Fast and Efficient Edge Fusing Network Architectures For Accurate Single Image Super-Resolution

by

Debjoy Chowdhury
Bachelor of Science, Chittagong University of Engineering and Technology, 2010

A thesis
presented to Ryerson University
in partial fulfillment of
the requirements for the degree of
Master of Applied Science
in the Program of
Electrical and Computer Engineering

Toronto, Ontario, Canada, 2019

Author's Declaration For Electronic Submission Of A Thesis

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my thesis may be made electronically available to the public.

FAST AND EFFICIENT EDGE FUSING NETWORK ARCHITECTURES FOR ACCURATE SINGLE IMAGE SUPER-RESOLUTION

Debjoy Chowdhury
Master of Applied Science
Electrical and Computer Engineering
Ryerson University, 2019

Abstract

Recovering a High-Resolution (HR) image from a Low-Resolution (LR) image is the main concept of image Super-Resolution (SR). Convolution Neural Networks (CNN) are becoming widely adopted in many applications including generation of HR images from LR images. Although CNNs are widely used with great performance improvements, there is still much room for improvement. There has always been a trade-off between the number of parameters and performance enhancement. This thesis presents a novel convolutional neural network architecture for high scale image SR inspired by the DenseNet and ResNet architecture. In particular, modifications can be made to the convolutional layers in the network: stacking the features and reusing the weight layers to increase the receptive field. It is shown how this method can be used to expand the receptive field and performance of super-resolution networks, without increasing the number of trainable parameters and sacrificing the computation time. These modifications can easily be integrated into any convolutional neural network to improve the accuracy by efficient high-level feature extraction while reducing training time and parameter numbers. Proposed methods are especially effective for the challenging high scale SR due to edge and texture recovery through the expanded network receptive field. Experimental results show that the proposed model outperforms the state-of-the-art methods.

Acknowledgments

I would like to convey my gratitude to my supervisor, Dr. Androutsos for being the first to inspire my interest in the field of Image Processing and Computer Vision, and for always encouraging me to push for the best. Also, many thanks to my parents for their unwavering support throughout both my academic and personal life.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Problem Context

The total number of pixels or dots in an image represents the resolution of that image [1]. It is represented as two dimensions of the image - Width and height. For instance, having the resolution of 500×500 means the image has a width of 500 pixels and a height of 500 pixels. High Resolution (HR) image is that which has a greater number of pixels compared to the Low Resolution (LR) image which consists of fewer pixels. In the spatial domain, HR images contain more details and visually clearer and are spectacular, and hence highly desirable to be used in many image applications. In terms of frequency domain, HR images contain high frequency components and details like edges, corners and contours. LR images lack those details and have smooth areas which represents the low frequency components. Having more details improves the pictorial information for human interpretation (i.e., aesthetics) and helps with representation for automatic machine perception (i.e., computer vision) [1, 2].

There are many types of image acquisition devices, which are capable to produce HR images. However, this is not always possible due to the expense, availability or transportation of the devices. Furthermore, HR is dependent on the intended use of the image. Image SR is categorized into two - Single Image Super Resolution (SISR)and Multi Image Super Resolution (MISR) [2]. SISR aims to recover a HR image from single LR version. MISR reconstructs the HR from multiple images. This is an inverse problem focused on generating a HR image from a LR one. This problem is highly ill-posed due to large number of unknown variables in HR image compared to LR. The main problem lies in recovering high frequency details like edges, texture, blobs and contours, since they are lost from the Ground Truth (GT) HR image by degradation, hardware limitation or data loss,

(a) (b) (c) (d)

Figure 1.1: Example of SISR. (a) The HR image with resolution 2040×1356. (b) LR image at $\times 2$ scale. (c) LR image at $\times 4$ scale (d) LR image at $\times 8$ scale.

especially with high upscaling factors. The term "scale" or "upscaling factor" commonly used in SR is the ratio of HR pixels with corresponding LR. If the LR image has a size of 500×500 and the HR image has a size of 2000×2000, then the SR scale is said to be $\times 4$. This is illustrated in Figure 1.1 where examples of the HR image and corresponding LR images at $\times 2$, $\times 4$, and $\times 8$ scales are shown. As the upscaling factor goes higher, the more challenging the reconstruction becomes because less data availability to recover the HR image from its counterpart LR image. When downscaling an image by a factor of $\times 2$, half of data are lost in the vertical direction and half in the horizontal direction, and in total, four times the original data are lost to use for building the HR image. In the same way, for the factor of $\times 8$ with only having $\frac{1}{64}^{\text{th}}$ of the data makes it much more complicated and difficult to reconstruct higher frequency details. The motto of SISR process is to increase the high frequency components and details as well as remove the degradation which may be caused by the low-resolution hardware or while processing image for various purposes such as image communication. While recovering the image with a high upscaling factors, many fine image structures, details, and textures are often missing due to lack of sufficient input data. Most of the state-of-the-art works in SISR can perform substantially high reconstruction accuracy with lower scaling factors, however fails to achieve higher accuracy with higher factors. Currently, researchers are moving more towards working on the higher scales such as $\times 4$ and $\times 8$ with more complex architectures which will be discussed in Chapter 2.

Many different techniques were proposed in the last 35 years for SR started with Tsai and Huang's first work in 1984 [3]. In traditional approaches, SR requires multiple low-resolution im-

ages, prior knowledge and assumptions to recover the high-resolution image [4]. The problem lies in an inadequate number of low-resolution images, ill-conditioned registration, unknown blurring, and a non-unique solution. Several methods are being adopted to regularize the inversion of this problem. Moreover, if the magnification factor is large, the SR algorithm decays rapidly. As a result, the output is smoother and lacks important high frequency details, which is an inherent problem with bicubic interpolation method. Edge and gradient based models recover sharpness. There are also background/foreground descriptors, gradient profile prior and learning dictionary patch pairs as the old fashioned approach [5, 6]. Recently, deep CNNs with the ability to learn complex mappings in image transformation and high representational power, have become very popular in SISR. Some techniques that were used to improve classification accuracy are also utilized for increasing the SISR reconstruction accuracy by many researchers [7, 8, 9, 10, 11, 12]. The trend has thus been towards constantly adapting the latest techniques from image classification and then building deeper models with more layers and parameters. Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity index (SSIM) are used to measure the accuracy of the architecture.

Besides higher SR reconstruction accuracy, deeper models come with a few drawbacks. Firstly, deeper layers need very large number of parameters, leading to a larger memory bandwidth requirement. Recursion has been successfully used in past works [13, 14] to increase depth while mitigating memory consumption. However, minimizing memory consumption as well as maintaining state-of-the-art accuracy needs a higher number of recursions and hence many layers. Using such many layers leads to sacrifice of speed for accuracy, even with the minimal memory consumption. Secondly, training the network is much more complex due to both the large number of parameters and layers. Finally, it requires careful design of the learning rate schedule and gradient clipping [13, 14, 15].

1.2 Scope and Contributions of this Thesis

In this thesis, a new direction and discussion is unveiled to increase the receptive field of SR networks without increasing the network depth or parameter count, to have a more efficient use of parameters while focusing on the more challenging high upscaling factors. In particular, three

different methods are proposed to expand the receptive field with better edge and texture map:

- Reusing residual blocks, motivated by the DenseNet [12]
- Masking (pointwise multiplication of feature vectors) of an independent parallel net with the residual block, inspired by Channel Attention Net [16]
- Finally, infusing Sobel, Laplacian edges to the input features

By reusing the weights, the receptive field is expanded without increasing the parameters and with masking and including the edge maps, the high-level features are preserved. All those techniques can be applied to SR networks to increase reconstruction accuracy while maintaining the optimal layers and parameter count, and without major sacrifices in speed unlike some current state-of-the-art methods. Several different convolution arrangement schemes are also explored to find the best match. Proposed Edge Fusing Network (EFNet) achieves state-of-the-art accuracy in terms of PSNR and SSIM and is easy to train by using low memory.

1.3 Overview of this Thesis

The content of this thesis is structured as follows: Chapter 2 reviews the main developments in SR from a deep learning and computer vision perspective related to the proposed methods. Chapter 3 describes the proposed model explorations and methodology in detail. In Chapter 4, implementation details and experimental results on benchmark SISR image data sets are presented. Conclusion and future research are discussed in Chapter 5.

Chapter 2

Related Work

In this section, the history of SR research is discussed, started with the pre-deep learning era in section 2.1. This part includes the old works done for image SR like interpolation and sparse coding-based methods. Section 2.2 discusses the background of CNNs and the networks used to predict results in the computer vision field. Section 2.3 analyzes the state-of-the-art methods using deep learning to super resolve the images and their SWOT (strength, weakness, opportunity and threat) investigation.

2.1 Pre-Deep Learning Super-Resolution

2.1.1 First Formulation

Tsai et al. first introduced the problem to obtain the original image from its down sampled LR version in 1984 [3]. They used the dataset from Landsat satellites. They developed the equation in their model based on frequency domain representation of the image considering the shift property of Fourier Transform.

2.1.2 Recursive Least Squares

Kim et al. modified the model introduced by Tsai et al. by incorporating noise and blur [17]. Their model solved the problem by being more computationally efficient. However, problems arose as they didn't introduce the ill-posedness of the problem. To overcome that, they extended their work later by the total least square method [18]. These frequency domain models were simple but

were not completely free from drawbacks; introduction of prior knowledge is tough and sometimes inappropriate. To address the shortcomings, spatial domain methods are initiated.

2.1.3 Interpolation Method

Interpolation technique predicts the missing pixel by means of weighted average of all neighboring pixels. It has played a vital role in image applications such as zooming, enhancement and restoration. Some popular methods are: nearest-neighbor, weighted average, least-squares plane fitting, iterative back-projection, bi-linear, bi-cubic etc. In nearest-neighbour method, the nearest value closest to the missing pixel is copied to interpolate. Iterative Back-Projection (IBP) approaches the HR image by iteratively back projecting the difference between the input LR image and the stimulated LR image. Stimulated LR image is generated by down sampling the initial HR image. Again, the initial HR image can be generated by decimating the initial LR image [19]. Bi-linear interpolation provides equal weighing of the diagonally closest four pixels [1]. Bi-cubic interpolation in contrast, considers closest sixteen pixels around the missing pixel.

2.1.4 Edge-Preserving Method

For better edge preservation in an image, edge-guided interpolation was implemented [20, 21]. Here, different neighboring pixels have different weights given and those non-linear weights are calculated by statistical structure. This technique interpolates a pixel by means of two observation sets in two orthogonal directions. Using the two estimations, edge sharpness can be preserved hence the upscaled image is not smoothed.

The basic principle of edge-guided interpolation is shown in Figure 2.1. In Figure 2.1a, the black pixels are the available samples from the LR image, and the unfilled white pixels are missing ones to be determined in the HR space. First, the odd indexed pixel values (red circle) are computed with the weighted average of four black pixels in the 45° and 135° direction. Next, the even indexed pixels are calculated by the weighted average of two black pixels and two red pixels in the 0° and 90° direction, respectively shown in Figure 2.1b. The sliding window technique is used here to compute every pixel. In the sliding window, if the covariance of the pixels is positive, there is no

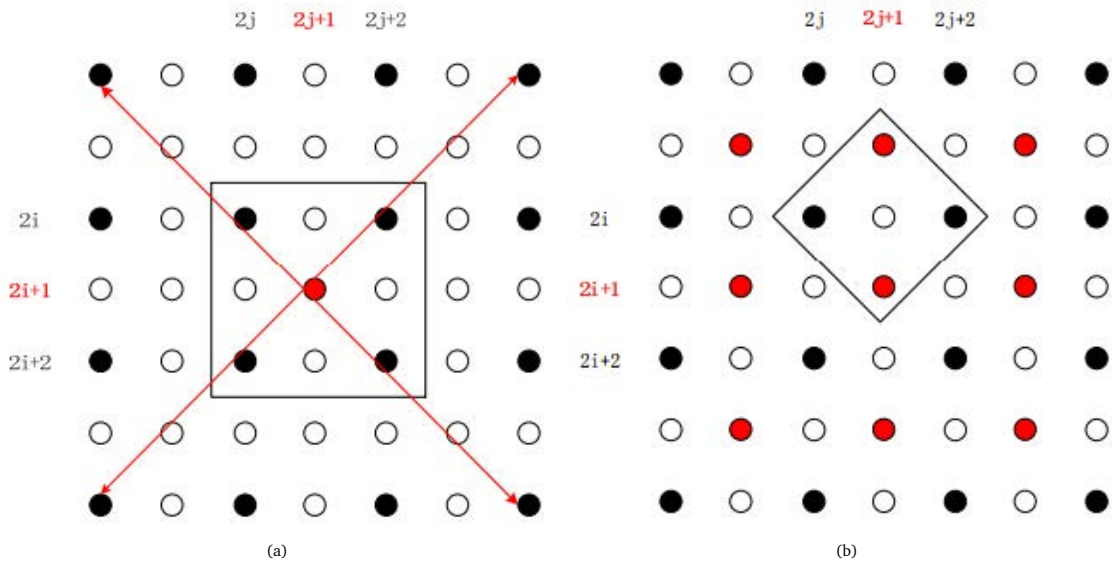


Figure 2.1: Illustrating the basic idea behind interpolation based methods. (a) Step 1 fill in the odd-indexed pixels using a weighted average of their four diagonal neighbors. (b) Step 2, fill in the even-indexed pixels using a weighted average of their two vertical and two horizontal neighbors.

edge and all the pixels are then given equal weighs. However, negative covariance represents a probable edge and in that case some pixels are given more emphasis than others. The problem with this method is a lack of high frequency details such as textures, which the covariance can't extract.

2.1.5 Sparse Coding Method

Sparse coding-based approaches address this limitation of Edge Preserving method. Here HR patches are represented as a sparse linear combination of dictionaries trained from external databases or recovered from self similar properties in the LR image itself at different locations [4, 22, 23, 24]. To ensure the same sparse representations, two dictionaries are learned simultaneously by concatenating them with proper normalization. The sparse signal is then learned from the training dataset which contains HR and LR patch pairs. This similarity in sparse representation is applied at the time of training.

The SR technique with these methods is shown in Figure 2.2. Here sparse representation of two coupled dictionaries for low resolution(X_l) and high resolution(X_h) is involved. For increasing the resolution of the image, the LR image patch is converted to sparse representation and matched to

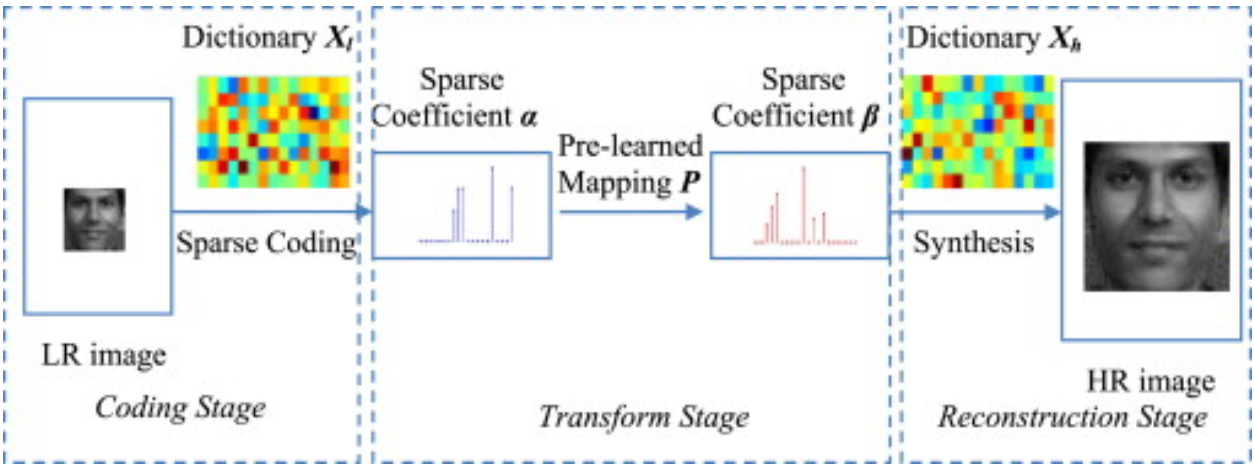


Figure 2.2: Illustrating the process of performing SISR using sparse signal dictionaries [25].

the closest one in the learned dictionary. According to the LR-to-HR mapping, the HR patch from X_h is thus recovered using the corresponding X_l. From the dictionary, the sparse representation selects the most relative patches with respect to the given low resolution and hence, leads to a qualitative and quantitative superior performance. These methods can generate sharper edges, clearer texture and robust to noise, while interpolation-based methods can't handle noise and SR simultaneously.

2.2 Convolutional Neural Networks

2.2.1 Basics of Convolutional Neural Networks

A Neural Network is a combination of algorithms which imitates the operation of the human brain to find the statistical relation in a given dataset. Neural Networks were not following efficient training since the introduction of Backpropagation algorithm in the mid 1980s [26]. Then in the 1990s, with the inclusion of Kernel methods, Neural Net gained fame and respect among the researchers. A kernel function is a computationally tractable operation that maps any two points in an initial space to the distance between these points in the target representation space, completely bypassing the explicit computation of the new representation [26].

CNN consists of several sets of filters/kernels whose weights are to be determined by a continuous training process [27]. A generic example is shown in Figure 2.3. The kernel has the dimension

of 3×3. With the weighted sum of first 3×3 pixels in the input image, it calculates the first output pixel. The kernel then slides through the rows and columns of the input, computes the next output pixels. Several stacks of such kernels make one layer and several layers lead to the complete CNN network. It is worth to mention that, all the kernels in a CNN network are unique. The output pixel is a function of weights and pixels of the previous layer and may be mathematically expressed as

$$z = \sum_{n} w_{\mathrm{n}} x_{\mathrm{n}}, \tag{2.1}$$

where w_{n} is the convolution kernel weight with corresponding input pixel x_{n}, and together they produce the single output pixel z. This output of the weighted average is called feature map. A kernel size of 3×3 has 9 parameters count and the stride is represented by how many slides a convolution kernel will make to compute the next output pixel. For example, stride of 2 means the filter will slide by 2 pixels in the horizontal and vertical direction to calculate the succeeding output pixel. It should be mentioned here that for a stride of 2, the size/resolution of the output is half of the input. For a stride of 4, the output turns quarter.

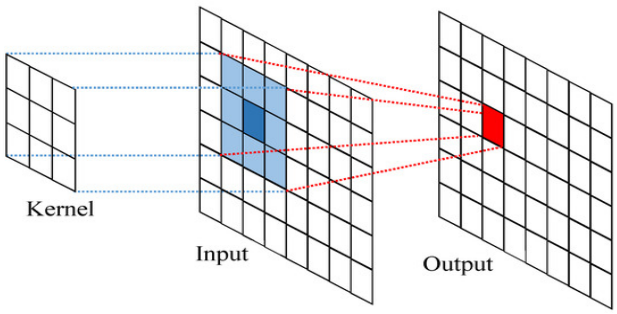


Figure 2.3: Illustrating the convolution operation.

Besides convolution layers, there are two more important parts in CNNs: activations and sub-sampling/pooling functions. The activation function does the non-linear mapping of the input to the output. Non-linear means that the output cannot be reproduced from a linear combination of the inputs. Without these activation functions, regardless of the number of layers present in a CNN, it will behave as a single layer model. Thus, this function enables CNNs to learn non-linear transformations by directly performing non-linear operations [27]. One important aspect to consider is that, the activation function must be differentiable, otherwise gradient-based methods can't be optimized. Several activation functions are Softmax, Elu, Selu, Tanh, Sigmoid,

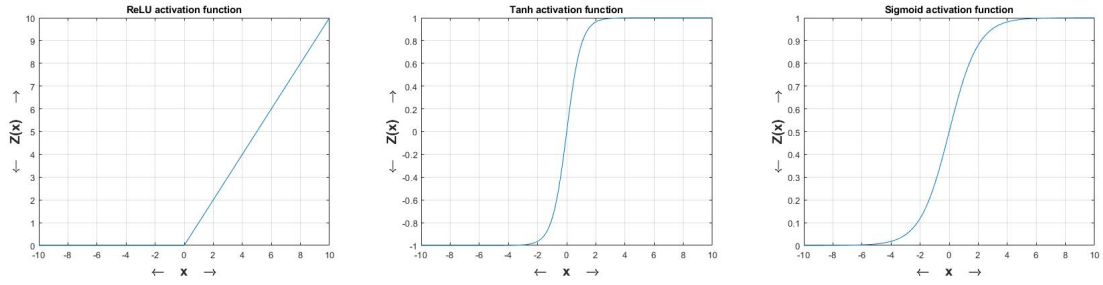
Hard Sigmoid, and the most commonly used is Rectified Linear Unit (ReLU) [28].

$$z(x) = max(0, x), \tag{2.2}$$

where x is the input value i.e., the input pixel and $z(x)$ is the output pixel. From the equation it is noticeable that, ReLU activates when the input is positive and disregards the negative values. ReLU activation function is shown in Figure 2.4a. Yann et al. used tanh or sigmoid [29], whereas ReLU is used most of the time with performance improvement [7, 8, 10, 11, 28]. Tanh function shown in Figure 2.4b ranges from -1 to 1, is a s-shaped function. The Sigmoid is an exponential function and can be expressed as

$$z(x) = \frac{1}{1 + e^{-x}}, \tag{2.3}$$

where x is the input value i.e., the input pixel and $z(x)$ is the output pixel. Sigmoid is highly preferred in the cases when the output generates a probability. In Figure 2.4c the graph is shown. Sigmoid function is slightly modified linear piece-wisely in the Hard Sigmoid function. It has a slope 0.2 for $-2.5 < x < 2.5$ and the slope is 0 elsewhere. So unlike Sigmoid, this function is a linear approximation and therefore, easier to calculate.



(a) (b) (c)

Figure 2.4: Illustrating Activation Functions. (a) ReLU activation function. (b) Tanh activation function. (c) Sigmoid activation function.

The pooling function, that is used to reduce the feature space of the input, can be assumed as a filter which slides through the input and provides a scalar output based on the equation given. The most widely used sub-sampling is max-pooling. This function picks the maximum value from the input within a given window. Max-pooling is very efficient for solving the classification problem as it picks the highest value from a feature map and those values represent the most important features.

