DOUBLE NETWORK SUPERRESOLUTION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

CEM TARHAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

JUNE 2019

Approval of the thesis:

DOUBLE NETWORK SUPERRESOLUTION

submitted by **CEM TARHAN** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar Dean, Graduate School of **Natural and Applied Sciences**	____________
Prof. Dr. İlkay Ulusoy Head of Department, **Electrical and Electronics Engineering**	____________
Prof. Dr. Gözde Bozdağı Akar Supervisor, **Electrical and Electronics Engineering, METU**	____________

Examining Committee Members:

Prof. Dr. Ziya Telatar Electrical and Electronics Engineering, Ankara University	____________
Prof. Dr. Gözde Bozdağı Akar Electrical and Electronics Engineering, METU	____________
Assoc. Prof. Dr. A. Oğuz Akyüz Computer Engineering, METU	____________
Assoc. Prof. Dr. H. Şakir Bilge Electrical and Electronics Engineering, Gazi University	____________
Assist. Prof. Dr. S. Figen Öktem Electrical and Electronics Engineering, METU	____________

Date:13.06.2019

Name, Surname: Cem Tarhan

Signature :

ABSTRACT

DOUBLE NETWORK SUPERRESOLUTION

Tarhan, Cem
Ph.D., Department of Electrical and Electronics Engineering
Supervisor: Prof. Dr. Gözde Bozdağı Akar

June 2019, 110 pages

As the social platforms became widespread, the image and video based materials are being shared continuously and increasingly each day. This not only brings an issue of storage but also internet bandwidth usage. In order for a user to effectively run a superresolution (SR) algorithm on a mobile device, a light-weight but good performing algorithm must be designed. In recent years, convolutional neural networks (CNNs) have been widely used for SR. Although their indisputable success, CNNs lack proper mathematical background on how and what they learn. In the first part of the thesis we prove that CNN elements act as inverse problem solvers that are optimal for the purpose. We show that the learned coefficients of a network obey a concept namely Representation-Dictionary Duality. We show the necessity of skip connections for convergence of the network.

The demand for high computational load for state of the art algorithms renders them unusable on a mobile platform. In the second part of the thesis, we propose a novel double network superresolution (DNSR) algorithm that requires dramatically low number of parameters. We propose the usage of two networks, trained with disjunct data. One network is responsible from reconstructing sharp transitions in an image where the other network is specialized for texture reconstruction. DNSR is not only able to learn SR solution with practically feasible number of operations but also able to obtain superior performance on the reconstruction of high frequency details with high fidelity.

Finally, we propose a Detail Fusion Interpolator (DFI), that combines optical flow estimation and motion compensation blocks within a small network. By extending DNSR to multi-frame approaches we compare its performance to state of the art Video SR algorithms and to single frame DNSR. We show that DFI is indeed able to compensate for motion and combined system performs better than single frame approach.

Keywords: Superresolution, Deep Neural Networks, Convolutional Neural Networks, Deep Learning, Inverse Problems, Sparse Representation, Optical Flow Estimation, Motion Compensation, Video Superresolution

ÖZ

ÇİFT AĞLI SÜPERÇÖZÜNÜRLÜK

Tarhan, Cem
Doktora, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Gözde Bozdağı Akar

Haziran 2019 , 110 sayfa

Sosyal platformların yaygınlaşmasıyla beraber günlük olarak paylaşılan video ve resim materyali hızla artmaktadır. Bu hem depolama hem de internet bantgenişliği konusunda sorunlara yol açmaktadır. Bir kullanıcının mobil cihazında bir SÇ algoritması kullanabilmesi için iyi sonuçlar üreten ve hafif bir algoritmaya ihtiyaç duyulmaktadır. Son yıllarda Evrişimli Sinir Ağları (ESA) SÇ alanında sıkça kullanılmıştır. Tartışılmaz başarılarına rağmen ESA'ların matematiksel anlaşılırlığı çok düşük seviyededir. Bu tezin ilk kısmında ESA elementlerinin esasında tersine sorunlara çözüm üreten optimum elementler olduğunu gösterdik. Öğrenilen ağ katsayılarının Temsil-Kütüphane İkililiği konseptine uyduğunu gösterdik. ESA eğitiminin yakınsaması için atlama bağlantılarının gerekliliğini gösterdik.

İhtiyaç duyulan yüksek işlem gücü gereksinimi güncel algoritmaları mobil sistemlerde kullanılamaz hale getirmektedir. Tezin ikinci kısmında özgün Çift Ağlı Süperçözünürlük (ÇASÇ) algoritmasını önererek bir ağın öğrenme kapasitesini düşürmeden ihtiyaç duyulan katsayı miktarını büyük ölçüde düşürmeyi başardık. Kullanılan iki ayrı ağ iki ayrılmış veri ile eğitilmektedir. Ağların bir tanesi resimlerdeki sert geçişleri yeniden kurmakla görevliyken diğer ağ desen içeren tüm yamaları yeniden kurmakla görevlendirilmiştir. ÇASÇ sadece çok az sayıda parametre ile SÇ problemine çözüm üretmekle kalmıyor aynı zamanda yüksek hata performansı gösterirken yüksek frekans detayları başarı ile yeniden yapılandırabiliyor. Son olarak Detay Kaynatma İnterpolasyon (DKI) isimli optik akış ve hareket telafisini birleştiren küçük boyutta bir ağ önerdik. Bu sayede çok kare işlemeye esnetilmiş olan ÇASÇ algoritmasını diğer Video SÇ algoritmalarıyla ve tek kare ÇASÇ ile kıyasladık. DKI'nin hareket tahmini ve telafisini başarıyla yerine getirdiğini ve birleştirilmiş sistemin tek kare uygulamalardan iyi sonuçlar ürettiğini gösterdik.

Anahtar Kelimeler: Süperçözünürlük, Derin Sinir Ağları, Evrişimli Sinir Ağları, Derin Öğrenme, Tersine Sorunlar, Seyrek Temsil, Optik Akış Tahmini, Hareket Düzeltme, Video Süperçözünürlük

Alles Vergängliche Ist nur ein Gleichnis

ACKNOWLEDGMENTS

First and foremost I would like to express my gratitude to my supervisor Prof. Dr. Gözde BOZDAĞI AKAR. Her guidance is always on point like precision scalpel that removes or adds the most important details to a work that makes it valuable. Without her encouragement and endless support this work would not have reached to a conclusion.

I would like to thank to my colleagues Güneş OTLU, Osman Ulaş ŞAHİN, Ramazan CİLASIN for all the conversations that we had. Those "mola"s made our working time infinitely richer.

I would like to thank Ufuk BÜYÜKŞAHİN, Nusret BAYHAN, Emin ZERMAN, Eda BAYRAM, Fatih ERKAN for being there, always, even though they are hundreds and thousands of kilometers away.

Lastly I would like to express my gratitude for my parents Ece TARHAN and Bülent TARHAN. Without them I would not have been who I am now, they are like the soil and the seed upon which my consciousness have sprouted.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

2D	2 Dimensional
3D	3 Dimensional
ADMM	Alternating Direction Method of Multipliers
ANR	Anchored Neighborhood Regression
CNN	Convolutional Neural Network
CON	Complete Orthonormal
CSC	Convolutional Sparse Coding
DBPN	Deep Backprojection Network
DNSR	Double Network Superresolution
DRCN	Deeply Recursive Convolutional Network
DRRN	Deep Recursive Residual Network
EDSR	Enhanced Deep Superresolution
ESPCN	Effective Sub-Pixel Convolutional Network
FCN	Fully Connected Neural Network
FFT	Fast Fourier Transform
FLOPS	Floating Point Operation per Second
FSRCNN	Fast Superresolution Convolutional Neural Network
GAN	Generative Adversarial Network
IDN	Information Distillation Network
IST	Iterative Shrinkage Thresholding
KL	Kullback-Liebler
LapSRN	Laplacian Superresolution Network
LISTA	Learned Iterative Shrinkage Thresholding Algorithm

LR	Low Resolution
LS	Least Squares
MC	Motion Compensation
MFSR	Multiframe Superresolution
MS-LapSRN	Multi-Scale Laplacian Superresolution Network
MSE	Mean Square Error
OF	Optical Flow
Ops/op	Operations per output pixel
PSNR	Peak Signal to Noise Ratio
RAISR	Rapid Accurate Image Superresolution
RELU	Rectified Linear Unit
RDD	Representation-Dictionary Duality
RDN	Residual Dense Network
SC	Spatial Coherency
SCHI	Separate Channel Interpolation
SGD	Stochastic Gradient Descent
SotA	State of the Art
SPMC	Subpixel Motion Compensation
SR	Superresolution
SRCNN	Superresolution Convolutional Neural Network
SSIM	Structural Similarity
SVD	Singular Value Decomposition
TNRD	Trainable Nonlinear Reaction Diffusion
VDSR	Very Deep Superresolution
VSR	Video Superresolution

CHAPTER 1

INTRODUCTION

1.1 Motivation and Problem Definition

Images and videos become the most time consuming media as social platforms are evolving. Each user is reaching for thousands of content daily and they are also creating content that will be seen by thousands of people. Not only the amount of storage utilized for social platforms and required internet bandwidth increase everyday, as smart phones become more widespread this rise is exponentially accelerating. One of many methods to address this issue is to reduce the resolution of the media on the storage side. To present the media to the user as similar to the original as possible, the resolution must be re-scaled to the original value. This step, which is also known as superresolution, should also be carried out on a mobile phone among many other actions. Not only image superresolution but also denoising, inpainting and reconstruction may be needed as examples are not limited by social platform usage. Military, medical, automotive and surveillance usage of imaging, requires more or less all of the aforementioned method to be utilized into their applications. Most of these areas are also limited in terms of hardware such as storage and computation capacity.

The broad term for the family of image reconstruction methods is inverse problems. An image is required to be reconstructed from a cascade of effects such as blurring, noise, downsampling, pixel loss. The inversion of such problems is almost always ill conditioned, which means that the procedure of reconstruction requires some regularizations and modelling assumptions.

The literature of inverse problem solutions reach back to decades ago. There are still new methods and algorithms that solve problems defined many years ago. Besides analytic methods that use mathematical approach, there is another method that is trending in all areas of imaging: the convolutional neural networks (CNNs). In the last decade, due to advancing technology, usage of CNNs is exploded in the literature, mostly overshadowing analytic methods. Although their undisputed success, the knowledge over CNNs is quite limited and lacking. Proposed methods are mostly emerging from experimentation rather than careful mathematical foundation.

In the intersection of emerging CNN technology and requirement of good and fast performing inverse problem solutions, specifically superresolution, we submit this thesis work.

1.2 Proposed Methods and Contributions

Our contributions are as follows:

- We show that CNN elements, i.e. neuron filters, solve for an inverse problem during training. The operator of the inverse problem is taken as a dictionary matrix constructed from input training data. The solution of the problem are the neuron filter coefficients. These coefficients are representation vectors of the target, which act similarly as described in manifold theory [16] where each neuron filter acts as a point in high dimensional space to form a manifold for a solution. Differently from the argument in the literature [17] [18] [19] that trained networks resemble or become inverse problem solvers, we claim that the training process is carried out in this fashion.

- By proposing *Representation Dictionary Duality* (RDD) we relate sparse representation and convolutional sparse coding [20] to the evolution of CNNs and provide a mathematical basis for necessity of skip connections [21] and residual learning [4].

- We propose a double network super resolution (DNSR) structure where the training set is separated according to texturedness measure. This enables separate networks to better adapt and learn presented information with dramatically lower parameter set.

- In order to decrease complexity even further, we propose different depths and interpolation methods for separate networks.
- Proposed double network topology handles high frequency detail reconstruction superior to state-of-the-art methods as compared in Figure 5.5
- We have proposed detail fusion interpolation (DFI) that incorporates a 3D CNN's ability to infer motion information from neighboring video frames. This is proposed against more complex motion compensation blocks [2][14][15]. DFI is used to extend our DNSR into video superresolution.
- In order to propose DNSR, we have carried extensive experiments and in the process we have introduced new concepts such as separate channel interpolation (SCHI) instead of sub-pixel accuracy interpolation [5]

1.3 The Outline of the Thesis

In the Chapter 2 a background on inverse problem solutions is given. In Chapter 3 mathematical properties of CNNs are analyzed. Mathematical proofs regarding CNN training and testing are given in this chapter. In Chapter 4 focus of the thesis is narrowed down to application of superresolution. The details of our proposed model DNSR are described in this section. In Chapter 5 experimental validations are carried out for mathematical assertions and our DNSR model. DNSR is compared to SotA methods and its superiority in pixelwise error, noise performance and high frequency reconstruction quality is given quantitatively. In Chapter 6 the extension of our DNSR to video superresolution is described. Theoretical and experimental results are given in this chapter. In Chapter 7 our thesis work is summarized, future works are listed and the thesis is concluded.

CHAPTER 2

BACKGROUND ON INVERSE PROBLEMS

The broad class of inverse problems in image processing contains problems that deal with recovery of a lost data within an observational system. The solution of such problems vary vastly in literature ranging from purely mathematical approaches [22] to almost completely experimental methods [4]. In this chapter, the problem is formulated and related solution methods are explained in two categories, namely analytic methods and data driven approaches.

2.1 Introduction

In many image processing applications, an observational system model is the initial step to a solution. An observation g can be modeled as an output of a system K(.) when stimulated by an input t. Finding the variable t for an observation g and system K is named as an inverse problem. Due to the nature of observation modelling in imaging applications, the inversion of system K is ill-conditioned. Small variations in the observed value causes huge swings in estimated input value. This perturbation is caused by the noise in the system which is denoted by n.

$$g = Kt + n \tag{21}$$

System K could be divided into multiple categories such as sampling, motion artifacts, blurring and downsampling. Figure 2.1 displays the observation process of a continuous scenery. Many areas of research deal with inverse problems under the name of denoising, deblurring, deconvolution, enhancement, restoration, single im-

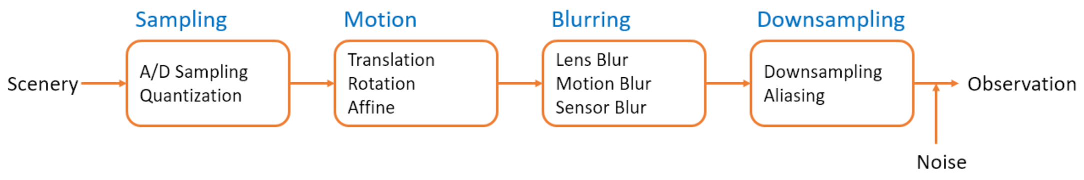


Figure 2.1: Image observation model

age superresolution, MRI and CT reconstruction from projections etc. These problems can be classified as inverse problems that are defined with convolution operations. For decades, analytic methods have been utilized for solving these problems. A deterministic approach is to minimize a data fidelity term

$$\hat{t} = \underset{t}{argmin}\{\Delta(Kt, g)\} \tag{22}$$

The $\Delta(.)$ can be $||\boldsymbol{g} - K\boldsymbol{t}||^2$ to obtain a Least Squares (LS) solution. Bold characters indicate a vector. Other methods include L_P norm solution, Kullback Liebler (KL) distance solution. Since this problem is ill-conditioned a regularization term can be added to the data fidelity term. Regularization, r(**t**), adds a mismatched function to the cost function that will balance the solution.

$$\underset{t}{argmin}\{\Delta(K\boldsymbol{t}, \boldsymbol{g}) + r(\boldsymbol{t})\} \tag{23}$$

Statistical inversion methods are another approach for an inverse problem. The observation vector **g** is assumed to be drawn from a probabilistic distribution p(**g**). For this approach, the cost function will become $\Delta(\boldsymbol{t}, \boldsymbol{g}) = p(\boldsymbol{g}|\boldsymbol{t})$ for a maximum likelihood approach. Stochastic approaches are out of scope for this work, therefore related literature is not included.

Analytic methods seek to successfully formulate the system model and find the optimum solution strategy with proper regularization. These methods require careful mathematical analysis and costly solutions.

Another class of methods that has been successfully used in inverse problem solutions is neural network approach, specifically convolutional neural networks (CNNs) for image processing applications. CNNs are trained by a set of images to learn a mapping from **g** to **t**. The structure of the network; number of layers and elements in each layer, interconnections between layers, special layers such as pooling or normalization changes depending on the specific application. However the learning process is generally based on stochastic gradient descent type coefficient updates. In most cases the cost function is chosen as mean square error (MSE). In some cases, such as generative adversarial networks and variational autoencoders, instead of end to end mapping from **g** to **t**, some feature discrimination is applied before calculating an MSE value together with a cross entropy loss function.

Recent advances on technology enabled training of bigger networks therefore providing better results [17][18][19]. A sample of methods available in literature show that CNN performance surpasses that of analytic methods in all aspects. This indicates that the research effort should be directed to using neural networks for the solution of the problem. CNN properties should be understood well in order to utilize its effectiveness with academic confidence. The learning process of CNNs should be clarified. After the learning is completed, the activation of the network for given novel input should also be understood clearly. The former work is about the training phase of the CNN and the latter is about the testing phase. The testing phase, also known as the forward pass, can be shown to be satisfying a set of equations for application specific cases and setups. Unfolded iterative shrinkage thresholding (IST) iterations [23], alternating direction method of multipliers (ADMM) iterations are calculated by network layers [24] and layer outputs are visualized as sparse representation vectors [3] or higher dimension manifold projections [16]. The training phase is not addressed as often in the literature to answer what a CNN actually solves during backpropagation. CNN related equations and explanations are going to be given in next chapter. This chapter includes solution approaches to inverse problems that will be crucial in explaining how CNNs operate in following chapters.

2.2 Analytic Solution Approaches

The analytic approach for solving the inverse problem involves minimizing a non-smooth convex cost function with additional regularizations that are impinged on the optimization to statistically tie the solution to the observations. Mathematically, most imaging problems such as debluring, denoising, SR are most appropriately formulated with variational equations. The objective of these equations is to incorporate a priori information and enforces coherency with the observations. Equation 23 is an example of a variational equation where r(**t**) term is continuous and differentiable function.

Pseudo inverse solution uses the inverse of the operator K^TK which is well defined if K^TK is a strictly positive operator. Regularization is used to stabilize the inverse problem solution, the functional in equation 23 to be minimized can be chosen as

$$\Delta(K\boldsymbol{t},\boldsymbol{g}) + r(\boldsymbol{t}) = ||K\boldsymbol{t} - \boldsymbol{g}||^2 + \lambda||\boldsymbol{t}||^2 \tag{24}$$

where λ is the regularization parameter. The minimizer can be found by

$$\hat{\boldsymbol{t}} = (K^TK + \lambda I)^{-1}K^T\boldsymbol{g} \tag{25}$$

where I is identity matrix and λ is the regularization parameter that is chosen according to the application. Although derivation of quadratic functions is easier especially for gradient descent type solvers, they are known to stuck to local minima occasionally which is a major problem for imaging applications where the cost function is almost always multimodal.

For this reason there are also nonquadratic constraints for regularizations, l^p-norm of **t** where $1 \leq p \leq 2$. For a given set of orthonormal basis functions φ_l of Hilbert space and set of weights **b**, a new functional, according to [25], can be defined as

$$\begin{aligned}\Phi_{\boldsymbol{b},\boldsymbol{p}}(\boldsymbol{t},\boldsymbol{g}) &= \Delta(\boldsymbol{t},\boldsymbol{g}) + r(\boldsymbol{t}) \\ &= ||K\boldsymbol{t} - \boldsymbol{g}||^2 + \sum_{\forall l} \boldsymbol{b_l}|\langle \boldsymbol{t}, \varphi_l\rangle|^p \end{aligned} \tag{26}$$

For the case where p is 2 and $b_l = \lambda\mathbf{1}$, where $\mathbf{1}$ is a vector of all ones, this equation reduces to the function minimized by equation 25. If p value is decreased towards 1 for this special case, then penalization of small ($|\langle \boldsymbol{t}, \varphi_l\rangle| < 1$) coefficients will be increased and penalization of larger ($|\langle \boldsymbol{t}, \varphi_l\rangle| > 1$) coefficients will be decreased. Therefore differently from the classical minimization in equation 25, there will be higher penalty for **t** with more but small elements in φ_l directions and there will be lower penalty for **t** with few nonzero components. When p is taken closer to 1, then the minimization of equation 22 promotes sparsity of the expansion of **t** w.r.t. φ_l. The minimization of equation 22 is, for example, used for denoising Guassian or Laplacian modelled noise since such r(**t**) is known to promote sparsity.

Daubechies [25] proposed the usage of Surrogate Functionals instead of functionals in equation 26. The presence of $K^T Kt$ in the minimization makes the solution problematic. Usage of surrogate functionals is proposed to cancel out the effects of $K^T Kt$ in the solution. For that purpose initial step is to choose a constant k where $||K^T K|| \leq k$ and a new operator is defined as $\Sigma(\boldsymbol{t}, \boldsymbol{a}) = k||\boldsymbol{t} - \boldsymbol{a}|| - ||K\boldsymbol{t} - K\boldsymbol{a}||^2$ which evolves with auxiliary variable **a** of Hilbert space. Similar solution strategies have been followed by many in literature [22] [26]. Since by definition $kI - K^T K$ is strictly positive, $\Sigma(\boldsymbol{t}, \boldsymbol{a})$ is strictly convex for all **a**. Without loss of generality assuming k = 1, two functionals $\Sigma(\boldsymbol{t}, \boldsymbol{a})$ and $\Phi(\boldsymbol{t}, \boldsymbol{g})$ are added together, as explained in [25]

$$\Phi^{SUR}_{\boldsymbol{b},\boldsymbol{p}}(\boldsymbol{t}, \boldsymbol{g}, \boldsymbol{a}) = \Phi_{\boldsymbol{b},\boldsymbol{p}}(\boldsymbol{t}, \boldsymbol{g}) + \Sigma(\boldsymbol{t}, \boldsymbol{a}) \tag{27}$$

$$= ||K\boldsymbol{t} - \boldsymbol{g}||^2 + \sum_{\forall l} \boldsymbol{b}_l |\langle \boldsymbol{t}, \varphi_l \rangle|^p - ||K\boldsymbol{t} - K\boldsymbol{a}||^2 \ + ||\boldsymbol{t} - \boldsymbol{a}|| \tag{28}$$

$$= ||\boldsymbol{t}||^2 - 2\langle \boldsymbol{t}, \boldsymbol{a} + K^T \boldsymbol{g} - K^T K \boldsymbol{a} \rangle$$
$$+ \sum_{\forall l} \boldsymbol{b}_l |\langle \boldsymbol{t}, \varphi_l \rangle|^p + ||\boldsymbol{g}||^2 + ||\boldsymbol{a}||^2 - ||K\boldsymbol{a}||^2 \tag{29}$$

The variational function is transformed into an iterative solution if auxillary variable **a** is taken as the previous value of **t** in an iterative scheme. If we take **b**=0, the functional becomes

$$\Phi^{SUR}_{\boldsymbol{0},\boldsymbol{p}}(\boldsymbol{t}^n, \boldsymbol{g}, \boldsymbol{t}^{n-1}) = ||\boldsymbol{t}^n||^2 - 2\langle \boldsymbol{t}^n, \boldsymbol{t}^{n-1} + K^T(\boldsymbol{g} - K\boldsymbol{t}^{n-1}) \rangle \tag{210}$$
$$+ ||\boldsymbol{g}||^2 + ||\boldsymbol{t}^{n-1}||^2 - ||K\boldsymbol{t}^{n-1}||^2$$

which yields the Landweber solution, whose convergence to solution of equation 21 is proven [25].

$$\boldsymbol{t}^n = \boldsymbol{t}^{n-1} + K^T(\boldsymbol{g} - K\boldsymbol{t}^{n-1}) \tag{211}$$

If **b** is taken as a constant vector and p=2 then the surrogate functional becomes

$$\Phi^{SUR}_{\boldsymbol{b},\boldsymbol{p}}(\boldsymbol{t}^n, \boldsymbol{g}, \boldsymbol{t}^{n-1}) = (1+\lambda)||\boldsymbol{t}^n||^2 - 2\langle \boldsymbol{t}^n, \boldsymbol{t}^{n-1} + K^T(\boldsymbol{g} - K\boldsymbol{t}^{n-1})\rangle \tag{212}$$

$$+ ||\boldsymbol{g}||^2 + ||\boldsymbol{t}^{n-1}||^2 - ||K\boldsymbol{t}^{n-1}||^2 \tag{213}$$

the solution is obtained from regularized Landweber iterations.

$$\boldsymbol{t}^n = \frac{1}{1+\lambda}(\boldsymbol{t}^{n-1} + K^T(\boldsymbol{g} - K\boldsymbol{t}^{n-1}))$$

For the case when p=1 which is more of interest to inverse problem solutions, specifically superresolution, the equation 27 is differentiable in $\langle \boldsymbol{t}, \varphi_l\rangle$ if $\langle \boldsymbol{t}, \varphi_l\rangle$ is nonzero, then the variational equation becomes

$$2\langle \boldsymbol{t}, \varphi_l\rangle + \boldsymbol{b_l} sign(\langle \boldsymbol{t}, \varphi_l\rangle) = 2(\langle \boldsymbol{a}, \varphi_l\rangle + \langle (K^T(\boldsymbol{g} - K\boldsymbol{a})), \varphi_l\rangle) \tag{214}$$

Equation 214 can be transformed into equation 215

$$\langle \boldsymbol{t}, \varphi_l\rangle = S_b(\langle \boldsymbol{a}, \varphi_l\rangle + \langle (K^T(\boldsymbol{g} - K\boldsymbol{a})), \varphi_l\rangle) \tag{215}$$

where the function S_b is defined as

$$S_b(x) = \begin{cases} x - b & if\, x \geq b \\ 0 & if\, |x| < b \\ x + b & if\, x \leq -b \end{cases} \tag{216}$$

As we shall see later symmetrical soft thresholding has an inherent connection to non-negative soft thresholding used in neural networks. The thresholding function is depicted in Figure 2.2 for b=0.5. Notice that $K^T(\boldsymbol{g} - K\boldsymbol{t}^{n-1})$ is the negative gradient of data fidelity term $\Delta(\boldsymbol{t}, \boldsymbol{g})$ in the original formulation 23. For the case when p = 1, a straightforward variational equation can be obtained in an iterative if previous iterations are plugged into the auxiliary variable **a**. The class of these equations are also known as iterative shrinkage thresholding (IST) in the literature [22]

$$\langle \boldsymbol{t}^n, \varphi_l\rangle = S_{\boldsymbol{b}}(\langle \boldsymbol{t}^{n-1}, \varphi_l\rangle + \langle K^T(\boldsymbol{g} - K\boldsymbol{t}^{n-1}), \varphi_l\rangle) \tag{217}$$

Daubechies proposed [25] that the iterations over the set of basis functions can be carried out in one formula

$$\boldsymbol{t}^n = Z_{\boldsymbol{b}}(\boldsymbol{t}^{n-1} + K^T(\boldsymbol{g} - K\boldsymbol{t}^{n-1})) \tag{218}$$

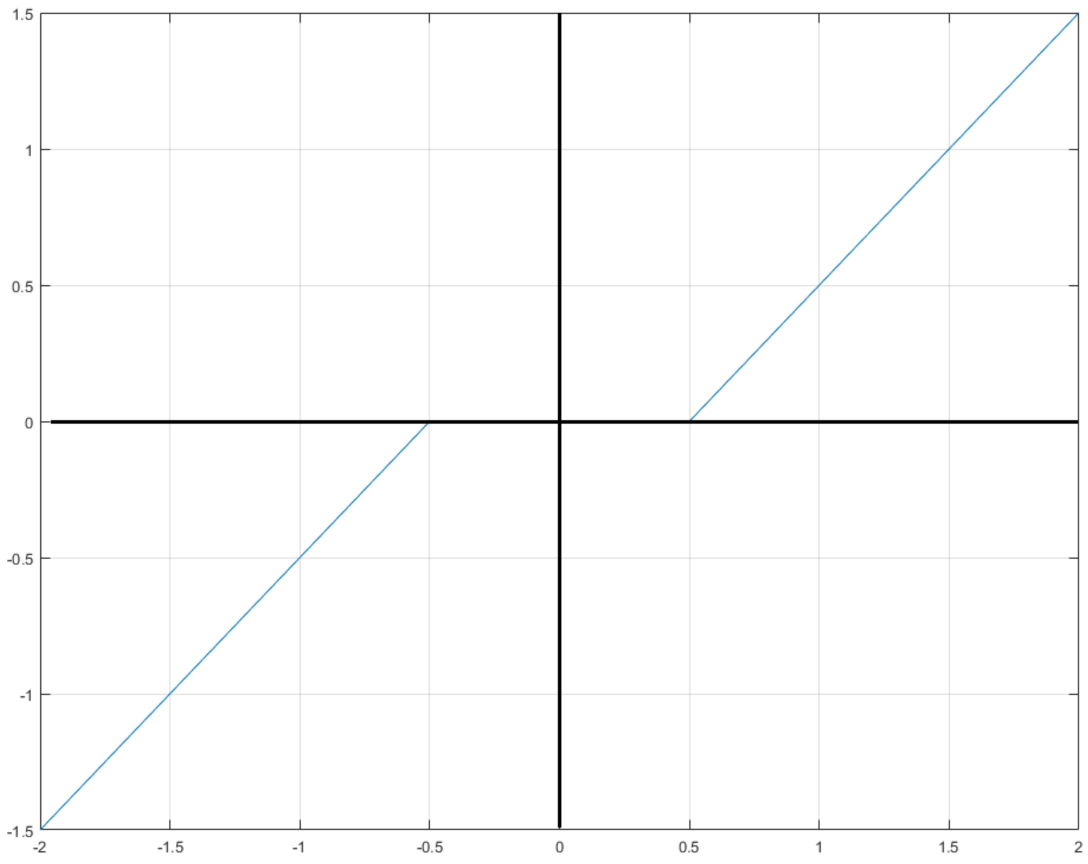


Figure 2.2: Symmetrical Soft Thresholding

where

$$Z_b(x) \doteq \sum_{l \in \Gamma} S_b(\langle x, \varphi_l \rangle)\varphi_l \tag{219}$$

which can be seen as a method to erode the elements of x in the direction of φ_l. Daubechies et. al. have proven that the solution obtained by iterating **t** reaches the global minimum of the solution space and solution method is stable. The solution will reach to an optimum point if K is a bounded operator satisfying $||K\boldsymbol{t}|| \leq k||\boldsymbol{t}||$ for any vector **t** and some constant k [25].

Although more research has been put into regularization methods later starting with Combettes & Wajs [22], the method of Daubechies et. al. is suitable for us to explain training of CNNs later in Chapter 3.

2.3 Data Driven Approaches and CNNs

Instead of approaching the inverse problem to directly invert an observation model, Data Driven Approaches (DDA) learn mappings from input and target training images. Methods either learn a compact dictionary [27][28][29][30] or train a model that fits the problem and learn parameters for the model [31][32].

Dictionary based DDA jointly solve for a compact dictionary and a representation vector. Sparse representation has been applied to the dictionary learning based SR problem where an input image is sparsely represented by a dictionary. The representation vector is applied to another library for reconstruction of output image. These algorithms both solve for creating dictionary and solve for a representation vector for any input.

The K-SVD algorithm [33] is one of the keystones of dictionary learning for the purpose of sparse representation. Aharon et. al. have proposed the usage of a compact dictionary D, from which a set of atoms (columns or dictionary elements) are to be selected via a vector **f** and the combination of these atoms is constrained to be similar to an observation patch (or image) **g** via $||\boldsymbol{g} - D\boldsymbol{f}||_p \leq \varepsilon$. If the dimension of **g** is less than that of columns of matrix D and if D is full-rank matrix then there are infinitely many solutions to the problem therefore a sparsity constraint is introduced [33].

$$\min_{f} ||\boldsymbol{f}||_0 \; s.t. \; ||\boldsymbol{g} - D\boldsymbol{f}||_2 \leq \varepsilon \tag{220}$$

The L_0 norm gives the number of entries in **f** that are non-zero.

The usage of compact dictionaries for SR problem is introduced in [27]. The authors have used the approach of K-SVD to learn representation vectors and dictionaries. Instead of using an L_0 normed regularization and L_1 normed regularization is used which still guaranties sparsity for the formulated problem. Such a problem is formulated using Lagrange multipliers [27].

$$\min_{f} \lambda||\boldsymbol{f}||_1 + \frac{1}{2}||\boldsymbol{g} - D\boldsymbol{f}||_2^2 \tag{221}$$

During learning phase the library D is initialized by random gaussian noise and an iterative algorithm between a batch representation matrix Z and dictionary D refines the dictionary while maintaining sparsity for representation vectors of training set. The estimation of sparse representation vectors are done by using basis pursuit methods. [27] uses two dictionaries, one for low resolution (LR) representation, one for high resolution (HR) reconstruction.

