1.0 dB) approaches. The CSF approach of [31], which also uses GCRFs, performs poorly (0.24 dB for $\sigma = 25$) compared to our deep network.

[1] When we tried training a single network for all noise levels, our training was mainly focusing on high noise data.

[2] We did not go beyond 8×8 due to memory and computation issues. We believe that bigger patches could further improve the performance.

[3] Optimizing the β values using a validation set may further improve our performance.

Test σ	10	15	20	25	30	35	40	45	50	55	60
ClusteringSR [7]	33.27	30.97	29.41	28.22	27.25	26.30	25.56	24.89	24.28	23.72	23.21
EPLL [40]	33.32	31.06	29.52	28.34	27.36	26.52	25.76	25.08	24.44	23.84	23.27
BM3D [6]	33.38	31.09	29.53	28.36	27.42	26.64	25.92	25.19	24.63	24.11	23.62
NL-Bayes [20]	33.46	31.11	29.63	28.41	27.42	26.57	25.76	25.05	24.39	23.77	23.18
NCSR [8]	33.45	31.20	29.56	28.39	27.45	26.32	25.59	24.94	24.35	23.85	23.38
WNNM [14]	**33.57**	31.28	29.70	28.50	27.51	26.67	25.92	25.22	24.60	24.01	23.45
CSF [31]	-	-	-	28.43	-	-	-	-	-	-	-
MLP [3]	33.43	-	-	**28.68**	-	**27.13**	-	-	**25.33**	-	-
$DGCRF_5$	33.53	**31.29**	**29.76**	28.58	**27.68**	26.95	**26.30**	**25.73**	25.23	**24.76**	**24.33**
$DGCRF_8$	**33.56**	**31.35**	**29.84**	**28.67**	**27.80**	**27.08**	**26.44**	**25.88**	**25.38**	**24.90**	**24.45**

Table 1: Comparison of various denoising approaches on 300 test images.

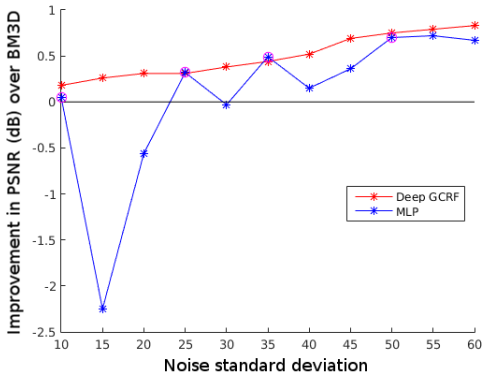

Figure 4: Sensitivity analysis of the MLP and the proposed approach. The noise levels for which MLP was trained are indicated using a circular marker.

When compared with MLP[3], which is the state-of-the-art deep networks-based denoising approach, we perform better for $\sigma = [10, 50]$, worse for $\sigma = 35$, and equally well for $\sigma = 25$. However, note that while [3] uses a different MLP for each specific noise level, we trained only two networks, each of which can handle a range of noise levels. In fact, our single low noise network is able to outperform the MLP trained for $\sigma = 10$ and perform as good as the MLP trained for $\sigma = 25$. This ability to handle a range of noise levels is one of the major benefits of the proposed deep network. Note that though we did not use the noise levels $\sigma = 10, 15, 20, 45$ during training, our networks performs very well for these σ. This shows that our networks are able to handle a range of noise levels rather than just fitting to the training σ. Also, our high noise network performs well for $\sigma = 55$ and 60 even though these values are out of its training range. This shows that the proposed *model-based* deep network can also generalize reasonably well for out-of-range noise levels.

We acknowledge that the comparisons in Table 1 may be biased since some of the competing methods are not designed for denoising quantized images. However, we believe that, for the denoising problem, using quantized images is a more realistic experimental setting than using unquantized images. Please refer to Table 2 for additional results on a benchmark dataset under the unquantized setting.