CNNs learn the parameters to transform the input to output in a most accurate way. The weight matrix of the convolution filters should be as optimal as possible to perform an end-to-end mapping. Generally, these weight values are initially set manually by some initialization technique. He et al. introduced a technique for random weight initialization which is very popular in the state-of-the-art methods [30]. Once the weights have been initialized, in the next stage CNN learns the optimum value by means of *Back-Propagation* algorithm [27]. This technique minimizes the loss by finding the global minimum of the high dimensional loss surface of a CNN. This algorithm is closely connected to the predefined loss function. Loss is defined as a function, that evaluates the predicted output with respect to a given input, which is called ground truth or sometimes labeled data. The goal is to view the loss as a multivariable function of all the parameters. *Back-Propagation* updates the weight matrix by applying the chain rule. Chain rule computes the derivative of a composite function. If *Forward-Propagation* is a series of nested equations, chain rule finds the loss with respect to any variable in that nested equation. Thus, using the greedy algorithm, *Back-Propagation* propagates error from loss function to first layer and updates the weights accordingly.

For CNNs, *Back-Propagation* is based on gradient descent/steepest descent process on convex optimization, which iteratively optimizes the network to find the minimum of a function [31]. The gradient descent algorithm is mathematically expressed as

$$w_{\mathrm{n}+1} = w_{\mathrm{n}} - \gamma \Delta \mathcal{L}(w_{\mathrm{n}}), \tag{2.4}$$

where loss function is denoted as $\mathcal{L}$, the network weights on the nth iteration as w_{n}, and γ is the learning rate. The first order partial derivative of the loss function with respect to the weight matrix is the gradient $\gamma \Delta \mathcal{L}(w_{\mathrm{n}})$, which is subtracted from the previous weight value to update the next weight. γ determines how fast the convergence will be. Lower γ results in higher number of iterations and hence slows down the training process and raises "Vanishing Gradient" problem. When the optimization can't progress from a certain minimum point due to too low gradient, it is called "Vanishing Gradient". However, high γ provides faster convergence also with a chance to overshoot the minimum. When the gradient becomes too large, it leads to an unstable network. This phenomenon is called "Exploding Gradient" problem. To prevent these problems "Gradient Clipping" is used in the state-of-the-art methods. In "Gradient Clipping" algorithm, a threshold of

gradient is set and when the gradient exceeds the threshold, it is scaled down to the standard. One important aspect of good model architecture is tuning the γ variable. It is important to mention that, for convex error function, convergence is done easily. However, in reality, condition of convexity doesn't hold and hence convergence is not guaranteed. Moreover, sometimes the convergence may become stuck to a local minimum not a global minimum. Despite this, is has been effectively used to solve computer vision and image processing tasks.

2.2.2 State-of-the-art Techniques in Convolutional Neural Networks

Presently CNNs draw inspiration from the techniques applied in image classification problems and utilize them to solve other computer vision problems. Some fundamental techniques, which are being used as baseline for many cutting edge methods in SISR domain, will be discussed here.

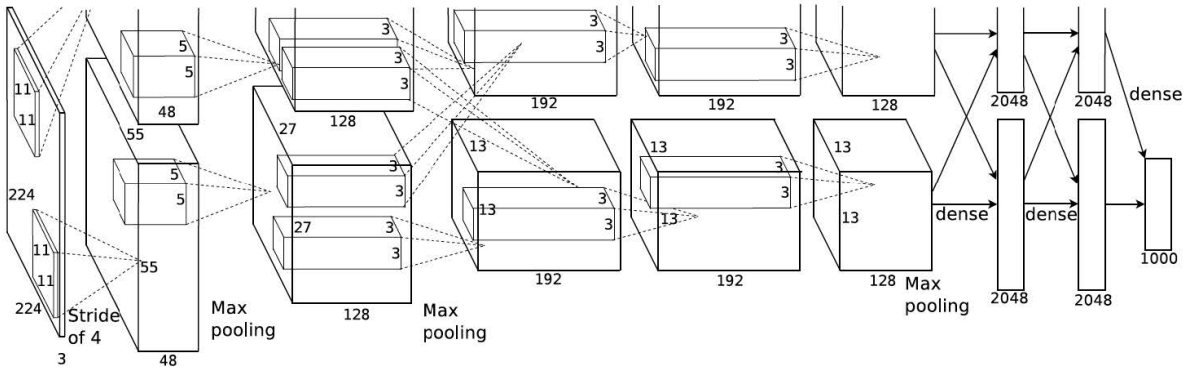


Figure 2.5: Illustrating The AlexNet architecture [7].

AlexNet architecture by Krizhevsky et al. was the first to utilize a deep network of CNN for large scale image classification. They trained their model using ImageNet challenge dataset [32, 33], which contains a large amount of labeled data. Due to insufficient memory as a result of higher number of parameters, they implemented the training on two parallel GPUs. They also replaced tanh activation function with ReLU for the non-linearity activation and since then ReLU has become a default activation function. They expanded the training by following data augmentation techniques, e.g., image translations, horizontal reflections, and mean subtraction, which also resulted in improved performance. In Figure 2.5 AlexNet architecture is shown.

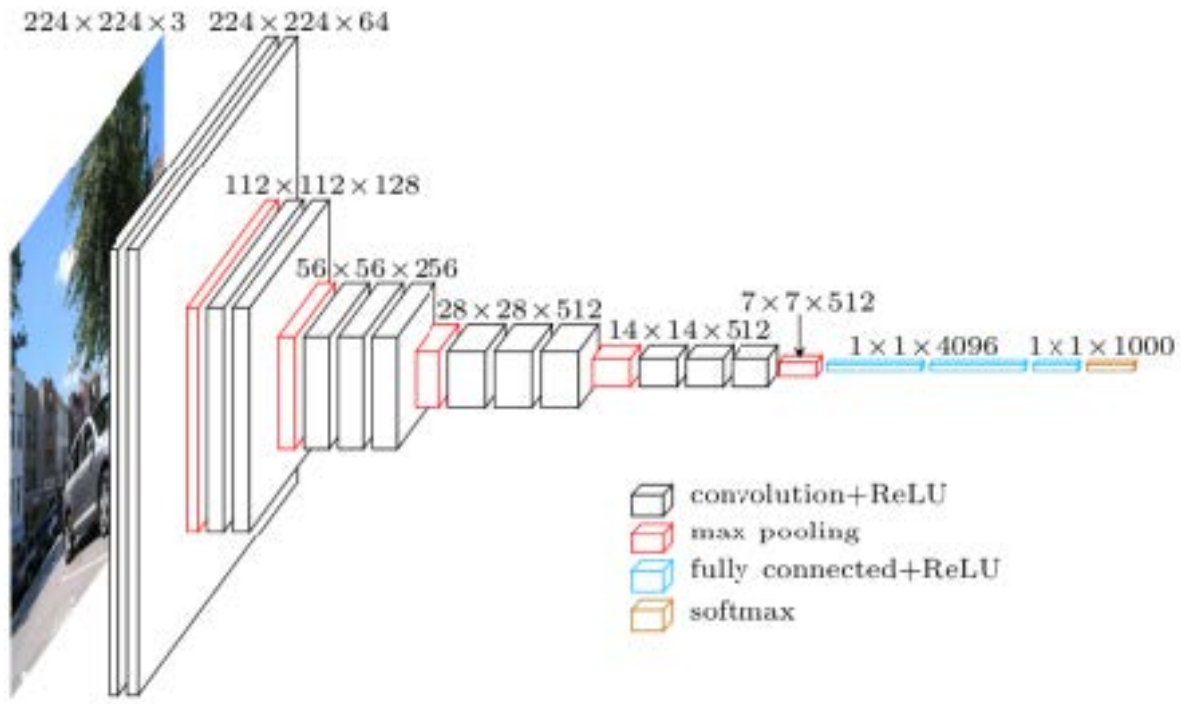


Figure 2.6: Illustrating The VGGNet architecture [34].

Simonyan et al. [8] extended the Alexnet by adding more layers named VGG (Visual Geometry Group) network (Figure 2.6). The largest VGG net has 19 layers, approximately 4 times more than Alexnet. Moreover, they significantly reduced the number of parameters by using kernels of size 3×3 as compared to AlexNet's first two layers, which are using 11×11 and 5×5. They proposed a novel theory of stacking several small size kernels which is equivalent to using a bigger size kernel. For example, two successive 3×3 convolution provides the same receptive field of one 5×5 convolution. Similarly, a single 7×7 convolution is equivalent to three successive 3×3 convolutions. This technique benefits the network computationally and also reduces the requirement of large memory requirement for deeper layers. Besides, there is a chance to include more ReLU non-linearity between the layers and hence, it makes the decision boundary more robust to outliers.

Both AlexNet and VGGNet established the concept that, deeper layers tend to produce better result. However, He et al. [10] showed that, deeper layer is not always better. Sometimes they aggravate the performance of the network too. It happens due to the "Vanishing Gradient" problem (explained in Subsection 2.2.1). In practice, due to weight decay, sometimes weight matrix of filters at a certain layer goes to zero. Because of the"Vanishing Gradient" problem, the weight

matrix doesn't update. As a result, the filters don't pass any feature vectors to the next layers. So, the following layers can't generate any results rather than increasing the memory requirement. To address this problem, they introduced skip connection in residual learning. A residual block is illustrated in Figure 2.7. The additive skip connection helps the deeper layers to have the access to the previous layers. Thus it propagates information throughout the network, and eliminates the chance of any layer having an output zero and hence solves the vanishing gradient problem [10,35]. Training is also benefited from skip connection for more efficient back-propagation. Such a way, ResNet first implemented deeper network having more than 100 layers.

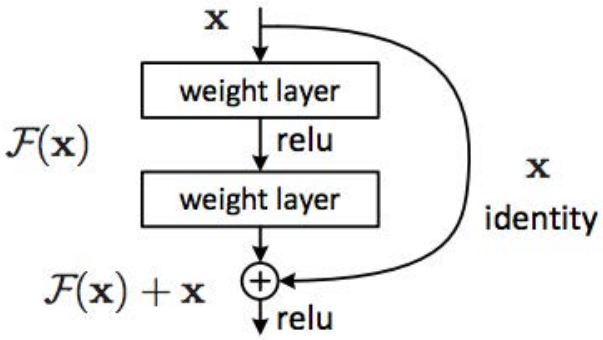


Figure 2.7: Illustrating Residual Block from ResNet [10].

Huang et al. introduced dense networks which generated substantially better results than ResNet [12]. They substituted the addition function in ResNet with the concatenation of features from the previous layers, so that, all the layers are connected to each other in a feed forward fashion. Feature maps from all the preceding layers are fed into each layer. DenseNet helps to reuse the features and propagates the features more efficiently. One complication is the growth rate becomes very high at a certain point and overshoots the allocated memory. To overcome this issue, bottleneck layer (1×1 convolution) is applied in the transition layer between two adjacent blocks. 1×1 convolution also performs weighted learned addition of feature maps, compared to the direct addition in ResNet. This enables the model to learn the features which have more priority over others. DenseNet successfully outperformed the previous state-of-the-art methods with less computations. A dense block is illustrated in Figure 2.8.

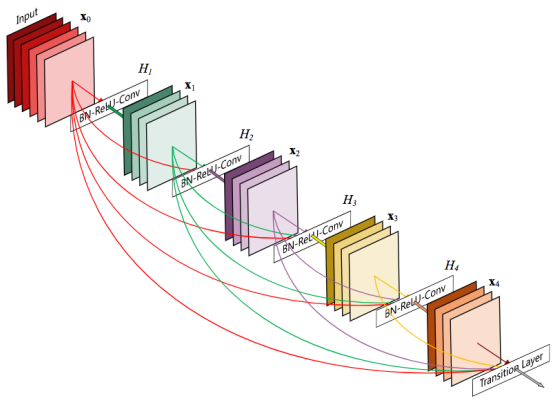


Figure 2.8: Illustrating Dense Block from DenseNet [12].

2.3 Deep Networks for Super-Resolution

2.3.1 Primary CNNs for Super-Resolution

Applying CNNs to super resolve image was first introduced by Dong et al [36], which directly learns an end-to-end mapping between low and high-resolution images, named Super-Resolution Convolutional Neural Network (SRCNN) [Figure 2.9]. They used the simplest architecture having only three layers with ReLU activation. Inspired by optimization, the convolution layers represent the patch extraction and aggregation. In the pre-processing part, LR image is first upscaled to a desired size by means of bicubic interpolation. Then, three operations are performed for mapping purposes. First, the overlapped patch is extracted from the image. Second, each high dimensional vector is mapped non-linearly on to another high dimensional vector. Finally, the mapped vectors are aggregated to form the HR image. The operations are named as Patch extraction and representation, Non-linear mapping and Reconstruction respectively.

They experimented many parameters such as filter size, number of train data, train on different color channels, and depth of the network. They showed increasing the filter size provides slightly better performance, while sacrifice the speed. They evaluated the result by training the model on different color channels and found RGB training patches provides the best result because of high cross correlation among the channels. They also showed that, increasing number of training patch results in improved performance. Their only drawback was utilizing the depth of the network. Increasing the depth of the network made the model performed worse. They came to a decision

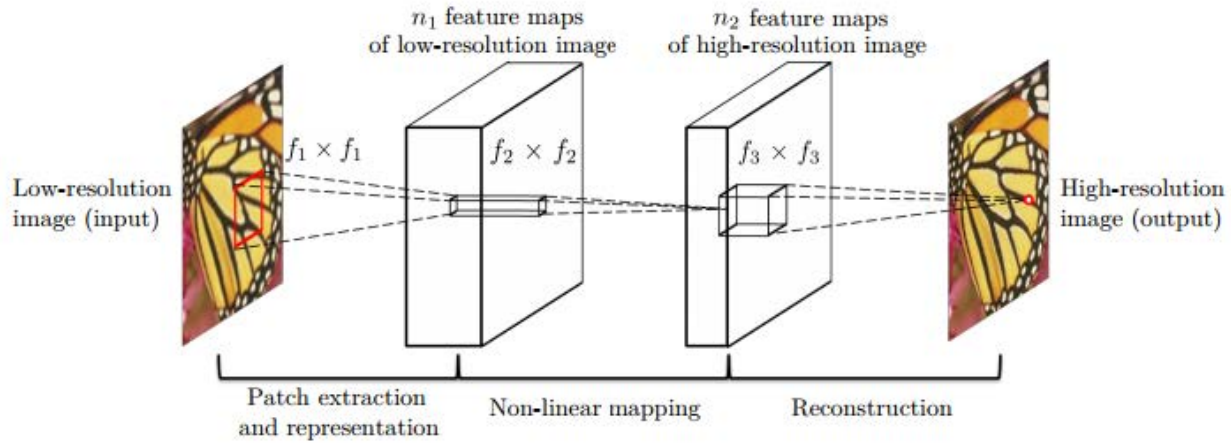


Figure 2.9: Illustrating SRCNN architecture [36].

that; deeper network tends to perform poorly. Later, this problem was resolved by the ResNet [10]. They have used Mean Squared Error as the loss function which can be expressed mathematically as follows

$$\mathcal{L} = \frac{\sum_{x=1}^{W} \sum_{y=1}^{H} (|Y_{\mathrm{i,j}} - X_{\mathrm{i,j}}|)^2}{WH}, \tag{2.5}$$

where Y is the ground truth HR image with width W and height H, X is the bicubic upscaled LR image.

Despite having some drawbacks, their method started a new era in the SR field which outperformed the previous state-of-the-art methods in terms of PSNR and SSIM.

2.3.2 Further Improved Deeper Networks

In SRCNN, stacking more layers produced worse results. To mitigate this issue, Kim et al. came up with the Very Deep Super-Resolution network (VDSR) network, which worked based on the ResNet concept [10]. Instead of learning unreferenced functions, residual was learned according to the input to make optimization simple. Residual is the difference between the output and input image. The equation can be written as

$$H(x) = F(x) + x, \tag{2.6}$$

where $H(x)$ is the output, $F(x)$ is the residual function to learn and x is the input. It is easier to learn the residual for mapping the input-output in a feedforward neural network. The residuals are also called "shortcut connections" as they skip several layers. Shortcut connections simply perform identity mapping. Identity shortcut connections add neither extra parameters nor computational complexity. End-to-end training was possible using the simplest back-propagation algorithm. So, for SR, the network first takes an interpolated (usually bi-linear) LR image (to the desired size) and predicts the residual. Once predicted, the residual is then added back to the input LR image to give the final HR image. The block diagram of VDSR is shown in Figure 2.10.

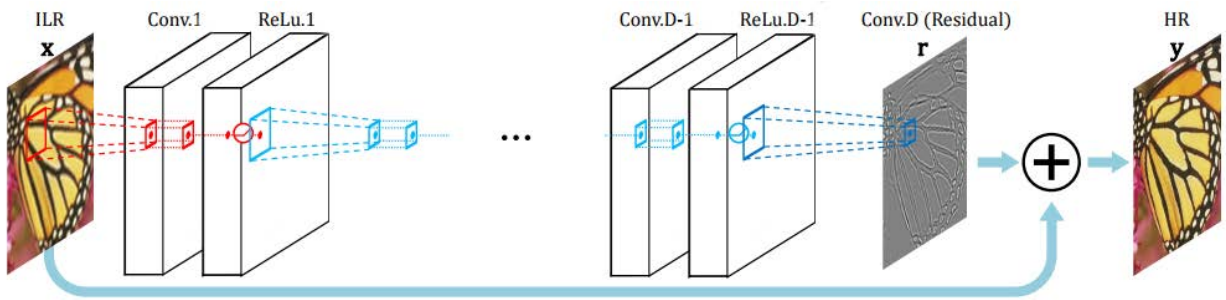


Figure 2.10: The VDSR architecture [14].

One thing worth noting is that, He et al. used a very high learning rate ($\gamma = 0.1$), which helped the network to converge fast. However, it can cause exploding gradient problem (explained in Subsection 2.2.1). To address this problem, the authors used gradient clipping in the optimization algorithm which prevents the gradient from going beyond a certain range. Moreover, they trained their network with multiple scales of data ($\times 2, \times 3, \times 4$) compared to the single scaling factor at a time in SRCNN. They used the same Mean Squared Error (MSE) loss function as in SRCNN. Their methodology showed significant improvement over Dong et al. technique. However, the problem of large amount of memory requirement for deeper networks was not completely solved.

Kim et al. proposed the Deeply Recursive Convolutional Network (DRCN) model [15] with a very deep recursive layer (up to 16 recursions) without introducing new parameters for additional convolutions which mitigates the drawbacks of VDSR. The general model consists of three sub-networks: embedding, inference and reconstruction networks (Figure 2.11). The embedding net maps the given image to a feature vector. The inference net analyzes the feature map from interference net by a single recursive layer. Each recursion repeats same convolution weights

followed by a rectified linear unit. This recursion can be applied multiple times to increase the network depth. Finally, in the reconstruction phase, feature maps from the inference net are fed into the reconstruction net to generate the output image.

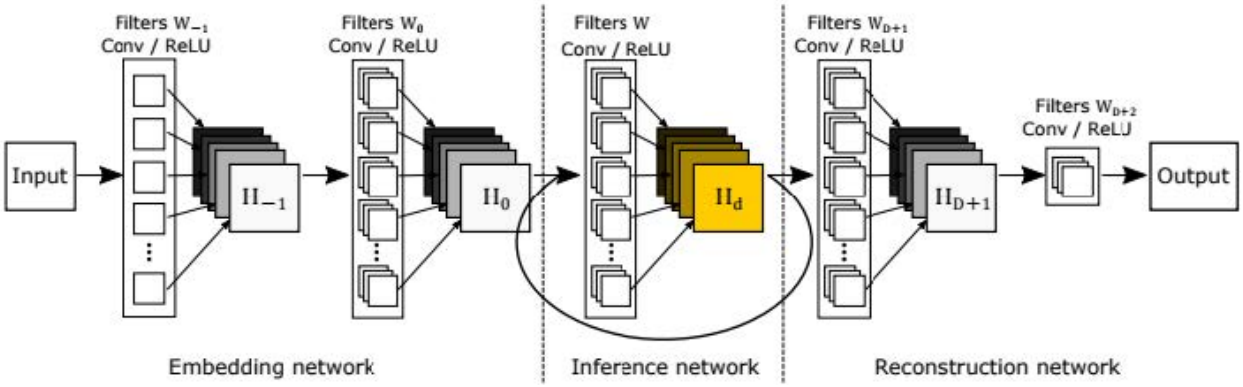


Figure 2.11: The DRCN architecture [15].

The authors also proposed two extensions: recursive-supervision and skip-connection. Recursive-supervision model utilizes every feature map produced by the recursive layers to reconstruct the HR image. These all predictions from the reconstruction net are simultaneously supervised during training to produce the final image. All those features from each layer are summed by weights, which are also learned during training. To maintain a high correlation between input and output image, skip-connections are also introduced from the input to the reconstruction net. With the same MSE loss function DRCN outperformed VDSR both visually and statistically.

2.4 Fast Super-Resolution Processing

Earlier methods used the bicubic upscaled image in the data pre-processing part. Due to processing of the higher resolution image, those methods were computationally inefficient by affecting the speed. In Fast Super- Resolution Convolutional Neural Network (FSRCNN) model [37], the authors discarded the unnecessary bicubic upscaled image and fed in the lower sized LR image directly in the architecture. That helped the network to train faster since less computation was needed for the lower size feature vector. In Figure 2.12 the schematic diagram of FSRCNN model is shown. After passing through several convolution layers, the feature maps are upscaled through strided transposed convolution [38] at the last layer to match the GT size. Transposed convolution is

sometimes named as deconvolution. It performs the upscaling through convolution technique. Overall, this type of architecture highly boosts the speed and the margin is prominent for higher scaling factors as higher scaling creates proportionately lower image size.

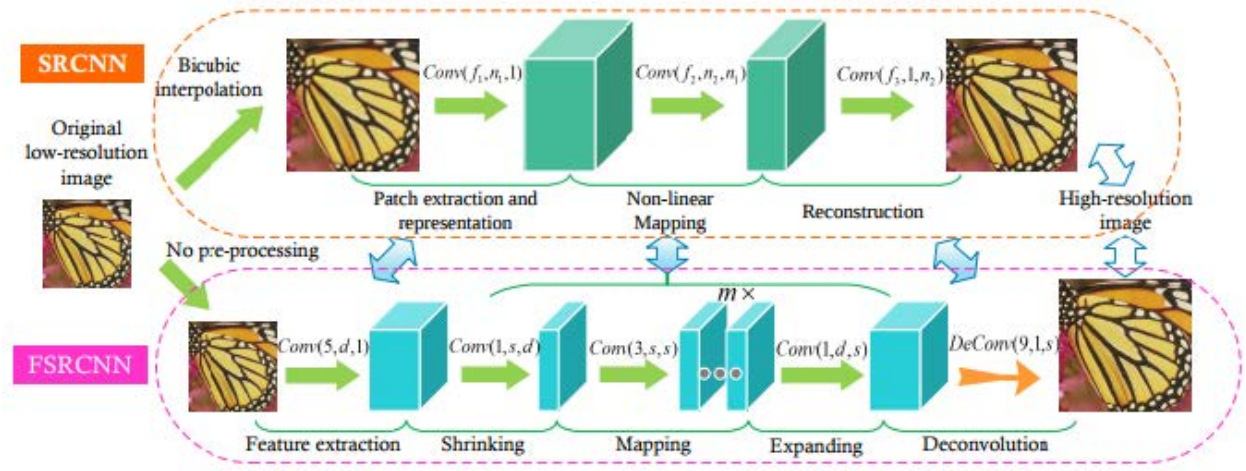


Figure 2.12: The FSRCNN architecture [37].