Timofte et. al. [29] have proposed the usage of L_2 norm instead of L_1 norm for even faster computations. Although usage of L_2 norm eliminated the sparsity constraint from the equation, it will play a role in understanding how CNNs work in later sections.

Trainable models are also used for inverse problems. A method, trainable nonlinear reaction diffusion (TNRD) [32], uses a training set to learn a set of filters and nonlinearities that are applied in a cascaded fashion to the input image. The resulting algorithm uses stochastic gradient descent type of learning. The minimized objective function uses a suitable proximal mapping function among previously inspected functions [22]. In [32], input image is filtered by a number of trained filters. Filtered images are then applied to a non-linear proximal function and then the output of nonlinearities are further filtered by the same number of filters. The result is obtained by summing all the result. The approach of TNRD is quite similar to how a CNN operates although the model is not weaved in a network fashion since it lacks the hidden layers which give CNNs its power to learn highly nonlinear surfaces (manifolds).

Another trainable model type method is proposed by Romano et. al. rapid and accurate image super resolution (RAISR) [31]. The algorithm learns a set of filters for training patches that are composed of differently oriented edges with various strength and coherencies. Separation of different content patches is carried out according to gradient information of input images. Even though RAISR uses a set of filters in a convolutional structure, the algorithm does not constitute a network structure.

In the last decade the usage of CNNs for inverse problems have dramatically increased, mainly due to advancing technology.

One of the first applications of inverse problems on CNNs was proposed by Gregor & LeCun [23] as learned iterative shrinkage thresholding algorithm (LISTA). The method deals with sparse coding problem that is formulated by 221. Each layer of neural network type model of LISTA represents an unfolded iteration of IST type of solver that was inspected by Combettes et. al. as iterative soft-thresholding [22]. The forward pass of the algorithm approximates iterations of an IST algorithm. Training of LISTA is carried out in gradient descent type learning. Later Bronstein et. al. [34] have addressed the same algorithm and showed that the resulting layers of LISTA are not mere approximations to an IST algorithm but fully functional sparse encoders in their own right.

In [16] authors have described representation learning as a manifold learning for which a higher dimensional data is represented compactly in a lower dimensional manifold. They have discussed that the variations in the input space is captured by the representations and each element of a network represents a higher dimensional coordinate point of the manifold. In the same paper authors have discussed the challenge of training deep networks; learning dynamics, convergence to good minimas of the cost function. The training of neural networks is not mathematically understood in current literature [16].

Schuler et. al. [35] have proposed to unroll the recovery steps required for a deblurring operations with a neural network such as estimation of the blur kernel and estimation of deblurred image. The blur kernel is estimated by using a separate network and then it is used in an other network to reconstruct the latent image. Two networks have been trained separately with different heuristics that the authors empirically found useful. The training set was composed of sharp and blurred image patches. Flat image patches that are distinguished by small gradient content was taken out from the training set. The reasons for formation of specific network structure with different activation functions was not discussed in the paper.

Xu et. al. [36] have proposed to improve end result by initializing the network parameters using information from estimated blur kernels. This was done using psuedo-inversion of separable kernels which is obtained by singular value decomposition. The network architecture is composed of two separate steps, deconvolution and outlier elimination, which are trained separately. The authors have shown that proper initialization of network parameters provided better results compared to random initializations.

Mao et. al. [21] have proposed the usage of symmetric convolutional-deconvolutional layers, hoping that the convolutional layers will encode the important features of the image while rejecting defects and deconvolutional layers will reconstruct the image without the defects. Their experiments have shown that proposed architecture yields better results compared to sole convolutional architectures. The authors have also used skip connections between convolutional and deconvolutional layers, expecting that it will cope with gradient vanishing problem for deeper networks.

Zhang et. al. [37] have proposed a denoising network by exploiting the similarities of execution of denoising CNNs to TNRD equations. The proximal function have been replaced with rectified linear unit (RELU) [38]. Also hidden layers have been added in between two-layer structure of TNRDs and batch normalizations at every second layer. Although the effects of additional operations to the mathematical justification of the network is not clear, the experimental results showed improvement compared to previous methods.

Yang et. al. [24] have proposed a network called ADMM-Net for optimizing a compressive sensing reconstruction problem, taking on from the iterative steps of ADMM algorithm [26]. Each step of the network is weaved accoring to split augmented lagrangian solver approach. This approach separates the method from many methods in literature where a CNN structure is used for various inverse problem solutions without mathematical reasoning. Training of cascaded stages are done end-to-end using a type of gradient descent method and the end result becomes unrolled iterations of ADMM algorithm.

The mapping between the high resolution (HR) and low resoultion (LR) images can also be found by convolutional networks for SR problem ([3], [4]).

The activation function plays an important role in neural network training. In many state-of-the-art algorithms major functions such as tanh and softmax have been replaced by rectified linear units (RELU) [38] that are linear approximations of mathematically complex and computationally heavy functions. Glorot et. al. [39] have empirically shown that by using rectified activations, the network can learn sparse representations easier. For a given input, only a subset of hidden neurons are activated as in Figure 2.3, leading to better gradient backpropagation for learning and

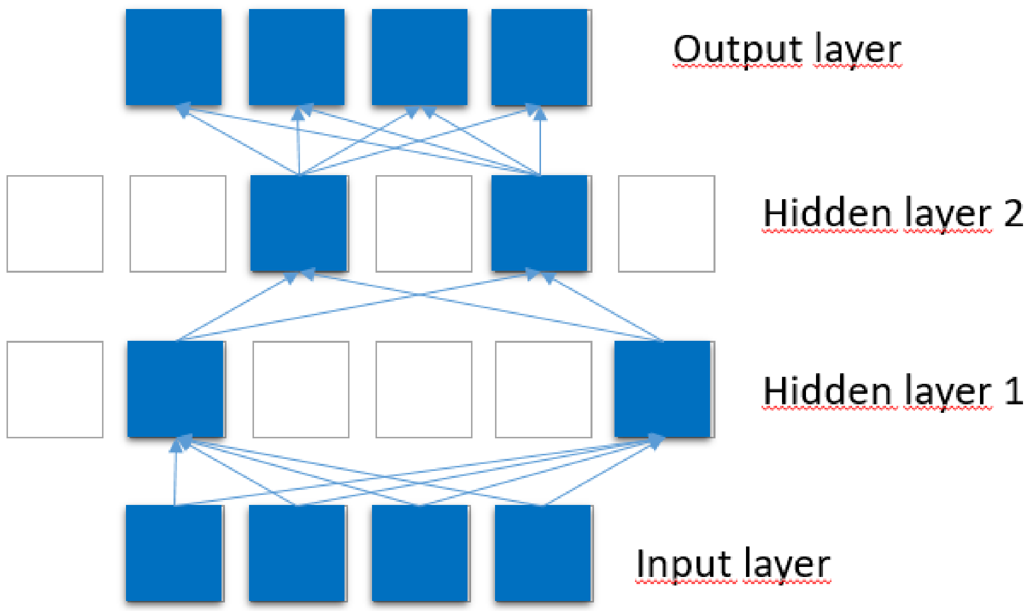


Figure 2.3: Sparse activation of neurons inside a neural network

better representations during forward pass. Especially sparse representation has been shown to be useful [39]. Sparsity constraint provides information disentagling which allows the representation vectors to be robust against small changes in input data. The equation for RELU, or nonnegative soft thresholding, is given in equation 222 and depicted in Figure 2.4

$$soft_b(x) = \begin{cases} x - b & if\ x \geq b \\ 0 & o.w. \end{cases} \tag{222}$$

Dong et. al. [3] have provided the earliest relation of CNNs to Sparse Representation. In their view outputs of the first layer of neurons constitute a representation vector for a patch around each pixel in LR image, second layer maps LR representations to HR representation vectors and the last layer reconstructs HR image using 5x5 sized filters (or atoms if we have used the jargon of sparse representations). Although this idea qualitatively maps CNNs as a solution method for sparse representation problem, in Chapter 3 we will show a more complete understanding with mathematical background.

Bruna et. al. [40] have used CNN for extraction of LR representation which would be used to reconstruct a proper HR image that is picked according to a gibbs distribution.

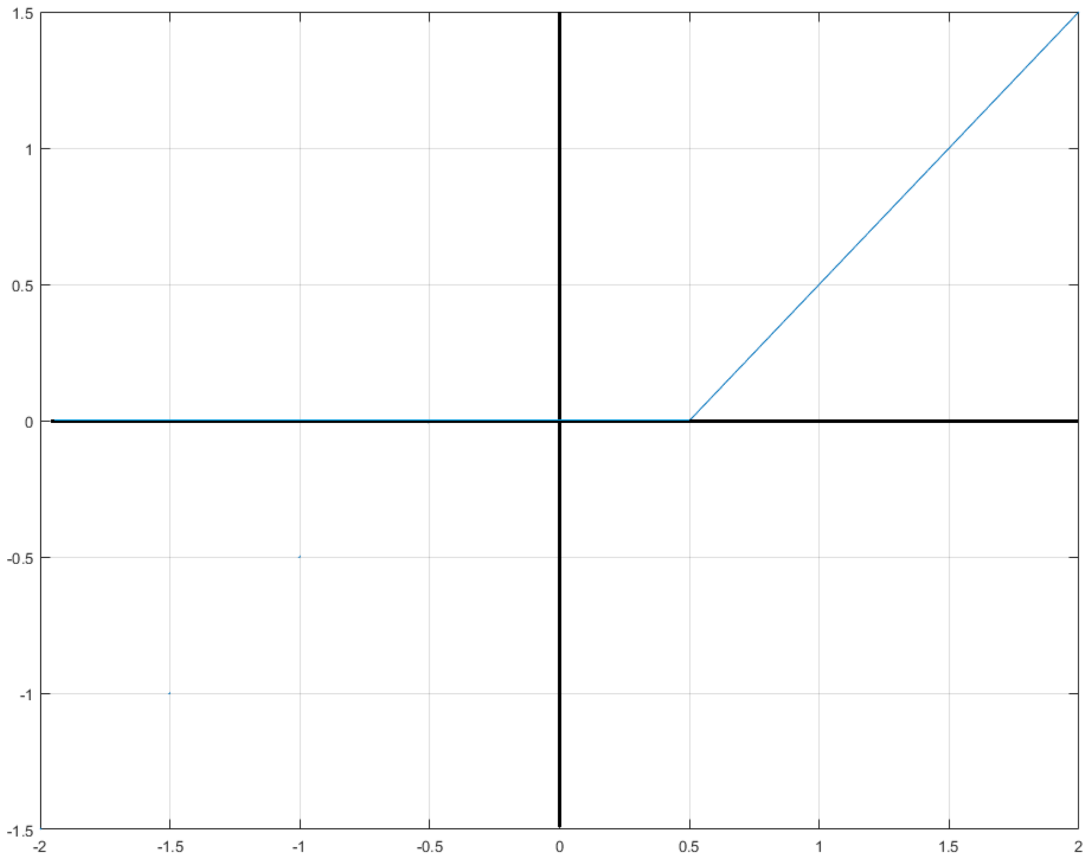


Figure 2.4: Nonnegative Soft Thresholding

Kim et. al. [4] have proposed usage of deep networks for SR problem and their unique approach of using residual learning helped the large network to converge in a reasonable time. Detailed literature survey on SR problem is going to be given in Chapter 4.

In a recent paper, Papyan et. al. [20] have discussed the connection of convolutional sparse coding and CNNs. By inspecting the activation of each layer of a trained network, it is proven that CNN layers become sparse representation dictionaries.

There is a clear distinction between analytic methods and data driven methods, which is the mathematical basis from which the algorithms have evolved. The inspected literature on inverse problems with CNNs contain no mathematical discussions or mostly qualitative discussions. For this reason on the next chapter, we are going to provide mathematical understanding over how CNNs learn, we will prove that CNN elements are optimum elements in solving inverse problems and explain the limitations of structure of CNN with guides on how to overcome the limitations.

CHAPTER 3

MATHEMATICAL PROPERTIES OF CNNS

It is as important to understand how a CNN learns as much as what a trained CNN represents or resembles. The latter is discussed in literature for many different areas of application, the former is lacking proper explanation and mathematical discussion. The main reason is that a mathematical model for a CNN cannot be obtained explicitly, the fastest method to analyze how a CNN learns is to let it learn. For this reason we start from the very beginning and formulate CNN operations for simplified cases. We show the optimality of CNN elements and conditions for optimality.

3.1 Convolutional Neural Networks

CNNs are different from fully connected neural networks (FCN). In a conventional FCN each neuron from any layer is connected to all the outputs from previous layer neurons. A connection diagram is depicted in figure 3.1 where there is a coefficient for each input to the neuron unit. In a CNN, neuron units are grouped to form a 2D filters and consequently their receptive field from an input is limited by their group size, such as 3x3 or 5x5. Differently from FCN, groups of neurons in a CNN are convolving the entire image as if they are filters.

To understand how an image is evolved through a CNN we can explain the operation of one of earliest algorithms, SRCNN [3]. The algorithm has three layers. First layer contains 64 filters of size 9x9. Second layer consists of 32 filters of size 1x1. Third layer consists of 32 filters of size 5x5. The hidden layers have to be considered differently from input and output layers. There will be 64 outputs presented at the output of the first layer. Third layer requires 32 inputs. The hidden layer in between

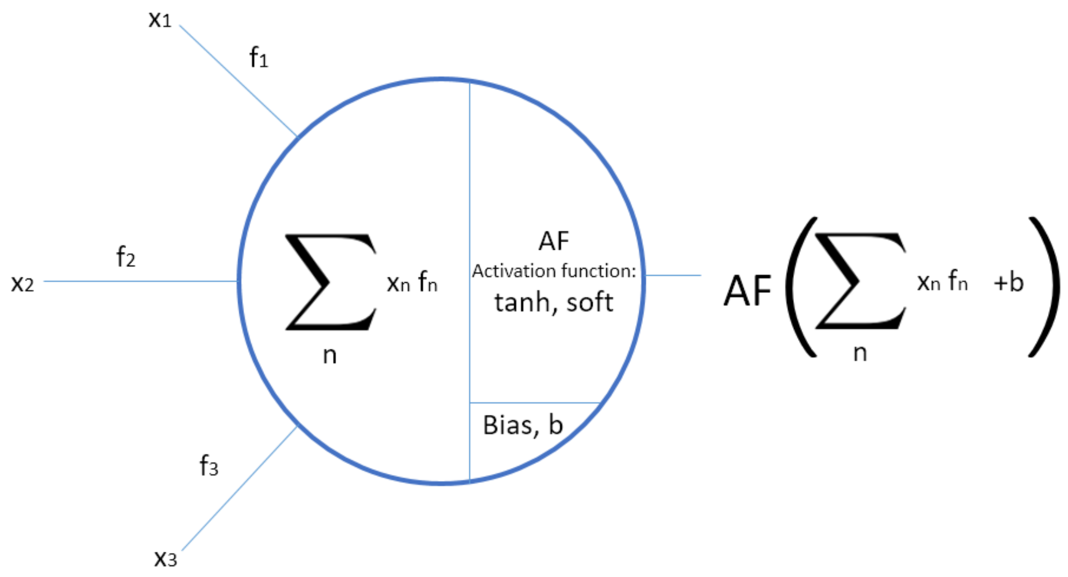


Figure 3.1: A Neuron depiction: three inputs (x), three coefficients (f), an activation function (tanh), a bias value (b).

needs to harbor 32 sets of filters, each set has to contain 64 filters in them. To explain filters and their operation, a detailed depiction in figure 3.2 is presented. Activation functions are omitted from the figure for simplicity.

With a different perspective, one can inspect the progression of one pixel inside the network. Convolutive nature of filters is going to cause the dispersion of information from one pixel to a larger area. By omitting activation functions for simplicity, Figure 3.3 displays the are where one pixel is dispersed through the network. The total amount of area where one output pixel is affected is named as receptive field of a network in the literature. Having a larger receptive field means that the reconstructed pixels are utilizing more input data to be reformed.

3.2 Training (Learning) Phase

For the training phase of CNNs, input images are fed into the network for forward pass. The resulting image from the network is compared against a ground truth (GT) image and the error is backpropagated. Since the input image is convolved by the neuron filter, its size should be larger than the size of the output to prevent boundary

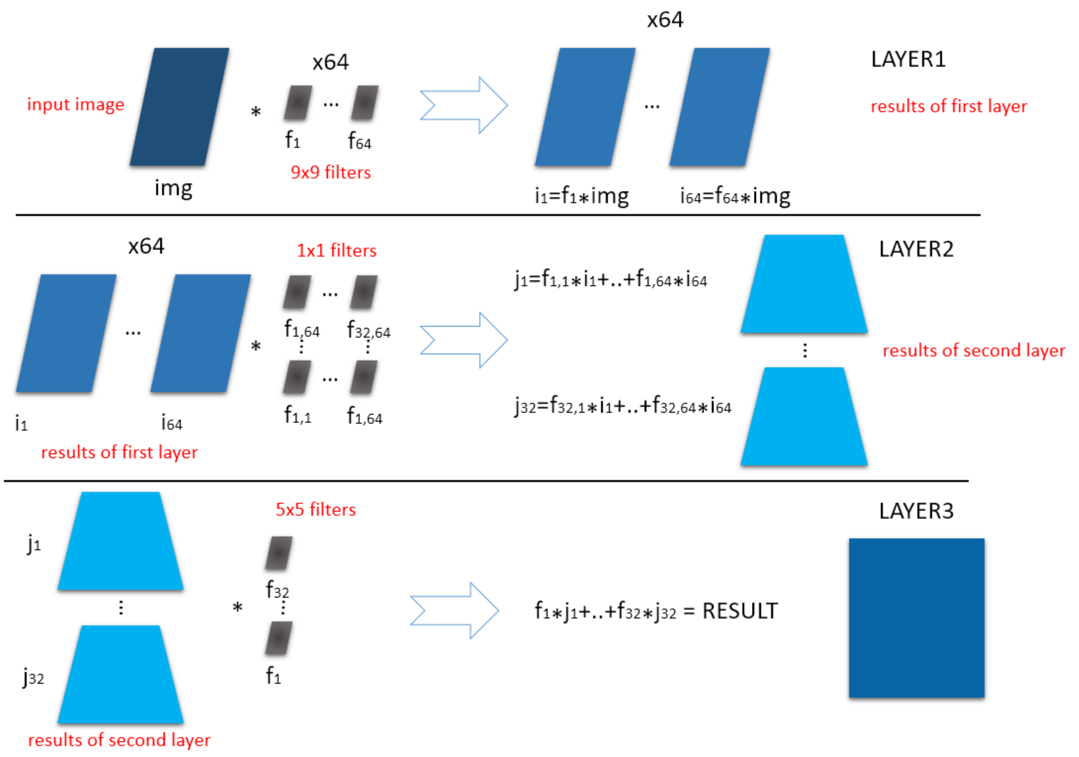


Figure 3.2: Path of an input image through a CNN. Based on the architecture of SRCNN [3].

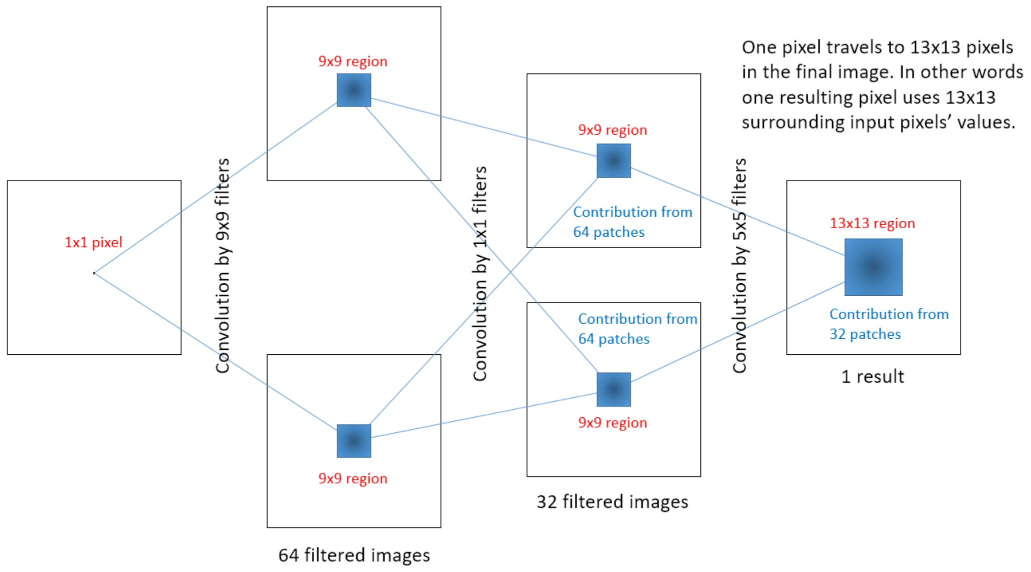

1x1 pixel
Convolution by 9x9 filters
9x9 region
9x9 region
64 filtered images
Convolution by 1x1 filters
9x9 region
Contribution from 64 patches
Contribution from 64 patches
9x9 region
32 filtered images
Convolution by 5x5 filters
13x13 region
Contribution from 32 patches
1 result
One pixel travels to 13x13 pixels in the final image. In other words one resulting pixel uses 13x13 surrounding input pixels' values.

Figure 3.3: Path of an input pixel through a CNN. Based on the architecture of SR-CNN [3].

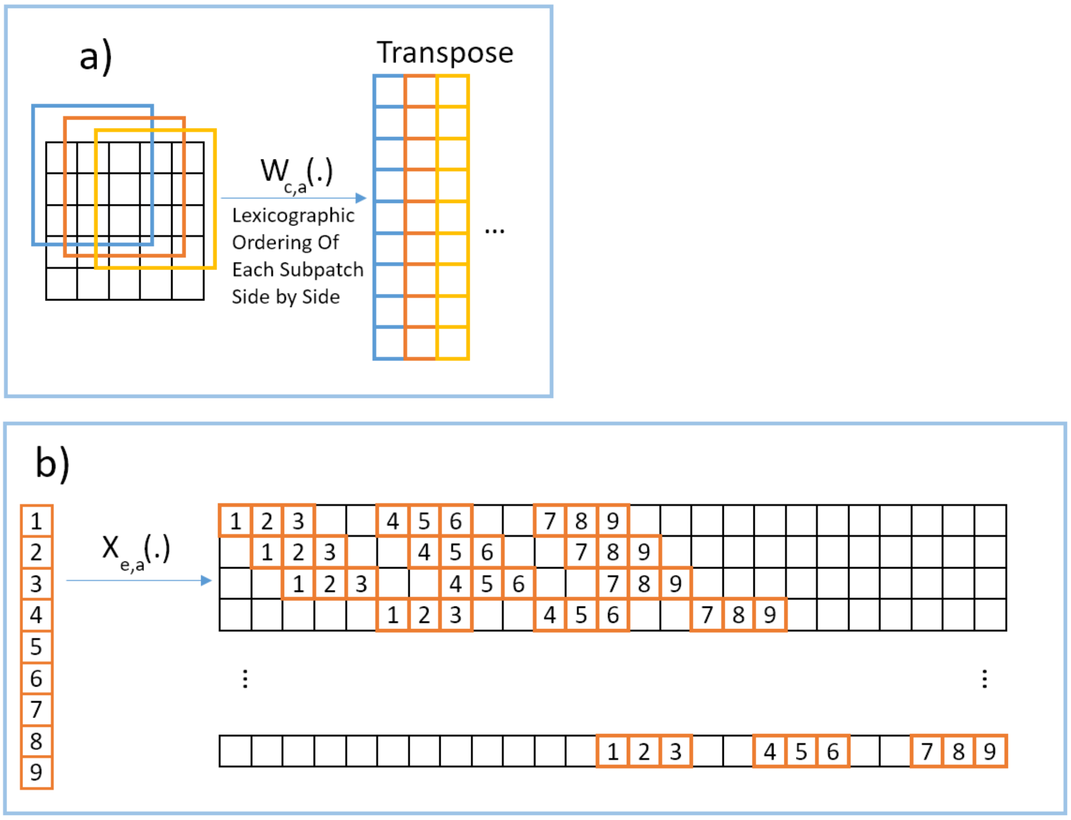


Figure 3.4: Operators: a) W(.) operator for lexicographical dictionary creation from a patch, b) X(.) operator for equivalent operation for an order swapping of cascaded convolutions

conditions. It is reported that boundary conditions do not cause a trouble in a residual learning environment [4].

Before moving on to explaining CNN operations we have to define a few operators. These operators will be useful in describing convolutional operations with algebraic equations. Take a filter, f, of size axa; an input patch, x_{k-1}, of size cxc. The valid part of the output, x_k, of convolution of f and x_{k-1} will have a size of $(c-a+1)$x$(c-a+1)$.

$$x_k = x_{k-1} * f$$

where $*$ is convolution operation. To show these operations algebraically we define a new operator $\mathrm{W}_{c,a}(.)$ which takes smaller patches (subpatches) from a larger patch (superpatch), orders them in lexicographical order then concatenates all vectors into

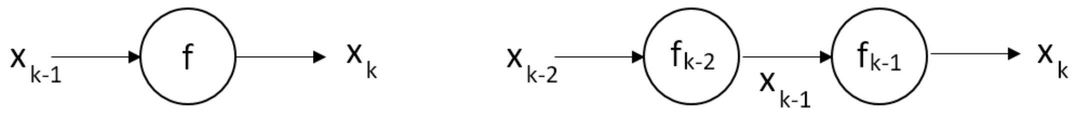


Figure 3.5: (left) Single neuron filter (right) Two cascaded neuron filters

a matrix. This operator is depicted in Figure 3.4 a. Subscripts indicate the sizes of x_{k-1} and f respectively as cxc and axa. The resulting matrix from W$_{c,a}$(.) operator will have a size of $(c - a + 1)^2$ x a^2.

$$\boldsymbol{x_{k-1}} * \boldsymbol{f} = W_{c,a}(x_{k-1}).\boldsymbol{f} \tag{31}$$

We denote lexicographically vectorized 2-D patches with bold characters. Notice in this case result of convolution $x_{k-1} * f$ is also vectorized. We also define an X$_{e,a}$(.) operator to denote an order swapping between two cascaded convolutions as shown in Figure 3.4 b. The index a indicates the size of the filter, axa, that is given as input to the operator. The index e indicates the size of the image patch, exe, with which the input filter is going to be convolved. Take an input x_{k-2} as in Figure 3.5 b, with size exe, and two filters f_{k-2} and f_{k-1} with sizes axa the resulting operations will be as follows

$$\begin{aligned} x_{k-2} * f_{k-2} * f_{k-1} &= x_{k-2} * f_{k-1} * f_{k-2} \\ &= X_{e,a}(f_{k-1}).W_{e,a}(x_{k-2}).\boldsymbol{f_{k-2}} \end{aligned} \tag{32}$$

To start analyzing a CNN we are going to take one CNN element, a neuron filter into consideration depicted in Figure 3.5 a.

$$x_k = soft_{\boldsymbol{b}}(x_{k-1} * f) \tag{33}$$

where $soft_b$, nonnegative soft threshdoling, is defined for each scalar input as in equation 34. For vector inputs same operations are applied to each element in the vector. The output of a neuron filter is outcome of nonnegative soft thresholding applied to the convolution result. Soft thresholding is formulized as in Equation 34 or equivalently 222.

$$soft_{\boldsymbol{b}}(\boldsymbol{x}) \doteq max(\boldsymbol{x} - \boldsymbol{b}, 0) \tag{34}$$

During training a target image, t, is used to calculate an error. The mean square error is used for gradient calculation and parameter updates. We define a dictionary for learning (training) phase as

$$\begin{aligned} D_{L,k-1} &\doteq W_{c,a}(x_{k-1}) && (35)\\ \boldsymbol{x_k} &= soft_{\boldsymbol{b}}(D_{L,k-1}\boldsymbol{f}) && (36)\\ error &= \boldsymbol{t} - soft_{\boldsymbol{b}}(D_{L,k-1}\boldsymbol{f}) && (37)\\ mse &= \frac{1}{2}||\boldsymbol{t} - soft_{\boldsymbol{b}}(D_{L,k-1}\boldsymbol{f})||^2 && (38)\\ \frac{\partial mse}{\partial \boldsymbol{f}} &= -(D^T_{L,k-1}(\boldsymbol{t} - soft_{\boldsymbol{b}}(D_{L,k-1}\boldsymbol{f}))) && (39) \end{aligned}$$

Moving forward with CNN learning operations, parameter updates are carried out by adding negative gradient on top of old value.

$$\begin{aligned} \boldsymbol{f}^n &= \boldsymbol{f}^{n-1} - \frac{\partial mse}{\partial \boldsymbol{f}^{n-1}} && (310)\\ &= \boldsymbol{f}^{n-1} + D^T_{L,k-1}(\boldsymbol{t} - soft_{\boldsymbol{b}}(D_{L,k-1}\boldsymbol{f}^{n-1})) && (311) \end{aligned}$$

where n is the iteration number. In Daubechies et. al. [25] (Remark 2.4) the learning rate was introduced with a matrix, G, that is diagonal in φ_l basis as $G\varphi_l = \eta_l\varphi_l$. The term η_l modifies equation 311 as

$$\boldsymbol{f}^n = \boldsymbol{f}^{n-1} + \frac{1}{\eta_l}D^T_{L,k-1}(\boldsymbol{t} - soft_{\boldsymbol{b}}(D_{L,k-1}\boldsymbol{f}^{n-1})) \tag{312}$$

Daubechies et. al. have commented that the solution can be followed by this equation but they have omitted it for simplicity of equations. It will affect the convergence speed in general but it was proven that the solution reaches to optimum point either way. Since we show that CNN gradient descent based learning yields the same solution as Daubechies' solution we also omit this term in equation 312 and use equation 311.

The f^n can be defined piece-wise as

$$\boldsymbol{f}^n = \begin{cases} \boldsymbol{f}^{n-1} + D_{L,k-1}^T(\boldsymbol{t} - D_{L,k-1}\boldsymbol{f}^{n-1} + \boldsymbol{b}) \\ \qquad\qquad\qquad\qquad if\ D_{L,k-1}\boldsymbol{f}^{n-1} > \boldsymbol{b} \\ \boldsymbol{f}^{n-1} + D_{L,k-1}^T\boldsymbol{t} \\ \qquad\qquad\qquad\qquad if\ D_{L,k-1}\boldsymbol{f}^{n-1} \leq \boldsymbol{b} \end{cases} \tag{313}$$

We want to collect the iterations inside the soft thresholding to modify equations towards Daubechies' solution.

$$\boldsymbol{f}^n = soft_{\boldsymbol{b'}}(\boldsymbol{f}^{n-1} + D_{L,k-1}^T(\boldsymbol{t} - D_{L,k-1}\boldsymbol{f}^{n-1})) \tag{314}$$

To have f^n definitions be equal, we would have to describe b' as

$$\boldsymbol{b'} = \begin{cases} -D_{L,k-1}^T\boldsymbol{b} & D_{L,k-1}\boldsymbol{f}^{n-1} > \boldsymbol{b} \\ -D_{L,k-1}^T D_{L,k-1}\boldsymbol{f}^{n-1} & D_{L,k-1}\boldsymbol{f}^{n-1} \leq \boldsymbol{b} \end{cases} \tag{315}$$

Although the split seems to describe four zoned thresholding function, this is not the case. Since we are dealing with natural images $D_{L,k-1}$ and t have positive values, which reduces the thresholding function to two zones again. Also it is important to emphasize that we are not proposing new functions for the training of a neuron. We are changing the equations to make them tractable.