To analyze the sensitivity of the *non-model* based MLP approach to the deviation from training noise, we evaluated it on noise levels that are slightly (± 5) different from the training σ. The authors of [3] trained separate MLPs for $\sigma = 10, 25, 35, 50$ and 65. As reported in [3], training a single MLP to handle multiple noise levels gave inferior results. Figure 4 shows the improvement of the MLP approach over BM3D in terms of PSNR. For each noise level, we used the best performing model among $\sigma = 10, 25, 35, 50, 65$. As we can see, while the MLP approach does very well for the exact noise levels for which it was trained, it performs poorly if the test σ deviates from the training σ even by 5 units. This is a major limitation of the MLP approach since training a separate model for each individual noise level is not practical. In contrast to this, the proposed approach is able to cover a wide range of noise levels just using two networks.

Please note that the purpose of Figure 4 is not to compare the performance of our approach with MLP on noise levels that were not used in MLP training, which would be an unfair comparison. The only purpose of this figure is to show that, although very powerful, a network trained for a specific noise level is very sensitive.

Apart from our test set of 300 images, we also evaluated our low noise $DGCRF_8$ network on a smaller dataset of 68 images [28] which has been used in various existing works. Tables 2 and 3 compare the proposed deep GCRF network with various approaches on this dataset under the unquantized and quantized settings, respectively. For each noise level, the top two PSNR values are shown in boldface style. As we can see, the proposed approach outperforms all the other approaches except RTF_5 [30] and MLP [3] under the quantized setting, and TRD [5] under the unquantized setting. However, note that while we use a single network for both $\sigma = 15$ and $\sigma = 25$, the MLP, TRD and RTF_5 approaches trained their models specifically for individual noise levels.

Test σ	ARF [2]	LLSC [22]	EPLL [40]	opt-MRF [4]	ClusteringSR [7]	NCSR [8]	BM3D [6]	MLP [3]	WNNM [14]	CSF [31]	RTF_5 [30]	TRD [5]	DGCRF_8
15	30.70	31.27	31.19	31.18	31.08	31.19	31.08	-	31.37	31.24	-	**31.43**	**31.43**
25	28.20	28.70	28.68	28.66	28.59	28.61	28.56	28.85	28.83	28.72	28.75	**28.95**	**28.89**

Table 2: Comparison of various denoising approaches on 68 images (dataset of [28]) under the unquantized setting.

Test σ	LLSC [22]	EPLL [40]	opt-MRF [4]	ClusteringSR [7]	NCSR [8]	BM3D [6]	NL-Bayes [20]	MLP [3]	WNNM [14]	CSF [31]	RTF_5 [30]	DGCRF_8
15	31.09	31.11	31.06	30.93	31.13	31.03	31.06	-	31.20	-	-	**31.36**
25	28.24	28.46	28.40	28.26	28.41	28.38	28.43	**28.77**	28.48	28.53	**28.74**	28.73

Table 3: Comparison of various denoising approaches on 68 images (dataset of [28]) under the quantized setting.

Training time: We trained each DGCRF_8 network for three weeks (around 250 limited memory BFGS iterations).

Denoising time: The proposed DGCRF_8 network takes 4.4s for a 321×481 image on an NVIDIA Titan GPU using a MATLAB implementation.

6. Conclusions

In this work, we proposed a new end-to-end trainable deep network architecture for image denoising based on a Gaussian CRF model. The proposed network consists of a parameter generation network that generates appropriate potential function parameters based on the input image, and an inference network that performs approximate Gaussian CRF inference. Unlike the existing discriminative denoising approaches that train a separate model for each individual noise level, the proposed network can handle a range of noise levels as it explicitly models the input noise variance. We achieved results on par with the state-of-the-art by training two deep GCRF networks, one for low input noise levels and one for high input noise levels. In the future, we plan to use this network for other full-image inference tasks like super-resolution, depth estimation, etc.

Appendix

In this appendix, we show how to back-propagate the loss derivatives through the layers of our deep GCRF network. Please refer to the supplementary material for detailed derivations. Let L be the final loss function.