Another upscaling technique is proposed by Shi et al. in the Efficient Sub-Pixel Convolutional Network (ESPCN) [39]. Like the FSRCNN, it also uses the LR space image as the input but use sub-pixel convolution at the end for upscaling. It is almost the same as the deconvolution, except a shuffling is done through the channels at the final stage. Two convolutions are followed by a reshaping function of preceding channels. Unlike deconvolution, which puts zeros in between pixels, it computes more convolutions in lower resolution and resize the resulting map into an upscaled image [40]. It also speeds up the process than the traditional deconvolution technique. An illustration of sub-pixel convolution is given in Figure 2.13. This process was also followed in [41].

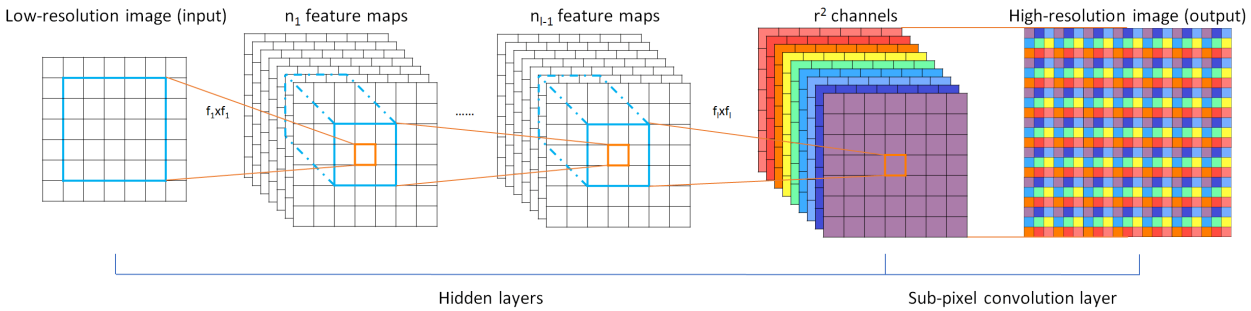


Figure 2.13: An illustration of sub-pixel convolution in ESCPN framework where r denotes the upscaling ratio [39].

Lai et al. combined the concept of VDSR and FSRCNN in Laplacian Pyramid Super-Resolution Network (LapSRN) by progressively assemble residuals in sub-bands [42] which is shown in Figure 2.14. The model has two branches: (1) feature extraction and (2) image reconstruction. Like a pyramid the feature extraction part consists of several levels. Each level (also called sub-network) consists of cascade of convolution layers to extract feature maps followed by a transposed convolution layer for up sampling the feature maps. Each sub-network takes the downscaled LR image as input and predicts the residuals progressively. There is also a skip connection added at the end of every sub-network like in the VDSR. Then, in the reconstruction, a convolution layer is used to predict the sub-band residuals from each level. Finally, the residuals are added to the input to efficiently reconstruct the HR image. LapSRN can perform progressive upscaling for a factor of 2. Thus, in one feed-forward pass the network generates multi-scale predictions of upscaling factors $\times 2, \times 4, \times 8$. Charbonnier loss function [43] is used to train the proposed LapSRN, which is robust to outliers than MSE or the Mean Absolute Error (MAE). Charbonnier loss function can be expressed mathematically as:

$$\mathcal{L} = \frac{\sum_{x=1}^{W} \sum_{y=1}^{H} \rho.(|Y_{\mathrm{i,j}} - X_{\mathrm{i,j}}|)^2}{WH}, \tag{2.7}$$

where $\rho = \sqrt{X^2{}_{\mathrm{i,j}} + \epsilon^2}$. The total loss is the sum of all individual sub-network loss.

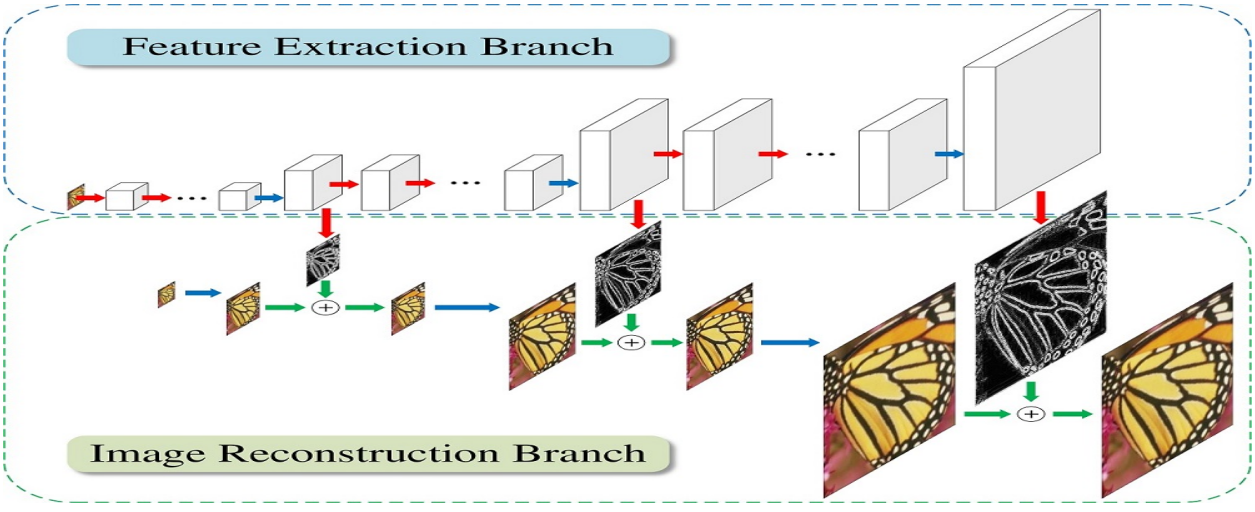


Figure 2.14: The Laplacian Super-Resolution Network (LapSRN) architecture [42].

Although having good PSNR and SSIM, the previous methods lack to recover high frequency details from LR images and hence provides smoother images. An edge guided CNN was introduced

by Yang et al. [44] is shown in Figure 2.15. The Deep Edge Guided Recurrent Residual Learning (DEGREE) network jointly infer the edge map and the recurrent residual learning to reconstruct the image. At the first stage, the prior edge map is extracted from the LR image by applying an off-the-shelf edge detector like Sobel filter. Then, the recurrent layers are provided with both the LR image and LR edge input. Unlike the previous models, this network predicts the edge map beside the feature map in the penultimate layer and use them both to predict the residual output which is mentioned as the high frequency component by the authors. The high frequency and low frequency components (the bicubic upscaled image in this case) are then added together to reconstruct the final image. The loss function is composed of the MSE based edge loss and normal reconstruction loss. DEGREE network has shown better performance both statistically and visually.

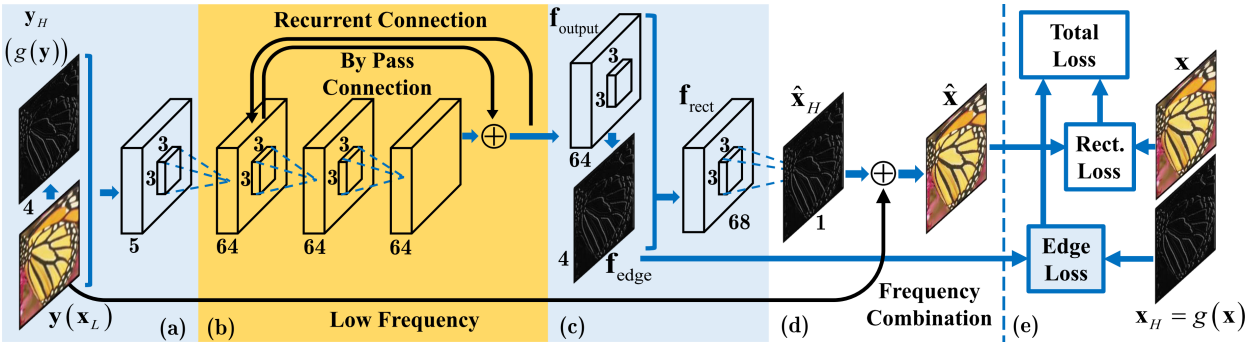


Figure 2.15: The framework of the DEGREE network for image SR [44].

Another similar type of network was surfaced by Liu et al. in the Attention based approach [45]. They adopted the same concept of extracting high-level features for better visual effect. It may be seen as combination of two different networks:

- Feature Reconstruction Network-Predicts the low-level features
- Attention Producing Network- Generates high-level features

Then outputs of both networks are multiplied point wise to compute the final residual edge map, which is finally added to the LR image in the HR space i.e., the bicubic upscaled one. As the Feature Reconstruction Network, the authors used their own DenseRes block which takes the downscaled image as input and at the end upscale features by means of sub-pixel convolution [39]. However, as Attention Producing Network, UNet was used, followed by a sigmoid activation function to create the mask, which enhances the features created by the other network. Besides PSNR and SSIM

development, marginal improvement was achieved through this process to recover the edge and texture like high-level features. The basic building block of Attention based approach is shown in Figure 2.16.

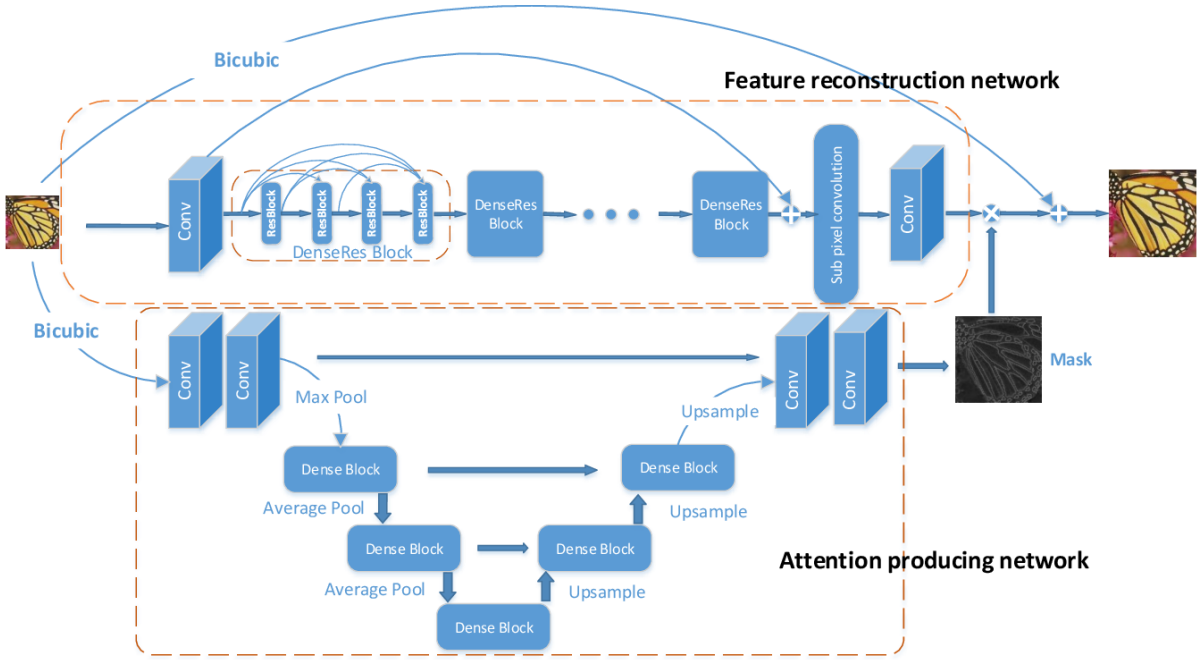


Figure 2.16: The framework of the Attention based approach for image SR [45].

2.5 Recent Development for Super-Resolution

Recently, further complex networks have been used to solve the SR problem. One of them is the Generative Adversarial Networks (GAN) [46]. A GAN is a combination of two parts: Generative part named as Generator and Adversarial part which is called Discriminator. The Generator network learns the distribution of a certain class based on certain label and generates the features. The features produced by the Generator are then passed to the Discriminator network, which detects them as true or false. It is a process, where Generator creates fake data and tries to fool the counterpart, where Adversarial part tries to catch the false data. Through such continuous training, the model becomes and expert to entirely fool the Discriminator and hence the output goes close to the GT. Ledig et al. used GAN operation with Super-Resolution Residual Network (SRResNet) as Generator [Figure 2.17]. Their Generator predicted the HR image, and the Discriminator learned to distinguish between the real HR image and the predicted one. Likewise, in [37], the input was

downscaled LR image, and sub-pixel convolution was applied at the end to upscale the features. They also adopted local residual [10, 11] and global residual [14] learning at the same time.

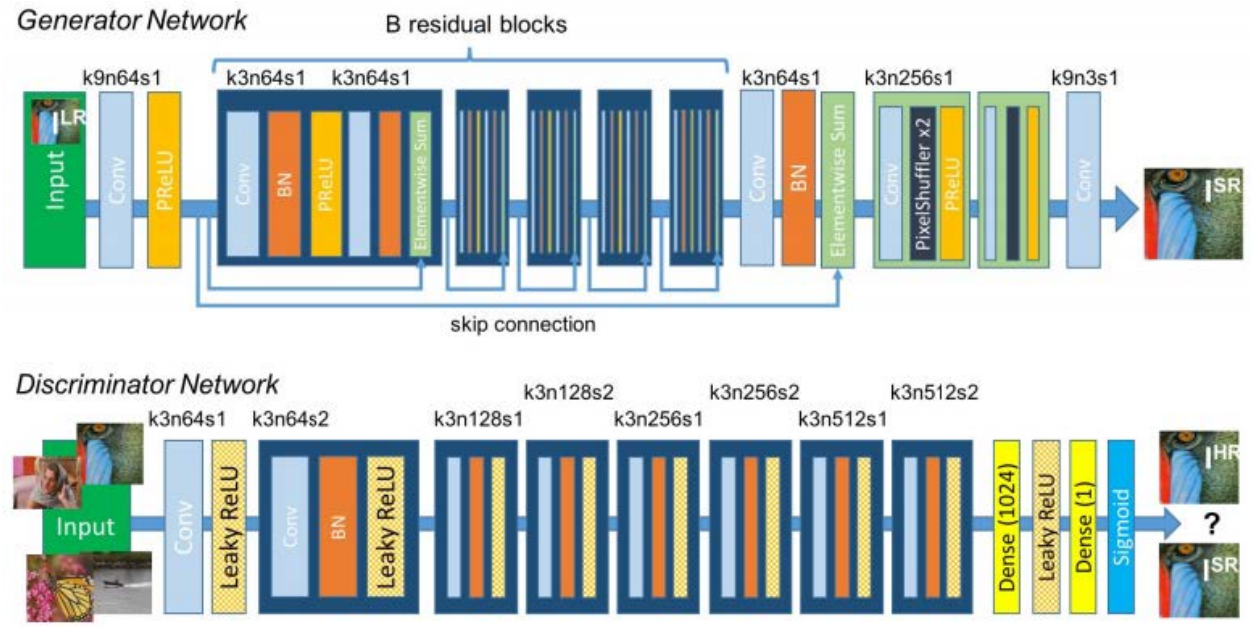


Figure 2.17: The SRResNet architecture with GAN [46].

Deep Recursive Residual Network (DRRN) by Tai et al. consisting of a very deep CNN model (up to 52 convolutional layers) use recursive learning to control the model parameters while increasing the depth [13]. DRRN has two key parameters: recursive block and the residual unit in each recursive block. In Figure 2.18, inside each recursive block, both the green and red convolution layers share the same weights. However, the weights are different in each recursive block. There is a local skip connection/identity path in each residual unit too. As a result, there are multiple paths between the input and output of the recursive block. The residual paths help to learn highly complex features and the identity paths help gradient flow during training. Several recursive blocks are stacked, followed by a convolution layer to reconstruct the residual between the LR and HR images. The residual image is then added to the global identity mapping from the input LR image. Such connections allow the DRRN model to have high depth with top performance while keeping the number of parameters very low. DRRN uses the bicubic upscaled image as input, which takes longer processing time, and a major drawback of this method. Each layer takes a higher number of feature maps unlike most previous works [14, 15, 36, 37, 39, 42, 46], which is also a reason for slowing down the speed.

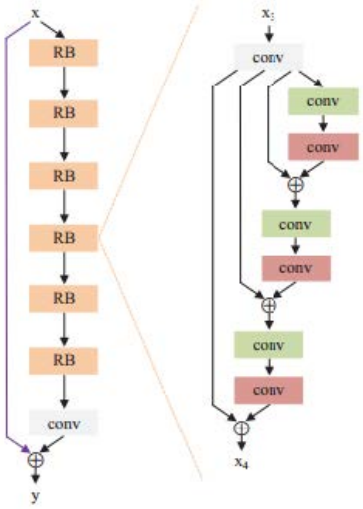


Figure 2.18: A recursive residual block used in the DRRN architecture [13].

Establishment of Dense Net motivated Tong et al. to apply the concept in SR field [47]. Features from each layer are connected to the subsequent layers through concatenation. At the time of reconstruction, both high-level and low-level features are combined. As the inputs are in LR space, upscaling is needed in the system. Authors used deconvolution to scale up the features. A term growth rate k is used here, which defines the number of features produced by every convolution layer in each dense block. If a block contains 8 convolution layers with k=16, a total $8k$=128 feature maps will be created by that block. In their extension work, the authors also used the local and global skip connection. To reduce high volume of the feature maps, bottleneck layer was introduced by 1×1 convolution. Despite the problem of requiring high memory allocation, SRDense Net surpassed previous state-of-the-art methods. A SRDense Net block diagram is shown in Figure 2.19.

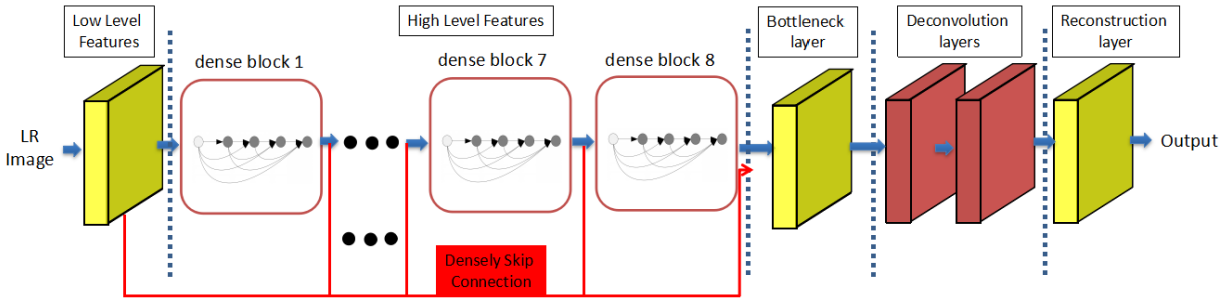


Figure 2.19: An illustration of SRDense Net [47].

A refinement of SRRes Net structure is done in Enhanced Deep Residual Networks for Single

Image Super-Resolution (EDSR) [48]. The batch normalization layer is removed from the block to speed up the computation and save memory allocation. Also, a new technique residual scaling is induced to stabilize the training for increasing number of feature maps. Besides single scale, authors proposed a multiscale model (MDSR), both are shown in Figure 2.20. The network optimizes the MAE loss function. Although it achieves good performance, since the layers don't share the weights, this model has more parameters compared to the recursion based models [13, 15].

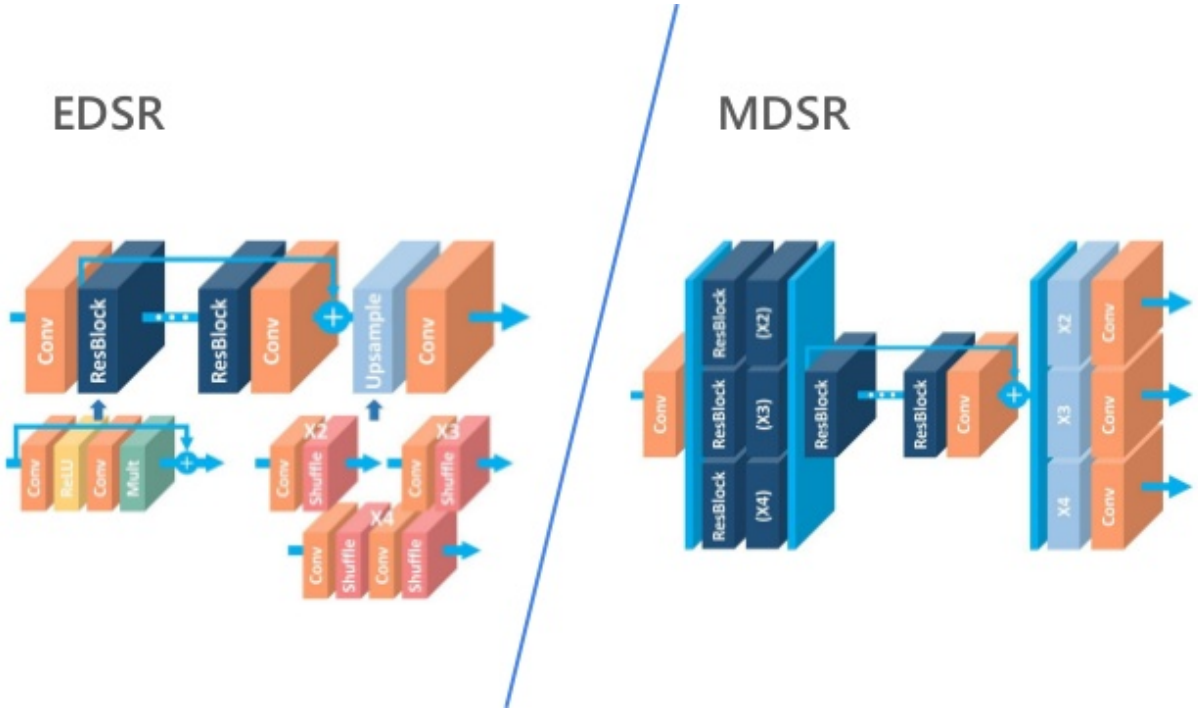


Figure 2.20: An illustration of EDSR and MDSR [48].