For simplicity define

$$\boldsymbol{e}^n \doteq \boldsymbol{f}^{n-1} + D_{L,k-1}^T(\boldsymbol{t} - D_{L,k-1}\boldsymbol{f}^{n-1}) \tag{316}$$

Introduce $\{\varphi_l\}$ a CON basis vector set of Hilbert space.

$$\langle \boldsymbol{f}^n, \varphi_l \rangle = \langle soft_{\boldsymbol{b'}}(\boldsymbol{e}^n), \varphi_l \rangle \tag{317}$$

Now define

$$h^n \doteq soft_{b_l}(\langle \boldsymbol{e^n}, \varphi_l \rangle) \tag{318}$$

Explicitly

$$\langle \boldsymbol{f}^n, \varphi_l \rangle = f^n[1]\varphi_l[1] + ... + f^n[M]\varphi_l[M] \tag{319}$$

$$h^n = e^n[1]\varphi_l[1] + ... + e^n[M]\varphi_l[M] - b_l \tag{320}$$

where M is the length of vector f. Since we actively control b' values for each computation, we know that $soft_{\boldsymbol{b'}}$ is active all times, only b' will change. Thus $f^n[i] = e^n[i] - b'[i]$ at all times. Therefore to have h^n equal to $\langle \boldsymbol{f^n}, \varphi_l \rangle$ we should have $b_l = \langle \boldsymbol{b'}, \varphi_l \rangle$.

$$\begin{aligned}
\boldsymbol{f}^n &= \sum_{\forall l} \langle \boldsymbol{f}^n, \varphi_l \rangle \varphi_l = \sum_{\forall l} soft_{b_l}(\langle \boldsymbol{e}^n, \varphi_l \rangle)\varphi_l \\
&= \sum_{\forall l} S_{b_l}(\langle \boldsymbol{f}^{n-1} + D^T_{L,k-1}(\boldsymbol{t} - D_{L,k-1}\boldsymbol{f}^{n-1}), \varphi_l \rangle)\varphi_l
\end{aligned} \tag{321}$$

where S_{b_l} is defined in equation 216. Since the basis vectors can be chosen with two different directions without loss of generality we can choose φ_l such that the innerproduct (or projection) always yields a nonnegative result. By doing so, we can use symmetric soft thresholding (S_b) instead of nonnegative soft thresholding ($soft_b$). Usage of symmetric thresholding enables CNN learning equations to become exact inverse problem solutions. We will replace operator K in equation 218 with $D_{L,k-1}$ in equation 321. From previous discussion we know that the operator needs to be bounded for the solution to exist for equation 218. In this case $D_{L,k-1}$ matrix which is composed of image patches is bounded. Therefore equation 321 shows that a neuron filter solves an inverse problem during training and that it is optimal and stable solution.

A neuron filter solves a system as in equation 322

$$\boldsymbol{t} = D_L \boldsymbol{f} + \boldsymbol{n'} \tag{322}$$

where t is the target image as before, D_L is the dictionary matrix constructed from input data and f is the neuron filter to be learned, n' is noise. The solution of such a system is given by equation 321. Notice that the training step is an intermediary step of using a CNN for the actual inverse problem of equation 21. As we will discuss in section 3.3, learned filters will change roles during testing.

The generalization of a single CNN element (neuron filter) to the entire network is mathematically cumbersome. We can make analysis on a subset of a network and then use generalized methods and theorems to understand what and how CNN learns. For that we analyze two neuron filters cascaded and show that how they learn is similar to a single unit. Two units are depicted in Figure 3.5. We use S(.) for soft thresholding in following equations for simplicity without denoting biases. Take a patch x_{k-2} with size exe; x_{k-1} with size cxc; f_{k-2} and f_{k-1}, with size axa.

$$\begin{aligned} x_{k-1} &= S_1(x_{k-2} * f_{k-2}) \\ x_k &= S_1(x_{k-1} * f_{k-1}) = S_1(S_2(x_{k-2} * f_{k-2}) * f_{k-1}) \end{aligned} \tag{323}$$

Update of f_{k-1} follows the same steps as discussed before. The update of f_{k-2} requires derivation of mse w.r.t. f_{k-2} where we will utilize the $X_{e,a}$ operator.

$$mse = \frac{1}{2}||\boldsymbol{t} - S_1(D_{L,k-1}\boldsymbol{f_{k-1}})||^2 \tag{324}$$

$$= \frac{1}{2}||t - S_1(X_{e,a}\boldsymbol{(f_{k-1})}S_2(D_{L,k-2}\boldsymbol{f_{k-2}}))||^2 \tag{325}$$

To calculate gradient, define a modified dictionary

$$P \doteq X_{e,a}(f_{k-1})D_{L,k-2} \tag{326}$$

Then we can calculate gradient of MSE. For readability issues we will stop using bold letters for vectors.

$$\frac{\partial mse}{\partial f_{k-2}} = -(P^T t - P^T S_1(X_{e,a}(f_{k-1})S_2(D_{L,k-2}f_{k-2}))) =$$

$$\begin{cases} -(P^T(t - Pf_{k-2} + X_{e,a}(f_{k-1})b_{k-2} + b_{k-1})) \\ \qquad if\ D_{L,k-2}f_{k-2} > b_{k-2} \\ \qquad and\ if\ X_{e,a}(f_{k-1})(D_{L,k-2}f_{k-2} - b_{k-2}) > b_{k-1} \\ -(P^T t) \quad o.w. \end{cases} \tag{327}$$

Then we calculate

$$f_{k-2}^{new} = f_{k-2} - \frac{\partial mse}{\partial f_{k-2}} =$$

$$\begin{cases} f_{k-2} + P^T(t - Pf_{k-2} + X_{e,a}(f_{k-1})b_{k-2} + b_{k-1}) \\ \qquad if\ D_{L,k-2}f_{k-2} > b_{k-2} \\ \qquad and\ if\ X_{e,a}(f_{k-1})(D_{L,k-2}f_{k-2} - b_{k-2}) > b_{k-1} \\ f_{k-2} + P^T t \quad o.w. \end{cases} \tag{328}$$

We can also define a new equivalent function as in previous case, by choosing bias value as given in equation 330.

$$f_{k-2}^{new} \doteq S_{b'}(f_{k-2} + P^T(t - Pf_{k-2})) \tag{329}$$

$$b' = \begin{cases} -X_{e,a}(f_{k-1})b_{k-2} - b_{k-1} \\ \qquad if\ D_{L,k-2}f_{k-2} > b_{k-2} \\ \qquad and\, if\ X_{e,a}(f_{k-1})(D_{L,k-2}f_{k-2} - b_{k-2}) > b_{k-1} \\ = -Pf_{k-2} \quad o.w. \end{cases} \tag{330}$$

Equation 329 leads to the same equation as we have reached before in equation 314. From this point we can generalize that a CNN can solve for an inverse problem during training by an IST algorithm. The aim of a neuron filter, or the entire network, is to match its output to a target image to minimize error in equation 37. The estimation of neuron filters in this way is analogous to basis pursuit of K-SVD algorithm for sparse representation estimation. Therefore the vector f is a representation vector in equation 37. The dictionary, D_L, upon which representation of target, t, is found is constructed out of subpatches from low resolution image (superpatch), this is only convenient because during testing (forward pass) the only information, from which the inverse problem is solved, is the input image itself.

3.3 Testing (Reconstruction) Phase

For the testing phase, a new representation - dictionary duality (RDD) concept is proposed. RDD concept states that the representation vectors learned during the training phase can be used as atoms of a dictionary for the testing phase. The cost function that is minimized by CNN training (learning) yields a representation vector as the neuron filter. During testing (scoring, reconstruction) phase, resulting representation vectors (filters) from a layer of neurons turn into a dictionary (later named as D_R) upon which the reconstruction of output image is carried out. We propose the idea that dictionaries and representations swap roles during training and testing. Also during training, inputs to each layer is perceived as a dictionary for the next layer. Following the idea of RDD, the neuron filter can be viewed as an atom of a dictionary consisting of

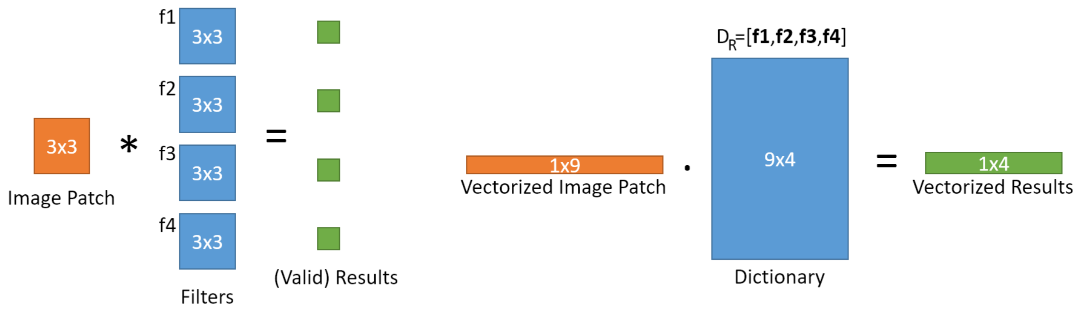


Figure 3.6: Formation of Reconstruction Dictionary (D_R)

many other neuron filters among the network layer. During testing period, the neuron filters are vectorized and concatenated to form the *reconstruction* dictionary matrix $\mathrm{D}_R = [\boldsymbol{f}_1; \boldsymbol{f}_2; \boldsymbol{f}_3...]$. A layer's output will be the *representation* vector of input image in terms of the dictionary atoms, i.e. the neuron filters. This is illustrated in Figure 3.6

In order to explain the representation problem analogy better we use ideas from two papers by Papyan et. al. [20][41]. The authors have described a convolutional sparse coding problem where an observation is represented by a dictionary and its representation vector is further represented by another dictionary for a number of layers. They have proven, in Theorem 1, that layered representations can be estimated by a CNN based forward pass where mentioned layered dictionaries are filters of each layer of CNN. (Theorem 10 in original text [20])

Theorem 1 Suppose **g = y + n** where **n** is noise whose the power of noise is bounded by ε_0 and **y** is a noiseless signal. Considering a convolutional sparse coding (CSC) structure where $D_{R,l}$ is the dictionary, constructed from l^{th} layer filters

$y = D_{R,1}x_1$

$x_1 = D_{R,2}x_2$

.

.

$x_{N-1} = D_{R,N}x_N$

Let $\hat{x}_i$ be a set of solutions obtained by running a convolutional neural network, or layered soft thresholding algorithm with biases b_i as $\hat{x}_i = soft_i\{D^T_{R,i}\hat{x}_{i-1}\} where\ \hat{x}_0 = g$. Denote $|x_i^{max}|$ and $|x_i^{min}|$ as absolute maximum and minimum entries of x_i. Then

assuming for $\forall 1 \leq i \leq N$
$||x_i||_0 < \frac{1}{2}(1 + \frac{1}{\mu(D_{R,i})}\frac{|x_i^{min}|}{|x_i^{max}|}) - \frac{1}{\mu(D_{R,i})}\frac{\varepsilon_{i-1}}{|x_i^{max}|}$
where $\mu(D_{R,i})$ is the mutual coherence of the dictionary then

1. The support of the solution $\hat{x}_i$ is equal to the support of x_i

2. $||x_i - \hat{x}_i||_2 \leq \varepsilon_i$
$where\ \varepsilon_i = \sqrt{||x_i||_0}(\varepsilon_{i-1} + \mu(D_{R,i})(||x_i||_0 - 1)|x_i^{max}| + b_i)$

The mutual coherence $\mu(D_{R,i})$ is defined as $\underset{n \neq m}{min}|d_{R,i,n}^T d_{R,i,m}|$ where $d_{R,i,n}$ is n^{th} column of $D_{R,i}$ [20]. Although as stated by the Papyan et. al. there are tighter conditions for mutual coherence calculation. Later in this section we are going to analyze a different approach to the proving condition of this theorem.

Using Theorem 1, we can relate sparse representation problem with CNN forward pass. Although image spatial coherency and dictionary (neuron filters') mutual coherence are two distinct measures they are correlated. Theorem 1 is used to show the stability of CNNs for data representation. There are two different stability concepts that are proven by this theorem, one is having bounded response for small perturbations in the input data which is trivial in context of CNNs, the other more important concept of stability is the accuracy of the results given an input data. The theorem is valid, depending on mutual coherence of dictionary elements. The condition is given as

$$||x_i||_0 < \frac{1}{2} + \frac{1}{\mu(D_{R,i})}\frac{1}{2|x_i^{max}|}(|x_i^{min}| - 2\varepsilon_{i-1}) \tag{331}$$

In this condition, at first glance it looks as $\mu(D_{R,i})$ is required to be as low as possible and $\frac{|x_i^{min}|}{|x_i^{max}|}$ as close to 1 as possible. But further analysis reveals that $\mu(D_{R,i})$ and x_i^{min}, x_i^{max} values are interdependent, therefore a straightforward upper bound may not be defined. If mutual coherence decreases, $\underset{i \neq j}{min}|d_{R,i,n}^T d_{R,i,m}|$ decreases. It means the eigenvalue spread of $D_{R,i}^T D_{R,i}$ increases. Given enough number of elements in a dictionary this means maximum and minimum values of $\hat{x}_i$ are also pushed apart. Assuming a stable solution exists, this means an increase in $|x_i^{max}|$ and a decrease in $|x_i^{min}|$, since difference of estimation and true value is bounded by the power of noise. The existence of a solution for noisy case CSC (Theorem 1) is not proven in

Papyan et. al. [20]. Therefore we have to look for conditions that would contradict with the existence of a non-trivial solution. For equation 331 to be valid for non-trivial solution, $\frac{1}{\mu(D_{R,i})}\frac{1}{2|x_i^{max}|}(|x_i^{min}| - 2\varepsilon_{i-1})$ should be greater than $\frac{1}{2}$. We can define a looser but easier bound:

$$|x_i^{min}| > 2\varepsilon_{i-1} \tag{332}$$

If we assume the existence of a stable solution we would have $||x_i - \hat{x}_i||_2 \leq \varepsilon_i$ where $\hat{x}_i = soft_i\{D_{R,i}^T \hat{x}_{i-1}\}$ then

$$|x_i^{min}| = \underset{n}{min}|x_i[n]| \leq \underset{n}{min}|\hat{x}_i[n]| + \sqrt{\varepsilon_i} \tag{333}$$

$$\leq \underset{n}{min}|d_{R,i,n}^T \hat{x}_{i-1}| + \sqrt{\varepsilon_i} \tag{334}$$

$$\leq \underset{n}{min}|d_{R,i,n}^T x_{i-1}| + \sqrt{\varepsilon_i} + \sqrt{\varepsilon_{i-1}} \tag{335}$$

$$= \underset{n}{min}|d_{R,i,n}^T D_{R,i} x_i| + \sqrt{\varepsilon_i} + \sqrt{\varepsilon_{i-1}} \tag{336}$$

$$\leq \underset{n \neq m}{min}|d_{R,i,n}^T d_{R,i,m}|.||x_1||_2 + \sqrt{\varepsilon_i} + \sqrt{\varepsilon_{i-1}} \tag{337}$$

$$= \mu(D_{R,i})||x_1||_2 + \sqrt{\varepsilon_i} + \sqrt{\varepsilon_{i-1}} \tag{338}$$

To satisfy equation 332

$$\mu(D_{R,i}) > \frac{2\varepsilon_{i-1} - \sqrt{\varepsilon_{i-1}} - \sqrt{\varepsilon_i}}{||x_i||_2} \tag{339}$$

If mutual coherence is not above a certain value (equation 339) then the theorem does not hold. Notice as a side note that right hand side in equation 339 cannot be negative. The iterative formulation of ε starts with noise power, in a system where pixel values range from 1 to 255 and each iteration multiplies that value with $sqrt||x_i||_0$ and $|x_i^{max}|$ which makes it always greater than 1. When conditions are not satisfied, $||x_i - \hat{x}_i||_2 < \varepsilon_i$ statement is no longer valid. Starting from $\hat{x}_0 = g$, The estimate $\hat{x}_1$ will be perturbed beyond the power of noise of the input. Recall that we have defined the noiseless signal y as $y = D_{R,1}D_{R,2}..D_{R,N}x_N$ where N is the number of layers, or stages. As the estimate of each stage drifts further from the original representation values, in MSE sense, the end result of N layered CNN becomes an inaccurate representation of the original data.

$$\hat{y} = D_{R,1}D_{R,2}..D_{R,N}\hat{x}_N \tag{340}$$

Although estimating y, in forward problem that is defined as g=y+n, is a denoising problem, which is an example of an inverse problem nonetheless, the stability discussion can be applied to generalized inverse problem for imaging as $g = Kt + n$. Then the equation in the theorem changes as $Kt = D_{R,1}D_{R,2}..D_{R,N}x_N$ and as CNN learns to invert the observation system K, equation becomes $x_N = t$.

Skip connections relay information from previous layers to deeper layers. One must consider both training and testing to understand the effect of skip connections on mutual coherence of layer elements and the stability of CNN overall. During forward pass (testing), information from previous layers are added on deeper layers' outputs. As prior layers' information is more similar to the original data compared to deeper layers' information, Because of less perturbation caused by equation 340. Skip connections help preserving the accuracy of the end result. During backpropagation (training), the total error is carried, via skip connections, to deeper layers. Bypassing the chain rule of gradient calculation enables the true gradient to influence deeper filter coefficients. The initialization of CNN coefficients are done via Gaussian noise. Initial coefficients has the lowest mutual coherence possible. Backpropagating an error that is calculated from difference of target data and input data that is lost within initial coefficients (as in modified dictionary in equation 326), without losing fidelity becomes almost impossible for deeper networks. This is the reason why the middle layers always have lowest mutual coherence (see Table 5.1 in Chapter 5). Skip connections bypasses effect of many layers on the true gradient therefore skip connections help updating deeper layers with more accurate gradients. Consider the modified dictionary in equation 326. With skip connections the gradient descent will use direct input information $D_{L,k-2}$ instead of $X_{e,a}(f_{k-1})D_{L,k-2}$ that is dictionary modified with potentially unreliable f_{k-1} at the initial stages of training. (To be more precise, with skip connection, gradient descent will use a mixture of $D_{L,k-2}$ and P). Thus with skip connections filter coefficients become more accurate in representing noiseless input data Kt, then output of CNN becomes more accurate in representing the target data t. Therefore we have shown the benefit of skip connections between hidden layers or input-output layers as in residual learning [4] which provide the network with necessary mutual coherence. Mao et. al. [21] have proposed the usage of skip connections between convolution and deconvolution layers to prevent input information to be lost inside the encoder-decoder structure where an image is downscaled and upscaled by multiple stride convolution operators. Differently our reasoning for the usage of skip connections is to keep mutual coherence of filter coefficients high. This will provide the trained network to produce accurate results according to the error criterion used. During training, each layer's output is used as a dictionary for the next layer. The mathematical analysis of the learning process revealed in equation 327 that initial filters are updated with modified dictionaries P that carry information of all the layers beyond it. Skip connections will reduce the variance of filters in different layers by carrying information across the network during forward and backward passes of training process. Although we have shown that CNNs benefit from skip connections exact structure is still subject to experimental refinement depending on the application.

This discussion is also in favor of residual learning which is a skip connection between input and output layers. It is proposed [4] that residual learning quickened training for deep networks. We can show few reasons for its usefulness. First reason is that the initial filters are not going to be updated by data that is backpropagated through many layers and modifications but directly from the output MSE. Secondly, RDD supports the explicit updating of initial layers that will help construction of a more definite filter set which is going to be used as feature selectors for the rest of the network during testing. Lastly, input data is added to the last layer's output which will become a last dictionary for the training of reconstruction filters for output stage.

CHAPTER 4

SUPERRESOLUTION USING CNNS

Single Image Superresolution (SISR), i.e. reconstructing a high resolution (HR) image from a given low resolution (LR) image, is an ill-conditioned inverse problem. In the last decade the usage of CNNs for inverse problems have dramatically increased, mainly due to advancing technology.

With the introduction of more complex structures, the pixel-wise error performance of the networks increased. As error performance increased, the computation and training time of the networks also increased. Parameter count for networks reached to as high as 43 Million [42] and training time reached to the order of weeks [43] for the best performing networks. As usage of shared parameters became wide-spread [44][45][46] the number of operations per output pixel (ops/op) has also become an important measure determining the true size and speed of a network.

Mobile systems and real-time operating commercial or military equipment consists of lower memory and lower processing power compared to high-end lab equipment such as multiple GPUs and CPUs. Considering that there are multiple algorithms and software running simultaneously within a system, an algorithm should perform well within a feasible number of operations and parameters.

4.1 Superresolution Literature with CNNs

Before the "era" of convolutional neural networks (CNNs), superresolution (SR) problem was addressed by a plethora of different methods. The most related method to the scope of this paper was dictionary based methods [27]. It is discussed in literature

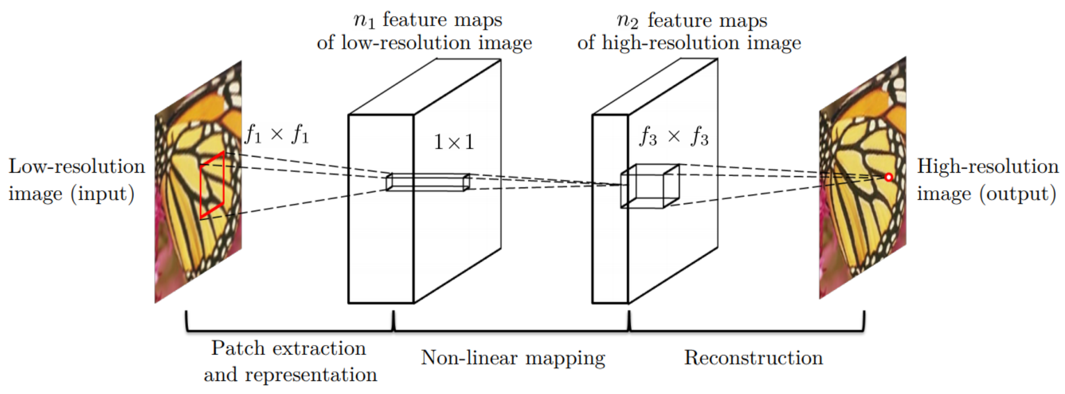

n_1 feature maps of low-resolution image
n_2 feature maps of high-resolution image
$f_1 \times f_1$
1×1
$f_3 \times f_3$
Low-resolution image (input)
High-resolution image (output)
Patch extraction and representation
Non-linear mapping
Reconstruction

Figure 4.1: SRCNN [3] block diagram

that the CNN based SR algorithms mimic the dictionary based algorithms [41] [47].

Super Resolution CNN (SRCNN) [3] was the first method to use CNNs for SR problem. The method visualized layers of a network as mapping functions of low resolution (LR) image to feature space and back from feature space to high resolution (HR) image as depicted in Figure 4.1 taken from respective paper. SRCNN had 57K parameters to be learned and it surpassed all previous Non-CNN methods up to date, in terms of PSNR. Later, an improvement on parameter number was proposed by Fast SRCNN (FSRCNN) [48] to reduce parameter count up to 12K and slightly increase error performance. On a later contribution based on FSRCNN, a discriminative learning based algorithm FSRGAN [49] improved over visual quality by using generative type networks. This was the lowest number of parameters proposed for a CNN based SR algorithm. Since then the number of parameters have increased at each generation of breakthroughs.

One breakthrough happened with Kim et. al. [4], when very deep networks with residual learning was proposed. It was discovered that the depth of the network had direct correlation with error performance. VDSR had 20 layers, with 64 filters at each layer with the size of 3x3. This amounted to 664K parameters to be trained. PSNR performance was dramatically increased with the cost of execution time. The proposed algorithms use bicubically upsampled LR images to obtain HR result. The training was completed within 4 hours on Nvidia Titan Z GPU. The algorithm was presented as in Figure 4.2.

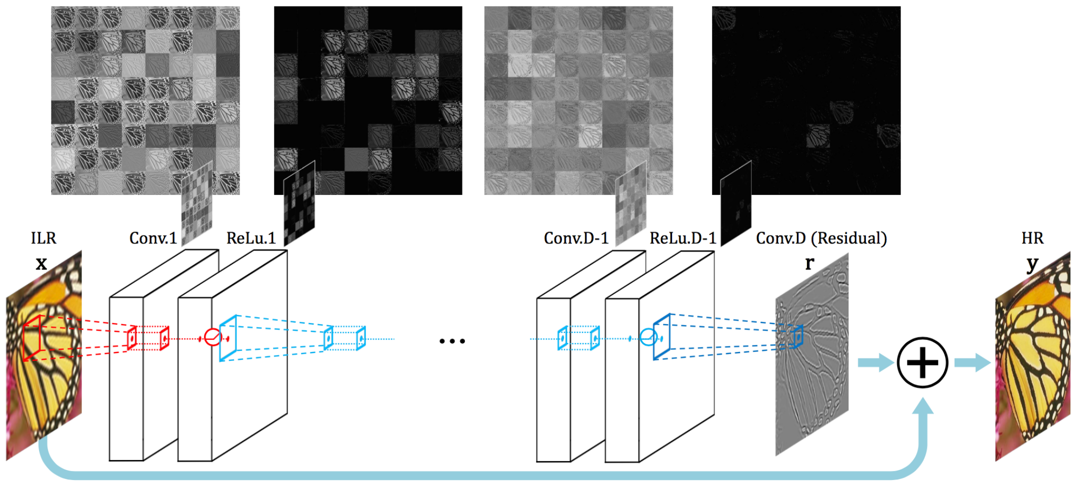


Figure 4.2: VDSR [4] block diagram

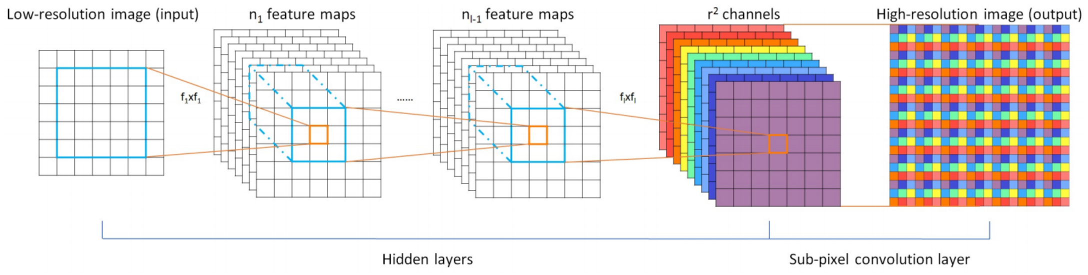


Figure 4.3: ESPCN [5] block diagram

Kim et al. [50] also proposed Deeply Recursive networks for SR problem (DRCN). DRCN had similar performance results compared to VDSR with 1.7M parameters to be learned. The training of DRCN took 6 days on Nvidia Titan X GPU.

ESPCN [5] was an exceptions in the timeline of ever-increasing number of parameters. The SRCNN topology is constructed with 3 layers, but with a novel sub-pixel interpolation block in the end. The LR image itself was propagated through the network until the last layer, where resulting n^2 images were used to generate HR image of upsampling ratio of n as depicted in Figure 4.3. Sub-pixel interpolation has not only increased the speed of the algorithm but also helped the network to learn its own interpolation filters. Training of ESPCN took 7 days on Nvidia K2 GPU.

In another method, namely LapSRN [51], a cascaded upsampling structure was proposed. For scaling ratios of 4 and 8, 2 and 3 cascades were used respectively. It is reported that their algorithm has 27 layers, which would amount to 922K parameters for 64 filters on each layer. LapSRN was trained in 3 days on Nvidia Titan X GPU. Shared coefficients are introduced to LapSRN algorithm for proposing MS-LapSRN [44]. Due to the introduction of shared parameters, the number of used parameters for following SotA algorithms have dropped. Even though shared algorithms are good for memory issues the number of operations required to obtain one output pixel, hence operations per output pixel (ops/op) is the true measure of an algorithm's size. Although independent parameter count of MS-LapSRN is reported to be 222K for D5R8 sub-method, its operations per input pixel is calculated to be 3M in total. The calculation of ops/op for an algorithm that recursively upscales an image is rather difficult. The ops/op for MS-LapSRN for upscaling ratio 2 is 3.3M/(2x2) that is 825K. The algorithm is used twice in cascade to obtain 4x upscaling and then bicubic downsampling is applied for 3x upscaling ratio in total. Therefore the ops/op for MS-LapSRN is calculated to be 825K/4 + 825K = 1M. Further division by 4 is due to pyramid structure of cascaded upscaling framework. The ops/op is a direct measure to compare running times of algorithms rather than the parameter count.

With ResNet [52], residual blocks consisting of two convolution layers and skip connections between its input and output was proposed. Later Ledig et. al. [43] have used this idea for a superresolution algorithm named SRResNet. The topology of residual blocks was changed by Lim et. al. in [42]. Enhanced Deep Residual Network (EDSR) was another breakthrough of error performance which used 32 residual blocks (or layers) with 256 filters of size 3x3 at each layer. This amounted to 43M parameters to be learned. Training of EDSR took 8 days on two Nvidia Titan X GPUs.

Same year, another method attempted to reduce the number of parameters in a network. Deep Recursive Residual Network (DRRN) [45] have utilized two layered residual blocks of ResNet [52], but instead of forwards progressing, DRRN carried initial information to be added at every two layers' output. Ops/op for B1U25 and B1U9 sub-methods is calculated. With 128 filters of size 3x3 at each layer, B1U9 method (with 9 residual blocks, amounting to 18 layers plus two layers for the beginning and ending) contains 2.6M ops/op and B1U25 method (25 residual blocks) contains 7.3M ops/op. DRRN was trained in 4 days with two Nvidia Titan X GPUs.

Tai et. al. have proposed a memory network (MemNet) for SR problem [46]. MemNet used recursive blocks called memory blocks to learn multi-level dependencies inside a single image. The base algorithm with 80 layers and 64 filters of size 3x3 per layer (M6R6) is trained in 5 days on Nvidia Tesla P40 GPU. It has 2.9M ops/op. Their advanced sub-method M10R10 has 212 layers and 7.7M ops/op.

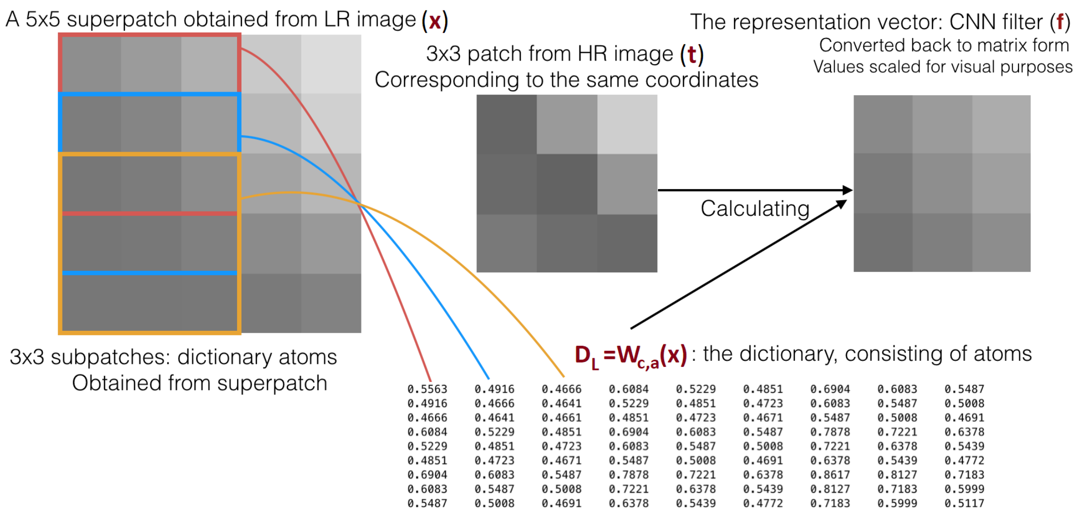
Another method that reduced number of parameters for a performance increase was IDN [1]. The authors have sliced and split layer outputs to be added on further layers, similar to skip connections. They have achieved good PSNR results for 663K parameters. Their model was trained in 1 day on Nvidia Titan X GPU. Total ops/op is calculated to be 73K for IDN since it processes LR data until the end of the network, where it is upsampled.

Two latest methods for SR problem are Deep Back-Projection Networks (DBPN) [53] and Residual Dense Network (RDN) [54]. DBPN used up-projection and down-projection blocks to preserve dependencies of HR estimate and LR input throughout the network while providing a solution for SR. There are 10M parameters to be trained that resulted slightly higher PSNR results compared to that of EDSR. Similarly RDN uses dense networks with 4.7M parameters (1.5M ops/op, since algorithm processes LR data and outputs 3 channels for RGB) to provide slight improvement over EDSR in terms of PSNR. RDN was trained for 1 day on Nvidia Titan Xp GPU.

Scaling of these algorithms to mobile applications, TV and industrial applications is more than necessary. It is evident that storing hundreds of thousands and even millions of parameters inside an embedded system and running these algorithms on real-time applications is impossible.

4.2 Analysis of Network Topology

To understand mechanism of CNN with simplified equations, in addition to discussion on Chapter 3, we can inspect cases where mathematical proving is easier. Let us now recall the psuedo-inverse solution, applied for CNN context $\boldsymbol{f_E} = (D_L^T D_L + \lambda I)^{-1} D_L^T \boldsymbol{t}$. Although psuedo-inverse solution is the estimator of a system with L_2 norm regularization, it is not going to effect our analysis after following assumptions.