Backpropagation through the combination layer: Given the derivatives $dL/d\mathbf{\Sigma}_{ij}$ of the loss function L with respect to the pairwise potential parameters $\mathbf{\Sigma}_{ij}$, we can compute the derivatives of L with respect to the combination weights γ_{ij}^k and the matrices $\mathbf{\Psi}_k$ using

$$\frac{dL}{d\gamma_{ij}^k} = \text{trace}\left(\mathbf{\Psi}_k^\top \frac{dL}{d\mathbf{\Sigma}_{ij}}\right), \quad \frac{dL}{d\mathbf{\Psi}_k} = \sum_{ij} \gamma_{ij}^k \frac{dL}{d\mathbf{\Sigma}_{ij}}. \tag{11}$$

Backpropagation through the quadratic layer: Given the derivatives dL/ds_{ij}^k of the loss function L with respect to the quadratic layer output s_{ij}^k, we can compute the derivatives of L with respect to the selection network parameters $(\mathbf{W}_k, b_k)$ and the input patches $\bar{\mathbf{x}}_{ij}$ using:

$$\begin{aligned}
\frac{dL}{d\mathbf{W}_k} &= \left(\mathbf{W}_k + \sigma^2\mathbf{I}\right)^{-1} \left(\sum_{ij} \frac{dL}{ds_{ij}^k} \frac{\bar{\mathbf{x}}_{ij}\bar{\mathbf{x}}_{ij}^\top}{2}\right) \left(\mathbf{W}_k + \sigma^2\mathbf{I}\right)^{-1} \\
\frac{dL}{db_k} &= \sum_{ij} \frac{dL}{ds_{ij}^k}, \quad \frac{dL}{d\bar{\mathbf{x}}_{ij}} = -\sum_k \frac{dL}{ds_{ij}^k} \left(\mathbf{W}_k + \sigma^2\mathbf{I}\right)^{-1} \bar{\mathbf{x}}_{ij}.
\end{aligned} \tag{12}$$

Backpropagation through the patch inference layer: Given the derivatives $dL/d\mathbf{z}_{ij}$ of the loss function L with respect to the output of a patch inference layer, we can compute the derivatives of L with respect to its input patches $\mathbf{y}_{ij}$ and the pairwise potential parameters $\mathbf{\Sigma}_{ij}$ using

$$\begin{aligned}
\frac{dL}{d\mathbf{y}_{ij}} &= \left(\mathbf{I} - \mathbf{G}^\top(\beta\mathbf{\Sigma}_{ij} + \mathbf{G})^{-1}\mathbf{G}\right) \frac{dL}{d\mathbf{z}_{ij}}, \\
\frac{dL}{d\mathbf{\Sigma}_{ij}} &= \beta\left(\beta\mathbf{\Sigma}_{ij} + \mathbf{G}\right)^{-1} \mathbf{G} \frac{dL}{d\mathbf{z}_{ij}} \mathbf{y}_{ij}^\top \mathbf{G}^\top \left(\beta\mathbf{\Sigma}_{ij} + \mathbf{G}\right)^{-1}.
\end{aligned} \tag{13}$$

We skip the derivative formulas for other computations such as softmax, extracting mean-subtracted patches from an image, averaging in the image formation layer, etc., as they are standard operations.

References

[1] F. Agostinelli, M. R. Anderson, and H. Lee. Adaptive Multi-Column Deep Neural Networks with Application to Robust Image Denoising. In *NIPS*, 2013. 3

[2] A. Barbu. Training an Active Random Field for Real-Time Image Denoising. *IEEE Transactions on Image Processing*, 18(11):2451–2462, 2009. 3, 8

[3] H. C. Burger, C. J. Schuler, and S. Harmeling. Image Denoising: Can Plain Neural Networks Compete with BM3D? In *CVPR*, 2012. 1, 3, 6, 7, 8

[4] Y. Chen, T. Pock, R. Ranftl, and H. Bischof. Revisiting Loss-specific Training of Filter-based MRFs for Image Restoration. In *GCPR*, 2013. 8

[5] Y. Chen, W. Yu, and T. Pock. On Learning Optimized Reaction Diffusion Processes for Effective Image Restoration. In *CVPR*, 2015. 3, 7, 8

[6] K. Dabov, A. Foi, V. Katkovnik, and K. O. Egiazarian. Image Denoising by Sparse 3-D Transform-Domain Collaborative Filtering. *IEEE Transactions on Image Processing*, 16(8):2080–2095, 2007. 1, 3, 6, 7, 8