Based on human brain working principle, Tai et al. proposed Persistent Memory Network (MemNet)for Image Restoration [49]. Basically, MemNet is a joint network of Resblock, Denseblock and recursion. In each memory block the authors designed several recursive units. One recursive unit has almost the same resblock structure [46] with some changes in the layer order. These recursive units share weights in the same memory block. Feature maps from each recursive unit are called short term memory, and those from previous memory block are named long term memory. All short-term memories from the current block and long-term memory from the previous blocks are concatenated followed by a gate unit which performs 1×1 convolution. Such memory blocks are stacked together till the reconstruction layer [Figure 2.21]. MemNet achieved high performance

for SISR as well as image degradation problem.

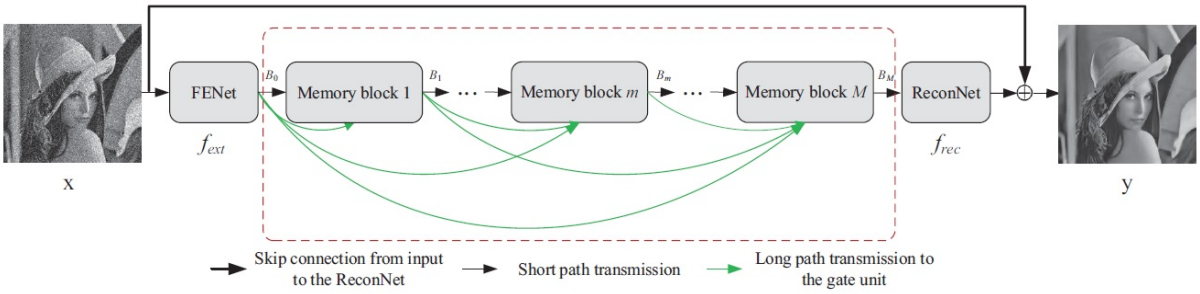


Figure 2.21: Basic MemNet architecture [49].

Another significant improvement is shown in Residual Dense Network (RDN) by Zhang et al. combining of SRResNet and DenseNet for image SR [50]. RDN learns local residuals by extracting abundant local features via dense connection which also allows direct connection from the preceding to the all subsequent layers. Like VDSR, RDN also learns global residuals, but this case, before the upscaling layer is induced. Each residual dense block (RDB) is an arrangement of several convolutions and ReLU activation functions followed by a bottleneck layer of 1×1 convolution at the end. The authors used the down sampled inputs, and then upscaled the features progressively using the method showed in ESPCN [39]. A basic block diagram of RDN is shown in Figure 2.22. The one caveat is that, RDN requires larger memory allocation for high number of parameters due to growing number of feature maps in each RDB.

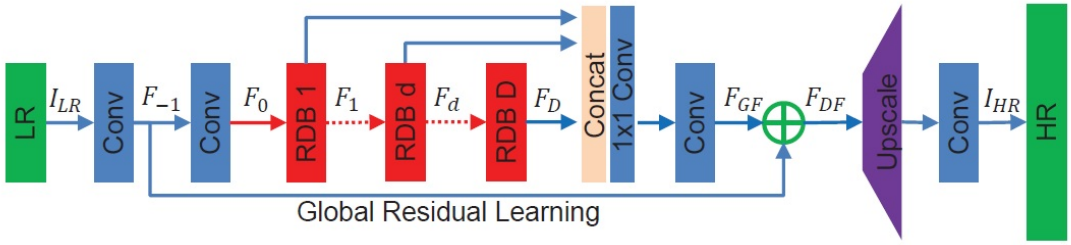


Figure 2.22: The architecture of Residual Dense Network (RDN) [50].

2.6 Main Drawbacks of Previous Methods

So far, a significant number of remarkable works have been implemented in the SR field to better super resolve the LR image. However, some solutions are very expensive in terms of computation

and memory consumption. Finding an optimum trade-off between expense, efficiency, and performance is still a matter of experiment. RDN [50], EDSR, MDSR [48], have better results with high number of parameters. Whereas, ESPCN [39], VDSR [14], DRRN [13], DRCN [15], LAPSRN [42] have a smaller number of parameters with less performance improvement compared to the previous ones. Expanding the network receptive field by stacking more unique convolution layer increase the parameter count of the network. However, recursion tends to affect the speed to achieve the same performance with more non-unique layers. Also, high-level features e.g., edges and textures are difficult to reconstruct especially for the higher scaling factors. As it will be shown, using a careful design of recursive block with dense connection, one-dimensional kernel and depth wise separable convolution, allows performance increasing in terms of PSNR, SSIM and network depth, while keeping lower parameter requirements.

Chapter 3

Technical Approach

In this chapter, design and different components of the proposed EFNet are described. To be specific, three different networks are presented here: one using simple recursive blocks, one using recursive blocks with features masking by an independent parallel architecture and the final one including Sobel and Laplacian edges while masking. In the beginning, various components and theoretical explanation of the EFNet is described in Section 3.1. How to increase the receptive field without affecting the parameter count is covered here. Then, inspection is done on different design spaces by rearranging the convolution layer orders. In chapter 4, further evaluation on different design schemes is done to find the best performing arrangement. The goal is to find the optimal parameter numbers with best performance both visually and statistically.

First, a baseline model is created, which is shown in Figure 3.1. Baseline structure is designed by restructuring the composition of RDB used in RDN [50]. This baseline model generates features in the same manner as some state-of-the-art methods [13, 37, 50]. The input is in LR space, and the features are upscaled by means of learned filters. Proposed model is not able to handle multiple scales in a single model simultaneously. To handle multiple scales as in [13, 14, 15], a different design scheme is created, which takes the bicubic/bilinear upscaled image in HR space. However, the performance is worse with a cost of higher computation complexity and time. In the offered structure, both global and local residual learnings are used like the previous state-of-the-art methods. But, for local residual learning, adding the skip connection like in [13] is discarded. Instead, concatenation of the preceding layer features is presented, followed by a 1×1 convolution layer to perform a weighted sum for computing the residuals. This way, the local residual generates the features, which has access to all the previous information and hence becomes more robust to the outliers. Each local residual block is a combination of three successive 3×3 convolutional

layer followed by only one ReLU at the end. As a result, the structure is simpler and more computationally efficient compared to RDN, where 8 successive unique convolutions and ReLU were used [50]. For global residual, traditional method of adding skip connection is followed as proposed in [10]. Only 4 residual blocks with 64 number of filters are used in all the convolution, except the last layer where there are only 3 filters to predict the final RGB image. Also, each unique residual block is reused thrice to apply the recursion and thus increase the depth. Experiments are made with the recursions more than thrice but didn't improve the performance.

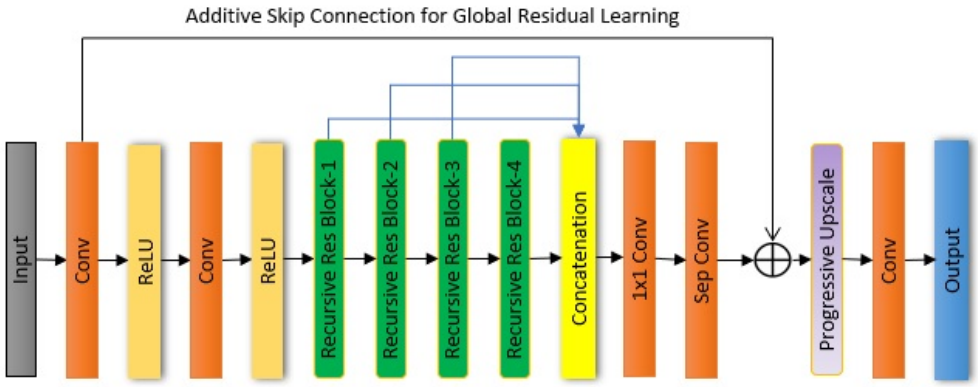


Figure 3.1: The architecture of proposed baseline model.

3.1 Expansion of Receptive Field

Receptive field is a term related to neuroscience, which is a region of a sensory space for each neuron. The definition is quite similar for CNNs as well. The area in the input to compute a single pixel in the output, is represented as the receptive field. In the Figure 2.3, a single 3×3 convolution layer is applied to calculate an output pixel. The spatial field of view or receptive field for that network 3×3. The higher the receptive field, the more information the network can use for a single output pixel. So, the main target should be expanding that in an efficient way. Several methods can be adopted to enlarge the receptive field. One possibility may be increasing the kernel size to have a bigger field of view. In the previous Figure 2.3, if a sliding window of size 9×9 is used, that would enlarge the area to compute a single output pixel. However, this technique requires more computations and memory bandwidth for the need of higher number of parameters. A single 9×9 kernel requires $9 \times 9 = 81$ parameters, compared to $3 \times 3 = 9$ numbers of a 3×3

filter. Computing a single output pixel, several multiplications and additions are required, which are equal to the quantity of parameters. So, compared to a 3×3 kernel, a 9×9 kernel has $\frac{81}{9} = 9$ times excess parameters and computations. As a result, memory bandwidth and runtime are higher for bigger sized kernels. Besides, training is also difficult for these types of models [8, 36].

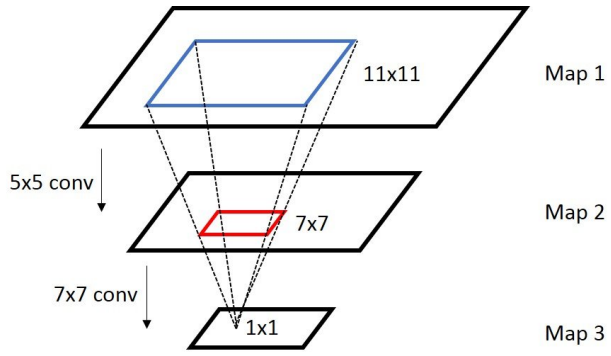


Figure 3.2: Stacking multiple small convolution layer for an efficient receptive field expansion.

Another expansion method is stacking several small sized kernels, proposed by Simonyan and Zisserman in the VGG network [8]. This technique overcomes the drawbacks of the bigger kernels. An example is illustrated in Figure 3.2. Here, two successive convolutions of 7×7 and 5×5 are applied. One pixel in $Map3$ is looking at 7×7 pixels in $Map2$. Similarly, in $Map2$, one pixel is used on a 5×5 map. In total, each pixel from $Map3$ is viewing $7 + 5 = 11 \times 11$ pixel area in $Map1$. Hence, stacking a 5×5 and 7×7 convolution is mathematically equivalent to using a single 11×11 convolutional layer. Now the question is, how these are efficient. A single 11×11 layer uses total $11 \times 11 = 121$ parameters, and also 242 $(121 + 121)$ additions and multiplications for each output pixel. However, the stacked smaller kernels need only $(5 \times 5) + (7 \times 7) = 25 + 49 = 74$ parameters and 148 $(74 + 74)$ computations in total for each output pixel. Thus, the later process provides enough savings with the same receptive field. Moreover, this specific architecture allows to introduce scaling factors [48] and multiple ReLU non-linearity to make the decision function more discriminative.

Many Computer Vision related issues experienced improved capability by increasing the receptive field [8, 10, 11, 51, 52, 53]. Having larger spatial context in a larger area helps the network to capture the complex mappings like edges, textures and blobs. As a result, reconstruction accuracy also improves.

Another memory efficient way to expand the receptive field involves applying recursion [13].

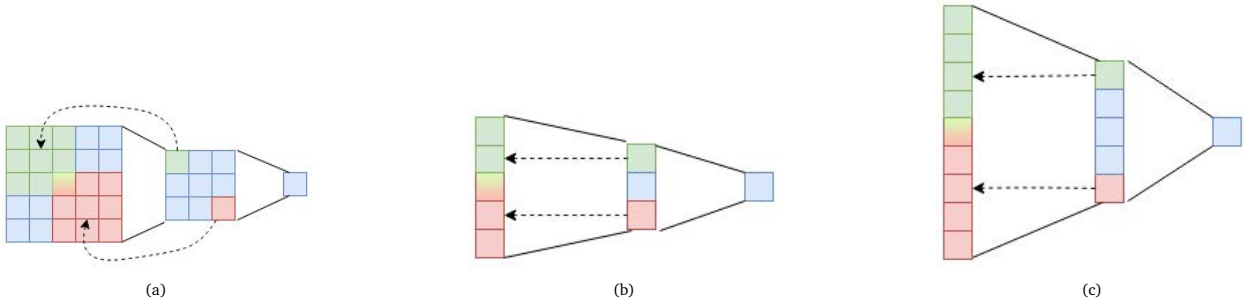
Stacking few convolution layers to obtain deeper layers comes with the adverse effect of more memory bandwidth requirements to save those large amounts of weights. By means of recursion, a network tends to have deeper layers with a constant number of parameters. A unique residual block is successively applied several times such that, memory consumption problem is mitigated, and different features are generated. But the process needs more computations to achieve the desired result. Moreover, a careful design is required as too many layers form difficulties to train the network.

One dimensional (1-D) kernel inspired by the separable kernel idea was successfully implemented in [53, 54]. This proved a very efficient theory for larger receptive field. Breaking down a 3×3 convolution into two successive 1×3 and 3×1 convolutions, produces the same result with less memory and computations.

For further memory efficient design, Chollet presented Xception net, which basically performs depth wise separable convolution [55]. The proposed technique will utilize separable convolution concept in addition to 1-D kernel with the baseline architecture. To produce a single feature map, depth wise separable convolution uses only a single unique filter for every given input feature. Thus, reduces the number of required filters as well as the quantity of weights. Experimental results will show how the 1-D kernel and separable convolution blend together to increase the accuracy while maintaining the parameter numbers with less computations.

3.2 Theory of 1-D Convolution and Depth wise Separable Convolution

1-D convolution was first introduced in the image SR field by Seif et al. [54]. 1-D convolution is called spatial separable convolution, which simply divides the kernel into two smaller kernels. For example, in Figure 3.3a, by means of two sequential 3×3 convolutional layers, a receptive field of 5 is created in both vertical and horizontal direction. Without the bias two layers utilize 18 parameters in total to cover receptive field. Now in Figure 3.3b, two back-to-back 3×1 convolutions make a receptive field of 5 in the vertical direction having only 6 parameters. Similarly, for horizontal direction 6 more parameters are needed. Hence, to gain the same receptive field by a 3×3 square kernel, 1-D convolution needs only 12 parameters, which is $\frac{1}{3}^{\text{rd}}$ times less.



(a) (b) (c)

Figure 3.3: Receptive field explanation by means of kernel [54]. (a) 3×3 convolution slides thrice to create a receptive field of 5 in both dimensions. (b) Receptive field of 5 in the vertical direction by means of two successive 3×1 convolution. (c) Receptive field of 9 in the vertical direction by means of two successive 5×1 convolution.

Another example is shown in Figure 3.3c. Two 5×1 convolutions are applied in the vertical direction to cover an area of 9 with 10 weights and 10 multiplications. Including the horizontal receptive field, a total of 20 parameters are required to have a receptive field of 9 in both directions. However, using the square requires a total of 50 parameters and same number of computations. So, the efficiency is more significant ($\frac{2}{5}$ $^{\text{th}}$) for the bigger kernels. Even, if 5×5 kernels are replaced by 3×3 square kernels to get a receptive field of 9, a total of 36 parameters and 36 computations (stacking 4 layers) are required, which is nearly twice than the 5×1 kernel.

1-D convolution is influenced by the separable filter concept in computer vision problems [1]. A square filter can be represented as a matrix multiplication of two 1-D filters and thus reduces the complexity while producing the exact same result. For example, an edge detecting filter such as Sobel operator ($G_{\text{x}}, G_{\text{y}}$) is represented by the following matrices

$$G_{\text{x}} = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \qquad G_{\text{y}} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \tag{3.1}$$

Here, G_{x} and G_{y} extracts the vertical and horizontal edges respectively. This filter has 9 parameters and requires 9 multiplications and additions to compute a single output pixel. Now,

this matrix can be separated into two one dimensional vectors for the ease of calculation.

$$G_{\mathrm{x}} = \begin{bmatrix} +1 \\ +2 \\ +1 \end{bmatrix} * \begin{bmatrix} -1 & 0 & +1 \end{bmatrix} \qquad G_{\mathrm{y}} = \begin{bmatrix} -1 \\ 0 \\ +1 \end{bmatrix} * \begin{bmatrix} +1 & +2 & +1 \end{bmatrix} \tag{3.2}$$

By representing the kernel in such a manner, the parameter numbers can be reduced, and accordingly the multiplications minimize from 9 to 6, although performing the same operation on the image. One constraint must be kept in mind is that, in CNNs 1-D kernel may or may not turn to be a separable one. Since, the network can't be enforced to train in such a way that, the learned kernels are always separable, there is a chance of performance degradation too. In this context, it is better to rely on the training and optimization method to utilize the parameters for a global minimum regardless the kernels are separable or not and proceed in an efficient way. Despite the uncertainty, better results were achieved efficiently in [54].

Depth wise separable convolution is a different version of factoring a convolution operation into multiple branches for efficient parameter use of a model. In the convolution layer of a CNN, the filters to be learned are in a three-dimensional state - width, height and channel. A single filter performs spatial correlation as well as cross channel correlation [55]. Spatial correlation is the most common height and width-wise convolution task. Cross channel correlation occurs when convolution is assigned through channels in a three-dimensional space. To perform both operations for a single output channel by an only one convolution, enormous parameters are needed. A depth wise separable technique factorizes these two steps in an efficient way. The entire process has two parts- depth wise convolution and pointwise convolution.

Consider the illustration of depth wise separable convolution in Figure 3.4. In Figure 3.4a, the depth wise convolution is shown. As can be seen, unlike normal convolutions, it uses only one filter for every input channel. This single filter slides over a single channel to compute one output channel. In the example shown in the figure, the input is RGB channel and only three 2-D (two dimensional) filters are required to produce all the spatial correlated output. In short, in the Figure 3.4a this step maps a $n \times n \times 3$ channel to another $n \times n \times 3$ channel with $3 \times 3 \times 3$ filter.

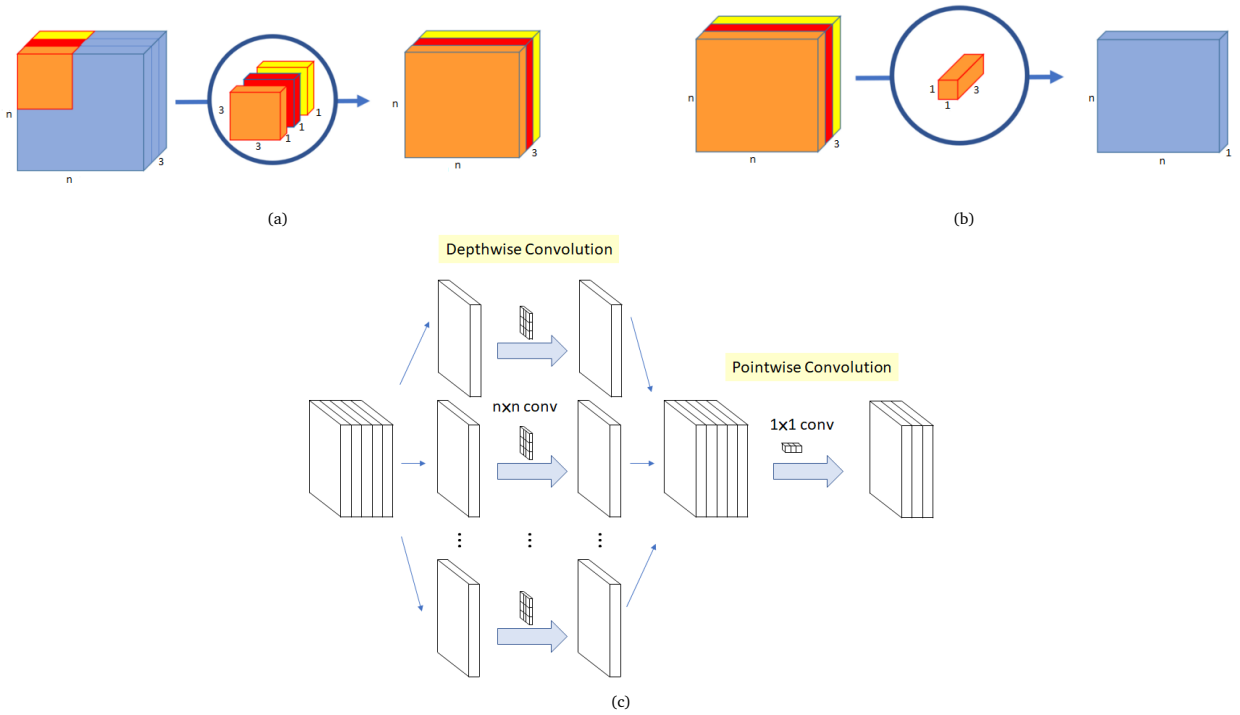


Figure 3.4: Convolution explanation by means of depth wise separable process [56]. (a) Depth wise convolution part performing spatial correlation. (b)Pointwise convolution part performing cross-channel correlation. (c) The full picture at a glance.

In Figure 3.4b, the second part, cross-channel correlation representation is shown. This is actually a 1×1 convolution. Kernel set has three dimensions and the depth is equal to the number of inserted channels. Since, in Figure 3.4b, the input has channels of 3, the depth is 3 for the kernel set having a dimension of $1 \times 1 \times 3$. All together, it maps the input to a $n \times n \times 1$ output. To have an output of n channels, it requires n number of $1 \times 1 \times 3$ filters and hence the output dimension will be $n \times n \times n$. For the normal convolution process, there will be three different filters (sometimes called a three-dimensional filter) for each generated feature map. To construct the same outputs shown in Figure 3.4, a normal convolution will need three $n \times n \times 3$ filters.

As this can be seen, depth wise separable convolution quite substantially expands the receptive field with fewer computations and parameters. Let's consider another example. Say, a $n \times n \times 64$ input channel is required to map into a $n \times n \times 64$ feature by using 3×3 kernel. For the regular convolution, 64 numbers of $3 \times 3 \times 64$ filters will be needed. Total required parameters are $64 \times 3 \times 3 \times 64 = 36,864$. Whereas, for the depth wise separable convolution, there involves only $64 \times 3 \times 3 + 1 \times 1 \times 64 \times 64 = 4,672$ parameters, which is only 10% of the original 2-D parameters. Thus, the network can perform more processing within the shortest possible time and saves a huge

memory consumption and processing time. Only caveat is that, due to substantially low number of parameters, the network might not learn properly and fail to converge. So, this technique is applied only in the middle layers of the proposed model to increase the network depth efficiently.

3.3 Design Space Exploration

In this section, various components of the proposed network architecture will be explored with the 1-D kernel and depth wise separable convolution. In the first stage, baseline model (EFNet-B) uses few recursive residual blocks to achieve the benchmark. Then is gradually upgraded from the baseline to EFNet through multiplication by a shallow network using 1-D kernel. And finally, in EFNet+, edge features are added to recover the textures and edges more prominently.