A 5x5 superpatch obtained from LR image (x)
3x3 patch from HR image (t)
Corresponding to the same coordinates
The representation vector: CNN filter (f)
Converted back to matrix form
Values scaled for visual purposes
Calculating
3x3 subpatches: dictionary atoms
Obtained from superpatch
D_L =W_c,a(x) : the dictionary, consisting of atoms

```
0.5563  0.4916  0.4666  0.6084  0.5229  0.4851  0.6904  0.6083  0.5487
0.4916  0.4666  0.4641  0.5229  0.4851  0.4723  0.6083  0.5487  0.5008
0.4666  0.4641  0.4661  0.4851  0.4723  0.4671  0.5487  0.5008  0.4691
0.6084  0.5229  0.4851  0.6904  0.6083  0.5487  0.7878  0.7221  0.6378
0.5229  0.4851  0.4723  0.6083  0.5487  0.5008  0.7221  0.6378  0.5439
0.4851  0.4723  0.4671  0.5487  0.5008  0.4691  0.6378  0.5439  0.4772
0.6904  0.6083  0.5487  0.7878  0.7221  0.6378  0.8617  0.8127  0.7183
0.6083  0.5487  0.5008  0.7221  0.6378  0.5439  0.8127  0.7183  0.5999
0.5487  0.5008  0.4691  0.6378  0.5439  0.4772  0.7183  0.5999  0.5117
```

Figure 4.4: Computation of filter from given LR and HR image patches.

In order to be able to use insights from this equation assume that all neurons in the network are activated for the inputs. The network filters can be convolved among themselves to produce an end point filter, $\boldsymbol{f_E}$, for the unrealistic case. Creating an $\boldsymbol{f_E}$ is feasible because when all neurons are activated, their linear unit outputs are going to be the convolution results minus a bias that can be added up at the end, simply enabling the convolution of all filters to be applied in a single instant. Figure 4.4 further elaborates on this analysis. A similar work is done by Mallat et. al. [55] to analyze linearization, projection and separability properties of sparse representations for deep neural networks.

The vector $\boldsymbol{f_E}$ is going to be a normalized projection of $\boldsymbol{t}$ onto input image domain. Considering the rows of D_L^T matrix, each row is a vectorized subpatch, thus each multiplication result from $D_L^T \boldsymbol{t}$ is going to be $\langle subpatch, t \rangle$ meaning the projection of target patch onto an input subpatch. $D_L^T D_L$ matrix have elements of inner products of subpatches such as $\langle subpatch_i, subpatch_j \rangle$. The diagonals of $D_L^T D_L$ matrix, therefore, are normed square of each subpatch. The inverted matrix is going to be mostly composed of diagonals that are inverted normed square values of subpatches. This means that the entire equation calculates the projection of target patch, $\boldsymbol{t}$, onto the input image domain. In other words, the result, $\boldsymbol{f_E}$, consists of scores which measure how similar $\boldsymbol{t}$ vector is to each subpatch from the entire superpatch. If the target image has content that cannot be recovered by using certain region of input image,

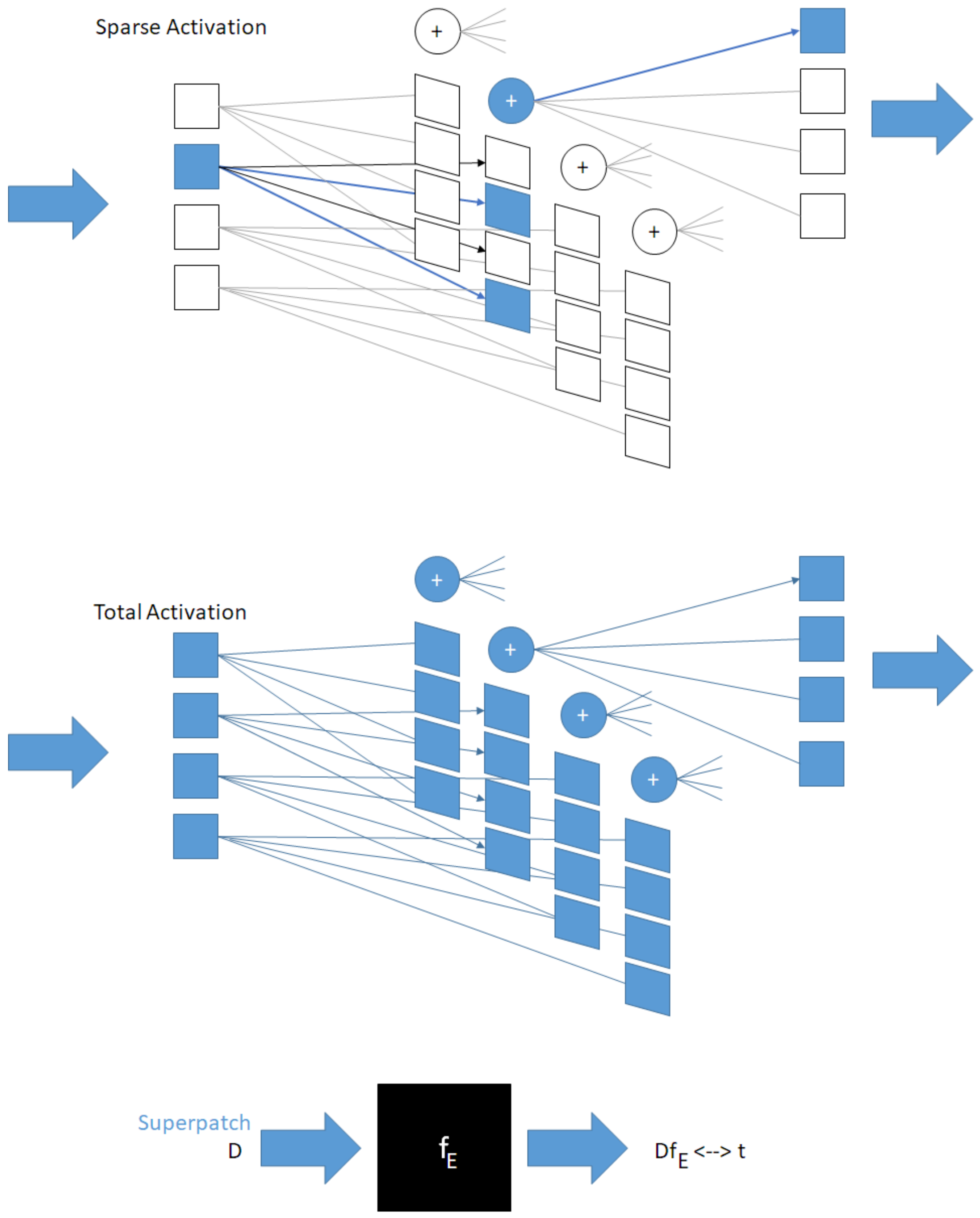


Figure 4.5: Illustating the Equivalent Filter f_E

the reconstructed image is going to be inferior. This is due to the fact that the inverse problem operates solely on an input image. Selection of a larger area for the reconstruction of certain target patches proves useful because of increased information included into the system.

The insight from discussion on extreme examples provides a method for determining how deep a network should be for certain features. For example when the super-patch and corresponding target region contains only irregular texture, which can be modeled as gaussian noise, the D_L matrix becomes linearly independent, meaning easily invertible. Consequently when the training set consists solely of textured images, shallow networks will perform as good as deep networks. Then for the testing phase, same filters are used to construct the D_L matrix and the result of the network is obtained by the same equation without normalization (without the inverse term) this time (since it is already normalized) i.e. projecting input image onto filters' domain. Notice that the error is generally not completely orthogonal to the input images because of iterative nature of equations. Therefore this is not going to be a meaningless operation. The representations that are learned during training can only be called complete if the data can be completely recovered using them [56].

In general the training set contains various features with different variances. Therefore the generalization of the new concepts that are introduced here are difficult. Training with different structures enables the constant evolution of neuron filters during training. However to have an activating branch for each feature either the network should have increased number of filters or the network will not converge which can be explained by the manifold hypothesis, as representations not covering the high dimensional input space [16].

We propose separating a CNN into two separate networks that will be trained with different data. The separation must be carried out according to texture information within training patches. Also the separation must be done in such a way that the disjunct training sets should present different informations to a network that is to be trained. Different metrics that can be used for the separation of data is inspected in section 4.3.1.1 in detail.

The verdict of this analysis is that the network depth should be different depending on

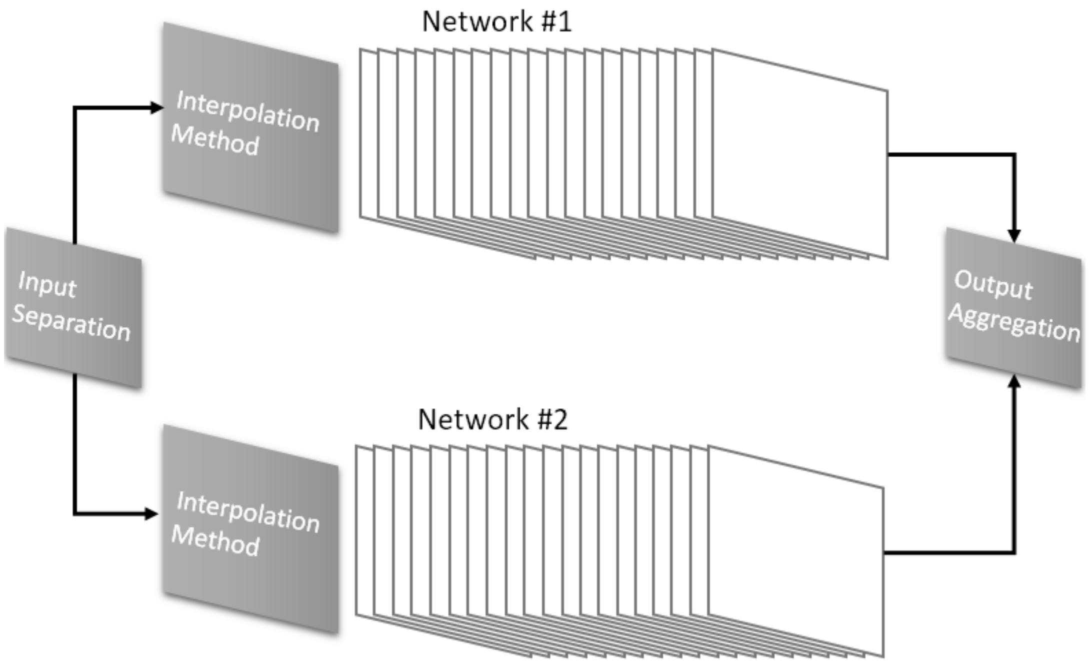


Figure 4.6: Framework for a double-network superresolution

the training data. Also the CNN becomes a high dimensional manifold upon which the input is projected [16]. Considering such complicated space, increased variety of features from the training set will increase training times and convergence speed greatly. Combining these ideas, we propose separation of the network into two or more networks which is carefully inspected in the next section.

4.3 Double Network Superresolution

To design a double network superresolution framework following questions must be answered:

- How to separate the training set?
- What interpolation method to use?
- How to merge the outputs of networks?
- How many layers and parameters to use?

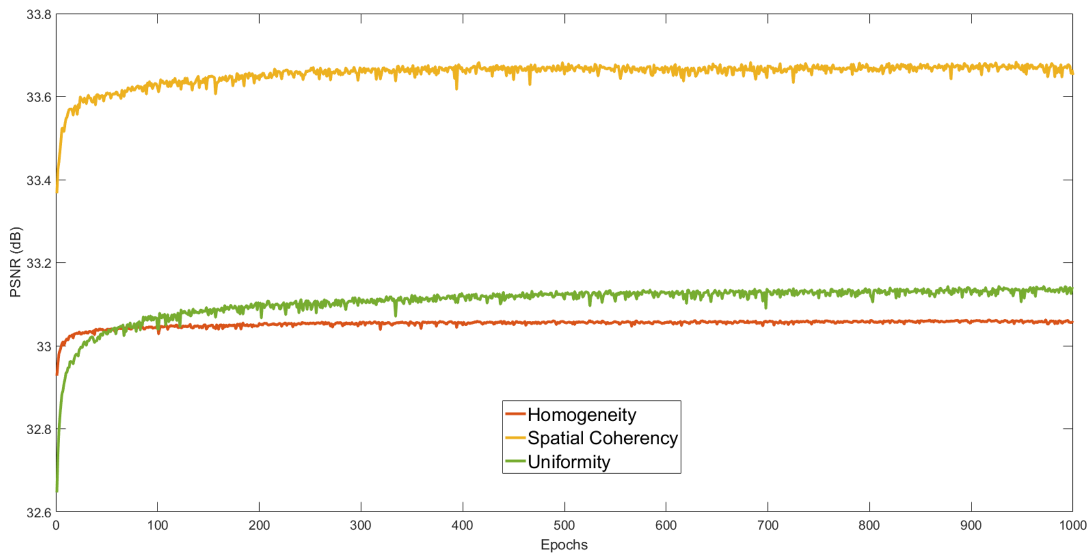


Figure 4.7: Comparison of the effect of different data set separation methods on the convergence of a network with a fixed data.

4.3.1 Experiments on Double Network Topology

To determine appropriate algorithm parameters, four different experiments are conducted all of which are done for 3x upsampling ratio and the results are also going to be given in the same ratio. 3x upsampling is chosen because it is a greater challenge to carry out a non-dyadic (not multiple of two) upsampling. The final test setup details are given in Chapter 5.

4.3.1.1 Experiments on Separation of the Networks

A single image superresolution algorithm can extract useful information from few cues such as color, edges and textures. Since this work is carried out on intensity level and not in color, utilization of texture and edge (gradient) information for separation of the training set is going to provide useful information for separate networks. For this reason three different methods are tested to separate training set according to its texturedness: Homogeneity and Uniformity measures obtained from a co-occurrence matrix [57] and a spatial coherency measure used by Romano et. al. [3] One of the most used measures for texturedness is co-occurrence matrices proposed by Haralick et. al. in 1973 [57]. Homogeneity measures the closeness of distribution of elements in a patch within small neighborhoods. Uniformity measures the energy of overall variation in a patch. The values for the entire set is calculated and the median value for two measures are chosen to be the value to separate training set. We have also used another method that was used by a similar purpose by Romano et. al. [31]. Eigenanalysis is performed on a patch of image that yields information regarding gradient angle, strength and spatial coherency. From a patch of size $\sqrt{M} \times \sqrt{M}$ gradient values in x and y direction are calculated. Vectorized gradient estimates are concatenated under a matrix G that has the size 2 by M. M is the number of pixels in a patch of image.

$$G = \begin{bmatrix} g_{x_1} & g_{y_1} \\ \vdots & \vdots \\ g_{x_n} & g_{y_n} \end{bmatrix}. \tag{41}$$

Eigenvalues of the matrix G^{T}G are calculated. SC is calculated by the normalized difference of larger eigenvalue $\sqrt{\lambda_1}$ and smaller eigenvalue $\sqrt{\lambda_2}$. Larger eigenvalue, λ_1, gives information about the strength of gradients in the patch. The difference of square rooted eigenvalues yields information about the spread of the local gradients. The SC measure can be calculated as in equation 42, as explained in [31].

$$\mu_k = \frac{\sqrt{\lambda_1} - \sqrt{\lambda_2}}{\sqrt{\lambda_1} + \sqrt{\lambda_2}} \tag{42}$$

Performance comparisons are done for each of the separation methods. Low textured images are prepared to train a 20 layer network with 16 filters at each channel that are of size 3x3. Figure 4.7 displays that best performance on training is achieved by separating data according to SC. Samples from real world images are shown on Figures 4.9 and 4.10 showing properties of low and high texture patches that are classified according to SC. The histogram of SC is calculated as in Figure 4.11, then the median value is chosen to separate training set into two.

Figure 4.8: Examples from mixed set of BSD100 [6], URBAN100 [7], DIV2K [8],[9] that high and low texture patch examples are shown in Figures 4.9, 4.10

4.3.1.2 Experiments on Interpolation Methods

There are two main approaches for the interpolation method in the literature. One is to interpolate the image with a known method such as bicubic or bilinear before the CNN [4][3]. The other method is to let the CNN learn its interpolation kernels [5]. Even though the latter seems to be the better option, there are advantages of using predetermined interpolation kernels (functions). For example the bicubic interpolation is known for its good localization and edge reconstruction. however bicubic interpolation fails to provide high frequency components such as texture details. Considering that there will separate image patches for training that are high and low textured, we propose the usage of bicubic pre-interpolation for low texture training set whereas for highly textured training set usage of learned interpolation is proposed.

To analyze the performance of the interpolation methods, we have used the separation method based on Spatial Coherency (SC) metric and compared the effect of bicubic interpolation on low and high texture patches. Figure 4.12 shows that bicubic interpolation creates unwanted high frequency components for textured images. Since high SC patch contains a single orientation edge and mostly flat regions, the bicubic interpolated image has less artifacts visible in the Fourier transform. We cannot compare a learned interpolation method with bicubic interpolation directly, because the

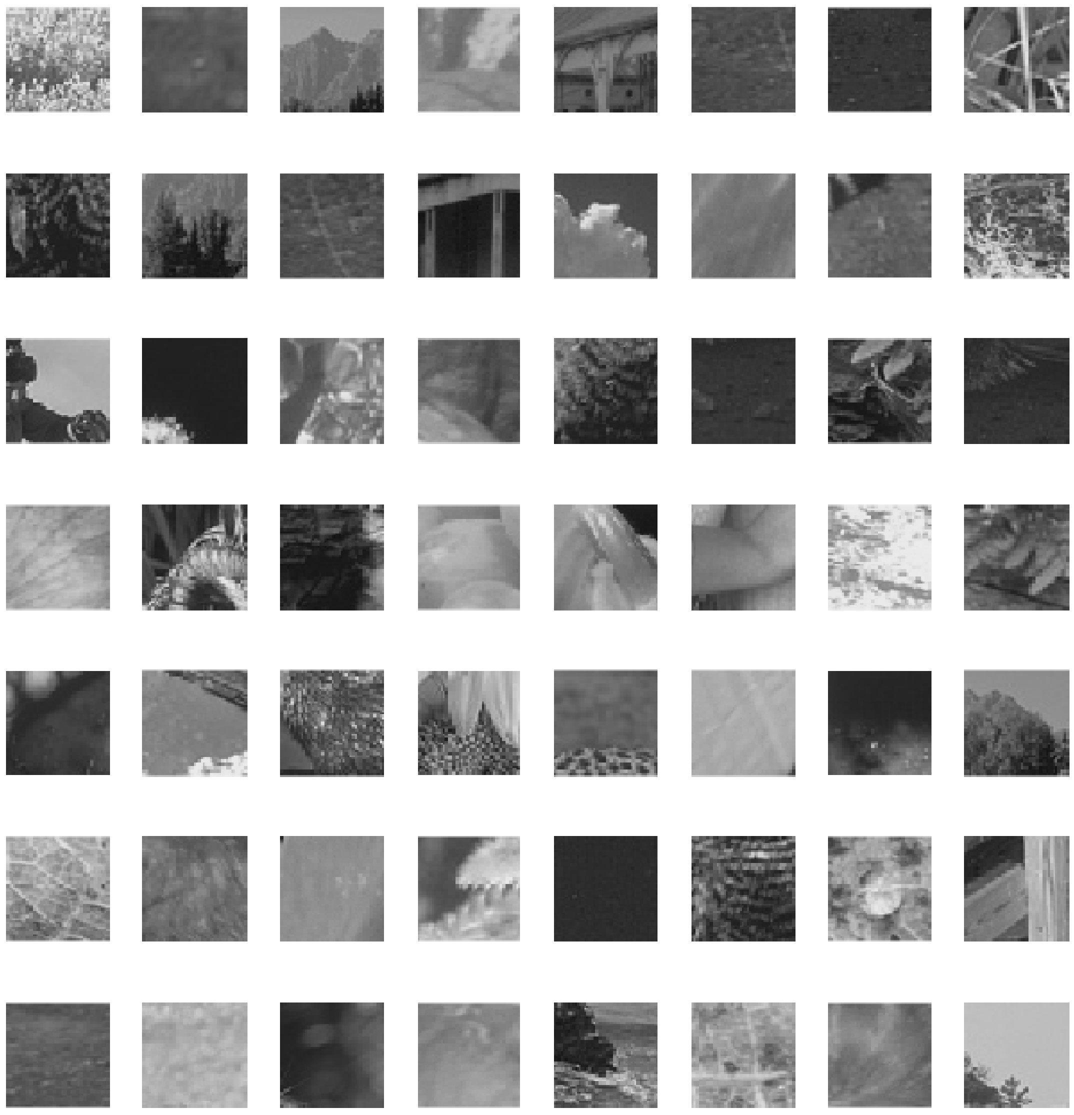


Figure 4.9: Examples of high texture image patches classified according to spatial coherency measure. High texture patches contain regular and irregular texture components.

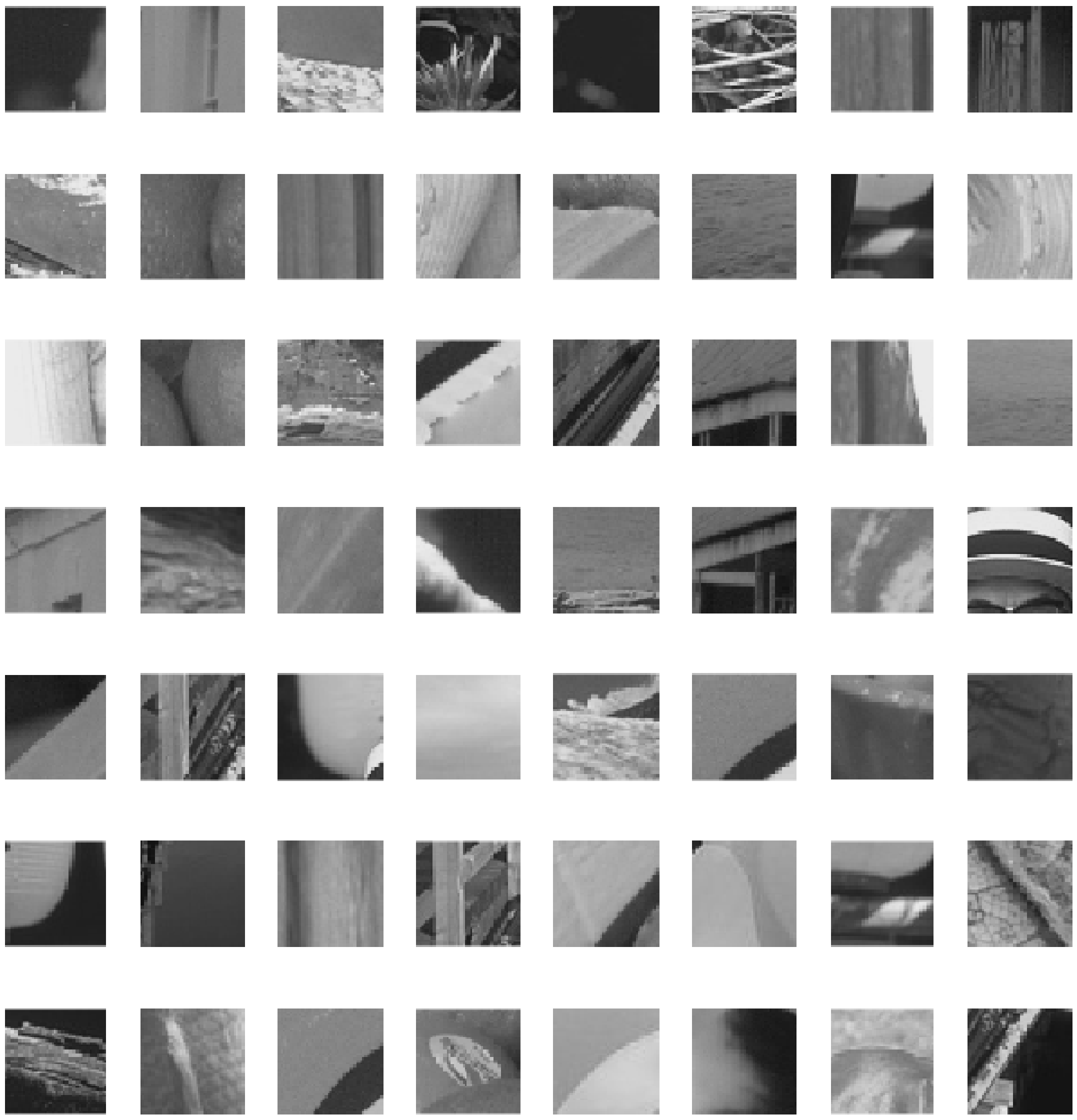


Figure 4.10: Examples of low texture image patches classified according to spatial coherency measure. Low texture patches are mostly comprised of flat regions with single orientation edges.

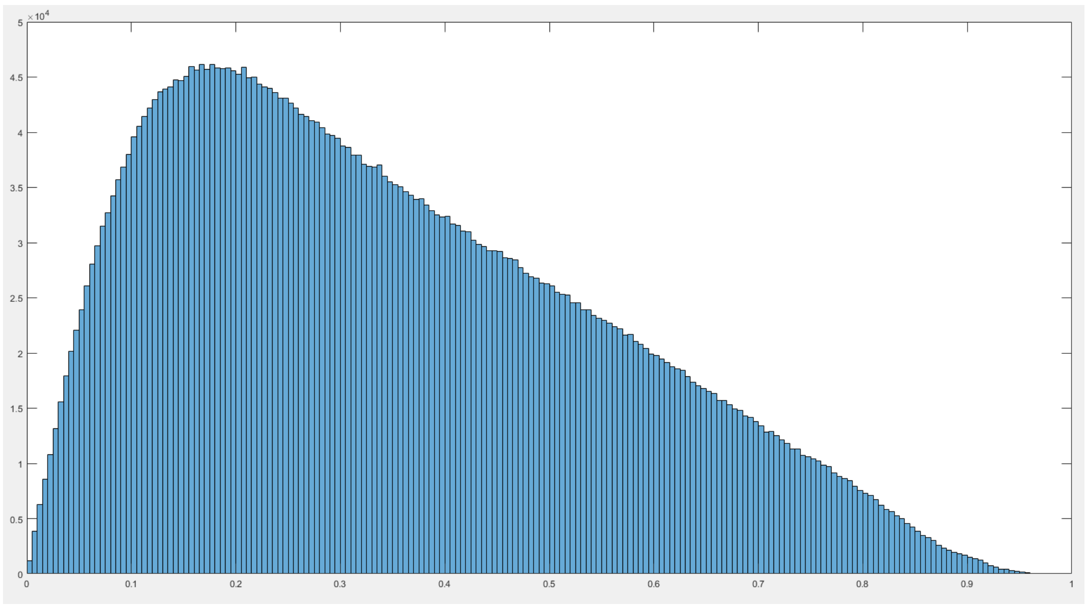


Figure 4.11: Spatial Coherency Histogram

Table 4.1: Comparison of the effect of different interpolation methods on reconstruction quality of different data sets

PSNR (dB) / Interpolation	Learned	Bicubic
Low Texture	32.49	33.05
High Texture	31.02	30.77

interpolation kernels are used at the end of a network. Instead of comparing bicubic versus learned interpolation, we have trained four networks. We have used low versus high SC data and bicubic versus learned interpolation methods. Low texture (High SC) image patches have higher PSNR score on a network that is trained with pre-interpolation. High texture (Low SC) image patches have higher PSNR score on a network that is trained with learned interpolation. Bicubic interpolation is a straightforward method compared to a learned interpolation. However, bicubic interpolation creates more artifacts for high textured image superresolution. Experiments have also validated that using learned interpolation for high textured images yields better error performance. This is the reason why learned interpolation was chosen to be used for high texture network and bicubic interpolation for low texture images.

The learned interpolation that is first used in ESPCN [5], is applied to the images

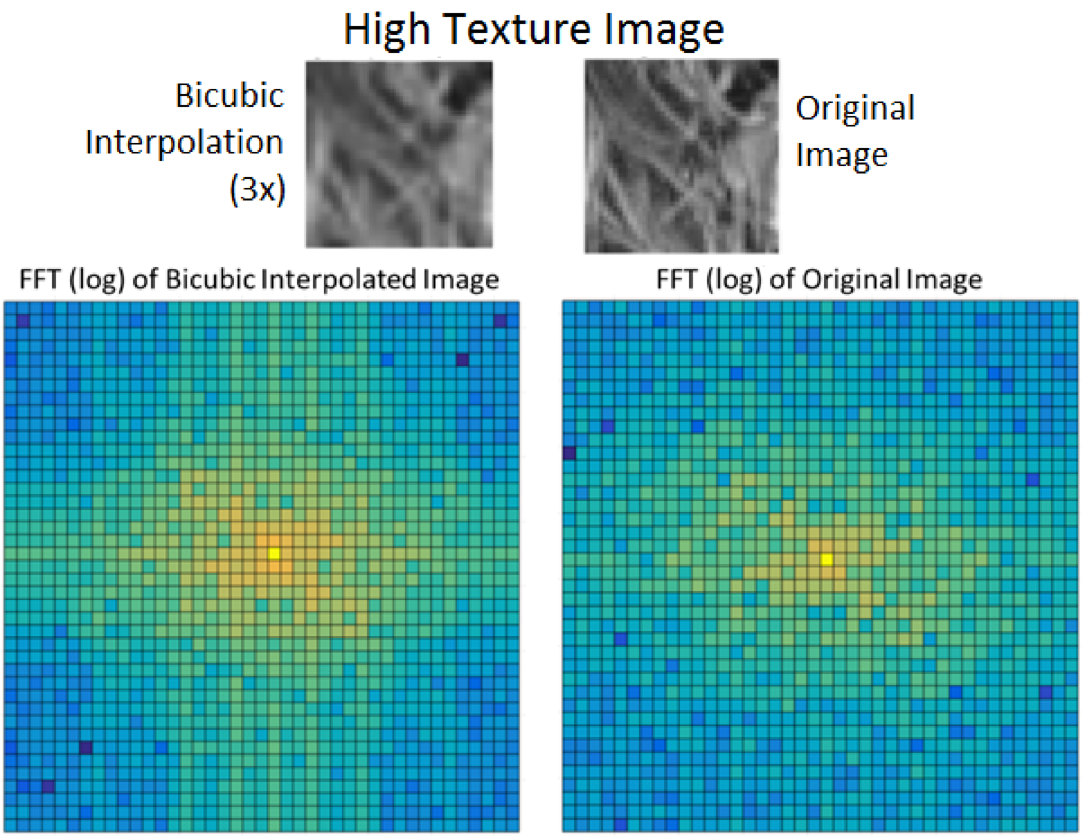


Figure 4.12: Comparison of FFT coefficients of bicubically interpolated low spatial coherency (high texture) data

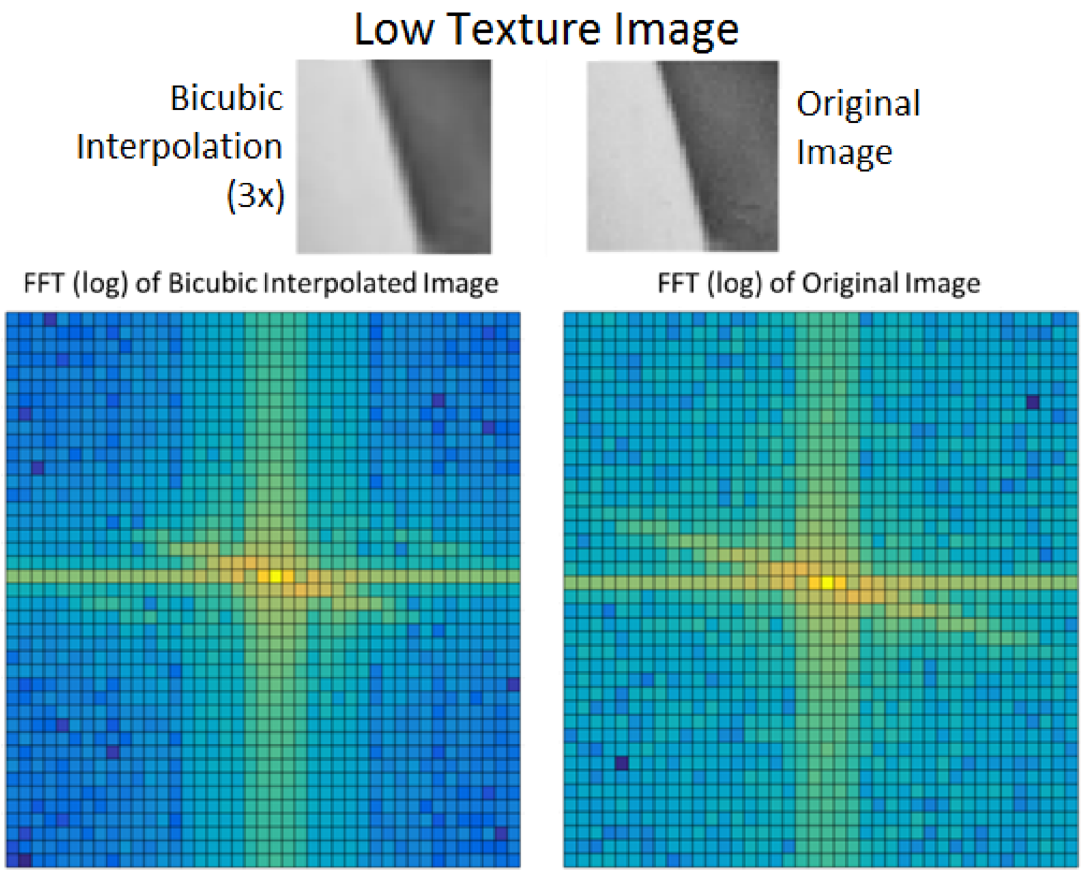


Figure 4.13: Comparison of FFT coefficients of bicubically interpolated high spatial coherency (low texture) data

at the end of a network. The network itself evolves coalesced with all sub-images that will turn into interpolated pixels in HR image. We have conducted an additional experiment to test out a new structure that was not proposed in the literature before. The fact that nxn pixels for n times interpolation, evolves inside the same network, lead us to separate the networks for each sub-pixel. The network structure is depicted in Figure 4.14. The resulting images from this network contained better texture components compared to low texture network on 3x scaling factor. Although apparent texture generation capability of the network is visualized in Figure 4.15, the convergence of proposed structure to a successful PSNR performance could not be achieved, within low parameter count limits that we have set. Therefore conventional learned interpolation method is chosen to be used for high texture training set. This method is left to be studied in future works.