[7] W. Dong, X. Li, L. Zhang, and G. Shi. Sparsity-based Image Denoising via Dictionary Learning and Structural Clustering. In *CVPR*, 2011. 3, 6, 7, 8

[8] W. Dong, L. Zhang, G. Shi, and X. Li. Nonlocally Centralized Sparse Representation for Image Restoration. *IEEE Transactions on Image Processing*, 22(4):1620–1630, 2013. 3, 6, 7, 8

[9] M. Elad and M. Aharon. Image Denoising via Sparse and Redundant Representations Over Learned Dictionaries. *IEEE Transactions on Image Processing*, 15(12):3736–3745, 2006. 3

[10] M. Everingham, S. M. A. Eslami, L. V. Gool, C. K. I. Williams, J. M. Winn, and A. Zisserman. The Pascal Visual Object Classes Challenge: A Retrospective. *International Journal of Computer Vision*, 111(1):98–136, 2015. 6

[11] D. Geman and C. Yang. Nonlinear Image Recovery with Half-quadratic Regularization. *IEEE Transactions on Image Processing*, 5(7):932–946, 1995. 2

[12] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *CVPR*, 2014. 1

[13] K. Gregor and Y. LeCun. Learning fast approximations of sparse coding. In *ICML*, 2010. 3

[14] S. Gu, L. Zhang, W. Zuo, and X. Feng. Weighted Nuclear Norm Minimization with Application to Image Denoising. In *CVPR*, 2014. 3, 6, 7, 8

[15] M. R. Hajiaboli. An Anisotropic Fourth-Order Diffusion Filter for Image Noise Removal. *International Journal of Computer Vision*, 92(2):177–191, 2011. 3

[16] J. R. Hershey, J. L. Roux, and F. Weninger. Deep unfolding: Model-based inspiration of novel deep architectures. *CoRR*, abs/1409.2574, 2014. 3

[17] V. Jain and H. S. Seung. Natural Image Denoising with Convolutional Networks. In *NIPS*, 2008. 3

[18] J. Jancsary, S. Nowozin, and C. Rother. Loss-specific Training of Non-parametric Image Restoration Models: A New State of the Art. In *ECCV*, 2012. 2

[19] D. Krishnan and R. Fergus. Fast Image Deconvolution using Hyper-Laplacian Priors. In *NIPS*, 2009. 2

[20] M. Lebrun, A. Buades, and J. M. Morel. A Nonlocal Bayesian Image Denoising Algorithm. *SIAM Journal on Imaging Sciences*, 6(3):1665–1688, 2013. 3, 6, 7, 8

[21] D. C. Liu and J. Nocedal. On the Limited Memory BFGS Method for Large Scale Optimization. *Mathematical Programming*, 45(1-3):503–528, 1989. 6

[22] J. Mairal, F. R. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Non-local Sparse Models for Image Restoration. In *ICCV*, 2009. 3, 8

[23] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics. In *ICCV*, 2001. 6

[24] P. Perona and J. Malik. Scale-Space and Edge Detection Using Anisotropic Diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639, 1990. 3

[25] G. Plonka and J. Ma. Nonlinear Regularized Reaction-Diffusion Filters for Denoising of Images With Textures. *IEEE Transactions on Image Processing*, 17(8):1283–1294, 2008. 3

[26] J. Portilla, V. Strela, M. J. Wainwright, and E. P. Simoncelli. Image denoising using scale mixtures of Gaussians in the wavelet domain. *IEEE Transactions on Image Processing*, 12(11):1338–1351, 2003. 3

[27] S. Ross, D. Munoz, M. Hebert, and J. A. Bagnell. Learning Message-passing Inference Machines for Structured Prediction. In *CVPR*, 2011. 3

[28] S. Roth and M. J. Black. Fields of Experts. *International Journal of Computer Vision*, 82(2):205–229, 2009. 3, 7, 8

[29] H. Rue and L. Held. *Gaussian Markov Random Fields: Theory and Applications*. Chapman & Hall, London, 2005. 1

[30] U. Schmidt, J. Jancsary, S. Nowozin, S. Roth, and C. Rother. Cascades of Regression Tree Fields for Image Restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, To appear, 2015. 3, 7, 8