3.3.1 Recursive Residual blocks

Baseline model consists of several residual blocks and recursive use of the blocks. Unlike the original residual block in ResNet architecture [10], batch-normalization and several ReLU are removed for computational efficiency. The residual block has 3 successive separable convolutions followed by a 1×1 convolution. In addition, to capture the correlation between the feature maps, all the convolution outputs are concatenated (including the initial input feature) prior to the final 1×1 convolution layer. As a result, the block can easily encapsulate important features from all the preceding layers, and thus a smooth flow of information from each channel is maintained. This layer also serves the purpose of local residual as adding the input features is removed in this case. Instead of plain addition, learned addition of input features are performed in the design. It is worth to be noted that, the ReLU non-linearity is not applied until the 1×1 convolution layer, which helps to independently process all the features with no loss of information. The applied ReLU hence serves as pre-ReLU for the next recursion block. Figure 3.5 illustrates several arrangements for the proposed resblock. In Figure 3.5a, baseline model for the residual block is shown. Rearrangement of the layers are experimented to see if performance improvement is possible. Figure 3.5b shows the diagram if the ReLU activation function is placed before the convolutions. The features maps are also processed through parallel branches instead of sequential convolution function (Figure 3.5c and 3.5d). This is inspected to observe if independent processing of features affects the spatial and

cross-channel correlation. After analyzing different arrangements, it is perceived that, the baseline design (scheme-a) performs the best. So, this architecture will be implemented throughout the rest of the experiments. Another design consideration is trading off performance and computational cost by using recursion blocks. After several experiments, a unique residual block is opted to use four times followed by concatenation of all the residual outputs and 1×1 convolution at the end.

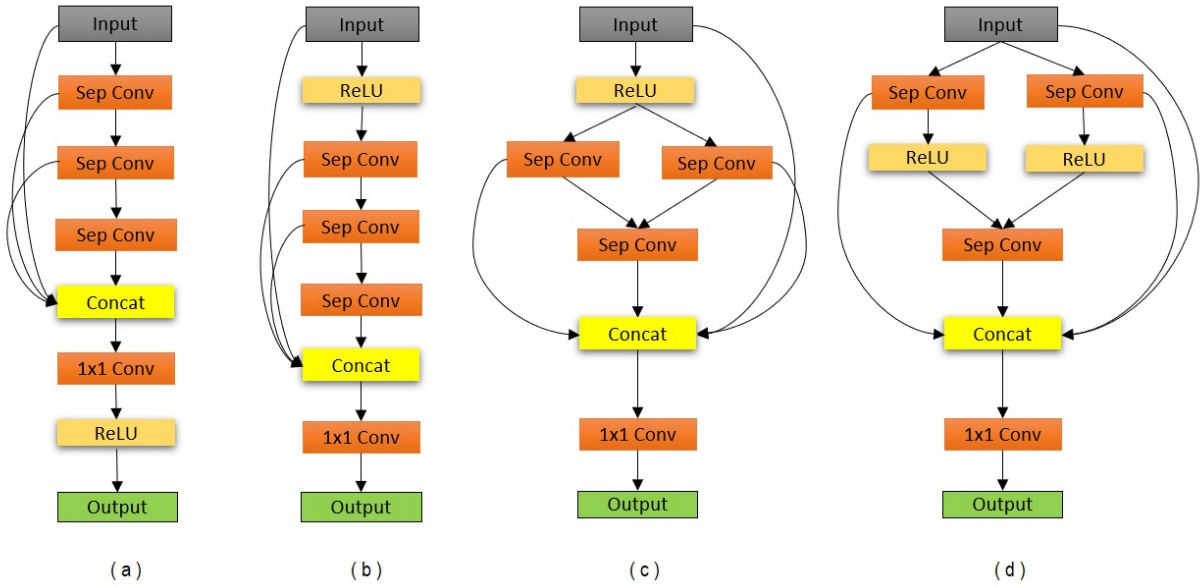


Figure 3.5: A basic Residual Block. (a) Baseline block setup. The ReLU is applied after the concatenation layer, for non-linear mapping of all the gathered features. (b)ReLU is shifted before the initial convolution layer to check if pre ReLU affects the performance. (c) & (d) The convolution is split into two different parallel branches to independently process the information with placing the activation function before and after the convolution respectively.

3.3.2 Shallow Multiplicative blocks for Attention

For masking [45] the features, shallow multiplicative blocks are incorporated in the architecture. The purpose of this part is to enhance the high-level features, which are most prominent for sharpening the image. Illustrated in Figure 3.6, 1-D depth wise separable kernel is applied to extract the high-level features by spreading the receptive field in a very efficient way. This block can expand the receptive field without any recursion and also able to introduce an extra non-linearity between two adjacent 1-D layers. Consequently, complex textures and edges are more visible in the final result. This scheme is inspired by the improved results in [54]. The purpose of this shallow Attention part is to independently increase the receptive field with bigger kernels having less parameters so that, the network can extract features losing a minimal number of information from

the input channels. However, bigger kernels also produce increased computations and sometimes redundant parameters. In [54], the authors have shown how to implement the bigger kernel concept with only a few computations, and also exhibited improvement in the SR field. Motivated by that concept, in total, four 1-D convolutions and four activation functions ReLU sequentially build the final block. Like the original ResNet structure [10], an additive skip connection is used to learn the local residual. An activation function Hard Sigmoid is applied at the end of this shallow masking architecture, enabling the model to amplify the area containing high feature maps.

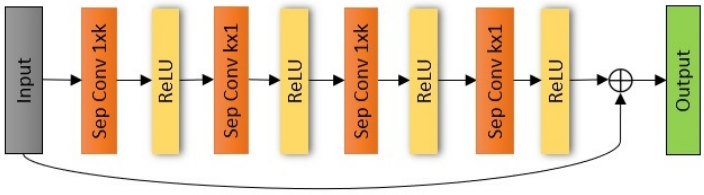


Figure 3.6: A basic multiplicative block for masking.

3.3.3 Combined Network Architecture

In the baseline EFNet-B model shown in Figure 3.1, recursion of residual blocks is used to enhance performance. Three successive recursions of each block yield the best result in test. The architecture is further extended with the shallow masking block for Attention mechanism. The EFNet structure is shown in Figure 3.7. From the figure, it is visible that, both the outputs of residual block and multiplicative block are passed through the sub-pixel convolution to upscale the features. Prior to that upscale layer, an additive skip connection enables the global residual learning. Then, the upscaled multiplicative features are passed through the Hard Sigmoid function. Masking the features then occurs, followed by a convolution which outputs three channels to synchronize with the initial RGB input.

Finally, incorporating the Sobel and Laplacian edge operators, the EFNet is further upgraded to EFNet+. It has an advantage over the previous model in terms of edge preservation while keeping fewer parameter count. In this case, first the edge features are extracted from the input by the Sobel and Laplacian operator. These edge features including the extracted features from the first convolution layer are passed into the middle part (i.e., recursive part) of the model. Then, the initial

edge maps are being upscaled in the bottom part using the technique described in ESPCN [39]. It is worth to mentioning that, for further enhanced edge performance, an edge loss function is applied in addition to the default loss function. Figure 3.8 explains the different parts of the final EFNet+ architecture.

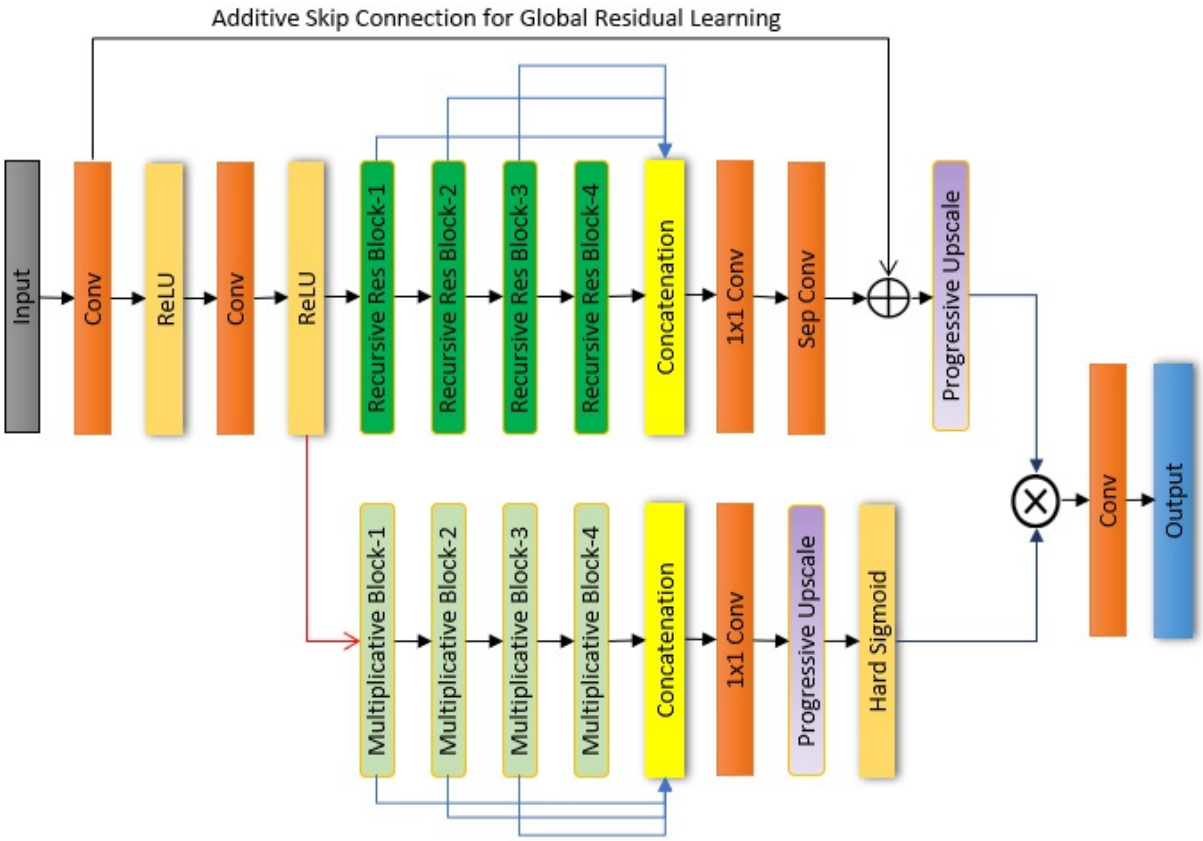


Figure 3.7: Illustration of proposed EFNet architecture.

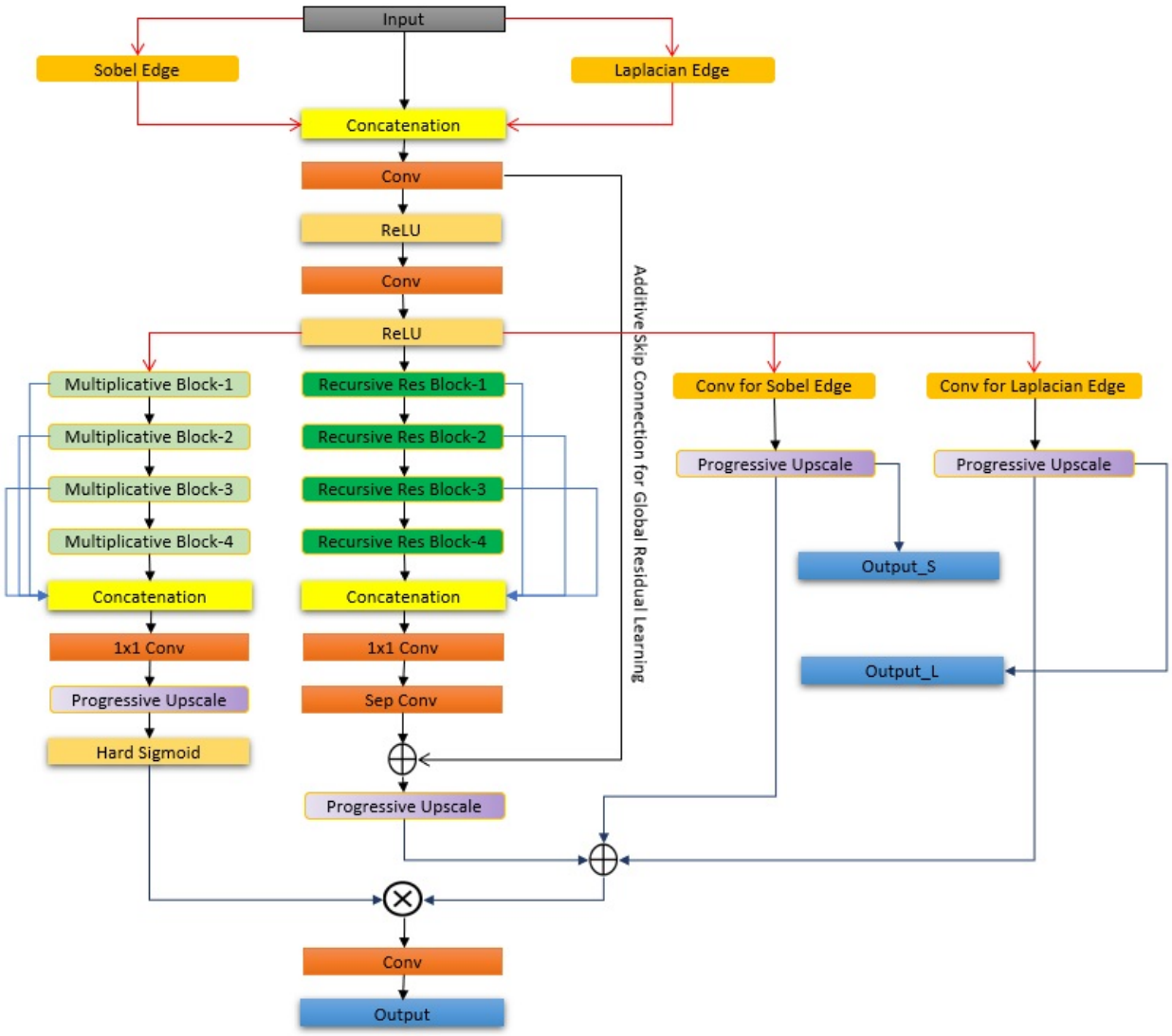


Figure 3.8: Illustration of proposed EFNet+ architecture.

Chapter 4

Experimental Results

In this chapter, evaluation of the proposed models is presented on various test dataset. Analysis of training data, measurement metrics, loss function, details the implementation of the system is shown in section 4.1. The next section 4.2 covers the statistical test results for different schemes of the proposed method. Finally, comparison of the results with the state-of-the-art methods is demonstrated in section 4.3.

4.1 Execution of Experiment

4.1.1 Training Dataset

Most widely used training dataset for SR purpose is the 291 imageset by Yang et al. [4] and the Berkeley Segmentation Dataset [57]. Recently, DIV2K dataset containing high quality image data [58], has been very popular for image restoration purpose. This dataset contains 800 HR (2k resolution) images for training and 100 images for testing and validation. This dataset has shown huge improvements in training over others in recent SISR works [59,60]. The New Trends in Image Restoration Enhancement (NTIRE) DIV2K dataset contains high quality diverse images. Those images have a minimal blur and noise, and effective to provide enough training data for image restoration-based architectures.

For training, first, the RGB images are cropped out into smaller patches of size 48×48 with the stride of 48, which implies no overlapping segment involved in the sub images. It is found, using patches of larger size doesn't improve the performance in contrast to the speed and runtime trade-off. Again, using too smaller patches limits the ability to use larger receptive field. Considering the above factors, a patch size of 48×48 is selected. Since, naturally HR & LR training pair doesn't

exist, the LR patch is artificially created from the corresponding HR one. In this experiment, the 48×48 HR patches are down sampled by the means of bicubic interpolation according to the SR scale. These down sampled patches are input to the system & original HR patches are the target output. This procedure is followed because it uses less memory space for computation efficiency as in some other state-of-the-art techniques [37, 39, 42, 48, 50].

To create additional data, augmentation of data is developed by random rotation and flipping [13, 14, 15, 42]. Both the training and evaluation is conducted in RGB color space. The HR image patches are downscaled with a scaling factor of x2, x3, x4, and hence proposed model can handle those scaling factors only. Final evaluation is performed on the predicted Y channel of YCBCr colour space in terms of PSNR and SSIM. Test is executed on DIV2K 100 validation images as well as Set5 [61], Set14 [62], BSDS100 [63], Urban100 [64] benchmark datasets.

4.1.2 Measurement Metrics

SISR performance is generally evaluated based on PSNR and SSIM [65]. PSNR measures the pixel by pixel accuracy and SSIM assesses the perceptual image quality. PSNR is the most commonly used metric based on MSE. MSE is very straightforward, makes clear mathematical sense, and convenient to optimize. It defines the distance between the predicted and reference pixel value, which can be presented as follows

$$MSE = \frac{1}{MN} \sum_{M,N} (\hat{Y}(m,n) - Y(m,n))^2, \tag{4.1}$$

where $\hat{Y}(m,n)$ is the predicted pixel, $Y(m,n)$ is the reference pixel and M, N are total number of rows and columns in the input image respectively. Now based on MSE, the PSNR of that window is defined as

$$PSNR = 10 \log_{10}(\frac{I^2}{MSE}), \tag{4.2}$$

where I is the maximum pixel value in that specific window of M rows and N columns. PSNR is a metric to quantify the reconstruction accuracy but doesn't suit perceived visual quality. It doesn't consider features from the image local context. To eliminate the shortcomings, in some cases

MSE has been adjusted to comply with human visibility. However, the best solution is SSIM, which measures the reconstruction performance according to human perception, hence considers the local intensity pattern into account. SSIM shows improved results to handle structural degradation besides recovering raw pixel. SSIM algorithm is explained in detail in [65].

4.1.3 Loss Function

Most widely used loss function is MSE (also called L2 loss). MSE penalizes substantially for higher deviation of the prediction with respect to the reference. Moreover, it is convenient for optimization. However, proposed networks are trained using L1 loss, since it is found by experiments that, L1 loss exhibits better results compared to L2. L1 loss (also called MAE) is defined as

$$MAE = \frac{1}{MN} \sum_{M,N} |\hat{Y}(m,n) - Y(m,n)| \tag{4.3}$$

In the EFNet-B and EFNet, L1 loss function is applied. In the EFNet+ model, a combination of losses from different layers are employed. The cost function for this architecture combines weighted sum of actual output loss, Sobel and Laplacian edge loss. Sobel and Laplacian loss are computed by applying Sobel and Laplacian operators on the ground truth image. So, the total loss is

$$\mathcal{L} = \frac{\alpha}{MN}|\hat{Y}(m,n) - Y(m,n)| + \frac{\beta}{MN}|\hat{E}_{\mathrm{s}}(m,n) - E_{\mathrm{s}}(m,n)| + \frac{\gamma}{MN}|\hat{E}_{\mathrm{l}}(m,n) - E_{\mathrm{l}}(m,n)|, \tag{4.4}$$

where $\hat{E}_{\mathrm{s}}(m,n)$ & $\hat{E}_{\mathrm{l}}(m,n)$ are Sobel & Laplacian edge map of the predicted image respectively, and α, β and γ denote the loss weight. Empirically, $\alpha = 0.7, \beta = 0.3, \gamma = 0.3$ are set to yield the best result.

4.1.4 Implementation Details

To explore the best combination and number of recursive blocks, initially the models are trained with first 100 images from DIV2K dataset on a scale of x2, x3, x4. Those images benchmark the baseline model besides achieving the state-of-the-art result. Once the baseline design scheme is firm, training is done with all the 800 images in DIV2K dataset on all the scale. Since, a single

model can't handle multiple scales, the same model is trained thrice with three different scales: x2, x3 & x4. ADAM optimizer [66] is used with an initial learning rate of 10^{-4}. The learning rate is halved in every 10 epochs with 200 numbers of epochs. To reduce the complexity of huge data and train time, total $64,000$ patches are implemented in each epoch. As a result, training time is significantly reduced to 3 days approximately. All the weights are initialized by the method stated in [11] with bias in effect. The batch size is set to 32. NVIDIA 1080Ti GPU is used throughout the whole process.

4.2 Litmus Test

4.2.1 Experimenting Baseline with Recursions

Scheme	Scale	EFNet-B PSNR/SSIM
A	x2	33.45/0.9347
	x3	30.10/0.8655
	x4	28.37/0.8071
B	x2	33.37/0.9340
	x3	30.08/0.8612
	x4	28.30/0.8055
C	x2	33.10/0.9290
	x3	29.75/0.8581
	x4	28.12/0.7957
D	x2	33.15/0.9336
	x3	29.93/0.8589
	x4	28.17/0.8067

Table 4.1: Evaluation result on NTIRE validation benchmark set according to PSNR/SSIM with various convolution schemes.

In this module, benchmark various arrangement scheme of the EFNet-B will be described. First, the best arrangement scheme of the convolutional layers in the residual block will be determined. In Figure 3.5, separate settings of layers are shown. Each of them is tested on a kernel size of 3 on the NTIRE validation set, which is shown in Table 4.1. The schemes are ordered from A to D, and it is clear from the table that, 3.5a achieves the best result among all of them. In scheme B,

placing the ReLU before the convolution deteriorates the performance substantially. In scheme C & D, parallelization of two independent convolutions don't improve the result either. Hence, future experiments are conducted using the scheme A to achieve the best results with the model.

Number of Recursions	Scale	PSNR/SSIM
0	x2	32.97/0.9299
	x3	29.15/0.8593
	x4	27.96/0.7968
1	x2	33.24/0.9297
	x3	29.88/0.8634
	x4	28.24/0.8049
2	x2	33.36/0.9329
	x3	30.01/0.8634
	x4	28.24/0.8058
3	x2	33.45/0.9347
	x3	30.10/0.8655
	x4	28.37/0.8071
4	x2	33.30/0.9122
	x3	29.83/0.8589
	x4	28.15/0.7991
5	x2	32.59/0.9038
	x3	29.52/0.8512
	x4	28.47/0.7819

Table 4.2: Evaluation result on NTIRE validation benchmark set according to PSNR/SSIM with various recursions.

Now, the question is how many recursions are needed for a residual block. Recursions are also responsible for higher number of computations. So, an optimal number of recursions must be determined for a better PSNR/SSIM with less calculations. In Table 4.2, the performance evaluation with higher order of recursions are shown. Observation depicts that, performance upgrades till the $3rd$ applied recursion. After that, decays gradually. So, 3 recursions are selected for each residual block.