4.3.1.3 Experiments on Structure of the Network

Following our work [47] we have chosen 10 layered network with skip connections for high texture data and 20 layered network with no skip connections (only residual learning) for low texture data. We know that a high spatial coherency (low texture) training set will yield learned filters of similar information content. This means that the neuron filters will be composed of mostly flat and singular orientation features thus filters already have higher mutual coherence. Also using skip connections for low texture network can introduce additional information that could decrease mutual coherence, thus further deteriorating the performance. For low spatial coherency (high texture) networks, the learned filters will exhibit textured features reducing the mutual coherence (reducing similarity) in the process. This is the reason why skip connections are used for high texture network. 3 and 4 parallel networks are also tested instead of two networks and the performance increase was found to be negligible compared to speed loss due to increased number of parameters.

Extensive experiments are conducted on the number of parameters required for the best performance/speed optimization. It is experimentally shown that low texture network requires less number of parameters where high texture network benefits more from increasing the number of parameters. This is due to the number of possible

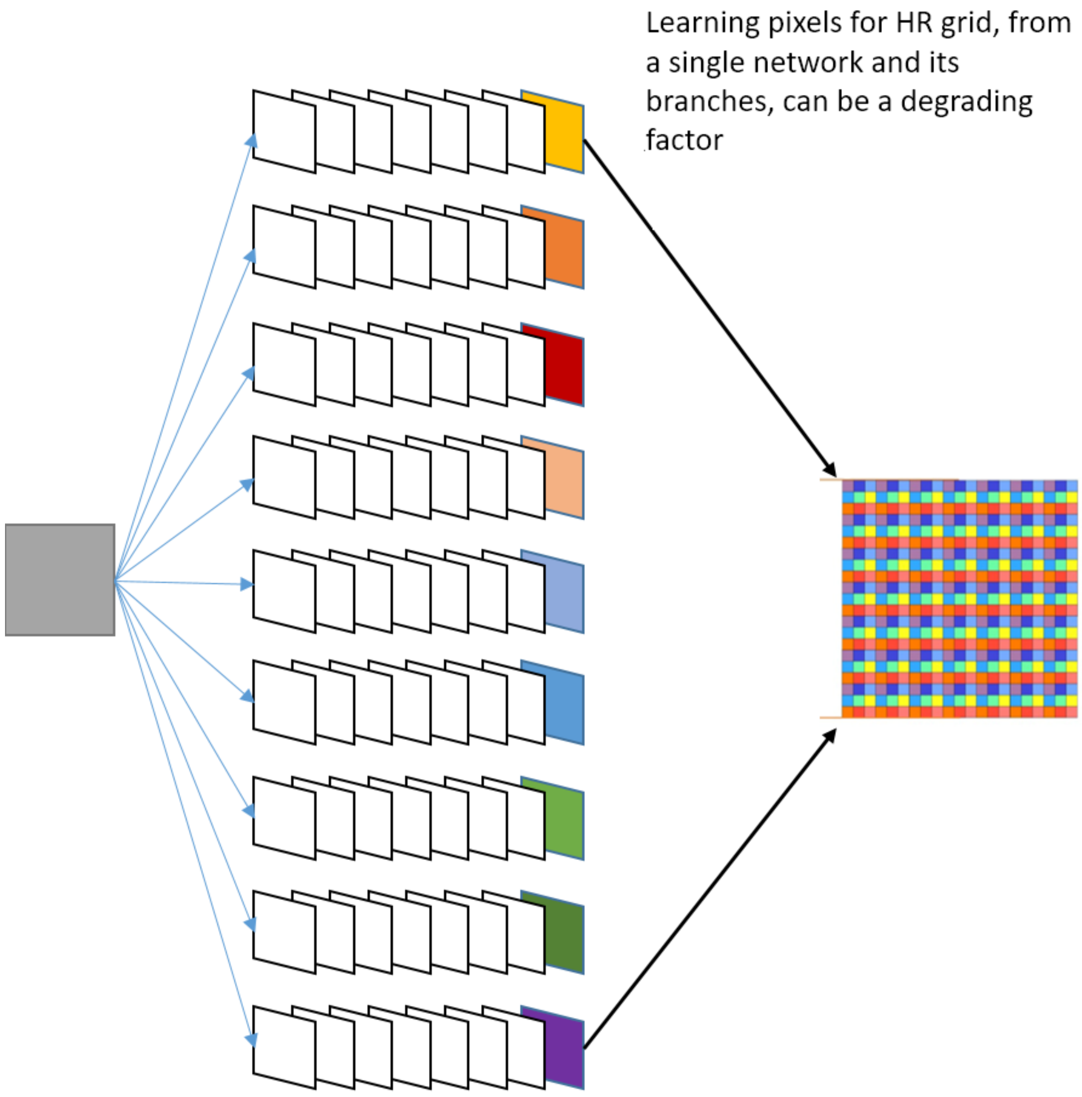

Learning pixels for HR grid, from a single network and its branches, can be a degrading factor

Figure 4.14: A new CNN structure for high texture set and learned interpolation namely separate channel interpolation (SCHI). Each pixel to be interpolated into HR grid is evolved through a different network, giving the network the ability to adapt recovered pixels in subpixel resolutions.

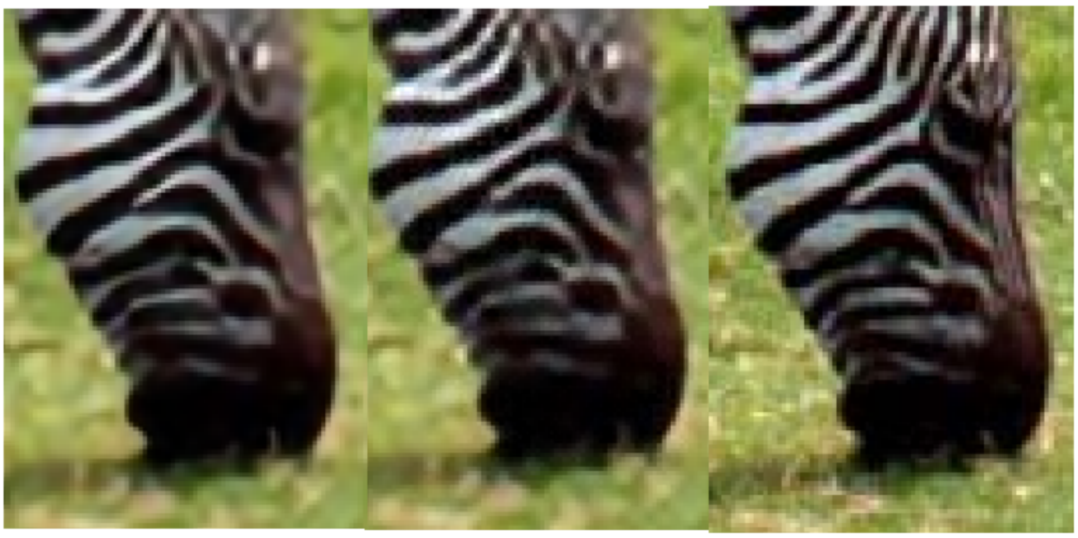


Figure 4.15: Comparison of low texture network result (left), separate channel interpolation (SCHI) method (middle), original image (right). SCHI method can recover more high frequency texture components compared to LT network. The fidelity of generated texture from SCHI is low, causing pixel wise error to increase.

variations on a training patch. Since low texture data set contains only single oriented edges and mostly flat patches, we have concluded that the dimensionality of the manifold that needs to span the training set was lower. The chosen number of parameters are 16 filters per layer for low texture network and 36 filters per layer for high texture network. This amounts to 93960 parameters for high texture network and 41760 features for low texture network. Considering that the high texture network processes LR image throughout the network, in terms of required operations per HR pixel, equivalent number of parameters of the final network becomes 52200.

Another set of experiments have been conducted on reduced number of parameters and layers for EDSR [42], SRResNet [43]. EDSR structure with 8 residual blocks and 64 filters per layer is tested. PSNR values for Set14 [11] images was 0.8 dB below that of high texture network of DNSR alone and the run time has increased by 6 times due to number of parameters increasing to 590K from 93K. We have also tested the same number of layers and parameters with SRResNet which contains extra batch normalization layers inside residual blocks and extra rectified linear unit (ReLU) layers. The PSNR performance did not improve significantly. DBPN-SS network (with 188K parameters) that is suggested in Haris et. al. [53] could not be replicated due to DBPN structure not being suitable for interpolation ratio that is non-dyadic (not multiple of two).

4.3.1.4 Aggregation of Network Outputs

After the training is finished, during testing phase either the input data needs to be separated for parallel networks or the outputs need to be aggregated. Although the former seems to be the better approach, it has few problems. Firstly the separation of input data pixel-wise or patch-wise requires too much processing power. Secondly CNN structure is not adapted to partially filled inputs and lastly it becomes as hard to aggregate results as it is to separate input. Therefore the topology to combine outputs of networks for whole image inputs is chosen.

In order to merge two images, we have trained a thin network with one hidden layer. Entire training set is passed through low and high texture network, results are used together with ground truth images to train a network with four layers. Two images from low and high texture networks are fused by early fusion. Resulting network is used at the end of our trained double networks. The PSNR score of combined algorithm was in between the scores of two networks, similar to averaging.

CNN structure, especially deep networks, lose fidelity to the original data. Methods such as residual learning [4] and skip connections [47] are used to overcome this problem, but it cannot be prevented nevertheless. To reduce the effect of deviating from the original information filtered backprojection is used. Only low texture network output is backprojected because it is deeper and lacks skip connections. Then two images are averaged. With this method the PSNR score of averaged image was higher than that of network outputs but also higher than the score of backprojected images alone. We have empirically found that 15 iterations were enough to increase PSNR performance by 0.07 dB for Set14[11] images. These operations are not included to parameter count that we have calculated earlier, because bicubic downsampling or upsampling uses 3x3 kernels and the error calculation has equivalent computation of a 1x1 filter. Since we iterate one channel 15 times the computation overhead is equivalent to 150 parameters, which is negligible. The final architecture of DNSR is depicted in Figure

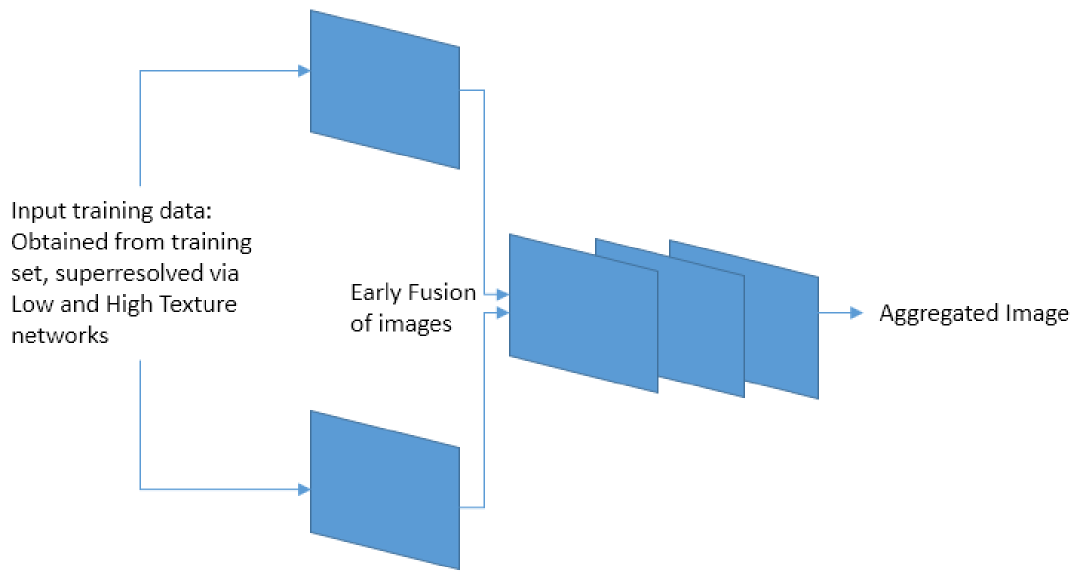

Input training data: Obtained from training set, superresolved via Low and High Texture networks
Early Fusion of images
Aggregated Image

Figure 4.16: Aggregation Network

4.17.

4.3.2 Ablation Study

Our proposed network is tested with different configurations. For fair comparison we have obtained four different networks with the same number of operations. Firstly Two separate networks are trained with entire training data, instead of respective high/low textured patches. This network was named as DNSR_AA. Secondly Two networks are trained with conflicting data, meaning high texture network is trained with low texture data and low texture network is trained with high texture data. This network was named as DNSR_CONF. Thirdly a single network that comprises the general structure of low texture network is trained, i.e. bicubic upsampled inputs, 20 layers with residual learning. Training was done with entire training set. The number of parameters were chosen to be near 52K ops/op therefore, each layer has 18 filters (18x18 for inner layers) instead of 16 filters. Output is not backprojected to input LR image because this is a feature of double network framework. This network was named as LTNSR. Fourth and last ablation test was conducted by training a network with general structure of high texture network, i.e. 10 layers with skip connections and learned interpolation at the end of the network. Training is done with the entire

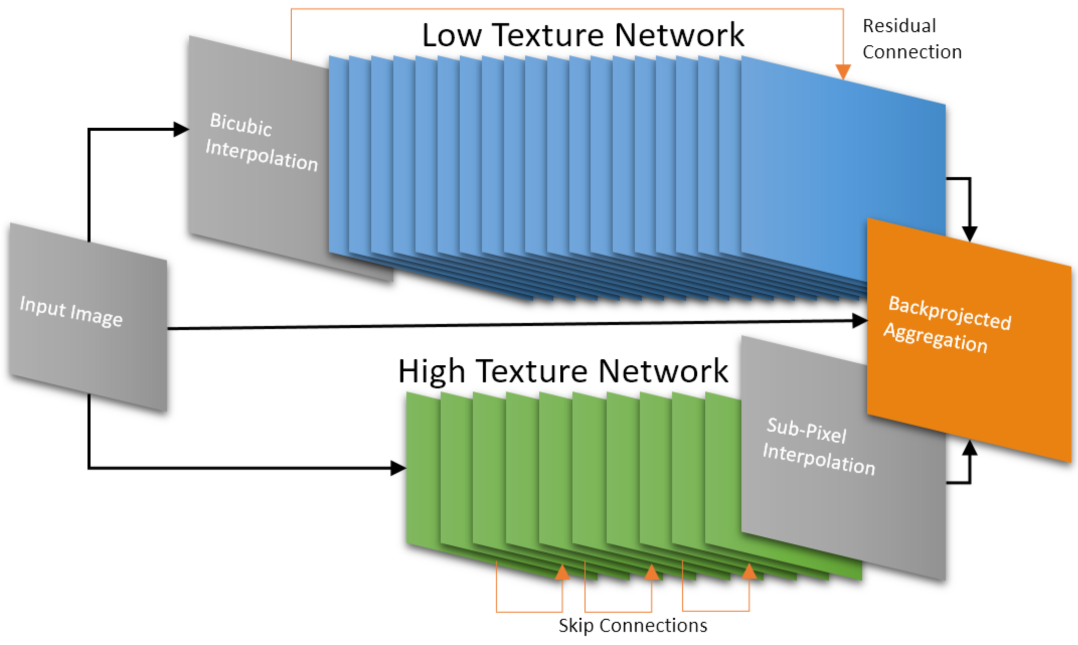


Figure 4.17: Architecture of proposed DNSR. Low texture network has 20 layers, 16 filters at each layer with size 3x3. High texture network has 10 layers, 36 filters at each layer with size 3x3. Skip connections are used between layers 1-3, 4-6, 7-9.

training set again. The number of filters per layer is chosen to be 80 instead of 36. This amounted to 473K parameters and one ninth of 473K ops/op that is 52K ops/op, because the scaling factor is chosen to be 3. This network is named as HTNSR. PSNR and SSIM scores are obtained for four networks together with original DNSR. BSD100 [6], Urban100 [7] datasets are used for comparison since they constitute a large variance of samples. In conclusion, original DNSR topology performed better than ablative networks as shown in Table 4.2.

Table 4.2: Ablation Study Results

	BSD100		URBAN100	
	PSNR	SSIM	PSNR	SSIM
DNSR_AA	28.757	0.8482	26.8546	0.8216
DNSR_CONF	28.7126	0.8493	26.7799	0.8206
LTNSR	28.732	0.8479	26.8391	0.8219
HTNSR	28.6931	0.8427	26.7636	0.8197
DNSR	**28.831**	**0.8527**	**27.1614**	**0.8293**

CHAPTER 5

EXPERIMENTAL VALIDATIONS

5.1 Introduction

Performance of DNSR is tested with extensive experimentation. PSNR and SSIM scores of our algorithm has been compared to SotA methods. Also we have tested our theoretical assertions from Chapter 3 and provided experimental validation.

5.1.1 Dataset

Throughout all the experiments we have used 91 images from Yang et. al. [28] and 200 images from Berkeley Segmentation Dataset (BSD) [6] as our training images. Similar to [4] we have additionally scaled our dataset between 0.8 and 0.6 and produced 90, 180, 270 degrees rotated versions of images. We have used only upsampling ratio 3 because we work on single scale upsampling.

5.2 Validation of Theoretical Assertions

We have validated our assertions with experiments to solidify our results. The experimental findings are summarized as

- Skip connections increase mutual coherence of middle layers, as well as other layers.
- Skip connections increase PSNR performance of the high texture network

Table 5.1: Comparison of mutual coherence of two networks. The training set data values was limited between 0 and 1.

Layer #	1	2	3	4	5	6	7	8	9
Skip MC	0,0034	0,00026	0,0031	0,0034	0,000099	0,00180	0,0026	0,0098	0,0038
NoSkip MC	0,0016	0,00018	0,0026	0,0031	0,000089	0,00099	0,0013	0,0027	0,0030

- Measured increase in values are subjected to T test and they are found statistically significant.

- Networks that are trained with lower spatial coherency data saturate in performance for shallower networks, while high spatial coherency data requires deeper network before converging in performance.

Skip connections are used on a 10 layered network with connections between 1st and 3rd layers, 4th and 6th layers, 7th and 9th layers. This network is trained with high texture data obtained by using SC metrix separation. Another network with same depth and parameters is trained without skip connections. Training is ended after both networks converged in their output error performance. Table 5.1 show that usage of Skip connection in a 10 layered network helped increasing mutual coherence. Especially deeper layers become less and less coherent as the information from the input becomes dispersed and altered due to non-linear activation functions. Skip connections help increase the lost mutual coherence.

Error performance of network with skip connections and without skip connections are compared. 100,000 patches have been used to test the error performance. The PSNR of the network without skip connections was averaged on 30.77 dB while skip connections helped the PSNR to increase to 30.88 dB. Bicubic PSNR average is calculated to be 29.2 dB. We have used the so called T test to measure the significance of this result. T test is formulated as in equation

$$T = \frac{mean_{set1} - mean_{set2}}{\sqrt{\frac{var_{set1}}{\#_{set1}} + \frac{var_{set2}}{\#_{set2}}}} \tag{51}$$

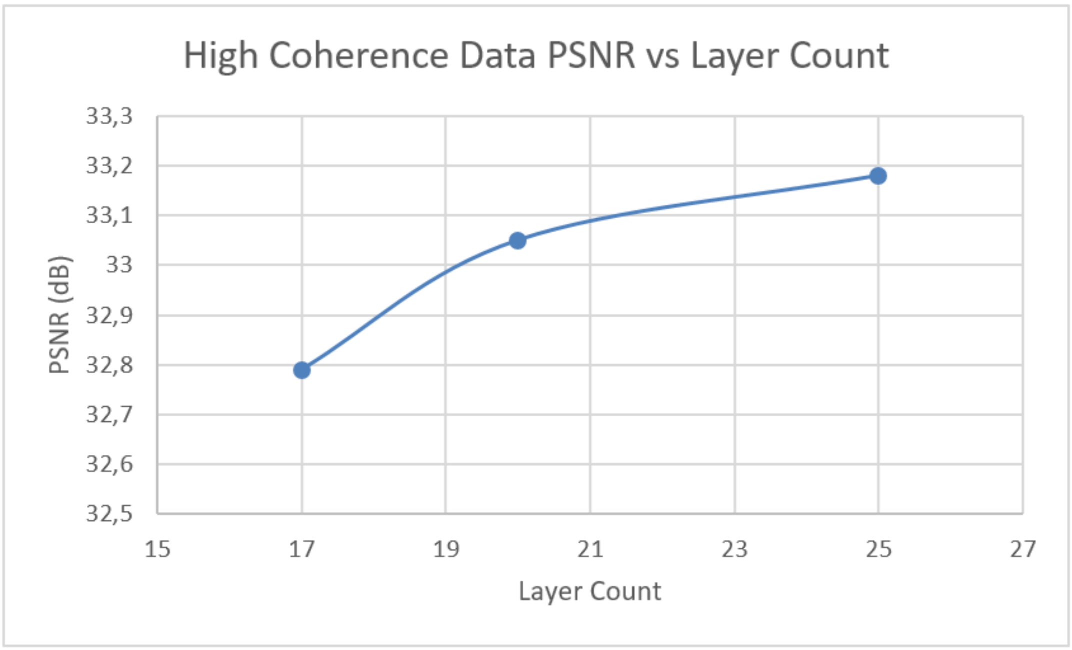


Figure 5.1: High spatial coherency network number of layers required for performance convergence. PSNR values are measured from single network, not the combined DNSR. A potential increase in performance of single network does not necessarily reflect to DNSR performance.

where set1 contains the results of network with skip connection and set2 contains results from network without skip connection. T value is calculated to be 7.07 which points that the improvement is quite significant.

We have trained multiple networks for high spatial coherency and low spatial coherency data sets. As we have pointed out with our discussion in Chapter 3, High spatial coherency network required deeper networks before converging in performance, while low spatial coherency networks converged on shallower networks. This is illustrated in Figures 5.1 and 5.2.

5.3 Performance of Proposed DNSR

The performance of DNSR is compared to SotA methods.

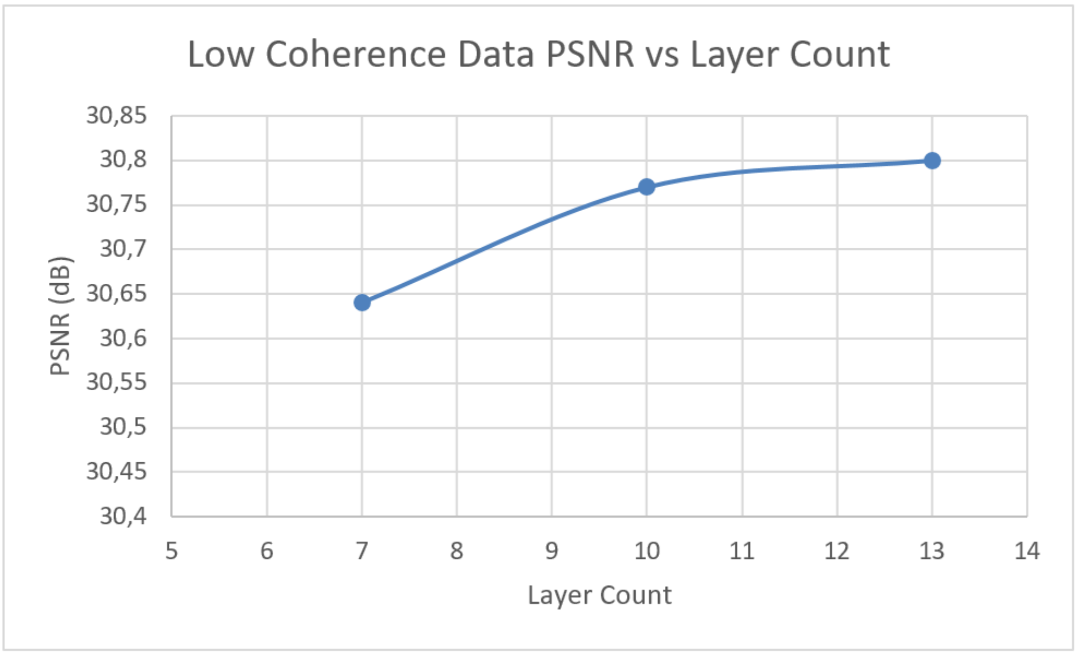


Figure 5.2: Low spatial coherency network number of layers required for performance convergence. PSNR values are measured from single network, not the combined DNSR. A potential increase in performance of single network does not necessarily reflect to DNSR performance.

Table 5.2: Operations per output pixel, training times, relative operating speeds of algorithms. Training times are calculated according to GFLOP/s capability of GPUs, projected to Nvidia Titan X level. Algorithms with ops/op less than 2 Million are compared.

Method	Ops/op	Params	Train time (hrs)	Relative Speed
SRCNN [3]	57K	57K	-	1.1
ESPCN [5]	1.6K	15K	59	0.03
VDSR [4]	664K	664K	2.8	12.77
DRCN [50]	1700K	1700K	144	32.69
LapSRN [51]	922K	922K	72	17.73
MS-LapSRN [44]	1M	6.6M	-	19.8
IDN [1]	73K	656K	24	1.4
RDN [54]	1.5M	4.7M	44	28.85
DNSR	52K	135K	8	1

5.3.1 Implementation Details

Networks are trained with stochastic gradient descent (SGD) based ADAM optimizer [58] with settings β_1=0.9, β_2=0.999, $\varepsilon = 10^{-8}$. We have initialized the parameters with Xavier initialization. A pre-training of 3 epochs has been applied.10 percent of both training sets are taken as such lowest textured patches from high texture set and highest texture from low texture set are collected. The pre-training was done with learning rate 0.001. This was done to reduce the effects of random ordering of data sets on initial backpropagations. Because we are separating the data set into low texture and high texture sets there is a small possibility of training the network with very high texture patches or almost flat patches after initialization. We wanted to prevent the network converging to a "prison" local minima at lower learning rates. After pre-training we have set the learning rate at 10^{-4}. For high texture network we have used 0.001 learning rate at every tenth epoch. We have found, empirically, that increasing learning rate in this fashion helped against convergence to a local minima. 41x41 sized image patches are used for low texture network. 21x21 and 63x63 sized pairs are used for high texture network.

Final architecture of Double Network Superresolution (DNSR) is depicted in Figure 4.17. The low texture network contains 20 layers with 16 features of size 3x3 at each layer. There is a residual learning connection from the input to the output of the network, similar to VDSR [4]. The high texture network contains 10 layers with 36 features of size 3x3 at each layer. There are 3 skip connections between every two layers (1-3,4-6,7-9) using our analysis from [47]. LR image is pre-interpolated with bicubic upsampling before low texture network. LR sized output of high texture network is interpolated using learned kernels at the sub-pixel interpolation block as proposed in ESPCN [5]. Two results from two networks are merged inside the new backprojected aggregation module. Our network has 135K parameters and 52K ops/op in total. The comparison of parameters to other methods are given in Table 5.2.

We have used Nvidia GTX760 GPU to display the ease of implementation for the network we propose. The training of two networks took 1 day. With comparison of Nvidia Titan X GFLOP/s capability this would amount to a 8 hours of training in total. Comparison of training times of other algorithms to that of DNS is shown in Table 5.2. We have used Caffe and MATLAB to train Low Texture network. For High Texture network we required the implementation of sub-pixel interpolation block [5] and Caffe did not have that library therefore we have used TenorFlow and TensorLayer to train High Texture network. Low Texture network is not carried onto TensorFlow framework as well due to the fact that tests on Caffe yielded slightly better PSNR results on more than one experiments.

5.3.2 Runtime Evaluation

We calculate the amount of floating point operations (FLOPS) required to use our algorithm. The equivalent parameter count is calculated as 52K at the HR resolution scale. A 1920x1080 resolution output requires 100 Giga FLOPS. Considering state of the art mobile platforms such as Nvidia Tegra X2 (2 TFLOPS capable) and Nvidia AGX Xavier (5 TFLOPS capable), our algorithm can reach 20 FPS and 50 FPS in these platforms respectively. The runtime comparison of DNSR with other algorithms

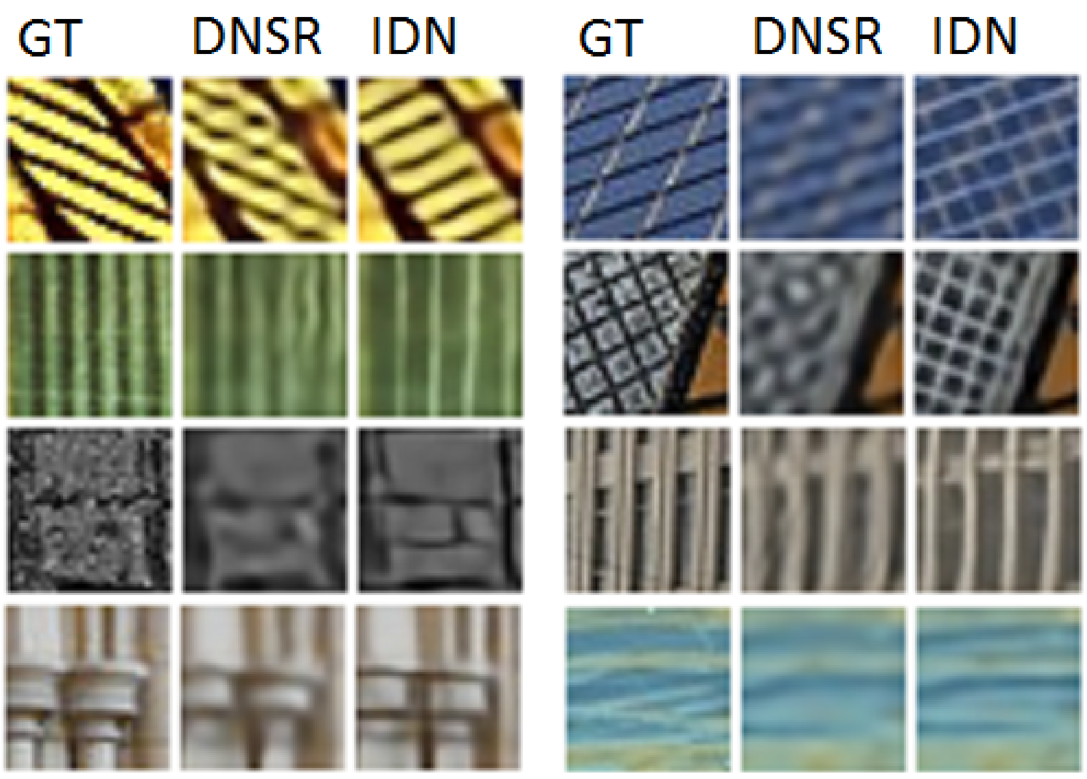


Figure 5.3: Comparison of DNSR and IDN[1] for fidelity in detail reconstruction, respectively ground truth, DNSR result and IDN[1] result.

are displayed on Table 5.2. DNSR is 1.4 times faster than the closest SotA method IDN [1]. Only exception is ESPCN [5] which is a very fast but not so well performing method in terms of PSNR and SSIM as displayed on Table 5.3. Similarly SRCNN [3] is an old network with low PSNR/SSIM score.