[31] U. Schmidt and S. Roth. Shrinkage Fields for Effective Image Restoration. In *CVPR*, 2014. 3, 6, 7, 8

[32] A. G. Schwing and R. Urtasun. Fully Connected Deep Structured Networks. *CoRR*, abs/1503.02351, 2015. 3

[33] E. P. Simoncelli and E. H. Adelson. Noise removal via bayesian wavelet coring. In *ICIP*, 1996. 3

[34] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. DeepFace: Closing the Gap to Human-Level Performance in Face Verification. In *CVPR*, 2014. 1

[35] M. F. Tappen, C. Liu, E. H. Adelson, and W. T. Freeman. Learning Gaussian Conditional Random Fields for Low-Level Vision. In *CVPR*, 2007. 1, 2

[36] Y. Wang, J. Yang, W. Yin, and Y. Zhang. A New Alternating Minimization Algorithm for Total Variation Image Reconstruction. *SIAM Journal on Imaging Sciences*, 1(3):248–272, 2008. 2

[37] J. Xie, L. Xu, and E. Chen. Image denoising and inpainting with deep neural networks. In *NIPS*, 2012. 3

[38] S. Zhang and E. Salari. Image Denoising Using a Neural Network-based Non-linear Filter in Wavelet Domain. In *ICASSP*, 2005. 3

[39] S. Zheng, S. Jayasumana, B. R.-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. S. Torr. Conditional Random Fields as Recurrent Neural Networks. In *ICCV*, 2015. 3

[40] D. Zoran and Y. Weiss. From Learning Models of Natural Image Patches to Whole Image Restoration. In *ICCV*, 2011. 1, 2, 3, 4, 6, 7, 8

Deep Gaussian Conditional Random Field Network: A Model-based Deep Network for Discriminative Denoising - Supplementary Material

Raviteja Vemulapalli
Center for Automation Research, UMIACS
University of Maryland, College Park

Oncel Tuzel, Ming-Yu Liu
Mitsubishi Electric Research Laboratories
Cambridge, MA

Section 1 of this supplementary material provides detailed derivations for the derivative formulas presented in the appendix of the main submission, and section 2 of this supplementary material provides an algorithmic description of the proposed deep Gaussian CRF network.

1. Backpropagation

1.1. Backpropagation through the combination layer

Forward step:

$$\mathbf{\Sigma}_{ij} = \sum_k \gamma_{ij}^k \mathbf{\Psi}_k. \tag{1}$$

Backward step:

$$\frac{dL}{d\gamma_{ij}^k} = \sum_{pq} \frac{dL}{d\mathbf{\Sigma}_{ij}(p,q)} \frac{d\mathbf{\Sigma}_{ij}(p,q)}{d\gamma_{ij}^k} = \sum_{pq} \frac{dL}{d\mathbf{\Sigma}_{ij}(p,q)} \mathbf{\Psi}_k(p,q) = \text{trace}\left(\mathbf{\Psi}_k^\top \frac{dL}{d\mathbf{\Sigma}_{ij}}\right). \tag{2}$$

$$\frac{dL}{d\mathbf{\Psi}_k(p,q)} = \sum_{ij} \frac{dL}{d\mathbf{\Sigma}_{ij}(p,q)} \frac{d\mathbf{\Sigma}_{ij}(p,q)}{d\mathbf{\Psi}_k(p,q)} = \sum_{ij} \frac{dL}{d\mathbf{\Sigma}_{ij}(p,q)} \gamma_{ij}^k \implies \frac{dL}{d\mathbf{\Psi}_k} = \sum_{ij} \gamma_{ij}^k \frac{dL}{d\mathbf{\Sigma}_{ij}}. \tag{3}$$

1.2. Backpropagation through the quadratic layer

Forward step:

$$s_{ij}^k = -\frac{1}{2} \bar{\mathbf{x}}_{ij}^\top \left(\mathbf{W}_k + \sigma^2 \mathbf{I}\right)^{-1} \bar{\mathbf{x}}_{ij} + b_k. \tag{4}$$