In the next step, experiments are conducted with the validation results by changing the filter size from 3 to 11. It is obvious that, increasing the kernel dimensions boosts behavior. It reflects

Filter Size	Scale	EFNet-B PSNR/SSIM
3	x2	33.45/0.9347
	x3	30.10/0.8655
	x4	28.37/0.8071
5	x2	33.49/0.9351
	x3	30.13/0.8659
	x4	28.39/0.8076
7	x2	33.52/0.9355
	x3	30.19/0.8666
	x4	28.45/0.8079
9	x2	33.43/0.9348
	x3	30.12/0.8653
	x4	28.36/0.8069
11	x2	33.35/0.9320
	x3	30.04/0.8613
	x4	28.23/0.8055

Table 4.3: Evaluation result on NTIRE validation benchmark set according to PSNR/SSIM with various kernel size.

the theory in the results of Table 4.3. The bigger the kernel size with keeping the layers fixed, the more the enhancement. EFNet-B is trained with different kernel sizes and the results are validated with NTIRE DIV2K validation set on RGB images. However, larger receptive field by increased filter size comes with more parameters and runtime, which is displayed in Table 4.4.

After observing Table 4.3, it is clear that both PSNR and SSIM improve gradually until kernel size of 7×7 and decrease afterwards. The same happens for all the scales. However, Table 4.4 proves the additional parameters by increased kernel size is not worth of the performance improvement as expected. So, in that case the better choice will be not to trade-off memory bandwidth for performance since improvements are not significant. Hence, kernel size of 3 is selected for the network architecture. In a nutshell, for a fair performance efficiency trade-off, convolution scheme A is opted with a filter size of 3 and 3 recursions for the rest of the experiments.

Filter Size	Scale	EFNet-B Parameters	EFNet-B Runtime
3	x2	418,243	5.39s
	x3	602,883	6.9s
	x4	565,955	5.45s
5	x2	503,235	5.53s
	x3	687,875	7.09s
	x4	650,947	5.57s
7	x2	630,723	6.08s
	x3	815,363	7.23s
	x4	778,435	6.11s
9	x2	800,707	6.29s
	x3	985,347	7.36s
	x4	948,419	6.40s
11	x2	1,013,187	6.45s
	x3	1,197,827	7.44s
	x4	1,160,899	6.51s

Table 4.4: Parameter number and Runtime result on NTIRE validation benchmark set with various kernel size.

4.2.2 Experimenting EFNet with various Factors

Now, importance is given on the construction of masking network in the EFNet. In the shallow multiplication block, 1-D kernels are used to extract the edges by means of efficient receptive field expansion. The components are chosen in such a way that it emphasizes the edge or texture pixels without a major increase in the parameter count. In Table 4.5, the results and parameter numbers by using kernel size from 3 to 11 are shown. Performance improves until a kernel size of 9 and then starts decreasing. Using a size of 9 gives better result compared to a little rise in the memory consumption. So, 1×9 & 9×1 convolution kernel is picked for the final network architecture. As the multiplication blocks and residual blocks work independently, their combined framework benefits the model to perform better than the baseline.

Filter Size	Scale	EFNet Parameters	EFNet PSNR/SSIM
3	x2	470,828	33.49/0.9360
	x3	655,518	30.13/0.8669
	x4	618,580	28.39/0.8071
5	x2	472,364	33.51/0.9367
	x3	657,054	30.14/0.8678
	x4	620,116	28.41/0.8085
7	x2	473,900	33.57/0.9370
	x3	658,590	30.18/0.8682
	x4	621,652	28.45/0.8092
9	x2	475,436	33.60/0.9376
	x3	660,126	30.20/0.8685
	x4	623,188	28.48/0.8097
11	x2	476,972	33.53/0.9320
	x3	661,662	30.14/0.8613
	x4	624,724	28.42/0.8055

Table 4.5: Evaluation result on NTIRE validation benchmark set according to PSNR/SSIM with various kernel size.

4.3 Comparison with the state-of-the-art Methods

Experimental results are compared with a few state-of-the-art methods such as Bicubic, SRCNN [36], VDSR [14], DRCN [15], LapSRN [42], DRRN [13], Memnet [49], EDSR [48], RDN [50] at a scale of x2, x3, x4. Results are validated on public benchmark dataset Set5 [61], Set14 [62], BSDS100 [63] and Urban100 [64]. For comparison, predicted RGB images are first converted to YCBCr space, and then PSNR/SSIM of the Y channel are computed only. This maintains a fair comparison against the state-of-the-art methods. In Table 4.6, the final test results on DIV2K dataset of all the methods using the full 800 training images are displayed. In this case, the evaluation is done on RGB images. The comparison with other methods, on statistical result of Y channels are shown in Table 4.7.

From the experimental data included in Table 4.7, some discussions can be made. By using the depth wise separable convolution, improved results are obtained consistently for all the scaling factors, while keeping the number of parameters minimal. Even the baseline, although having the

Architecture	Scale	DIV2K Val. PSNR/SSIM
EFNet-B	x2	33.63/0.9453
	x3	30.24/0.8760
	x4	28.49/0.8179
EFNet	x2	33.69/0.9492
	x3	30.31/0.8795
	x4	28.55/0.8205
EFNet+	x2	33.65/0.9489
	x3	30.26/0.8793
	x4	28.59/0.8219

Table 4.6: Performance Evaluation on NTIRE validation set using all the 800 DIV2K training images by the three different models.

simplest structure, exceeds the state-of-the-art performance. Moreover, proposed architecture eases the training with the Adam optimizer having no gradient clipping in action. However, performance improvement with the Urban100 dataset is not quite significant on x4. This is obvious, as major edges and textures are lost there because of the higher down sampling factor. Those features on Urban100 dataset are very challenging to predict and reconstruct. Yet, the baseline performance is aligned with the recent results, although a very basic error function like MAE is used in this context. One interesting observation is that, for scaling factor x2 & x3, EFNet performs slightly better than EFNet+, whereas in x4, EFNet+ exceeds all others. This may be due to the capability of extracting high-level features e.g., edges in EFNet+ architecture. One more reason is, the edge features are extracted from the down sampled image and then upscaled by sub-pixel convolution. This technique helps the network to separately and independently reconstruct the high-level features, and hence improves the performance for higher scaling factors. On the contrary, for the lower scaling factors like x2 & x3, information lose is insignificant, and the convolutional filters can better extract the edge or texture features than the edge operators. Therefore, EFNet+ performance turns detrimental and does nothing but increase parameter count in x2 & x3 scale.

Comparison is also done with the other state-of-the-art methods with respect to network depth, parameters and runtime. Figure 4.1, 4.2, 4.3 plot the depth vs parameter number on x2, x3, x4 scales. In all cases, the proposed networks perform depth expansion with a very few parameter requirements alongside rise in PSNR and SSIM. The improvement is even remarkable compared

to those having more parameters and layers. With the EFNet+ architecture in scale x4, the enhancement is more distinct. This result indicates that, EFNet+ performs evidently with the higher down sampled images. These plots show how efficient use of recursions and depth wise separable convolution deepen the network and improves the performance efficiently, while keeping the parameter count as low as possible.

Further analysis is done on the statistical result PSNR with the other state-of-the-art models against the parameter numbers. In Figure 4.4, 4.5, 4.6, comparison is shown between the PSNR and parameters, with Set14 dataset on x2, x3, x4 scales respectively. Achievement is greater performance gain with a very few parameters. Only DRRN has a smaller number of parameters. However, their performance is not up to the mark compared to EFNets. Since, EDSR and RDN has more parameters compared to reduced PSNR, they are skipped from the plot for the clear visualization. Finally, runtime vs PSNR plot is compared against the prominent methods. For a fair comparison, plots in Figure 4.7, 4.8, 4.9 are conducted on the same Set14 dataset. In the all cases, proposed models show improvement will less computation time. SRCNN, LapSRN are faster, but have less PSNR. It is also notable that, EFNet+ runs slightly slower than the DRRN, MemNet and VDSR due to having a masking part and an edge fusion part. As a result, more computations are made to better extract edges and textures. Moreover, SRCNN, LapSRN and DRRN have 3, 24(on x4 scale), 54 layers respectively, compared to 100 convolution layers of the EFNet+, and in turn they tend to run faster. However, due to efficient use of parameters in the proposed networks, the difference is not very big. So, it can be said, as an interference performance trade-off, proposed models outperform the previous models substantially in all aspects.

Visual qualitative comparison among the methods are also presented. Figure 4.10 to 4.24 show some x4 & x3 upscaling results on images from Set5 [61], Set14 [62], BSDS100 [63], Urban100 [64] database using different methods. It is evident that, proposed architecture can detect the edges and textures better than the other methods. The sharpness of the edges and improved clarity of the images are clearly seen even in the higher x4 scaling factor, where other methods tend to produce smooth pictures. The complex images from BSDS100 and Urban100 have very fine details and textures and reconstructing the SR image doesn't show the expected improvement qualitatively. However, EFNets performances are aligned with the other state-of-the-art methods and such results are included in Figure 4.16 to 4.20. Proposed methods still perform better with a less noticeable

difference in between.

Based on all the aforementioned quantitative and qualitative results, decision can be made that, proposed all three models perform agreeably well over the state-of-the-art methods on all the scales. In addition, they are efficient to utilize the parameters, even with the baseline, through the inclusion of separable convolution technique. Inclusion of shallow masking network and edge fusion makes the network more robust to the textures and edges, with a slight increase in the inference time. The technique offers better PSNR and SSIM over the state-of-the-art models and also very efficient in using the parameters and speed.

Architecture	Scale	Set5		Set14		BSDS100		Urban100	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Bicubic	x2	33.66	0.9299	30.24	0.8688	29.56	0.8431	26.88	0.8403
SRCNN [36]		36.66	0.9542	32.45	0.9067	31.36	0.8879	29.50	0.8946
VDSR [14]		37.53	0.9590	33.05	0.9130	31.90	0.8960	30.77	0.9140
DRCN [15]		37.63	0.9588	33.04	0.9118	31.85	0.8942	30.75	0.9133
LapSRN [42]		37.52	0.9591	33.08	0.9130	31.08	0.8950	30.41	0.9101
DRRN [13]		37.74	0.9591	33.23	0.9136	32.05	0.8973	31.23	0.9188
MemNet [49]		37.78	0.9597	33.28	0.9142	32.08	0.8978	31.31	0.9195
LRFNet-S [54]		-	-	-	-	-	-	-	-
EDSR [48]		38.11	0.9602	33.92	0.9195	32.32	0.9013	32.93	0.9351
RDN [50]		38.24	0.9614	34.01	0.9212	32.34	0.9017	32.89	0.9353
EFNet-B		38.29	0.9617	34.07	0.9215	32.37	0.9019	32.93	0.9356
EFNet		38.32	0.9619	34.09	0.9216	32.41	0.9024	32.95	0.9357
EFNet+		38.31	0.9617	34.06	0.9216	32.40	0.9022	32.91	0.9355
Bicubic	x3	30.39	0.8682	29.76	0.7742	27.21	0.7385	24.46	0.7349
SRCNN [36]		32.75	0.9090	29.30	0.8215	28.41	0.7863	26.24	0.7989
VDSR [14]		33.67	0.9210	29.78	0.8320	28.83	0.7990	27.14	0.8290
DRCN [15]		33.82	0.9226	29.76	0.8311	28.80	0.7963	27.15	0.8276
LapSRN [42]		-	-	-	-	-	-	-	-
DRRN [13]		34.03	0.9244	29.96	0.8349	28.95	0.8004	27.53	0.8378
MemNet [49]		34.09	0.9248	30.00	0.8350	28.96	0.8001	27.56	0.8376
LRFNet-S [54]		-	-	-	-	-	-	-	-
EDSR [48]		34.65	0.9280	30.52	0.8462	29.25	0.8093	28.80	0.8653
RDN [50]		34.71	0.9296	30.57	0.8468	29.26	0.8093	28.80	0.8653
EFNet-B		34.75	0.9301	30.62	0.8473	29.31	0.8099	28.83	0.8657
EFNet		34.79	0.9306	30.69	0.8476	29.35	0.8108	28.88	0.8662
EFNet+		34.77	0.9301	30.66	0.8473	29.31	0.8104	28.87	0.8663
Bicubic	x4	28.42	0.8104	26.00	0.7027	25.96	0.6675	23.14	0.6577
SRCNN [36]		30.48	0.8628	27.50	0.7513	26.90	0.7101	24.52	0.7221
VDSR [14]		31.35	0.8830	28.02	0.7680	27.29	0.0726	25.18	0.7540
DRCN [15]		31.53	0.8854	28.02	0.7640	27.23	0.7233	25.14	0.7510
LapSRN [42]		31.54	0.8850	28.19	0.7720	27.32	0.7270	25.21	0.7560
DRRN [13]		31.68	0.8888	28.21	0.7721	27.38	0.7284	25.44	0.7638
MemNet [49]		31.74	0.8893	28.26	0.7723	27.40	0.7281	25.50	0.7630
LRFNet-S [54]		31.91	0.8900	28.44	0.7780	27.47	0.7330	25.70	0.7730
EDSR [48]		32.46	0.8968	28.80	0.7876	27.71	0.7420	26.64	0.8033
RDN [50]		32.47	0.8990	28.81	0.7871	27.72	0.7419	26.61	0.8028
EFNet-B		32.51	0.8994	28.88	0.7875	27.79	0.7425	26.65	0.8032
EFNet		32.56	0.8997	28.91	0.7878	27.82	0.7429	26.68	0.8035
EFNet+		32.61	0.8999	28.95	0.7882	27.86	0.7435	26.72	0.8038

Table 4.7: Evaluation result on public benchmark set according to PSNR/SSIM. Red indicates the best.

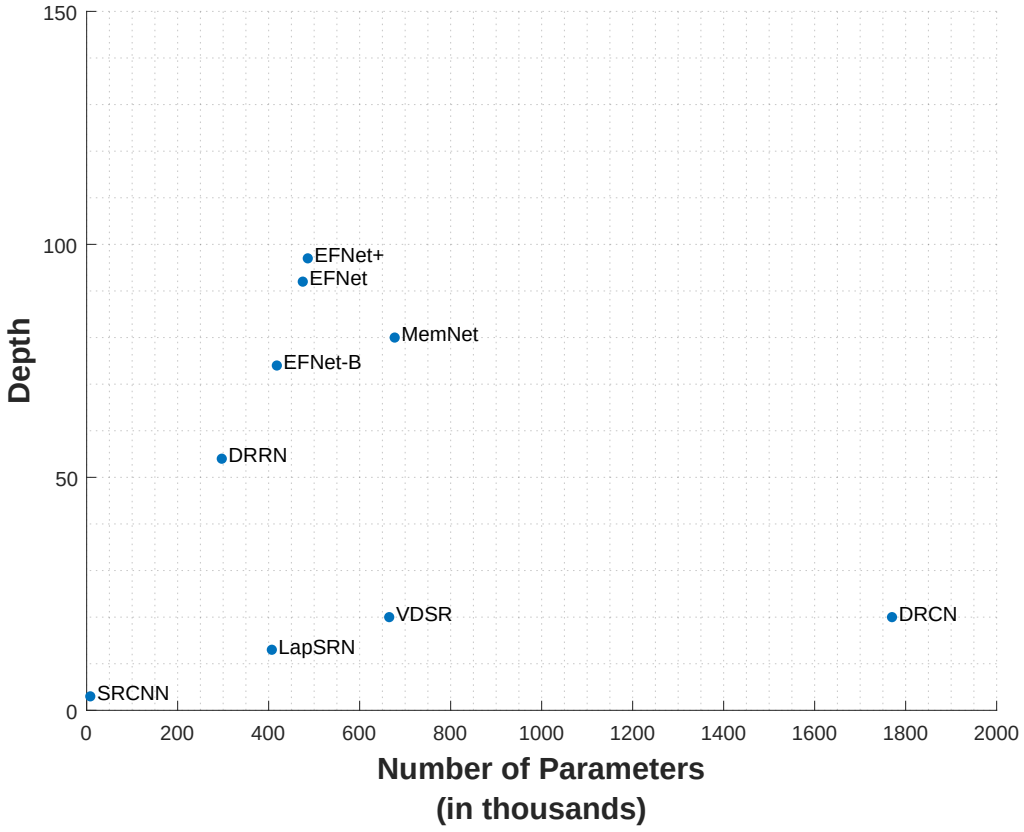


Figure 4.1: Plot of Network Depth vs Parameters against some state-of-the-art and proposed models on scale x2.

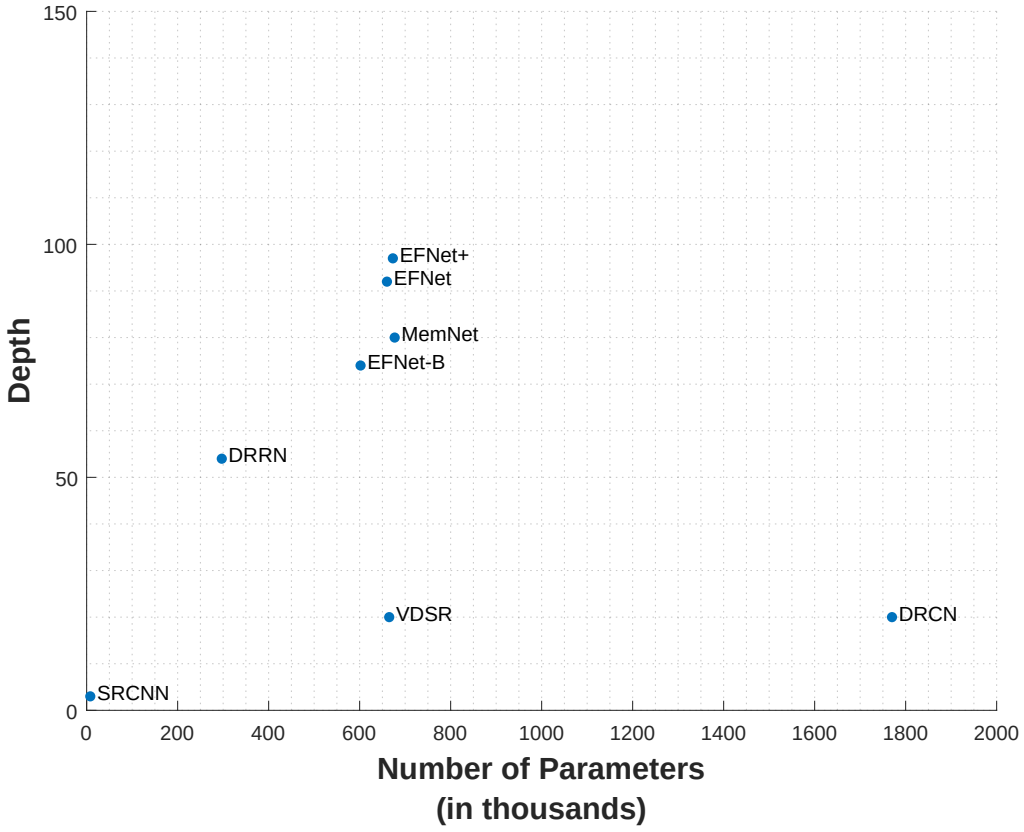


Figure 4.2: Plot of Network Depth vs Parameters against some state-of-the-art and proposed models on scale x3.

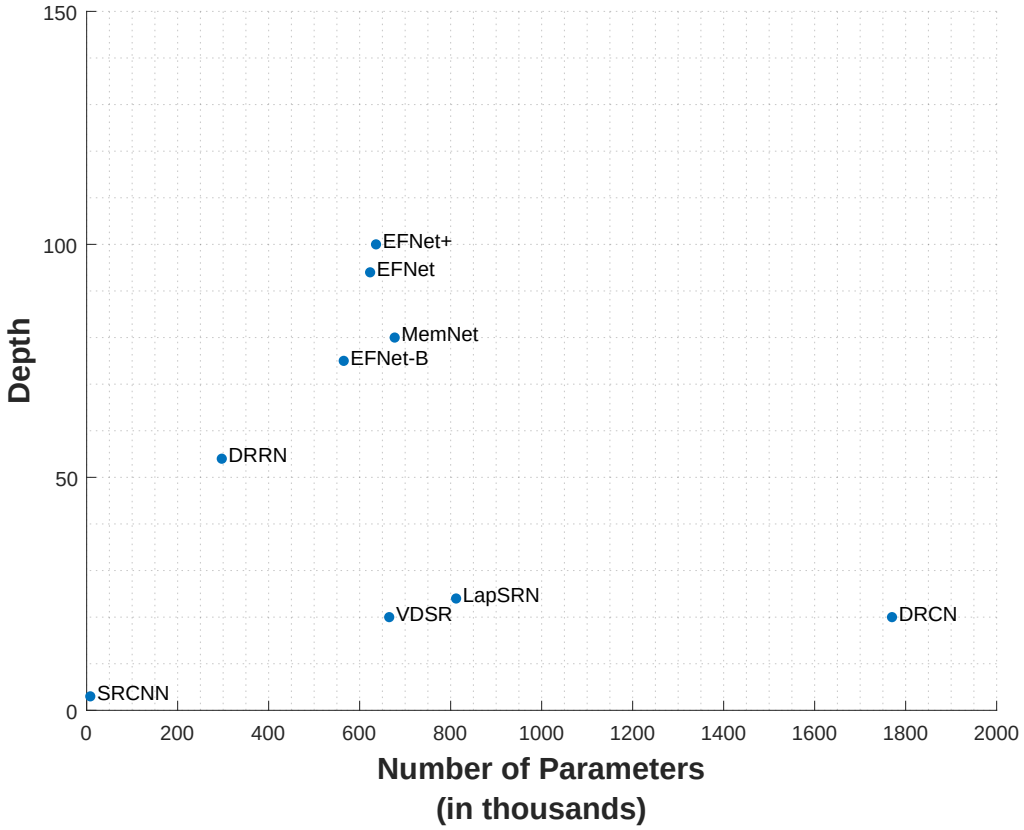


Figure 4.3: Plot of Network Depth vs Parameters against some state-of-the-art and proposed models on scale x4.