5.3.3 Evaluation of Test Sets

We have tested our results using Set5 [10], Set14 [11], BSD100 [6], Urban100 [7]. Table 5.3 displays numerical results. Details of some results can be seen in the Figure 5.5. Our network competes with SotA methods that have reasonably close running times (on the order of 10 times more and lower). The only method that is closer in terms of running speed to DNSR is IDN [1]. IDN performs better on overall PSNR and SSIM scores. Although PSNR is a widely-used metric for comparing the general quality of a proposed algorithm, it tends to favor pixelwise consistency over visual quality and fidelity of the reconstruction. Our DNSR is able to reconstruct high

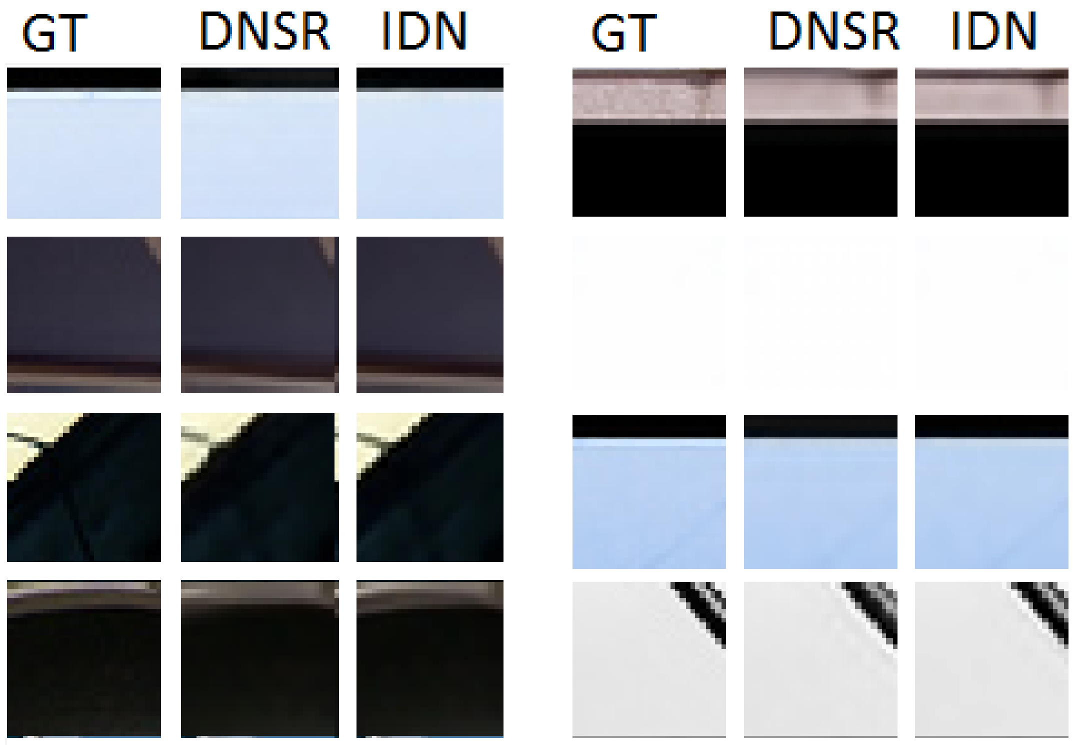


Figure 5.4: Comparison, in conjunction with Figure 5.3, of patches with highest score difference where IDN [1] is better compared to DNSR. This clearly shows that DNSR is better around patches where there is meaningful information such as textures and edges. IDN's success mainly stem from flat patches with minimal variation.

Table 5.3: Numerical Results and Comparison of DNSR Performance with algorithms that have ops/op less than 1 Million for scaling factor of 3.

Dataset	Result	Bicubic	SRCNN [3]	ESPCN [5]	VDSR [4]	LapSRN [51]	IDN [1]	DNSR
Set5	PSNR	30.39	32.75	33.00	33.66	**33.81**	**34.11**	33.68
Set14	PSNR	27.55	29.28	29.42	29.77	29.79	**29.99**	**29.83**
BSD100	PSNR	27.21	28.41	-	28.82	28.82	**28.95**	**28.83**
Urban100	PSNR	24.46	26.24	-	27.14	27.07	**27.42**	**27.16**
Set5	SSIM	0.8682	0.9090	-	0.9213	0.9244	**0.9253**	**0.9446**
Set14	SSIM	0.7742	0.8209	-	0.8314	**0.8325**	**0.8354**	0.8319
BSD100	SSIM	0.7835	0.7863	-	0.7976	0.7980	**0.8013**	**0.8527**
Urban100	SSIM	0.7973	0.7989	-	0.8279	0.8275	**0.8359**	**0.8293**

Table 5.4: DNSR error performance on high and low texture patches compared to IDN[1]. (DNSR-IDN) value is given, negative values indicate that IDN performs better

Image Set	Percent of Patches	PSNR Difference	Std. Dev. of Difference
BSD100 [6]	40.8	0.5249dB	0.7256
URBAN100 [7]	37.7	0.7582dB	0.8758
BSD100 [6]	58.2	-0.1695dB	0.3787
URBAN100 [7]	62.3	-0.2694dB	0.5586

frequency details and textures better compared to other algorithms within our scope of "fast" algorithms. Displayed images on Figure 5.5 show that DNSR is able to handle aliased data better. This is due to the novel double network structure of DNSR. High texture network handles textured recovery better and low texture network handles edge recovery better. When aggregated with backprojected averaging, DNSR excels at recovery of high frequency components.

We conducted an experiment to distill patches with high frequency components and to measure their error performance. Low-High and High-Low frequency components (in two dimensional FFT), that are multiple edges in the same direction are scored as the highest on SC measure. High-High frequency components, that are multiple edges on multiple orientations and irregular texture are scored as the lowest on SC measure. We have obtained patches from BSD100 and URBAN100 test sets with highest and lowest SC. We have put a lower limit over the strength of gradients (using λ_1 value from SC calculation). This is done to prevent nearly flat patches entering into comparison. We have distilled 30x30 non overlapping patches, 28370 out of 75257 for URBAN100 set and 5151 out of 12601 total number of patches from BSDS100 set have fallen within our category. From these patches we have calculated that DNSR performs on average 0.76dB better in PSNR compared to IDN [1] as in table 5.4. Standard deviations of selected patches and remaining patches are given. Given values indicate that, although somewhat higher, selected patches are not outliers in any means. It shows that the important portions of images which contain texture and edges are reconstructed better using DNSR as visually demonstrated in

Figures 5.3 and 5.4 and Table 5.4. Samples from selected patches are given in Figure 5.3. IDN is powerful in creating visually pleasing sharp edges. However most of the images are deviated from the original image in terms of structural shapes, location and width of edges, textural components. Data fidelity is an important issue not only for commercial image reconstruction but also for medical usage such as MRI and CT scan imaging. From that perspective DNSR is able to reconstruct details better qualitatively and quantitatively compared to other SotA methods. Bigger images of detailed examples on Figure 5.5 are given in Figure 5.6. Further examples from Set5 [10] and Set14 [11] are given in Figures 5.7, 5.8, 5.9 to illustrate the detail recovery improvement.

5.4 Further Validations and Experiments

Although theoretical and practical assertions from previous chapters have been covered in aforementioned experiments, there are implicit validations that need to be done. For example the proposed representation-dictionary duality needs experimental validation. Also the stochastic gradient descent (SGD) based learning algorithms such as ADAM [58] are used in our algorithm. The proven optimality of neuron filters are based on SGD. An implicit validation requirement appears from learning algorithms. For this reason we have carried out additional experiments regarding other branches of SR literature such as Generative Adversarial Networks [59], perceptual loss [60] and content loss functions [43]. Also multiple scale adaptation of our DNSR is tested. The reasoning for 3x scaling ratio was discussed as estimation of 9 pixels being a bigger challenge than dyadic upsampling ratios which can be cascaded for more options.

5.4.1 Noise Immunity

One of the weakest points of the CNNs is their dependence on input data models. In all of the methods in literature, except possibly one [61], the training is carried out with known downsampling models such as bicubic or bilinear. The main reason for this is that the learned coefficients of the network adapt to enhance bicubically

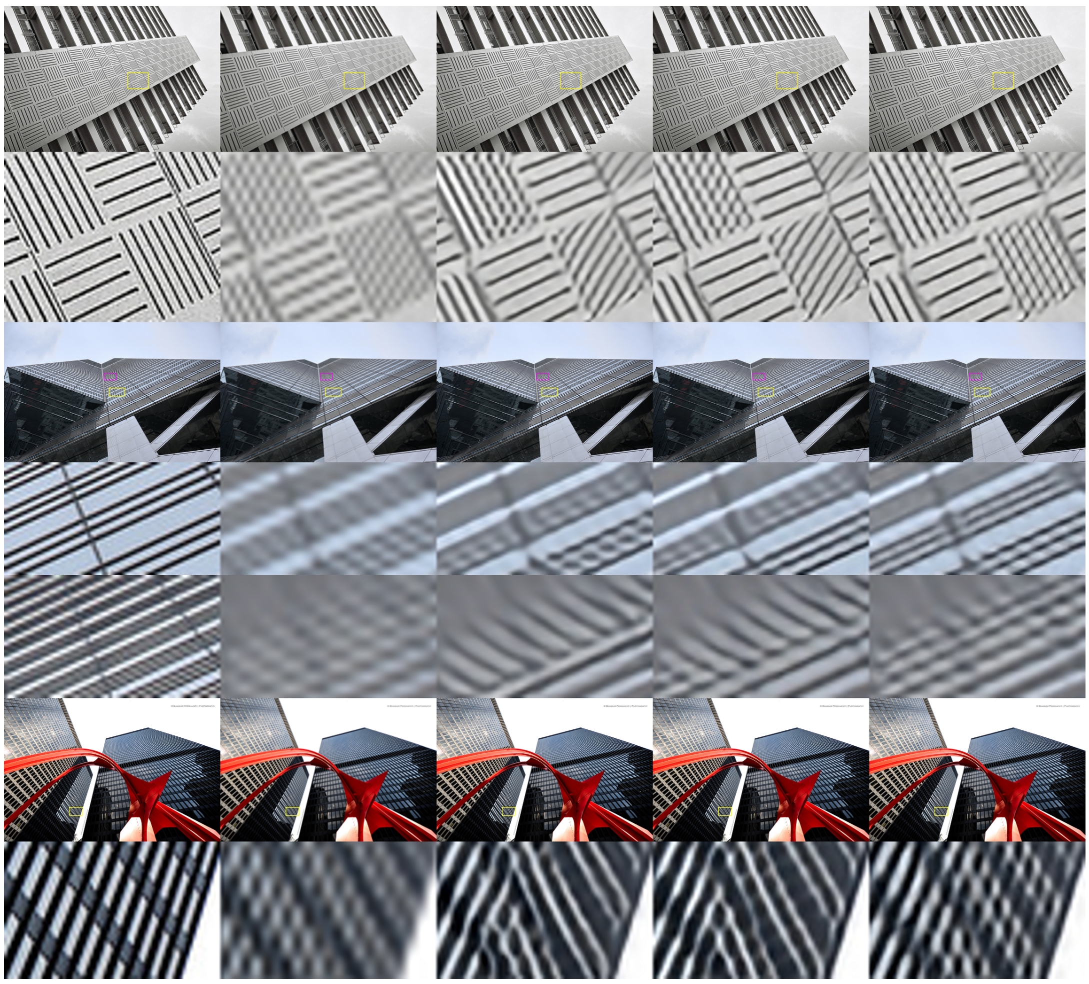


Figure 5.5: Visual comparison of our results. It can be observed that DNSR is superior in reconstructing high frequency details of an image better compared to other algorithms. From left to right: Original, Bicubic, VDSR [4], IDN [1], DNSR(ours). From top to bottom: img_092.png, img_059.png, img_062.png from URBAN100 [7]

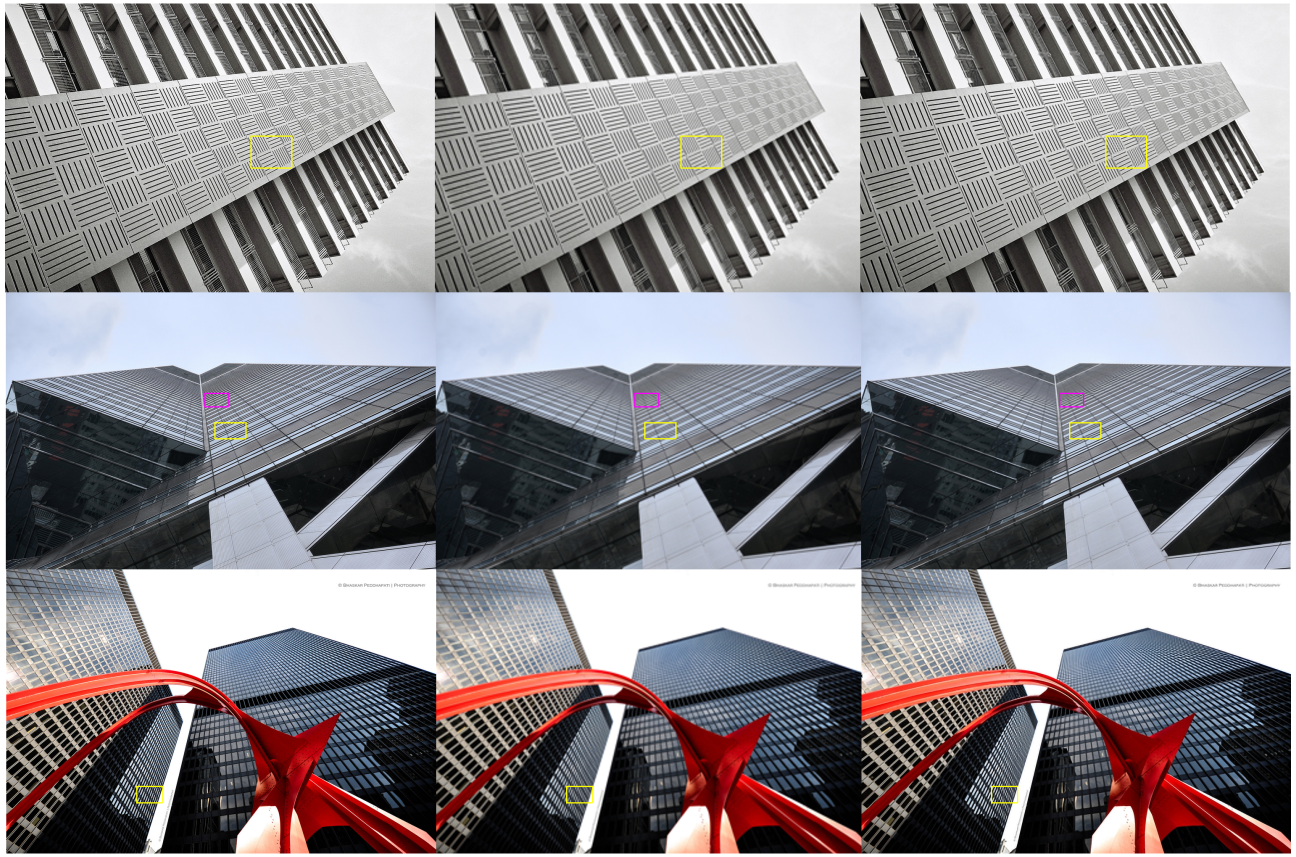


Figure 5.6: Visual examples from DNSR result for scaling ratio of 3 from Figure 5.5. GT (left), Bicubic (middle), DNSR (right)

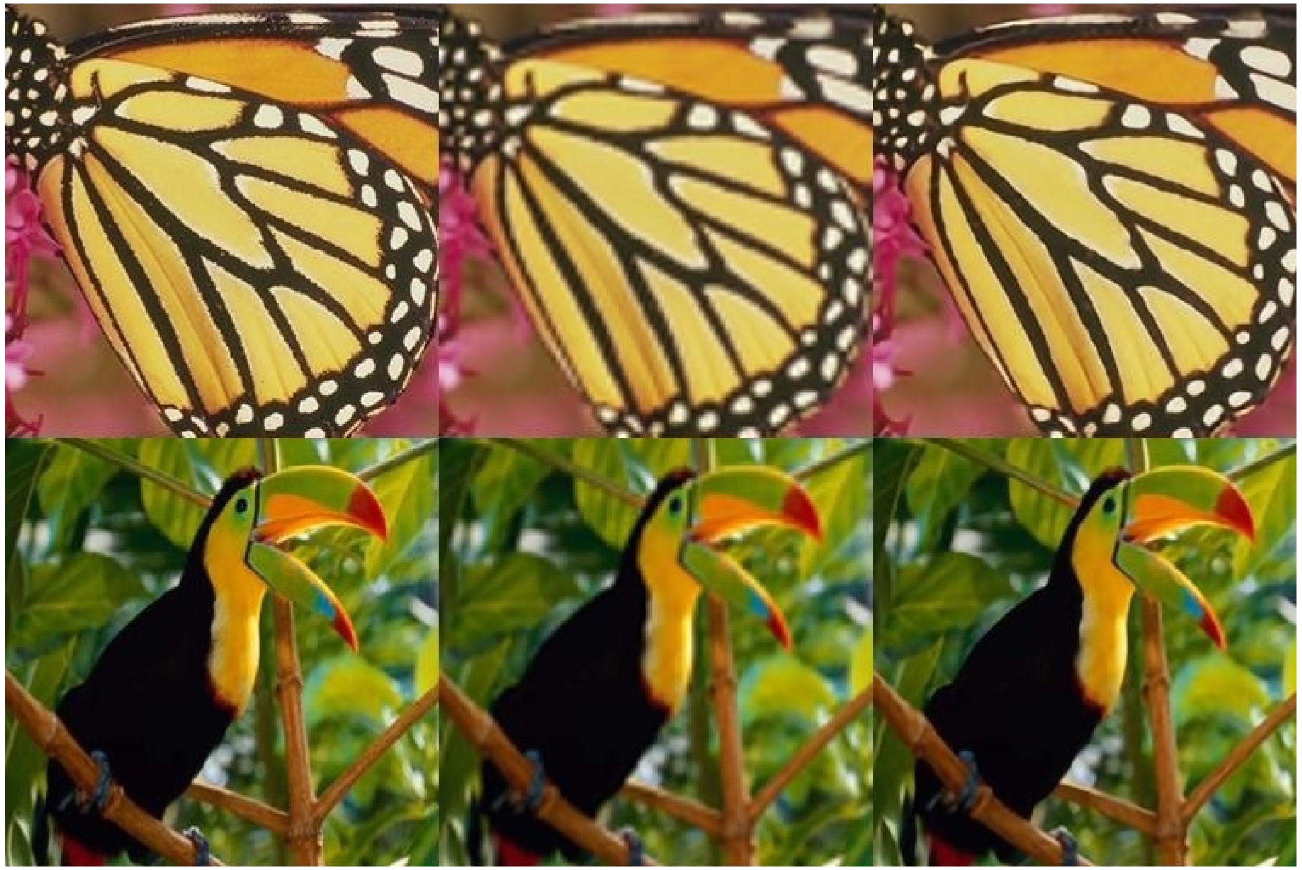


Figure 5.7: Further visual examples from DNSR result for scaling ratio of 3 on Set5 [10] images. GT (left), Bicubic (middle), DNSR (right)

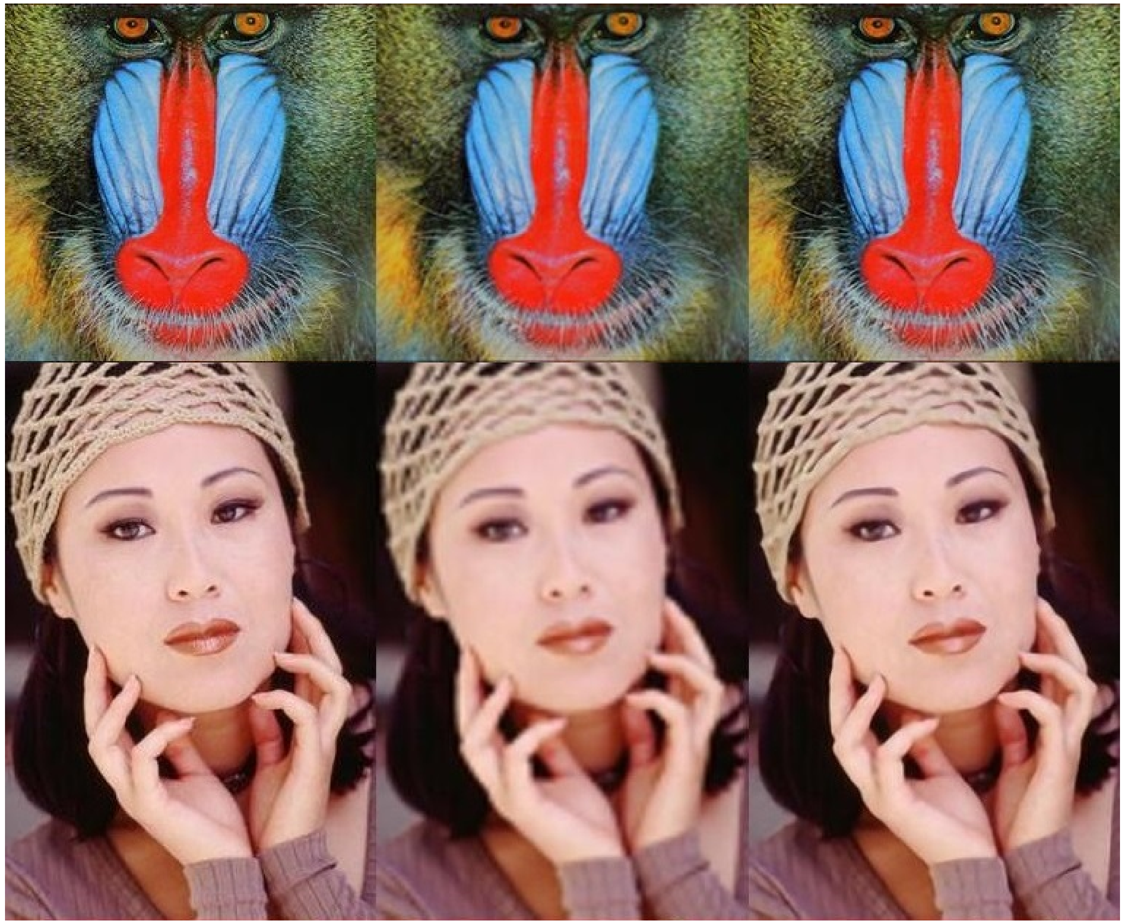


Figure 5.8: Further visual examples from DNSR result for scaling ratio of 3 on Set5 [10] and Set14 [11] images. GT (left), Bicubic (middle), DNSR (right)

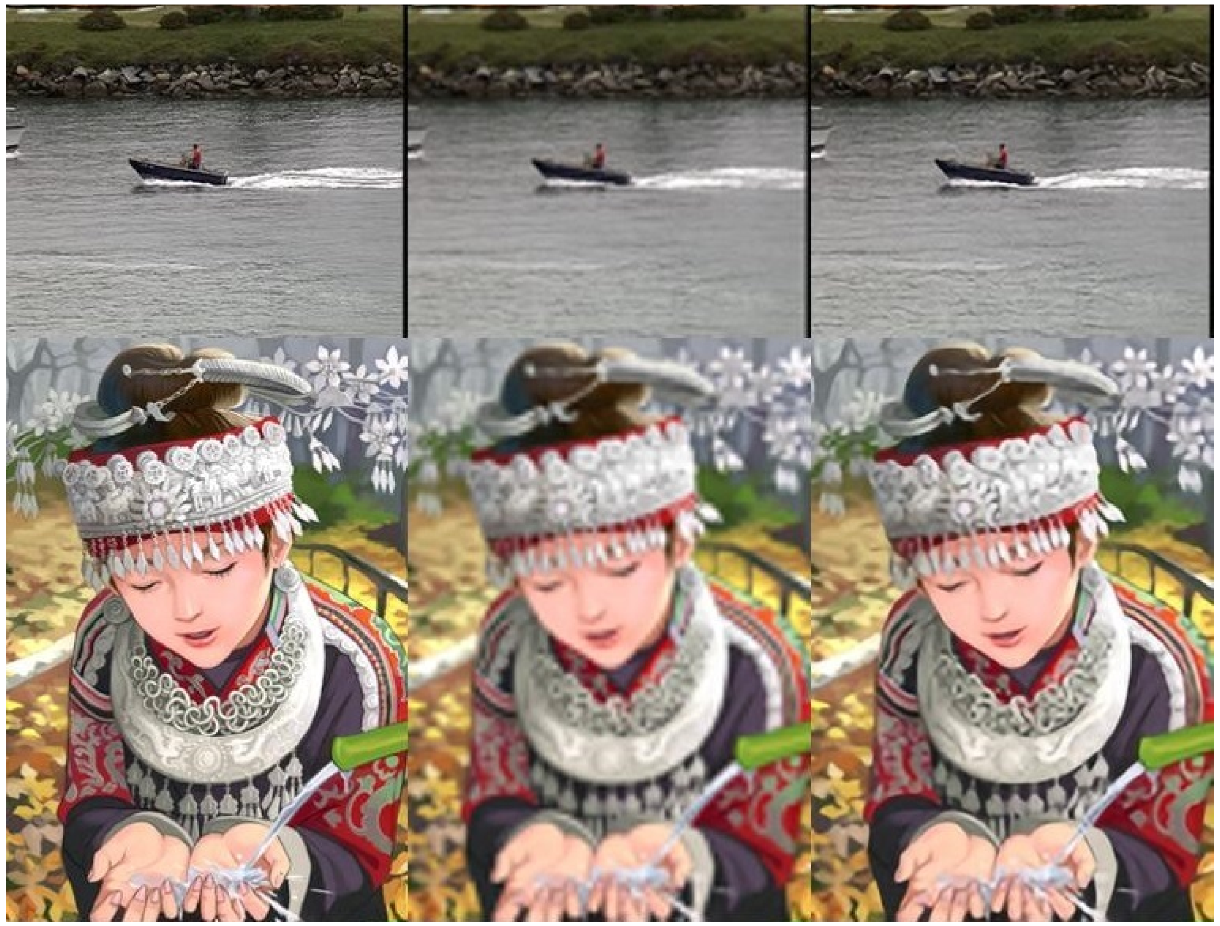


Figure 5.9: Visual examples from DNSR result for scaling ratio of 3 on Set14 [11] images. GT (left), Bicubic (middle), DNSR (right)

Table 5.5: Noise immunity test of DNSR vs IDN[1]. DNSR outperforms IDN in presence of noise even though IDN results were better on average without noise.

		Noised		Not Noised	
		DNSR	IDN	DNSR	IDN
BSD100	PSNR(dB)	**25.04**	24.96	28.83	**29.99**
	SSIM	**0.7254**	0.6663	**0.8527**	0.8013
URBAN100	PSNR(dB)	**26.38**	26.35	27.16	**27.42**
	SSIM	**0.6503**	0.6388	0.8293	**0.8359**

downsampled images to HR scale. When a novel input is presented to a network that is generated by a different model the result of CNN deteriorates.

This is also true for the presence of noise for networks. If a network is presented with a slight addition of noise its pixelwise error performance drops significantly, more so than that of known interpolation methods such as bicubic interpolation. For this reason we tested our DNSR for its noise immunity levels. Since DNSR consists of two different networks that are trained with different data we expect it to be more immune to noise then other SotA methods. The LR image is accepted as the input to the SR system. This is why instead of adding noise to the ground truth image and then downsampling, we have contaminated LR test sets with noise. The noise mean levels are held at 2 percent of highest value available for the images (0.02 to be precise, for the 291 image set that is used). Monte-Carlo simulations are utilized such that noise is generated in 100 realizations and the 100 sets of results are averaged. PSNR and SSIM results of noised reconstruction of DNSR is compared to IDN [1]. Results are displayed in Table 5.5. The decrease in performance of DNSR in presence of noise is smaller compared to IDN [1].

5.4.2 Multiscale Adaptation of DNSR

The structure of DNSR does not allow the networks to be trained with multiscale data such that algorithm can be used for many ratios at once. The learned interpolation can only be trained for a single ratio therefore the overall framework can only be used for

Table 5.6: Multiscale adaptation results of DNSR for 2x and 4x by using Bicubic intertpolation. PSNR/SSIM results are given

Dataset	Scale	Bicubic	SRCNN [3]	VDSR [4]	LapSRN [51]	IDN [1]	DNSR
BSD100 [6]	x2	29.56/0.8431	31.36/0.8879	31.90/0.8960	31.80/0.8952	32.08/0.8985	31.74/0.8923
	x4	25.96/0.6675	26.90/0.7101	27.29/0.7251	27.32/0.7275	27.41/0.7297	27.18/0.7289
Urban100 [7]	x2	26.88/0.8403	29.50/0.8946	30.76/0.9140	30.41/0.9103	31.27/0.9196	30.57/0.9188
	x4	23.14/0.6577	24.52/0.7221	25.18/0.7524	25.21/0.7562	25.41/0.7632	25.06/0.7573

single scaling ratio. The application of DNSR for multiple scale application requires optimization of network structure for each scaling ratio. Instead we have used DNSR to increase the resolution by 3 and adapt the upscaled image to required resolution by bicubic interpolation.

The results for large data sets (Urban and BSD) are given for comparison in table 5.6. A compromise in error performance is expected because of data model shift, such as in the case of noise immunity experiment. Although DNSR fares slightly below SotA in performance, this experiment indicates that if the structure of network is adapted and a new training is carried out, DNSR would be able to perform in the SotA performance.

5.4.3 Angled Data and RDD

In our Representation-Dictionary Duality we proposed that during training phase, neuron filters assume the role of representation vectors. After the training is finished (or during forward pass of training), these filter coefficients assume the role of a dictionary upon which the output image is reconstructed. This assertion required, in principle, that the learned filters to present the prevalent features from the training set such as edges and repeating patterns.

The spatial coherency computation method provided us with the opportunity to visually validate RDD. The arc tangent of y and x values of stronger eigenvalue yields the orientation of the strongest edge inside the image as in equation 52 as described in [31]. Utilizing this information we have obtained patches with orientation between

40 and 50 degrees as in Figure 5.10.

$$\theta = arctan(\lambda_{1,y}, \lambda_{1,y}) \tag{52}$$

After training a network with the dataset consisting of similar orientation edges, we have observed filters from the first layer. Picking strongest filters for better visualization we have observed that the trained filters also had similar orientation in Figure 5.11. Which visually verified RDD, showing that the neuron filters extract prominent features from the training set and they can be regarded as a representation vector and a dictionary interchangeably.

5.4.4 Perceptual Loss Function and Performance of SGD

Standard procedure in neural network learning is to calculate an error value from an iteration and to backpropagate the gradient of error to the network coefficients. This is called stochastic gradient descent (SGD) type learning. Normally the error function is chosen as MSE of difference of network output NET(I^{LR}) and I^{HR} where I^{LR} is low resolution input image and I^{HR} is the high resolution target image. The error is calculated as

$$MSE = \frac{1}{R}\sum_{\forall r}(NET(I_r^{LR}) - I_r^{HR})^2 \tag{53}$$

where R is total number of pixels on a patch. There have been other type of learning algorithms in the literature. The proposal in [60] and later [43] was to use another function instead of plain squared error calculation. Since methods such as SRGAN [43] and Style Transfer [60] require visually pleasing outcomes they have utilized pretrained layers of huge classification algorithms such as VGG-16 [62]. The VGG algorithm is a CNN based network that uses hundreds of filters at each layer to gradually reach to a binary or few digit outcome that will classify and image as depicted in Figure 5.12.