Backward step:

$$\frac{dL}{db_k} = \sum_{ij} \frac{dL}{ds_{ij}^k} \frac{ds_{ij}^k}{db_k} = \sum_{ij} \frac{dL}{ds_{ij}^k}. \tag{5}$$

$$\frac{dL}{d\bar{\mathbf{x}}_{ij}} = \sum_k \frac{dL}{ds_{ij}^k} \frac{ds_{ij}^k}{d\bar{\mathbf{x}}_{ij}} = -\sum_k \frac{dL}{ds_{ij}^k} \left(\mathbf{W}_k + \sigma^2 \mathbf{I}\right)^{-1} \bar{\mathbf{x}}_{ij}. \tag{6}$$

$$\begin{aligned}
\frac{dL}{d\mathbf{W}_k(p,q)} = \sum_{ij} \frac{dL}{ds_{ij}^k} \frac{ds_{ij}^k}{d\mathbf{W}_k(p,q)} &= -\frac{1}{2} \sum_{ij} \frac{dL}{ds_{ij}^k} \bar{\mathbf{x}}_{ij}^\top \frac{d(\mathbf{W}_k + \sigma^2 I)^{-1}}{d\mathbf{W}_k(p,q)} \bar{\mathbf{x}}_{ij} \\
&= \frac{1}{2} \sum_{ij} \frac{dL}{ds_{ij}^k} \bar{\mathbf{x}}_{ij}^\top (\mathbf{W}_k + \sigma^2 I)^{-1} \frac{d(\mathbf{W}_k + \sigma^2 I)}{d\mathbf{W}_k(p,q)} (\mathbf{W}_k + \sigma^2 I)^{-1} \bar{\mathbf{x}}_{ij} \\
&= \frac{1}{2} \sum_{ij} \frac{dL}{ds_{ij}^k} \left[(\mathbf{W}_k + \sigma^2 I)^{-1} \bar{\mathbf{x}}_{ij} \bar{\mathbf{x}}_{ij}^\top (\mathbf{W}_k + \sigma^2 I)^{-1}\right](p,q)
\end{aligned} \tag{7}$$

From (7), we have

$$\frac{dL}{d\mathbf{W}_k} = \frac{1}{2}\sum_{ij}\frac{dL}{ds_{ij}^k}\left[(\mathbf{W}_k+\sigma^2 I)^{-1}\,\bar{\mathbf{x}}_{ij}\,\bar{\mathbf{x}}_{ij}^\top\,(\mathbf{W}_k+\sigma^2 I)^{-1}\right]$$
$$= (\mathbf{W}_k+\sigma^2 I)^{-1}\left(\sum_{ij}\frac{dL}{ds_{ij}^k}\frac{\bar{\mathbf{x}}_{ij}\,\bar{\mathbf{x}}_{ij}^\top}{2}\right)(\mathbf{W}_k+\sigma^2 I)^{-1} \tag{8}$$

1.3. Backpropagation through the patch inference layer

Forward step:

$$\mathbf{z}_{ij} = \left(\mathbf{I}-\mathbf{G}^\top\left(\beta\boldsymbol{\Sigma}_{ij}+\mathbf{G}\mathbf{G}^\top\right)^{-1}\mathbf{G}\right)\mathbf{y}_{ij}. \tag{9}$$

Backward step:

Let $\mathbf{A}_{ij} = \mathbf{I}-\mathbf{G}^\top\left(\beta\boldsymbol{\Sigma}_{ij}+\mathbf{G}\mathbf{G}^\top\right)^{-1}\mathbf{G}$. Let $\mathbb{I}_{pq}$ be a matrix with (p,q) element as one and all other elements as zero. Note that the matrix $\mathbf{G} = \mathbf{I}-\frac{1}{d^2}\mathbf{1}\mathbf{1}^\top$ satisfies $\mathbf{G}\mathbf{G}^\top = \mathbf{G}$.