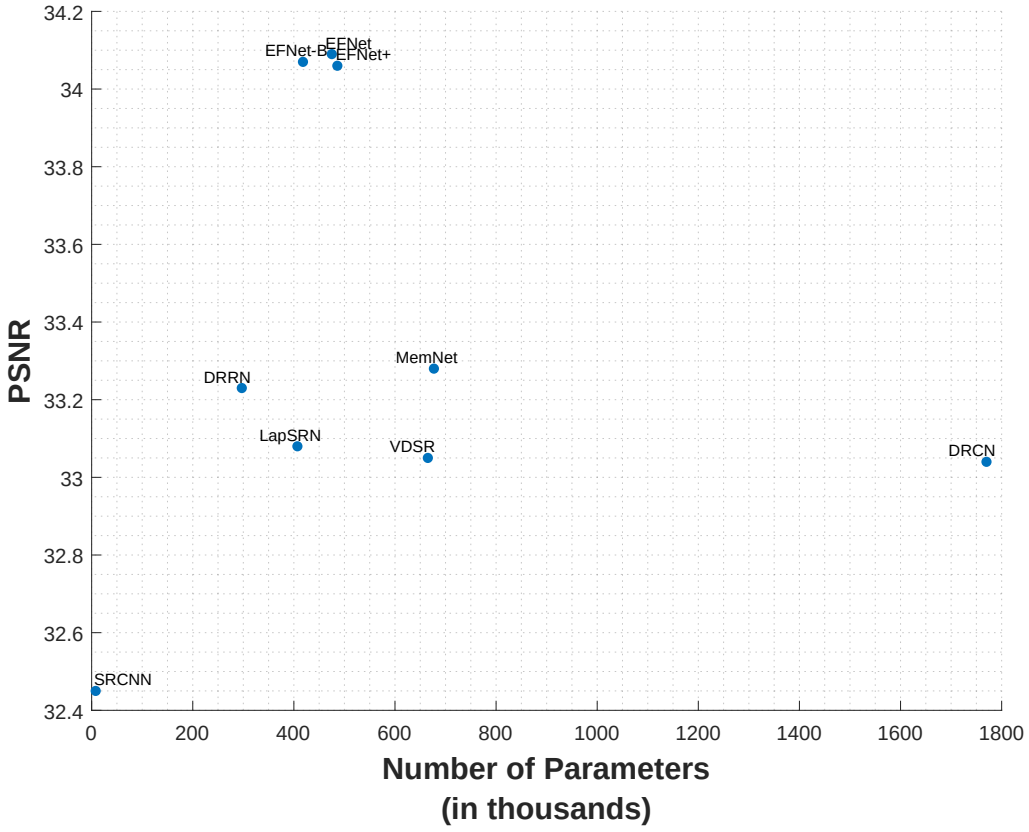


Figure 4.4: Plot of PSNR on Set14 dataset vs Network Parameters against some state-of-the-art and proposed models on scale x2.

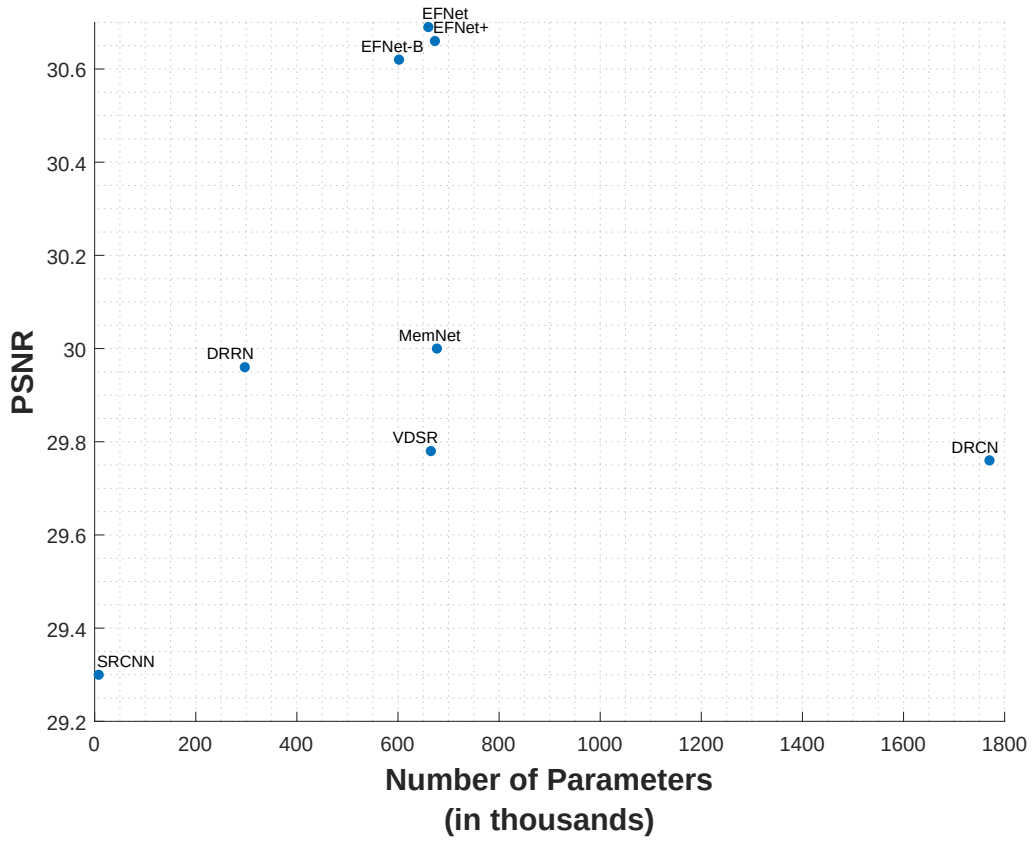


Figure 4.5: Plot of PSNR on Set14 dataset vs Network Parameters against some state-of-the-art and proposed models on scale x3.

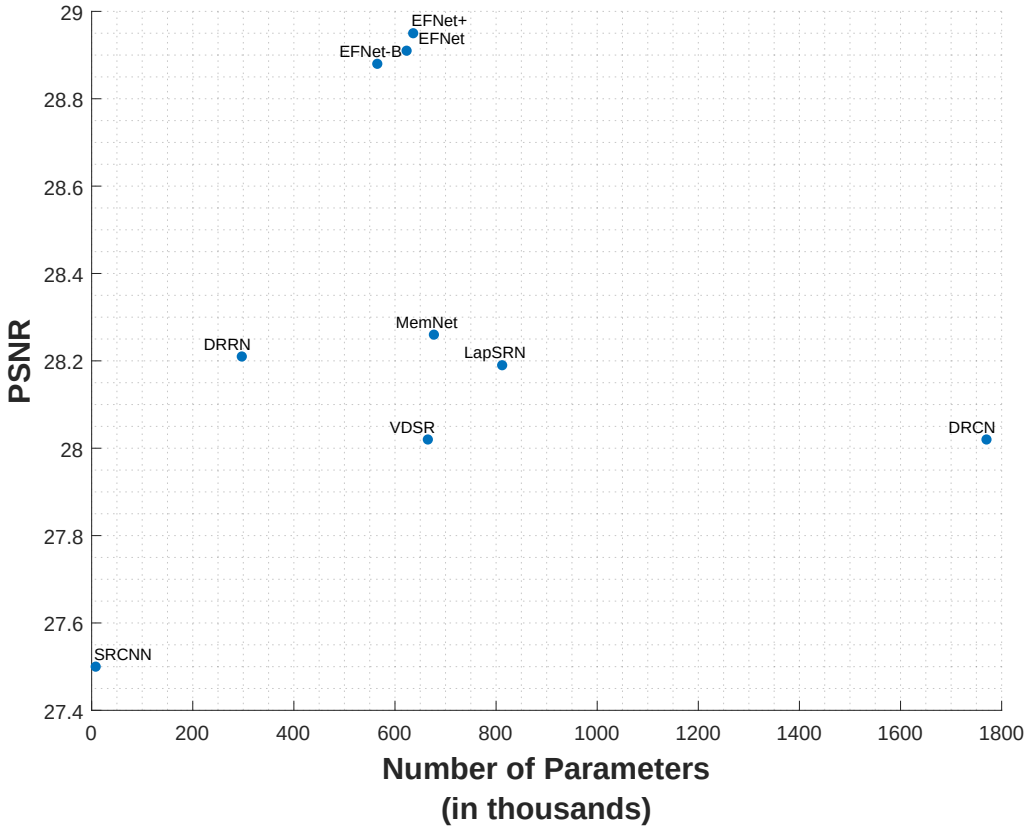


Figure 4.6: Plot of PSNR on Set14 dataset vs Network Parameters against some state-of-the-art and proposed models on scale x4.

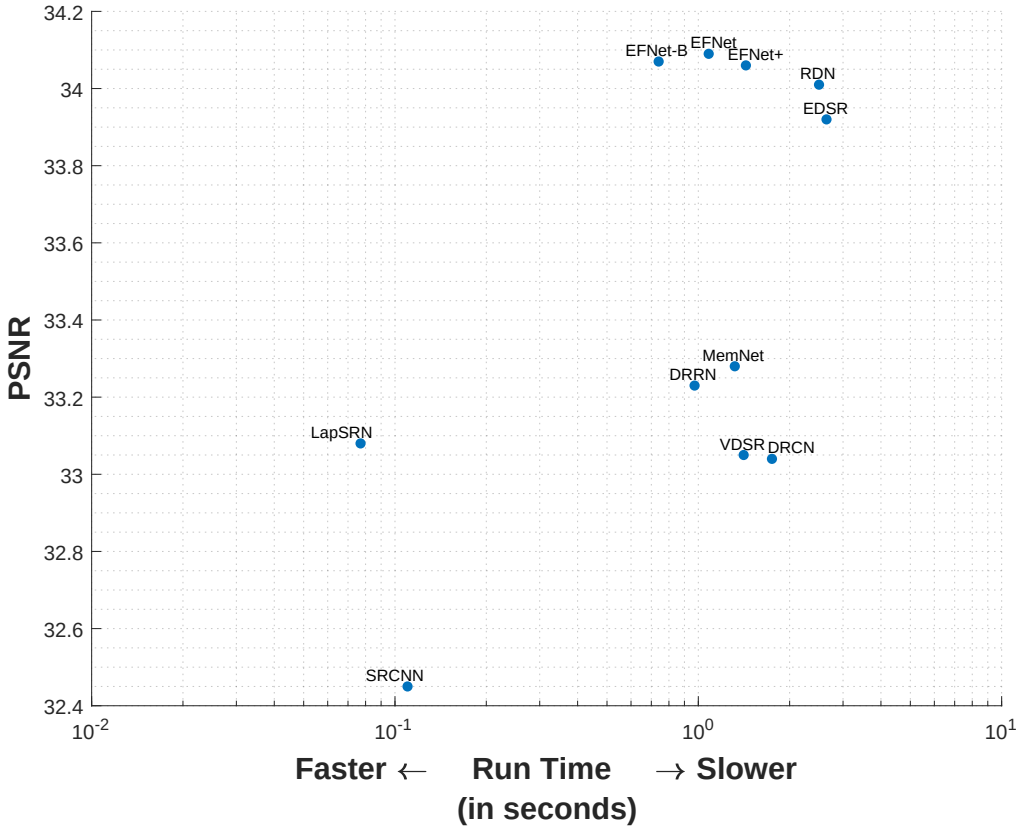


Figure 4.7: Plot of PSNR vs Network Runtime against some state-of-the-art and proposed models on scale x2 on Set14 dataset.

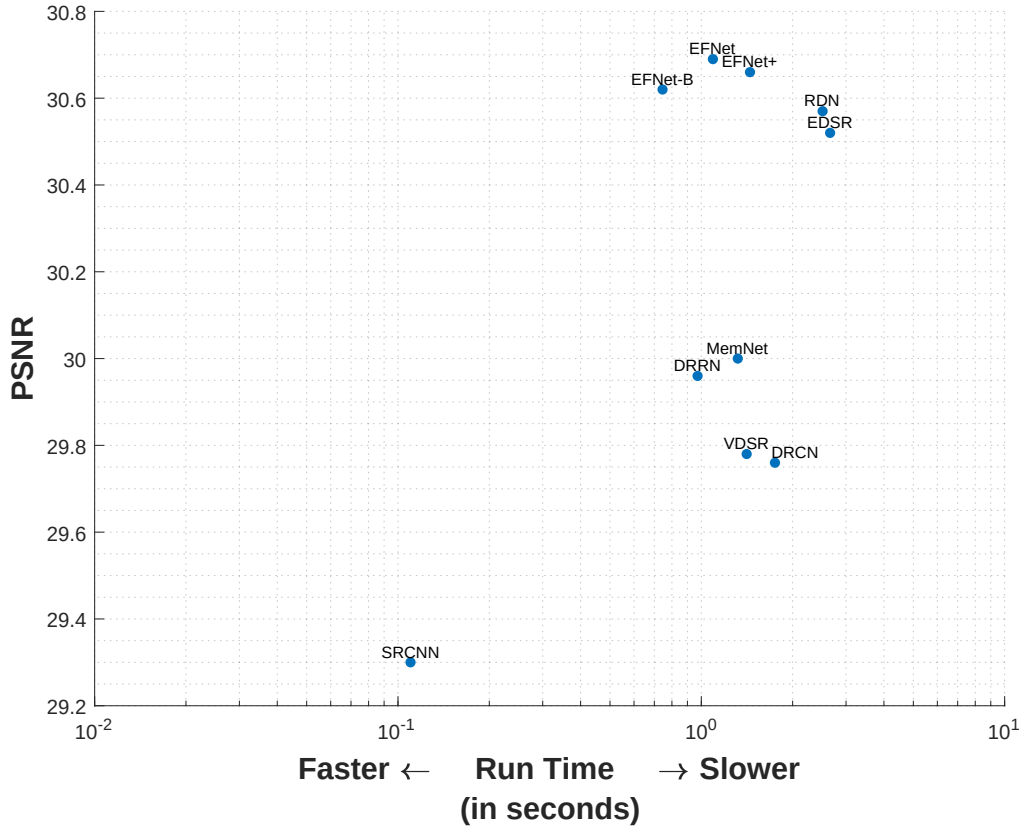


Figure 4.8: Plot of PSNR vs Network Runtime against some state-of-the-art and proposed models on scale x3 on Set14 dataset.

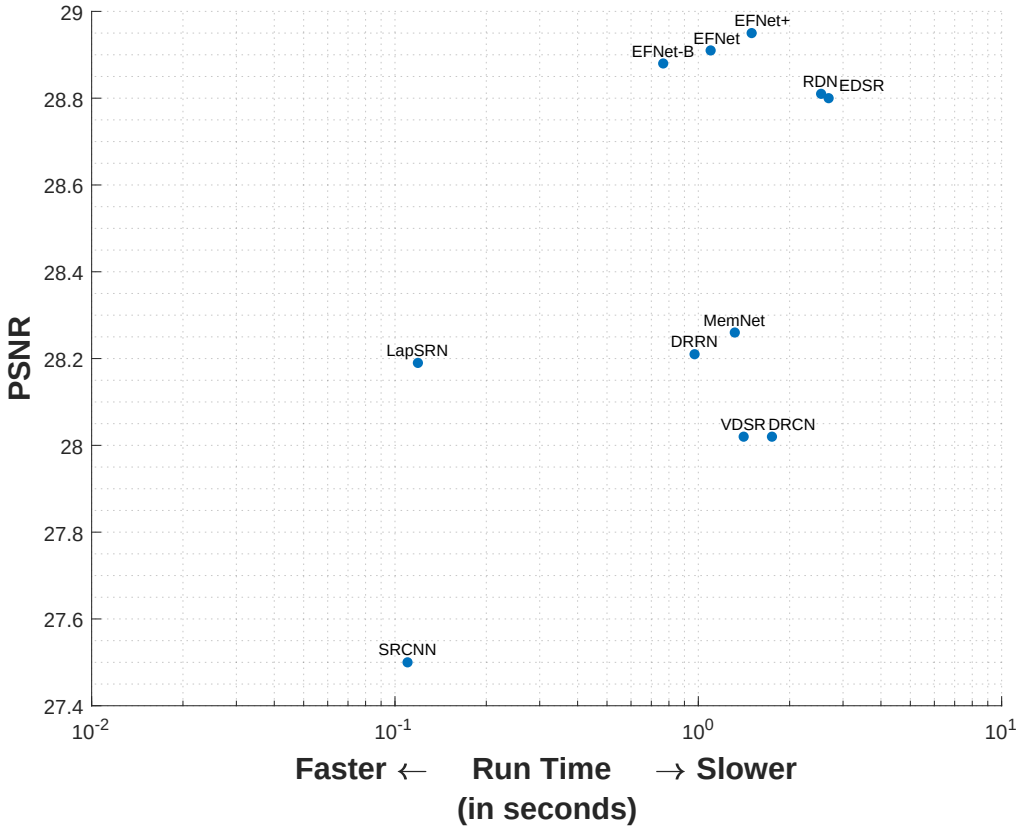


Figure 4.9: Plot of PSNR vs Network Runtime against some state-of-the-art and proposed models on scale x4 on Set14 dataset.

Figure 4.10: Visual comparison of the EFNet+ architecture with others on scale x4.

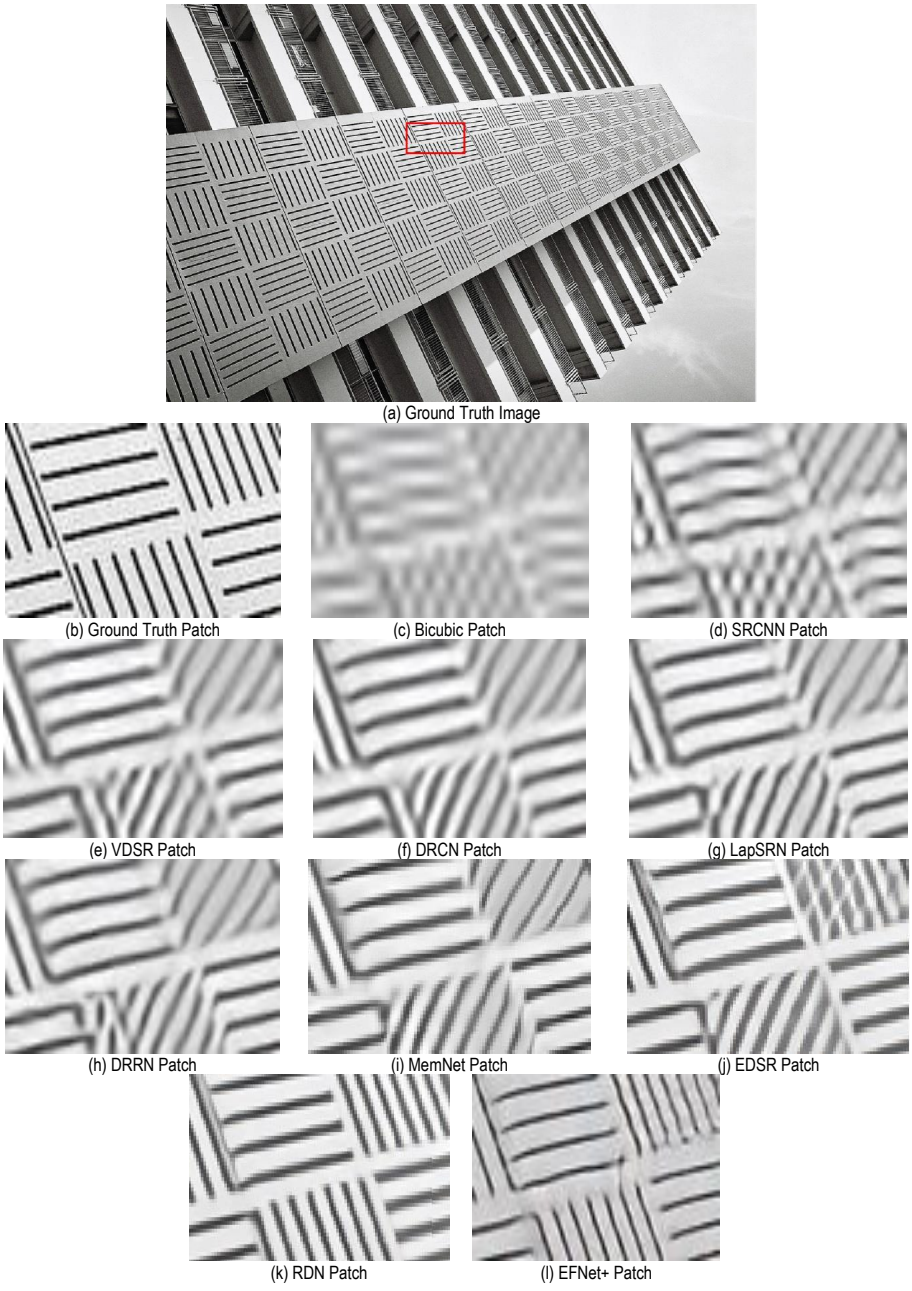


Figure 4.11: Visual comparison of the EFNet+ architecture with others on scale x4.

Figure 4.12: Visual comparison of the EFNet+ architecture with others on scale x4.

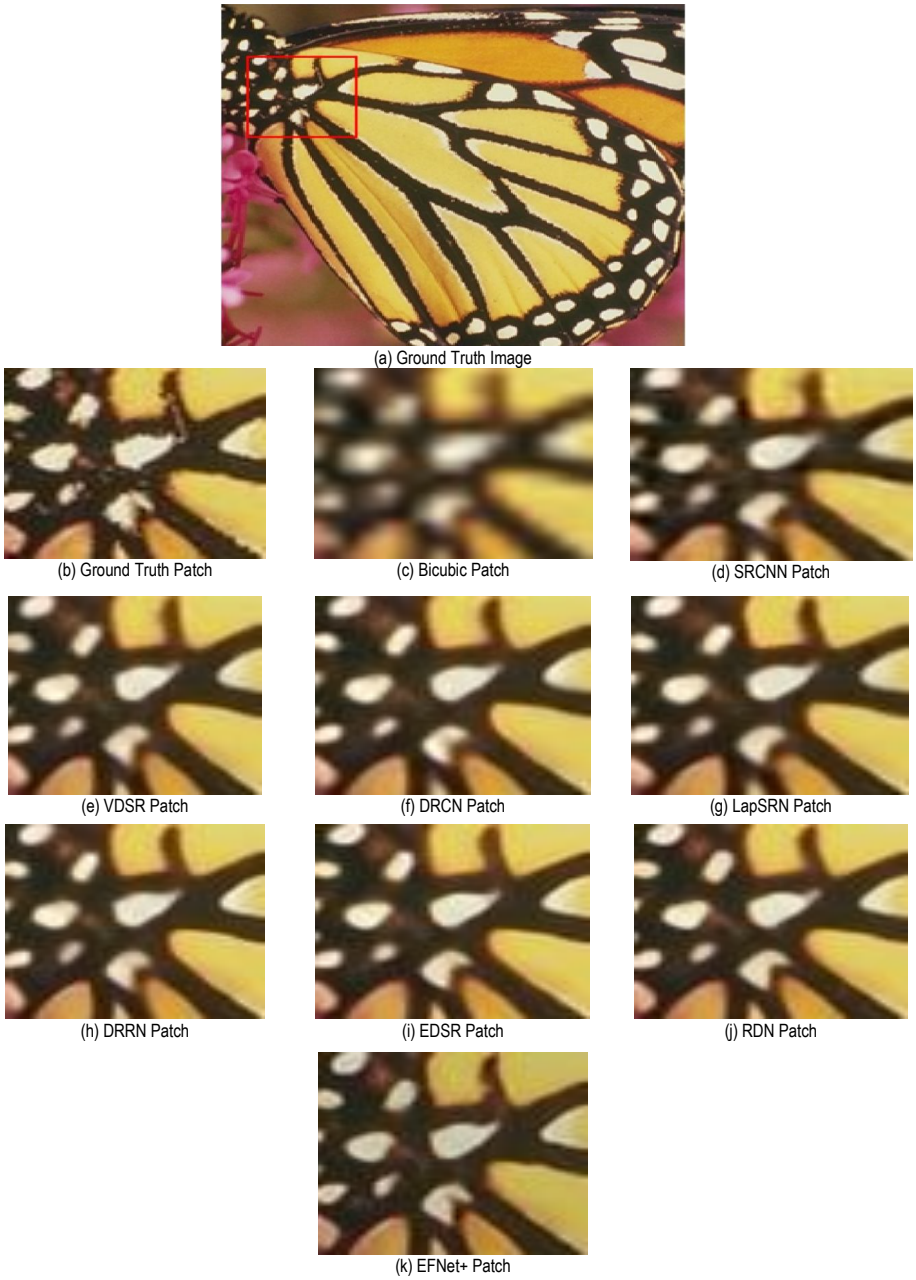


Figure 4.13: Visual comparison of the EFNet+ architecture with others on scale x4.