Classification networks are trained with millions of high quality images differently from superresolution networks where the standard training set contains averagely

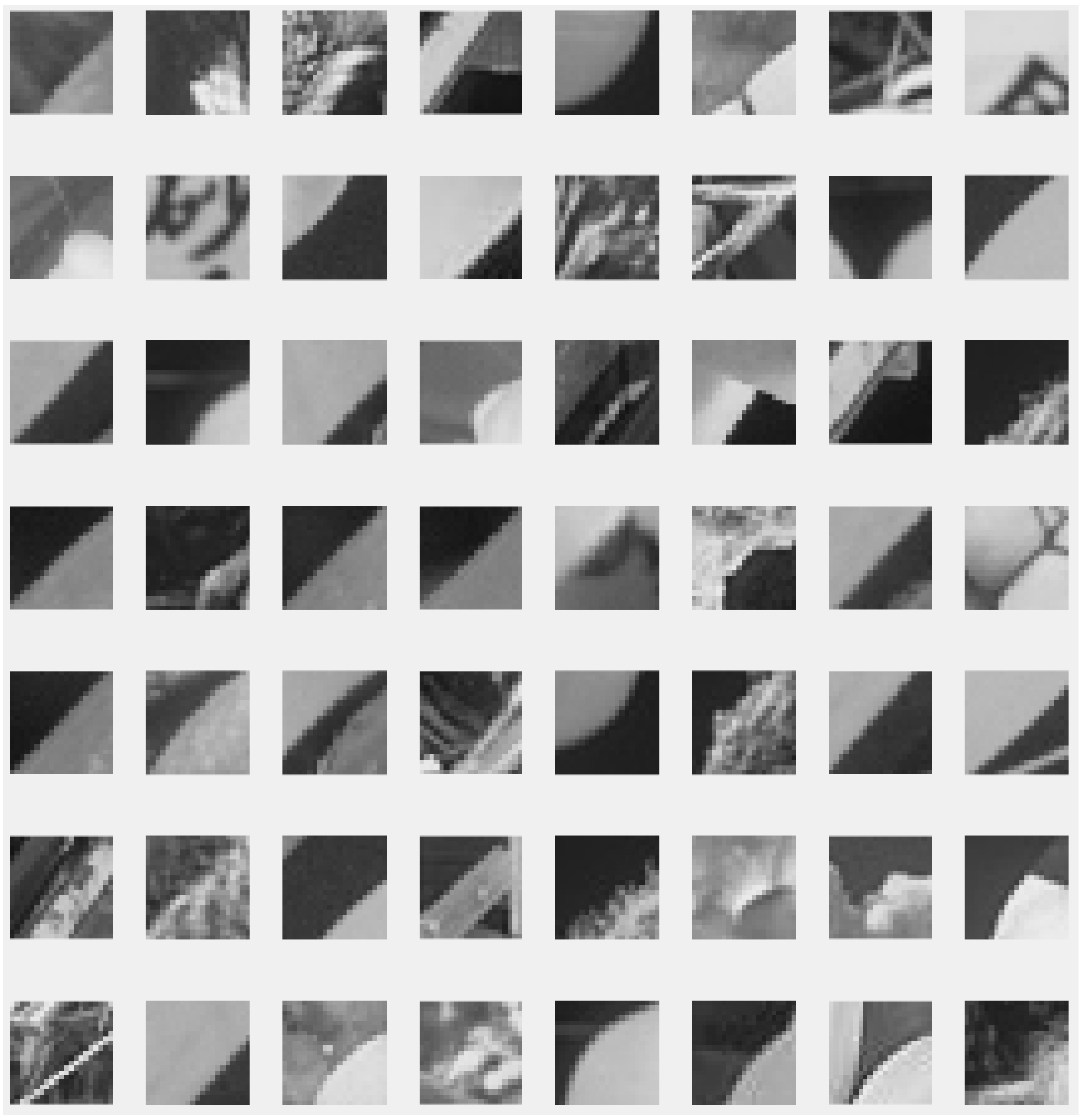


Figure 5.10: Image patches with angles between 40 and 50. Examples taken from BSD100 [6]

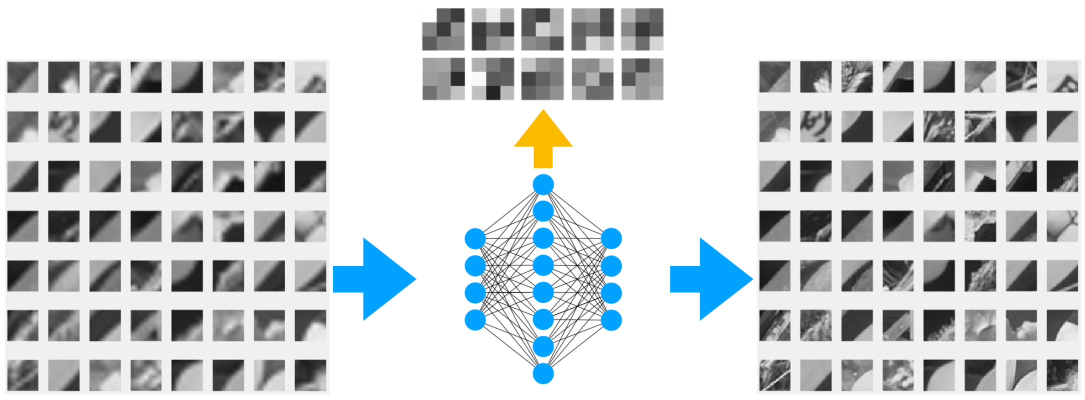


Figure 5.11: First layer filters from the network that is trained with image patches that have an orientation between 40 and 50 degrees. Indicated with yellow arrow.

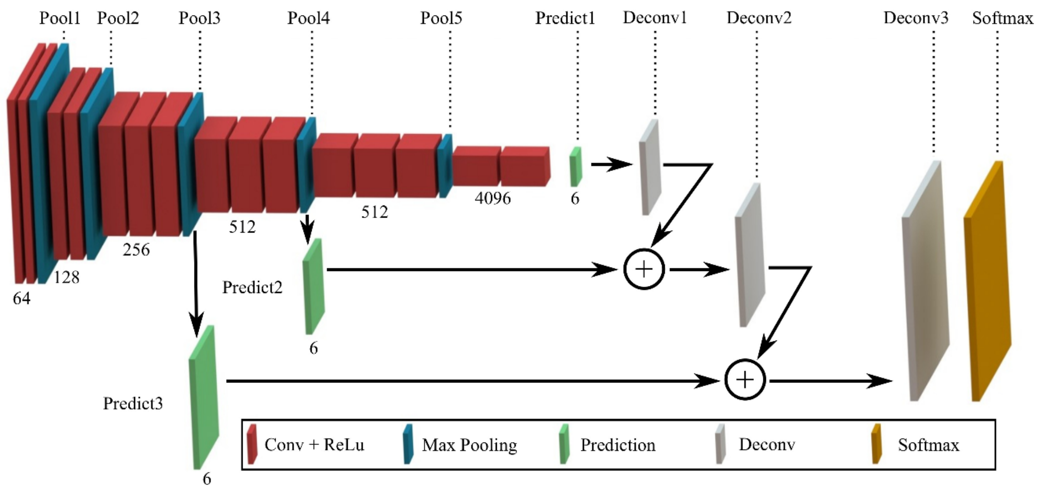


Figure 5.12: VGG-16 Network Topology

sized 291 images. Usage of vast library of images enables these networks to absorb prominent features from natural images. The idea of perceptual loss is to utilize some portion of this assimilated information inside the error calculation so that the network can effectively focus on prominent features of the training set instead of calculating error equivalently from all input pixels. Therefore the classical error computation becomes as

$$MSE = \frac{1}{R'} \sum_{\forall r} (\rho_i(NET(I_r^{LR})) - \rho_i(I_r^{HR}))^2 \tag{54}$$

where ρ_i stands for ith layer of the VGG network and R' stands for number of pixels at the output of ith layer of VGG network. The output of the SR network (NET) and target image is subjected to a layer of VGG network. This error value, the so called content loss [43], is used in conjunction with classical MSE calculation because the content loss cannot be used by itself to create superresolved images with fidelity to the original data.

The usage of content loss was limited to generative adversarial networks that generate visually sharp but low fidelity images. The MSE based algorithm SRResNet proposed in the same paper [43] does not use the content loss. One of the major setbacks of content loss is the huge amount of filters that needs to be used during training of the network. Second disadvantage is that content loss tends to smooth out the image causing localization errors for sharp edges. For these purposes we experimented on a different idea stemming from perceptual loss.

The Gabor filters were used, before the advent of CNN, inside the classification and detection algorithms [63]. Gabor filters are known for their feature discriminating properties. We have designed a simple network that will select dominant features from an image by using Gabor filters. Instead of using content loss that utilizes VGG network we have used 40 Gabor filters for the same purpose.

Although using Gabor filters for edge detection and feature emphasis is a good idea in general, the learning process of CNN was not feasible to incorporate its usefulness. After some testing we had to abandon this experiment due to halos and artifacts created by Gabor filters. Although it was much faster compared to the VGG based content loss it did not provide better results compared to our experiments with content loss + MSE based training.

Similarly using entropy loss function [43] together with MSE based error measurement did not provide better results. This concludes that SGD based learning methods suit best for SR applications.

CHAPTER 6

VIDEO EXPANSION OF DNSR

DNSR is intended for usage in realtime systems. Its lightweight formulation ensures realtime operability, while its reconstruction quality, fidelity and noise immunity guaranties its success.

There is an additional information that can be utilized for video applications, that is the temporal information. Since a video has almost always have movement, even if it is in the background, the subpixel information from neighboring frames can be harnessed.

6.1 Video Superresolution

Of course as there are advantages of using multi-frame approach to SR problem, there are also challenges to it. The list of challenges are as follows

- Motion information should be estimated for neighbor frames
- Video training and test sets must be prepared by authors as there is no consensus sets
- Neighbor frames should be motion compensated
- Motion compensated multiple frame information should be fused
- Learning based algorithms should be adapted to new structure

The literature of Video SR is quite limited compared to single image SR research. It is due the listed problems that the problem becomes multi-aspected, such as optical flow

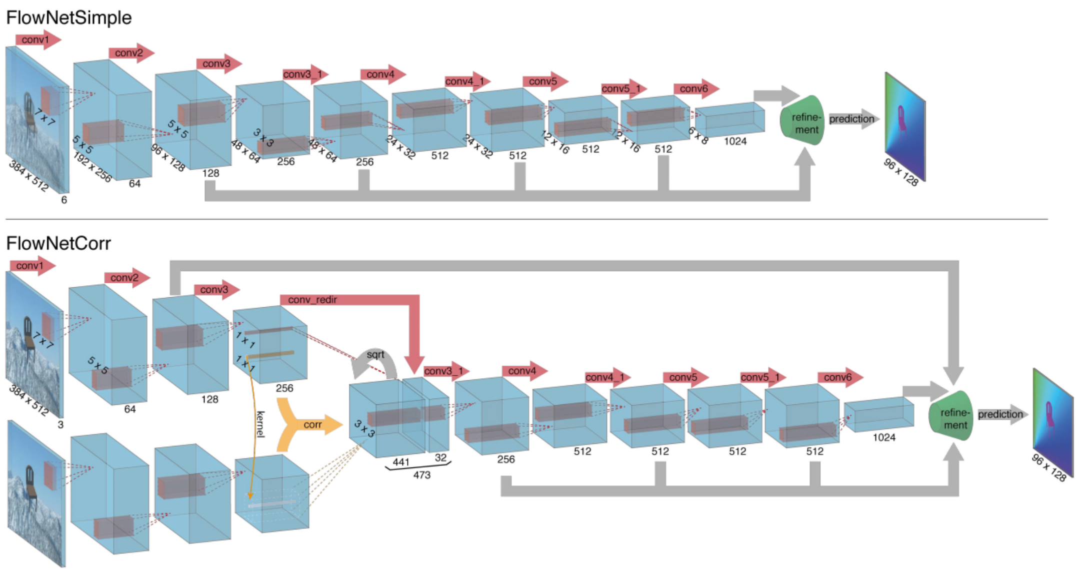


Figure 6.1: Flownet[12] Topology

estimation, image warping and SR at the same time. SotA methods that address Video SR (VSR) problem generally try to tackle these problems with simpler approaches [14][2][15] as we will discuss in this chapter.

6.1.1 Optical Flow and Motion Estimation

The main challenge of VSR is optical flow (OF) estimation. Although the literature is vast in terms of methods that can be utilized for OF estimation we limited our scope for CNN based methods. The reason for this is that in most of the cases CNN based approaches require joint training when they utilize cascaded models.

One of the most widely known methods is Flownet, that utilizes hundreds of filters at each layer to estimate the optical flow. Their framework is depicted in Figure 6.1. The cascaded pooling layers reduce the size of outputs from layers as it allow for the usage of more and more filters at each stage. This network topology is similar to classification methods such as VGG16 [62]. In another work pyramidal structure of coarse to fine flow is proposed for OF estimation [64]

Later Flownet's performance is improved by the introduction of Flownet2, which made it the SotA method for best performing OF estimators among CNN based

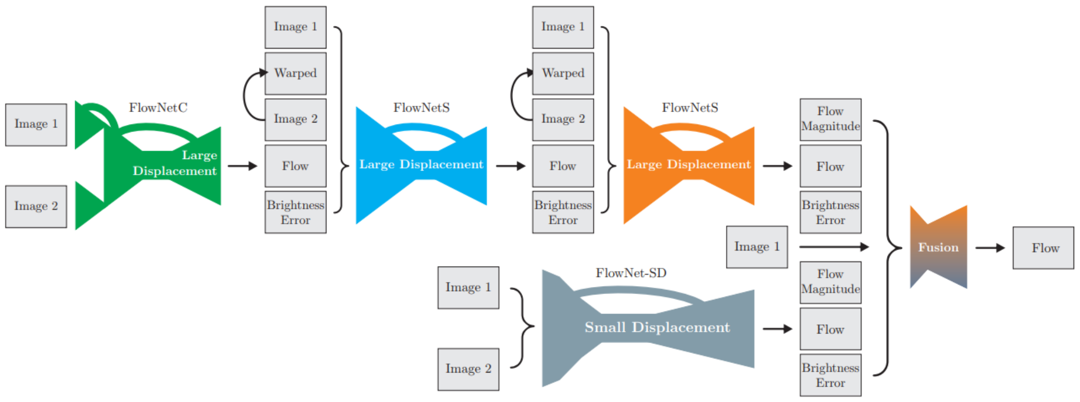


Figure 6.2: Flownet 2.0[13] Topology

methods. Their topology utilizes different structures of so called Pyramid structure. The final algorithm uses 160Million parameters to estimate OF and the model is trained in two weeks.

There have been attempts to reduce the number of operations required for OF estimation without sacrificing from performance [65] [66] [67] [68]. Although the fastest among them, the LiteFlownet [68], still utilizes 10 Million parameters making it gigantic compared to our DNSR that has only 52 thousand ops/op. Therefore fully-grown OF estimators must be out of the scope for a real-time capable VSR algorithm.

6.1.2 Motion Compensation and Warping

One of the first methods that use end-to-end trainable approach to VSR problem is VSRnet [14]. The main focus of the paper is how to fuse motion compensated frames by using neural networks. Three main methods have been inspected namely 3D filters, Slow fusion and Early fusion as depicted in 6.3. Although initially 3D convolution kernels seem to be the best approach to utilize temporal information in SR problem, it has been demonstrated that early fusion is the fastest and best performing method of information fusion.

Later another methods called Video ESPCN (VESPCN) [2] has been proposed by the same group that has proposed sub-pixel interpolation method[5]. VESPCN is targetted for real-time operability therefore their sub-blocks were light weight. VESPCN

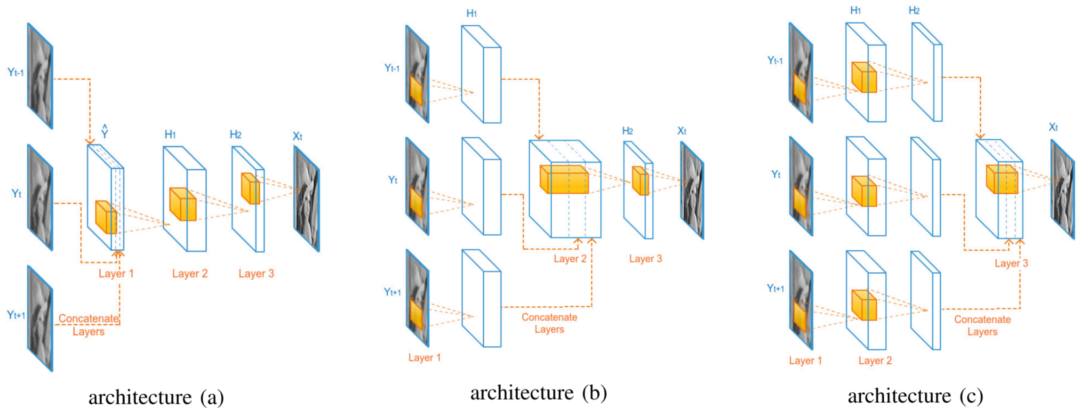


Figure 6.3: VSRnet[14] Topology

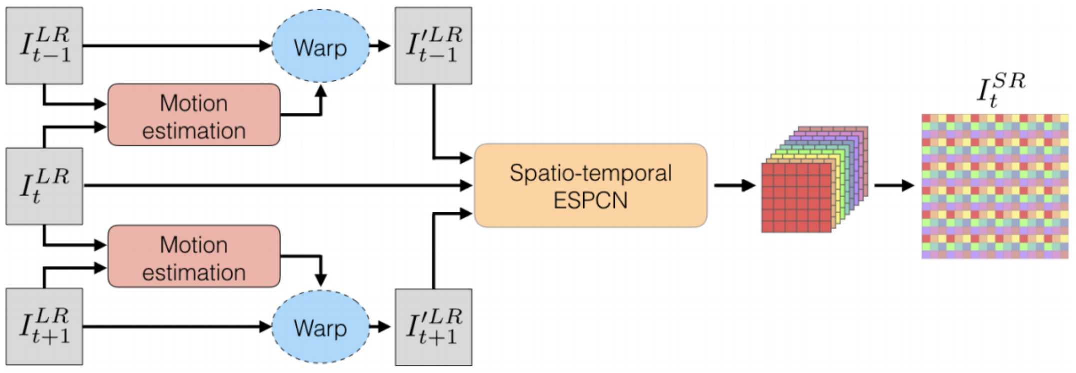


Figure 6.4: VESPCN[2] Topology

method is depicted in Figure 6.4. Similar to VSRnet, authors of VESPCN also inspected 3D convolutions, Slow fusion and Early fusion. The early fusion is found to be the best suitable method of fusing motion compensated images. Therefore the spatio-temporal ESPCN block depiceted in Figure 6.4 is nothing but ESPCN [5] topology. [2][15] The Motion compensation (MC) block of VESPCN is more important than its enhancement segment. Proposed MC block utilizes a coarse to fine architecture that is composed of total 12 layers of CNN as in Figure 6.5 and summarized in Table 6.1. By varying the size of filters and using strided filters, motion estimation and compensation is achieved at the same time. Although the performance of proposed MC block has not been measured outside the entire algorithm.

Later another methods has been proposed for VSR problem that utilizes a novel Subpixel Motion Compensation (SPMC) block that estimates the motion and interpolates

Layer	Coarse flow	Fine flow
1	Conv k5-n24-s2 / ReLU	Conv k5-n24-s2 / ReLU
2	Conv k3-n24-s1 / ReLU	Conv k3-n24-s1 / ReLU
3	Conv k5-n24-s2 / ReLU	Conv k3-n24-s1 / ReLU
4	Conv k3-n24-s1 / ReLU	Conv k3-n24-s1 / ReLU
5	Conv k3-n32-s1 / tanh	Conv k3-n8-s1 / tanh
6	Sub-pixel upscale $\times 4$	Sub-pixel upscale $\times 2$

Table 6.1: Motion compensation block of VESPCN [2]. CNN layers are summarized as their kernel size (k), number of filters (n) and stride (s). The activation functions are either RELU or tanh.

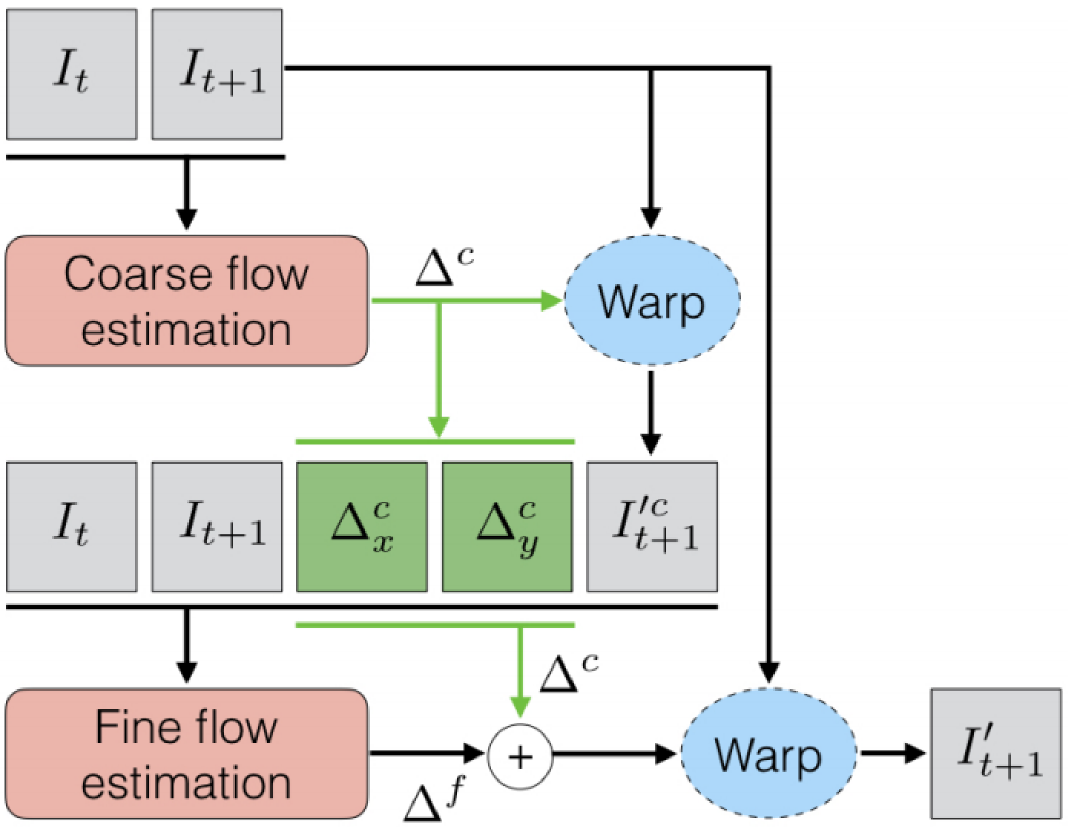


Figure 6.5: VESPCN Motion Compensation Topology

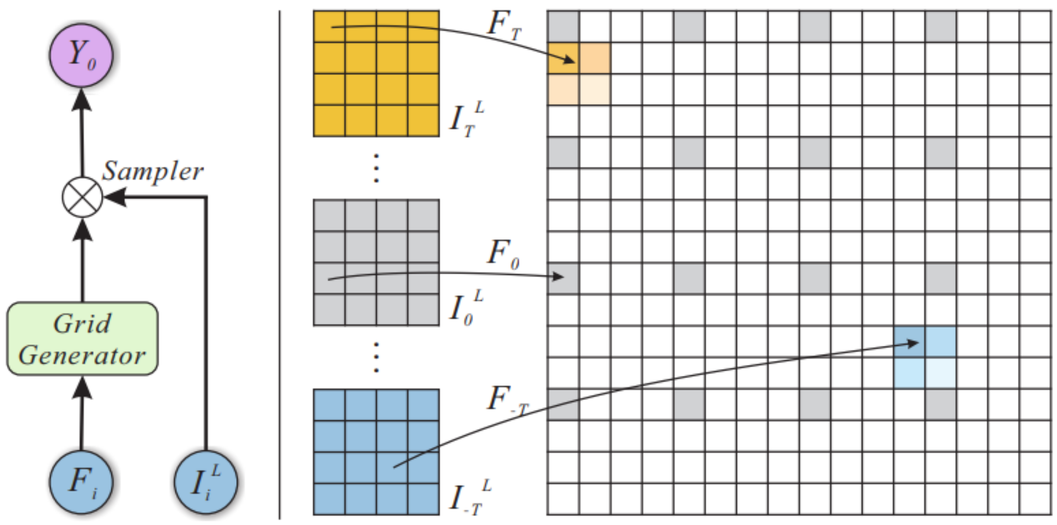


Figure 6.6: SubPixel Motion Compensation Block

an image according to subpixel motions [15]. The method uses, similar to all the previous methods, three frames for SPMC, two of the frames being the neighboring frames to the central frame.

Other methods include Huang et. al. [69] which uses bidirectional recurrent framework where networks are chosen to be shallow and motion compensation is done without explicit estimation. Similarly Jo et. al. [70] uses dynamic upsampling filters that are estimated recurrently for every new frame. By doing so the requirement for explicit motion estimation have been averted. Liu et. al. [71] proposed to learn temporal dynamics in a network that uses previously estimated motion information on following frames to reduce the operation weight. Sajjadi et. al. [72] used warped frame of previous computation for following frame calculations in a frame recurrent architecture.

We have tested SPMC as our initial candidate for MC for VSR algorithm. We have used SPMC in front of our DNSR as depicted in Figure 6.7

However the problem that occurred in the noise immunity test also appeared in VSR test. Since SPMC created images with slightly different sharpness compared to bicubic interpolation, the results of SPMC+DNSR were not as successful as expected as in Figures 6.8 and 6.9. Joint optimization of SPMC+DNSR was not possible as trainable files of SPMC were not presented online. Therefore we have chosen to propose

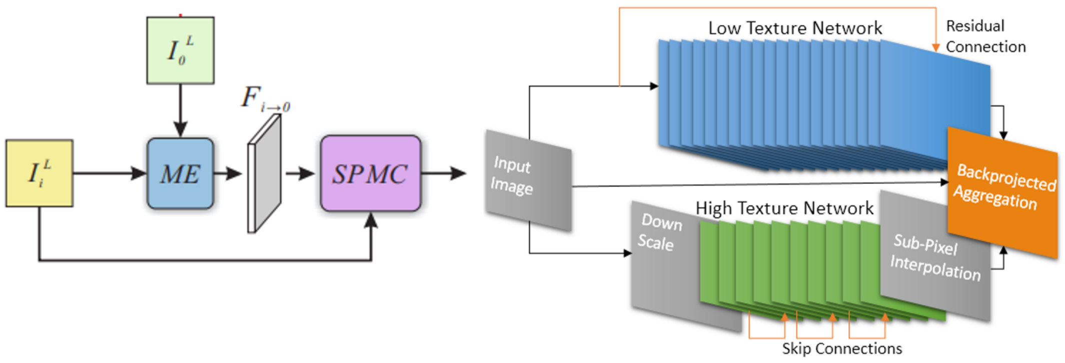


Figure 6.7: Usage of SPMC[15] together with our DNSR

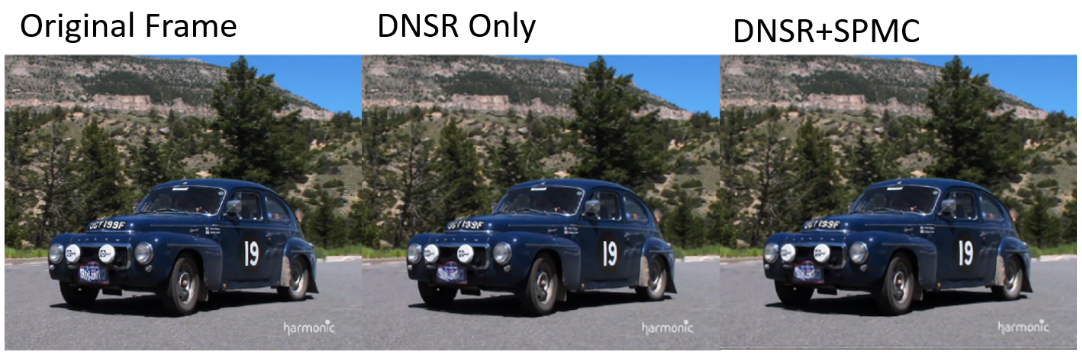


Figure 6.8: DNSR only compared to VSR usage of SPMC+DNSR

our own method for end-to-end trainable VSR algorithm.

6.1.3 Detail Fusion Interpolation

The learning ability of CNNs for any given task is incredible looking at plethora of research areas from scene understanding to generative networks. Although most of the times it is mathematically unproven, using enough number of filters and with correct choice of building blocks any imaging problem can be taught to a CNN. For this reason we have proposed an architecture that we hoped would account for motion estimation and motion compensation at the same time.

Usage of 3D filters have been discussed in literature, including the methods in our limited scope of VSR. Utilizing temporal information is easier by the usage of 3D filters since they can also learn motion together with spatial variations. However proposing a CNN based motion estimator should have some assumptions regarding

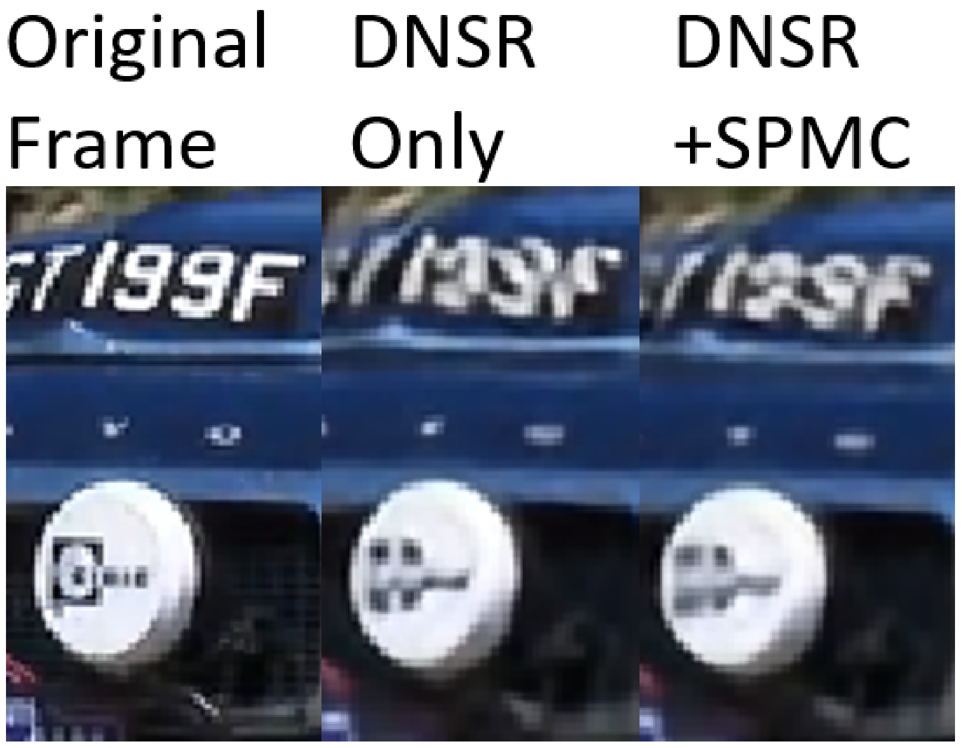


Figure 6.9: Details of comparison of DNSR to VSR usage of SPMC+DNSR

the speed and variation of motion. For this reason by assuming mild panning, affine transformation and minimal occlusions we have proposed using a network whose initial layer consists of 7x7 filters.

As we have proven in Chapter 3 learned coefficients of a network follow Representation-Dictionary Duality. This meant that the prominent features of the training set were absorbed by the filters in similar shapes. Using this line of thought in formulation of multi-frame approach we propose that by using enough number of filters the CNN could capture the motion of prominent features, therefore could compensate for assumed motion statistics. 256 filters per frame at the initial layer have been used. Second layer of the network was used to fuse outputs of the first layer with 64 filters of size 3x3 and the last layer is used for the interpolation kernels similar to subpixel interpolation block of [5]. This adds 17K ops/op to the total number of operations which is not comparable to 160 Million of parameters that are utilized in SotA methods.