$$\mathbf{z}_{ij} = \mathbf{A}_{ij}\mathbf{y}_{ij} \implies \frac{dL}{d\mathbf{y}_{ij}} = \mathbf{A}_{ij}^\top\frac{dL}{d\mathbf{z}_{ij}} = \left(\mathbf{I}-\mathbf{G}^\top\left(\beta\boldsymbol{\Sigma}_{ij}+\mathbf{G}\mathbf{G}^\top\right)^{-1}\mathbf{G}\right)\frac{dL}{d\mathbf{z}_{ij}} = \left(\mathbf{I}-\mathbf{G}^\top\left(\beta\boldsymbol{\Sigma}_{ij}+\mathbf{G}\right)^{-1}\mathbf{G}\right)\frac{dL}{d\mathbf{z}_{ij}}. \tag{10}$$

$$\mathbf{z}_{ij} = \mathbf{A}_{ij}\mathbf{y}_{ij} \implies \frac{dL}{d\mathbf{A}_{ij}} = \frac{dL}{d\mathbf{z}_{ij}}\mathbf{y}_{ij}^\top \tag{11}$$

$$\frac{dL}{d\boldsymbol{\Sigma}_{ij}(p,q)} = \sum_{r,s}\frac{dL}{d\mathbf{A}_{ij}(r,s)}\frac{d\mathbf{A}_{ij}(r,s)}{d\boldsymbol{\Sigma}_{ij}(p,q)} = \text{trace}\left(\left(\frac{dL}{d\mathbf{A}_{ij}}\right)^\top\frac{d\mathbf{A}_{ij}}{d\boldsymbol{\Sigma}_{ij}(p,q)}\right) \tag{12}$$

$$\begin{aligned}\frac{d\mathbf{A}_{ij}}{d\boldsymbol{\Sigma}_{ij}(p,q)} &= -\mathbf{G}^\top\frac{d\left(\beta\boldsymbol{\Sigma}_{ij}+\mathbf{G}\mathbf{G}^\top\right)^{-1}}{d\boldsymbol{\Sigma}_{ij}(p,q)}\mathbf{G}\\ &= \mathbf{G}^\top\left(\beta\boldsymbol{\Sigma}_{ij}+\mathbf{G}\mathbf{G}^\top\right)^{-1}\frac{d\left(\beta\boldsymbol{\Sigma}_{ij}+\mathbf{G}\mathbf{G}^\top\right)}{d\boldsymbol{\Sigma}_{ij}(p,q)}\left(\beta\boldsymbol{\Sigma}_{ij}+\mathbf{G}\mathbf{G}^\top\right)^{-1}\mathbf{G}\\ &= \mathbf{G}^\top\left(\beta\boldsymbol{\Sigma}_{ij}+\mathbf{G}\mathbf{G}^\top\right)^{-1}(\beta\mathbb{I}_{pq})\left(\beta\boldsymbol{\Sigma}_{ij}+\mathbf{G}\mathbf{G}^\top\right)^{-1}\mathbf{G}\end{aligned} \tag{13}$$

From (12) and (13), we have

$$\begin{aligned}\frac{dL}{d\boldsymbol{\Sigma}_{ij}(p,q)} &= \text{trace}\left(\left(\frac{dL}{d\mathbf{A}_{ij}}\right)^\top\left(\mathbf{G}^\top\left(\beta\boldsymbol{\Sigma}_{ij}+\mathbf{G}\mathbf{G}^\top\right)^{-1}(\beta\mathbb{I}_{pq})\left(\beta\boldsymbol{\Sigma}_{ij}+\mathbf{G}\mathbf{G}^\top\right)^{-1}\mathbf{G}\right)\right)\\ &= \text{trace}\left(\left(\beta\boldsymbol{\Sigma}_{ij}+\mathbf{G}\mathbf{G}^\top\right)^{-1}\mathbf{G}\left(\frac{dL}{d\mathbf{A}_{ij}}\right)^\top\mathbf{G}^\top\left(\beta\boldsymbol{\Sigma}_{ij}+\mathbf{G}\mathbf{G}^\top\right)^{-1}(\beta\mathbb{I}_{pq})\right)\end{aligned} \tag{14}$$