There is no consensus training and test set for video superresolution. For this purpose

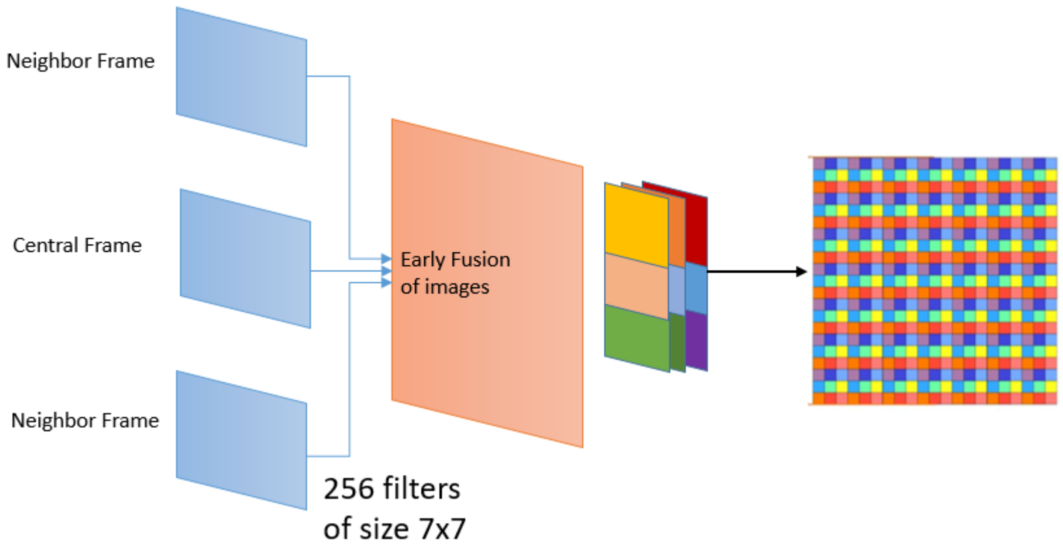


Figure 6.10: Detail Fusion Interpolator (DFI)

we have used the same set that has been used by [15] taken from [73] [74]. The proposed 3 layered network, as in Figure 6.10, has been trained for 6 hours with the same setup as our DNSR, described in Chapter 5. Resulting network, the Detail Fusion Interplator (DFI), is an SR algorithm on its own, although its main purpose is to fuse multiple frames together simultaneously estimating and compensating motion.

6.1.4 DNSR + DFI

Comparison of DFI to bicubic interpolation and DNSR is shown in Figure 6.11

To test the motion estimation capability of DFI we have submitted repeated frames as neighboring frames to the algorithm. It is clearly seen in Figure 6.12 that the DFI can capture information from neighboring frames. Similarly we wanted to test our proposal of a CNN's ability to capture motion information of prominent features. Since the training set contained similar adjacent frames, we have trained a variation of DFI that is able to accept 5 frames. The filters from initial layer of DFI showed signs of motion capture of some features shown in Figure 6.13. Motion information absorption capability of DFI can only be verified, other than visualizing the filters, by using DFI for the purpose it was trained for and quantitatively testing the end results.

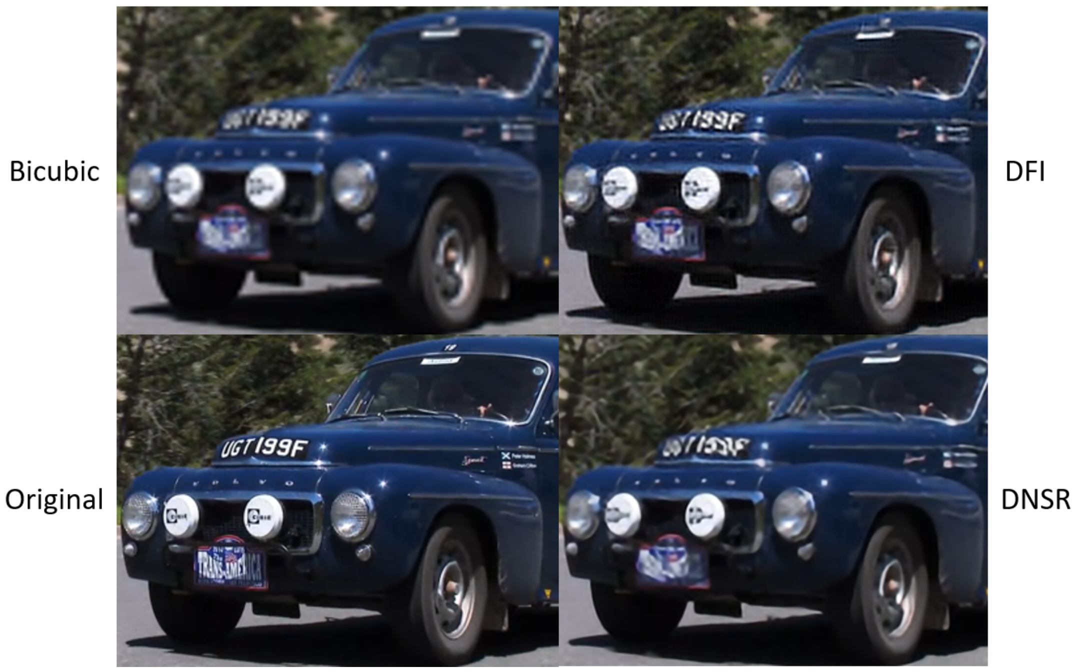


Figure 6.11: Comparison of Bicubic interpolation, DNSR and DFI

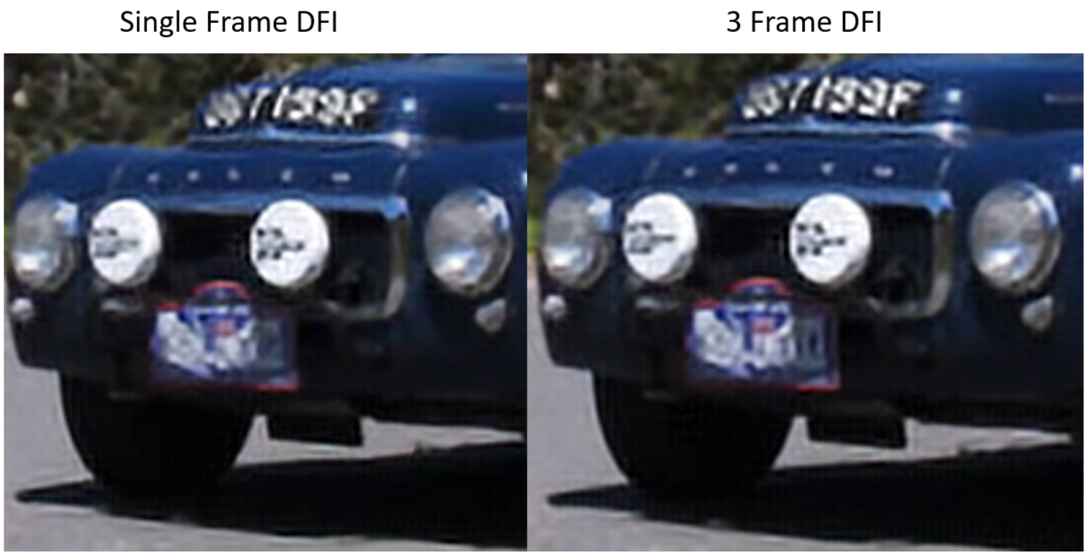


Figure 6.12: DFI tested with repeated single frame and neighboring three frames. This test confirms the motion compensation ability of DFI.

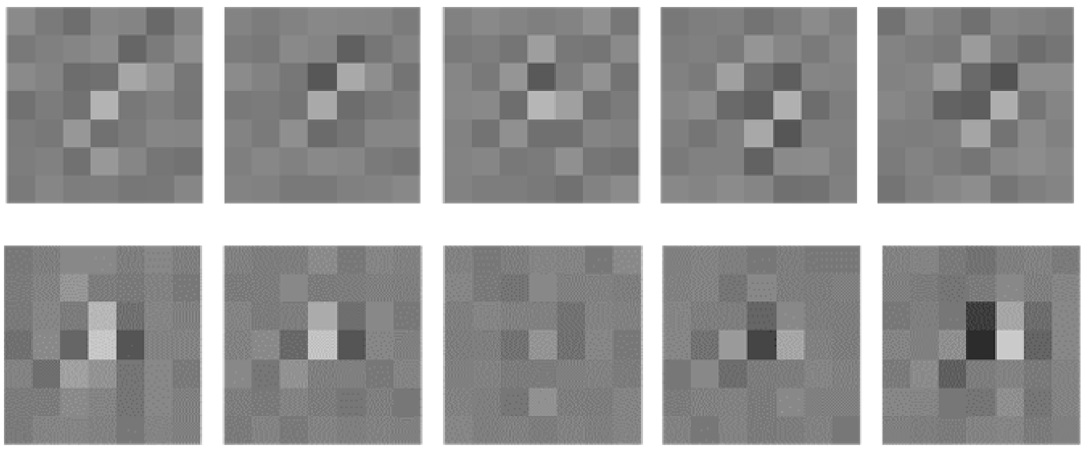


Figure 6.13: 2 sets of filters from initial layer of DFI to visualize motion capture capacity for prominent features

After the training of DFI is completed, it is used together with DNSR. Similary to SPMC case there have been artifacts in the resulting image due to the modelling change of DNSR input. Therefore we have jointly optimized DFI and DNSR as shown in Figure 6.15. We have used pretrained coefficients from low texture network. The high texture network accepts DFI output after bicubic downsampling. This operation cancels our opportunity to use pretrained coefficients. Three options have been discussed for the solution of this

- Training High Texture network with video training data set
- Training a learned downsampling layer to couple DFI and HT network in a jointly trainable fashion
- To use pretrained coefficients as is

By experimenting with three options we have concluded that total performance was not improving in retraining strategies. Training a high texture network with only video training set was not successful because of limited number of examples (30 sets to be exact). Training a downsampling network had its effect on the total performance, subtracting what re-training added to the performance. Therefore we have chosen to use trained high texture network coefficients as is. This is not a problem as it was in other cases because of bicubic downsampling. The resizing of an image with a known

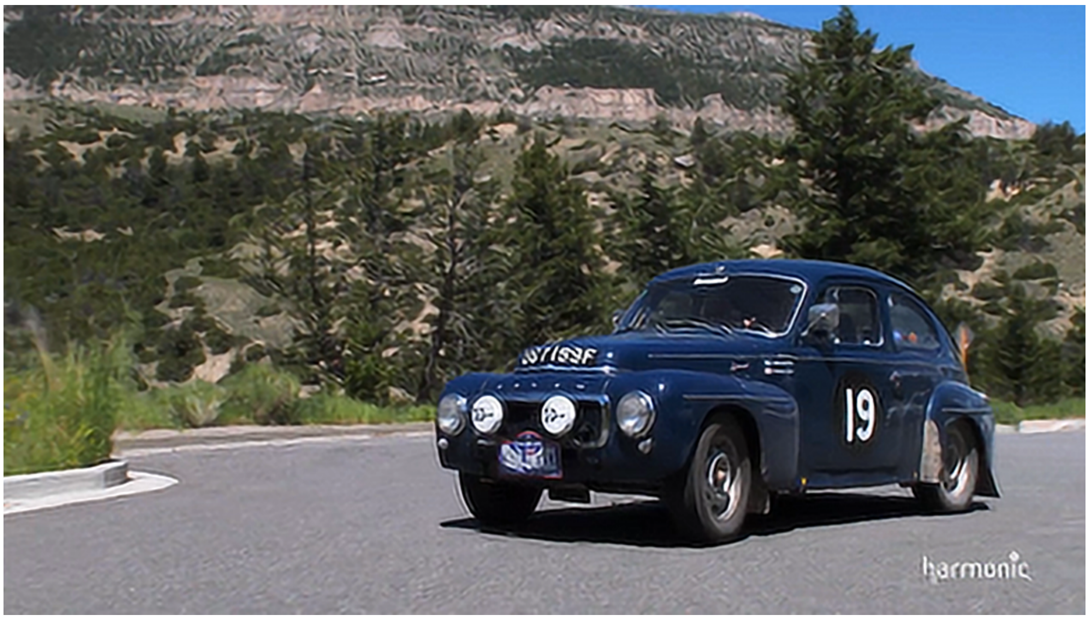


Figure 6.14: Ringing artifacts caused by using DFI together with DNSR without joint optimization

operator (known to the trained coefficients) removed the effects of statistical changes to the input image.

The result of DFI+DNSR is displayed in Figure 6.16. A numerical comparison is done for SotA VSR methods in Table 6.2. DFI+DNSR is not the best performing method in the literature. However we have demonstrated that the motion can be estimated and compensated at the same time by approximately 17 thousand ops/op as DFI+DNSR score is higher than that of DNSR solely. Also DFI+DNSR has 68K ops/op whereas SotA VSR method [70] has 2.7M parameters and 675K ops/op which

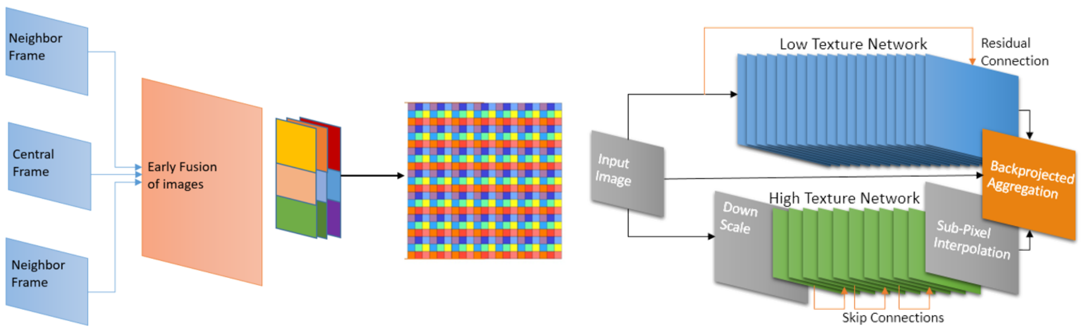


Figure 6.15: Joint DFI DNSR Topology

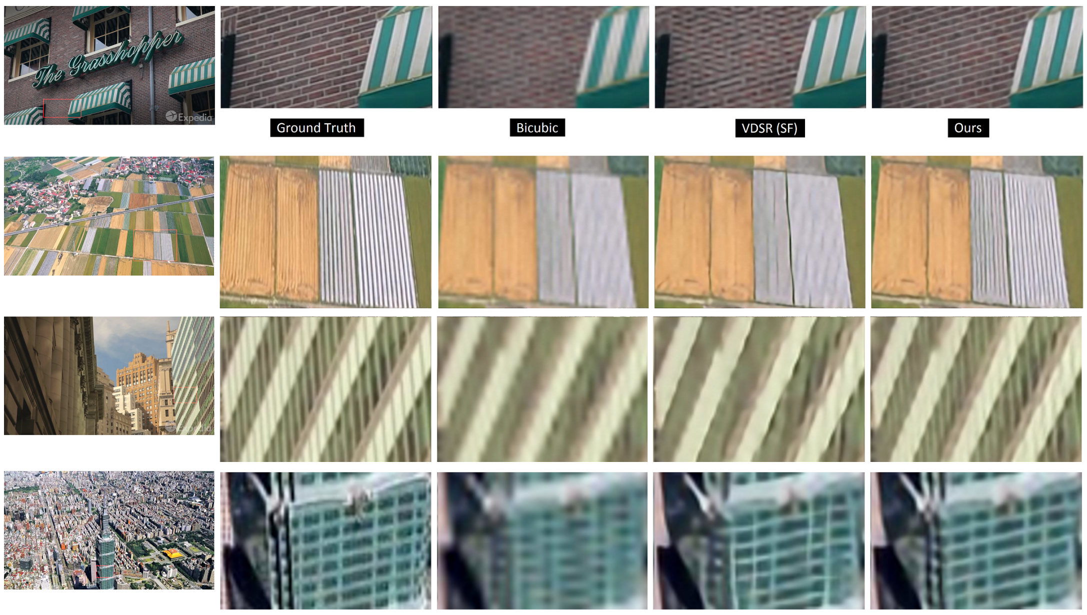


Figure 6.16: DFI+DNSR result compared with single frame methods. VSR comparison could not be done because test codes for 3x scaling was not available online.

Table 6.2: DFIDNSR is compared to SotA methods that have reported 3x scaling results.

		Bicubic	VSRnet [14]	VESPCN [2]	Tao et. al. [15]	Jo et. al. [70]	Ours
SPMCS [15]	PSNR	28.85	28.55	-	31.92	-	31.68
	SSIM	0.82	0.85	-	0.90	-	0.89
Vid4 [75]	PSNR	25.64	25.31	27.25	27.49	28.90	27.41
	SSIM	0.80	0.76	0.84	0.84	0.89	0.85

is 10 times more computational load. Therefore DFI+DNSR is a lightweight and fast algorithm that performs well within its computational category.

CHAPTER 7

CONCLUSION

In this thesis a mathematical analysis of CNNs have been carried out. The learning process of neurons have been proven to be optimal for inverse problems. The filters of CNNs have been shown to follow Representation-Dictionary Duality (RDD). Skip connections are proven to be useful and even required for CNNs to converge in performance, especially for high textured images.

Using mathematical background that we have provided and combining multiple ideas we have carefully constructed a context based double network superresolution (DNSR) algorithm. The DNSR consists of two separate networks that are trained individually. One of the networks, namely low texture network, is responsible from constructing edges and corners inside the image. It is trained with low texture image patches to ensure it is performing its assigned duty. Other network is specialized for high textured image reconstruction. High texture network is trained with low spatial coherency data.

The data separation is done by using a method called spatial coherency measure. Low texture images consist of edges and mostly flat regions. High texture images consist of regular and irregular textures, multiple orientation edges. Using RDD and Skip connection analysis we have utilized skip connections for low coherency (high texture) network. The depths of individual networks are also chosen according to our mathematical analysis. High texture network is chosen shallow but with more filters per layer and low texture network is chosen to be deeper with less parameters in total.

We have shown that DNSR is competing with SotA method. In terms of speed DNSR is 1.4 times faster than the fastest algorithm and more than 10 times faster than the average of inspected methods. DNSR also performs well in terms of pixel wise error performance. Except one method IDN[1], DNSR scores are also better than SotA methods. The IDN method is subjected to extensive testing against DNSR and it has been found that the informatively rich portions of an image such as textured and edged regions are recovered, on average, better compared to IDN.

Noise immunity of DNSR is tested and it has been found to be more immune to noise than other SotA methods. Such tests are important in terms of determining an algorithm's overall success. A method should also succeed outside limited scope of training/test sets. The surest method to determine this is to measure noise immunity.

An extension of DNSR to video SR is also proposed. The multiple frame processing comes with its challenges such as optical flow estimation, motion compensation, pixel registration. Instead of using huge constructs designed solely for optical flow estimation we have proposed a light-weight joint motion estimation-compensation block namely Detail Fusion Interpolator (DFI). We have tested the performance of DFI alone and together with DNSR. Usage of DFI together with DNSR improved DNSR results for video sequences that are used in the literature for video SR algorithms.

7.1 Future Work

We have proposed many useful method and ideas during this thesis that stood incomplete. In the future we plan on completing tests and analyses on these methods.

We have proposed Separate Channel Interpolation (SCHI) for the purpose of subpixel accuracy interpolation. The idea behind SCHI was to utilize a network's capacity to evolve one pixel towards n^2 many pixels where n is the upsampling ratio. All of the methods in the literature use a single network to bring about all of the interpolated pixels. As it can be further discussed with manifold theory [16] this would mean using the same low dimensional manifold to enhance subpixels which is a downgrading factor. Although pixelwise performance of our proposed algorithm was not as high as expected we tie this result to our limited scope of real-time operable network. In the future we will test SCHI idea with heavier networks to see its full usefulness.

We have tested some of the mostly used texture classification method to separate the training set into two. We also compared texture cue to other cues such as strength of edge information. But we still think there could be more methods to try for separation of data into informatively meaningful portions. One of such methods could be to train an individual network to do the separation.

From our analysis on network structure in Chapter 4, we have concluded that networks that are trained with almost flat patches should increase their receptive fields. We have proposed to use deeper networks to include more information inside the CNN, which can also be done by using dilated convolutions [76]. Dilated convolutions can also capture a larger area during convolution operation and effectively increase the receptive field of a network. We plan a further research on different network structures that can be tested against our proposed DNSR networks. Also different training strategies that do not only follow MSE based gradients but also SSIM based gradients are planned to be investigated.

The video expansion of DNSR can be carried further. The DFI is still size limited and can be evolved. Using more layers and more filters we plan on experimenting more on DFI structure for joint estimation and compensation of motion. Also, as it is in the case of inverse problems, the mathematical foundation of using CNN for optical flow estimation and also motion compensation is lacking and requires further research.

REFERENCES

[1] Z. Hui, X. Wang, and X. Gao, “Fast and accurate single image super-resolution via information distillation network,” *CoRR*, vol. abs/1803.09454, 2018.

[2] J. Caballero, C. Ledig, A. P. Aitken, A. Acosta, J. Totz, Z. Wang, and W. Shi, “Real-time video super-resolution with spatio-temporal networks and motion compensation,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 2848–2857, 2017.

[3] C. Dong, C. C. Loy, K. He, and X. Tang, “Learning a deep convolutional network for image super-resolution,” in *ECCV (4)*, vol. 8692 of *Lecture Notes in Computer Science*, pp. 184–199, Springer, 2014.

[4] J. Kim, J. K. Lee, and K. M. Lee, “Accurate image super-resolution using very deep convolutional networks,” in *CVPR*, pp. 1646–1654, IEEE Computer Society, 2016.

[5] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, “Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network,” in *CVPR*, pp. 1874–1883, IEEE Computer Society, 2016.

[6] D. R. Martin, C. 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*, pp. 416–425, 2001.

[7] J. Huang, A. Singh, and N. Ahuja, “Single image super-resolution from transformed self-exemplars,” in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pp. 5197–5206, 2015.

[8] E. Agustsson and R. Timofte, “Ntire 2017 challenge on single image super-resolution: Dataset and study,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[9] R. Timofte, E. Agustsson, L. Van Gool, M.-H. Yang, L. Zhang, B. Lim, *et al.*, “Ntire 2017 challenge on single image super-resolution: Methods and results,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[10] M. Bevilacqua, A. Roumy, C. Guillemot, and M. Alberi-Morel, “Low-complexity single-image super-resolution based on nonnegative neighbor embedding,” in *British Machine Vision Conference, BMVC 2012, Surrey, UK, September 3-7, 2012*, pp. 1–10, 2012.

[11] R. Zeyde, M. Elad, and M. Protter, “On single image scale-up using sparse-representations,” in *Curves and Surfaces - 7th International Conference, Avignon, France, June 24-30, 2010, Revised Selected Papers*, pp. 711–730, 2010.

[12] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, “Flownet: Learning optical flow with convolutional networks,” in *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pp. 2758–2766, 2015.

[13] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, “Flownet 2.0: Evolution of optical flow estimation with deep networks,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 1647–1655, 2017.

[14] A. Kappeler, S. Yoo, Q. Dai, and A. K. Katsaggelos, “Video super-resolution with convolutional neural networks,” *IEEE Trans. Computational Imaging*, vol. 2, no. 2, pp. 109–122, 2016.

[15] X. Tao, H. Gao, R. Liao, J. Wang, and J. Jia, “Detail-revealing deep video super-resolution,” in *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pp. 4482–4490, 2017.

[16] Y. Bengio, A. C. Courville, and P. Vincent, “Representation learning: A review

and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, 2013.

[17] Y. LeCun, Y. Bengio, and G. E. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[18] M. T. McCann, K. H. Jin, and M. Unser, "Convolutional neural networks for inverse problems in imaging: A review," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 85–95, 2017.

[19] A. Lucas, M. Iliadis, R. Molina, and A. K. Katsaggelos, "Using deep neural networks for inverse problems in imaging: Beyond analytical methods," *IEEE Signal Process. Mag.*, vol. 35, no. 1, pp. 20–36, 2018.

[20] V. Papyan, Y. Romano, and M. Elad, "Convolutional neural networks analyzed via convolutional sparse coding," *CoRR*, vol. abs/1607.08194, 2016.

[21] X. Mao, C. Shen, and Y. Yang, "Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections," in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 2802–2810, 2016.

[22] P. L. Combettes and V. R. Wajs, "Signal recovery by proximal forward-backward splitting," *Multiscale Modeling & Simulation*, vol. 4, no. 4, pp. 1168–1200, 2005.

[23] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pp. 399–406, 2010.

[24] Y. Yang, J. Sun, H. Li, and Z. Xu, "Deep admm-net for compressive sensing MRI," in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 10–18, 2016.

[25] I. Daubechies, M. Defrise, and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Communications on Pure and Applied Mathematics*, vol. 57, no. 11, pp. 1413–1457, 2004.

[26] D. Gabay and B. Mercier, "A dual algorithm for the solution of nonlinear variational problems via finite element approximation," *Computers & Mathematics with Applications*, vol. 2, no. 1, pp. 17–40, 1976.

[27] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution as sparse representation of raw image patches," in *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008), 24-26 June 2008, Anchorage, Alaska, USA*, 2008.

[28] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *Trans. Img. Proc.*, vol. 19, pp. 2861–2873, Nov. 2010.

[29] R. Timofte, V. D. Smet, and L. J. V. Gool, "Anchored neighborhood regression for fast example-based super-resolution," in *ICCV*, pp. 1920–1927, IEEE Computer Society, 2013.

[30] R. Timofte, V. D. Smet, and L. J. V. Gool, "A+: adjusted anchored neighborhood regression for fast super-resolution," in *ACCV (4)*, vol. 9006 of *Lecture Notes in Computer Science*, pp. 111–126, Springer, 2014.

[31] Y. Romano, J. Isidoro, and P. Milanfar, "RAISR: rapid and accurate image super resolution," *IEEE Trans. Computational Imaging*, vol. 3, no. 1, pp. 110–125, 2017.

[32] Y. Chen and T. Pock, "Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1256–1272, 2017.

[33] M. Aharon, M. Elad, and A. Bruckstein, "Ksvd: An algorithm for designing overcomplete dictionaries for sparse representation," *Trans. Sig. Proc.*, vol. 54, pp. 4311–4322, Nov. 2006.

[34] A. M. Bronstein, P. Sprechmann, and G. Sapiro, "Learning efficient structured sparse models," in *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*, 2012.

[35] C. J. Schuler, M. Hirsch, S. Harmeling, and B. Schölkopf, "Learning to deblur," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 7, pp. 1439–1451, 2016.

[36] L. Xu, J. S. J. Ren, C. Liu, and J. Jia, "Deep convolutional neural network for image deconvolution," in *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pp. 1790–1798, 2014.

[37] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep CNN for image denoising," *IEEE Trans. Image Processing*, vol. 26, no. 7, pp. 3142–3155, 2017.

[38] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[39] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *AISTATS*, vol. 15 of *JMLR Proceedings*, pp. 315–323, JMLR.org, 2011.

[40] J. Bruna, P. Sprechmann, and Y. LeCun, "Super-resolution with deep convolutional sufficient statistics," *CoRR*, vol. abs/1511.05666, 2015.

[41] V. Papyan, J. Sulam, and M. Elad, "Working locally thinking globally: Theoretical guarantees for convolutional sparse coding," *IEEE Trans. Signal Processing*, vol. 65, no. 21, pp. 5687–5701, 2017.

[42] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops, Honolulu, HI, USA, July 21-26, 2017*, pp. 1132–1140, 2017.

[43] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 105–114, 2017.

[44] W. Lai, J. Huang, N. Ahuja, and M. Yang, "Fast and accurate image super-resolution with deep laplacian pyramid networks," *CoRR*, vol. abs/1710.01992, 2017.

[45] Y. Tai, J. Yang, and X. Liu, "Image super-resolution via deep recursive residual network," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 2790–2798, 2017.

[46] Y. Tai, J. Yang, X. Liu, and C. Xu, "Memnet: A persistent memory network for image restoration," in *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pp. 4549–4557, 2017.

[47] C. Tarhan and G. B. Akar, "Convolutional neural networks analysed via inverse problem theory and sparse representations," *IET Signal Processing*, vol. 13, no. 2, pp. 215–223, 2019.

[48] C. Dong, C. C. Loy, and X. Tang, "Accelerating the super-resolution convolutional neural network," in *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II*, pp. 391–407, 2016.

[49] C. G. Turhan and H. S. Bilge, "Single image super resolution using deep convolutional generative neural networks," in *26th Signal Processing and Communications Applications Conference, SIU 2018, Izmir, Turkey, May 2-5, 2018*, pp. 1–4, 2018.

[50] J. Kim, J. K. Lee, and K. M. Lee, "Deeply-recursive convolutional network for image super-resolution," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 1637–1645, 2016.

[51] W. Lai, J. Huang, N. Ahuja, and M. Yang, "Deep laplacian pyramid networks for fast and accurate super-resolution," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 5835–5843, 2017.

[52] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 770–778, 2016.

[53] M. Haris, G. Shakhnarovich, and N. Ukita, "Deep back-projection networks

for super-resolution," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[54] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, "Residual dense network for image super-resolution," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[55] S. Mallat, "Understanding deep convolutional networks," *CoRR*, vol. abs/1601.04920, 2016.

[56] M. Ranzato, C. S. Poultney, S. Chopra, and Y. LeCun, "Efficient learning of sparse representations with an energy-based model," in *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*, pp. 1137–1144, 2006.

[57] R. M. Haralick, K. S. Shanmugam, and I. Dinstein, "Textural features for image classification," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 3, no. 6, pp. 610–621, 1973.

[58] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[59] I. J. Goodfellow, "NIPS 2016 tutorial: Generative adversarial networks," *CoRR*, vol. abs/1701.00160, 2017.

[60] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II*, pp. 694–711, 2016.

[61] A. Shocher, N. Cohen, and M. Irani, ""zero-shot" super-resolution using deep internal learning," *CoRR*, vol. abs/1712.06087, 2017.

[62] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[63] R. Mehrotra, K. R. Namuduri, and N. Ranganathan, “Gabor filter-based edge detection,” *Pattern Recognition*, vol. 25, no. 12, pp. 1479–1494, 1992.

[64] A. Ranjan and M. J. Black, “Optical flow estimation using a spatial pyramid network,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 2720–2729, 2017.

[65] A. Ranjan and M. J. Black, “Optical flow estimation using a spatial pyramid network,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 2720–2729, 2017.

[66] E. Shelhamer, J. Long, and T. Darrell, “Fully convolutional networks for semantic segmentation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, pp. 640–651, 2017.

[67] D. Sun, X. Yang, M. Liu, and J. Kautz, “Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume,” in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 8934–8943, 2018.

[68] T. Hui, X. Tang, and C. C. Loy, “Liteflownet: A lightweight convolutional neural network for optical flow estimation,” in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 8981–8989, 2018.

[69] Y. Huang, W. Wang, and L. Wang, “Bidirectional recurrent convolutional networks for multi-frame super-resolution,” in *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 235–243, 2015.

[70] Y. Jo, S. W. Oh, J. Kang, and S. J. Kim, “Deep video super-resolution network using dynamic upsampling filters without explicit motion compensation,” in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 3224–3232, 2018.

[71] D. Liu, Z. Wang, Y. Fan, X. Liu, Z. Wang, S. Chang, X. Wang, and T. S. Huang,

"Learning temporal dynamics for video super-resolution: A deep learning approach," *IEEE Trans. Image Processing*, vol. 27, no. 7, pp. 3432–3445, 2018.

[72] M. S. M. Sajjadi, R. Vemulapalli, and M. Brown, "Frame-recurrent video super-resolution," in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 6626–6634, 2018.

[73] R. Liao, X. Tao, R. Li, Z. Ma, and J. Jia, "Video super-resolution via deep draft-ensemble learning," in *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[74] Z. Ma, R. Liao, X. Tao, L. Xu, J. Jia, and E. Wu, "Handling motion blur in multi-frame super-resolution," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[75] C. Liu and D. Sun, "A bayesian approach to adaptive video super resolution," in *The 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011, Colorado Springs, CO, USA, 20-25 June 2011*, pp. 209–216, 2011.

[76] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

VITA

EDUCATION

Ph.D Student of Electrical and Electronics Engineering at Middle East Technical University 2014-present, Thesis Title: Double Network Superresolution

MSc. Student of Electrical and Electronics Engineering at Middle East Technical University 2011-2014, Thesis Title: Real-Time Single Frame Superresolution

Undergraduate Student of Electrical and Electronics Engineering at Middle East Technical University 2007-2011, Area: Telecommunications

EMPLOYMENT

Senior Electronics Design Engineer at ASELSAN 2019-present

Expert Electronics Design Engineer at ASELSAN 2015-2019

Electronics Design Engineer at ASELSAN 2011-2015

PUBLICATIONS

C. TARHAN, C. U. UNGAN, Telsiz Sualtı Akustik Haberleşme, SSTKON, ASELSAN 2011

C. TARHAN, G. BOZDAĞI AKAR, Real-Time Single Frame Superresolution, MSc Thesis at METU Library 2014

C. TARHAN, G. BOZDAĞI AKAR, Convolutional Neural Networks Analyzed via Inverse Problem Theory and Sparse Representations, IET Signal Processing Journal, Volume 13, Issue 2, April 2019, p. 215 – 223

C. TARHAN, G. BOZDAĞI AKAR, Context Based Double Network Superresolution, IEEE Transactions on Image Processing, submitted April 2019.

C. TARHAN, G. BOZDAĞI AKAR, Video Superresolution Using Detail Fusion Interpolator and Twin Network Superresolution, IEEE MLSP 2019, submitted May 2019