From (11) and (14), we have

$$\begin{aligned}\frac{dL}{d\boldsymbol{\Sigma}_{ij}} &= \beta\left(\left(\beta\boldsymbol{\Sigma}_{ij}+\mathbf{G}\mathbf{G}^\top\right)^{-1}\mathbf{G}\left(\frac{dL}{d\mathbf{A}_{ij}}\right)^\top\mathbf{G}^\top\left(\beta\boldsymbol{\Sigma}_{ij}+\mathbf{G}\mathbf{G}^\top\right)^{-1}\right)^\top\\ &= \beta\left(\beta\boldsymbol{\Sigma}_{ij}+\mathbf{G}\mathbf{G}^\top\right)^{-1}\mathbf{G}\left(\frac{dL}{d\mathbf{A}_{ij}}\right)\mathbf{G}^\top\left(\beta\boldsymbol{\Sigma}_{ij}+\mathbf{G}\mathbf{G}^\top\right)^{-1}\\ &= \beta\left(\beta\boldsymbol{\Sigma}_{ij}+\mathbf{G}\right)^{-1}\mathbf{G}\frac{dL}{d\mathbf{z}_{ij}}\mathbf{y}_{ij}^\top\mathbf{G}^\top\left(\beta\boldsymbol{\Sigma}_{ij}+\mathbf{G}\right)^{-1}\end{aligned} \tag{15}$$

2. Algorithmic description of the proposed deep Gaussian CRF network

Algorithm 1 Deep Gaussian CRF Network

Input: Noisy image $\mathbf{X}$, noise variance σ^2

Initialize the network input $\mathbf{Y}^0$ with the noisy input image $\mathbf{X}$.

for $t = 1$ to T **do**

Parameter generation network (PgNet):

Patch extraction layer: Extract the $d \times d$ mean subtracted patches $\{\bar{\mathbf{y}}_{ij}^{t-1}\}$ from the image $\mathbf{Y}^{t-1}$ at all pixels (i, j).

Selection network (parameters $\{(\mathbf{W}_k, b_k)\}$): Compute the combination weights $\{\gamma_{ij}^k\}$ for each patch $\bar{\mathbf{y}}_{ij}^{t-1}$ using

$$s_{ij}^k = -\frac{1}{2}\left(\bar{\mathbf{y}}_{ij}^{t-1}\right)^\top \left(\mathbf{W}_k + \sigma^2\mathbf{I}\right)^{-1} \bar{\mathbf{y}}_{ij}^{t-1} + b_k, \quad \gamma_{ij}^k = e^{s_{ij}^k} / \sum_{p=1}^{K} e^{s_{ij}^p}. \tag{16}$$

Combination layer (parameters $\{\mathbf{\Psi}_k\}$): Compute the potential parameters $\mathbf{\Sigma}_{ij}$ for each patch $\bar{\mathbf{y}}_{ij}^{t-1}$ using

$$\mathbf{\Sigma}_{ij} = \sum_{k=1}^{K} \gamma_{ij}^k \mathbf{\Psi}_k. \tag{17}$$

HQS layer of the inference network (InfNet):

Patch inference layer (PI): Compute the auxiliary patches $\{\mathbf{z}_{ij}\}$ using the patches $\{\mathbf{y}_{ij}^{t-1}\}$ extracted from the image $\mathbf{Y}^{t-1}$, and the pairwise potential parameters $\{\mathbf{\Sigma}_{ij}\}$ given by the PgNet:

$$\mathbf{z}_{ij} = \left(\mathbf{I} - \mathbf{G}^\top \left(\beta_t \mathbf{\Sigma}_{ij} + \mathbf{G}\mathbf{G}^\top\right)^{-1} \mathbf{G}\right) \mathbf{y}_{ij}^{t-1}, \text{ where } \mathbf{G} = \mathbf{I} - \frac{1}{d^2}\mathbf{1}\mathbf{1}^\top \text{ is the mean subtraction matrix.} \tag{18}$$

Image formation layer (IF): Compute the new clean image estimate $\mathbf{Y}^t$ using the auxiliary patches $\{\mathbf{z}_{ij}\}$ and the original noisy image $\mathbf{X}$:

$$\mathbf{Y}^t(i,j) = \frac{\mathbf{X}(i,j)}{1+\beta_t\sigma^2} + \frac{\beta_t\sigma^2}{(1+\beta_t\sigma^2)d^2} \sum_{p,q=-\lfloor\frac{d-1}{2}\rfloor}^{\lceil\frac{d-1}{2}\rceil} \mathbf{z}_{pq}(i,j). \tag{19}$$

Output: Clean image $\mathbf{Y} = \mathbf{Y}^T$.