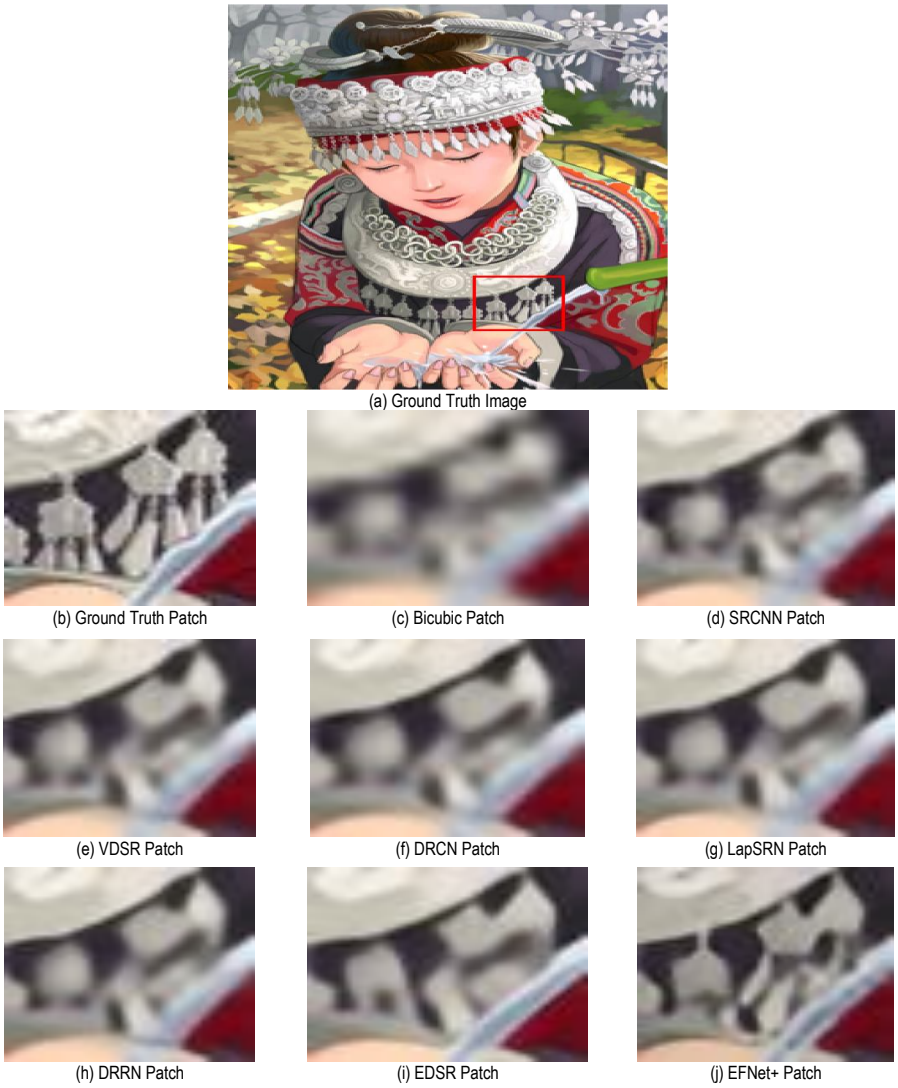


Figure 4.14: Visual comparison of the EFNet+ architecture with others on scale x4.

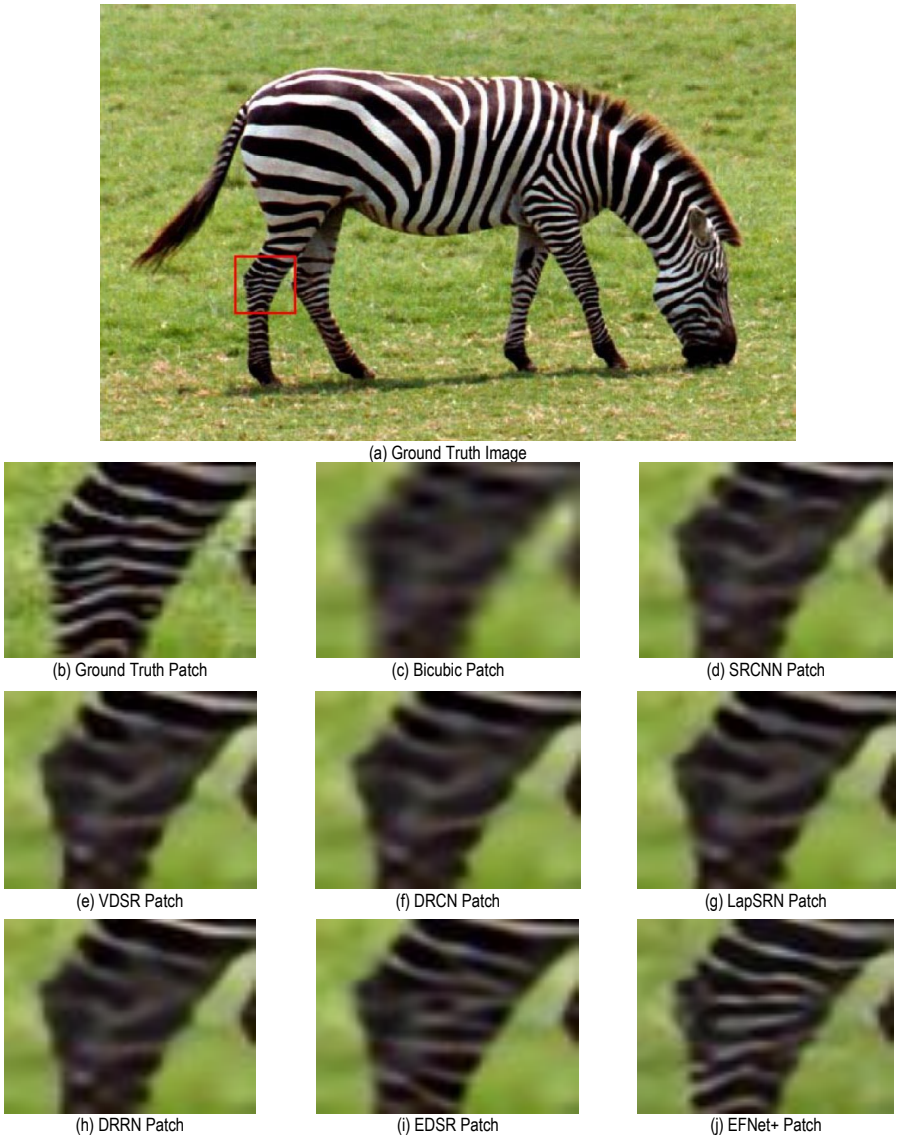


Figure 4.15: Visual comparison of the EFNet+ architecture with others on scale x4.

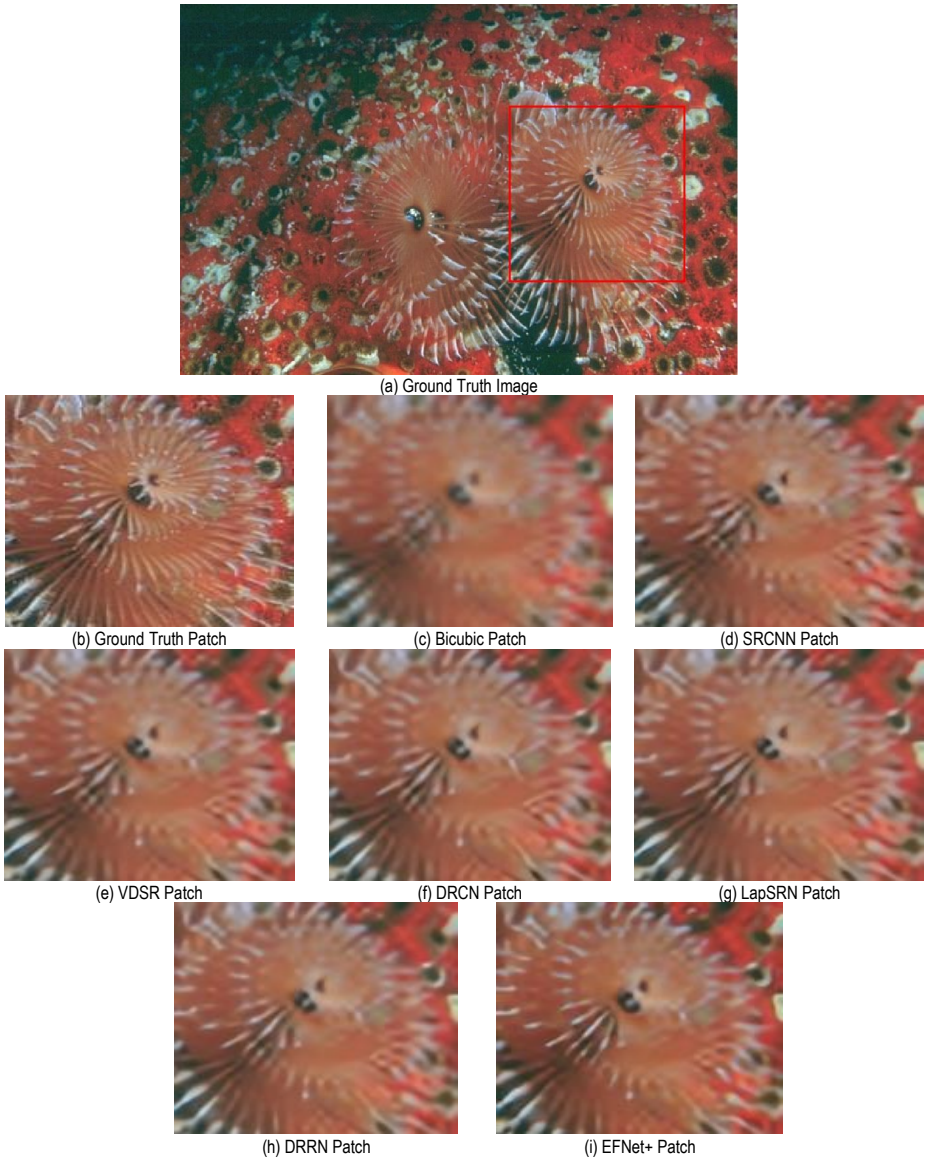


Figure 4.16: Visual comparison of the EFNet+ architecture with others on scale x4.

Figure 4.17: Visual comparison of the EFNet+ architecture with others on scale x4.

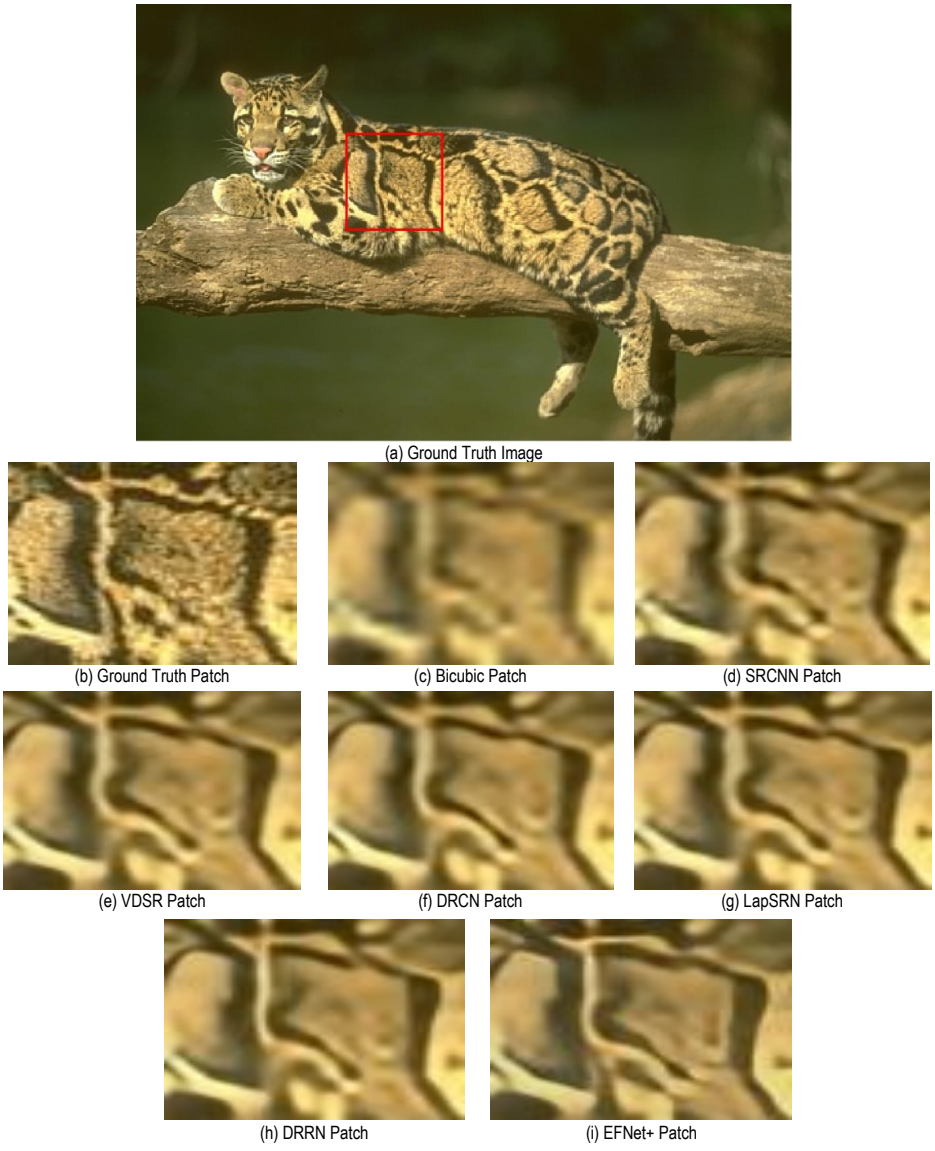


Figure 4.18: Visual comparison of the EFNet+ architecture with others on scale x4.

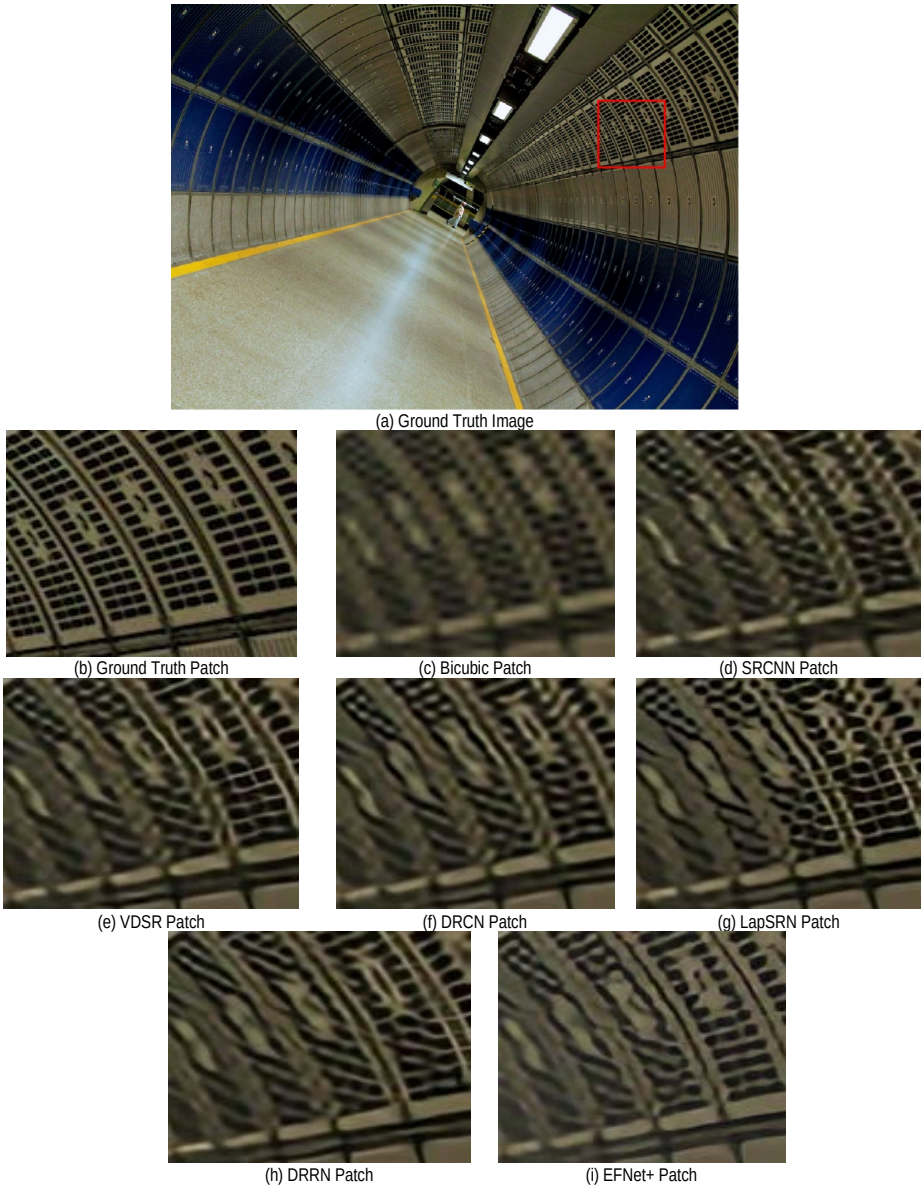


Figure 4.19: Visual comparison of the EFNet+ architecture with others on scale x4.

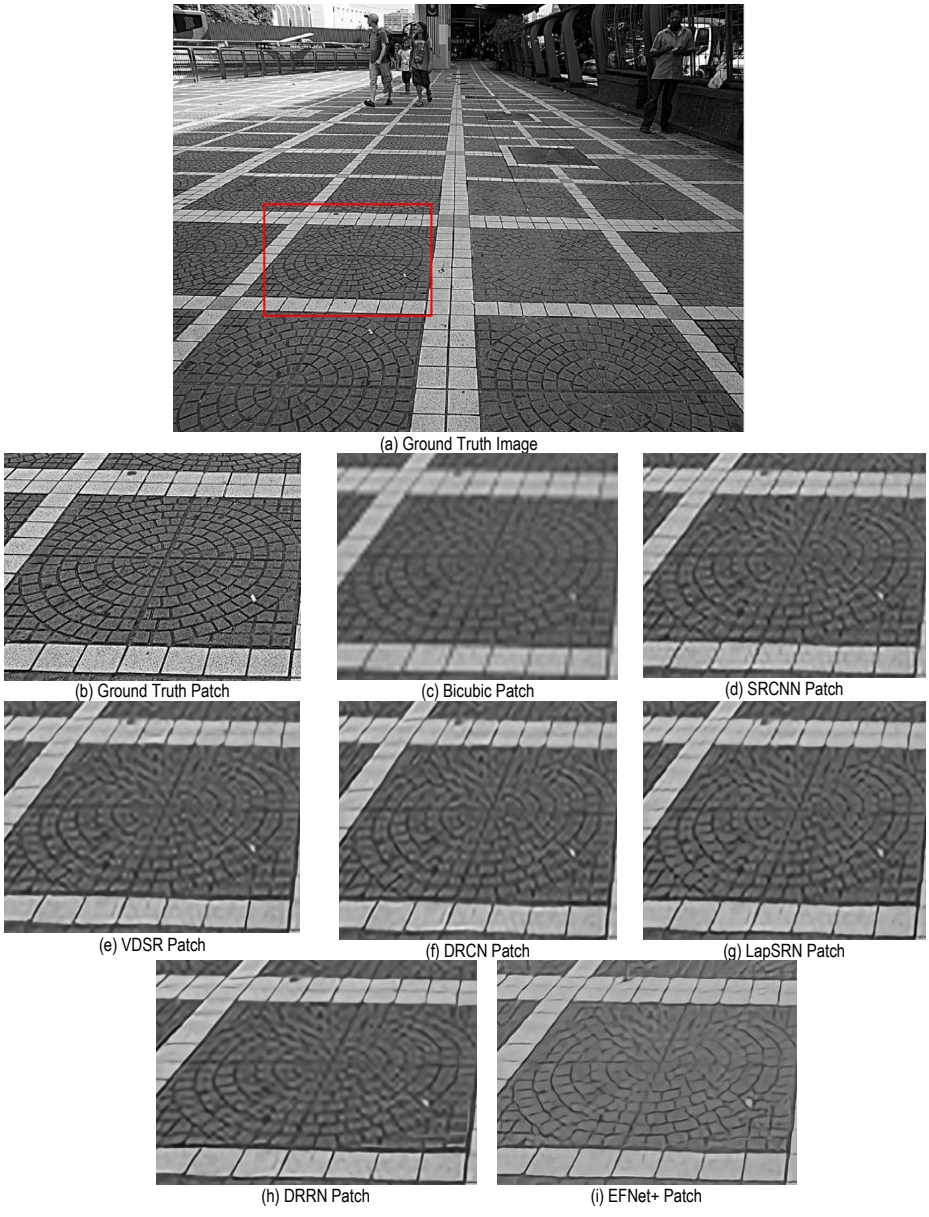


Figure 4.20: Visual comparison of the EFNet+ architecture with others on scale x4.

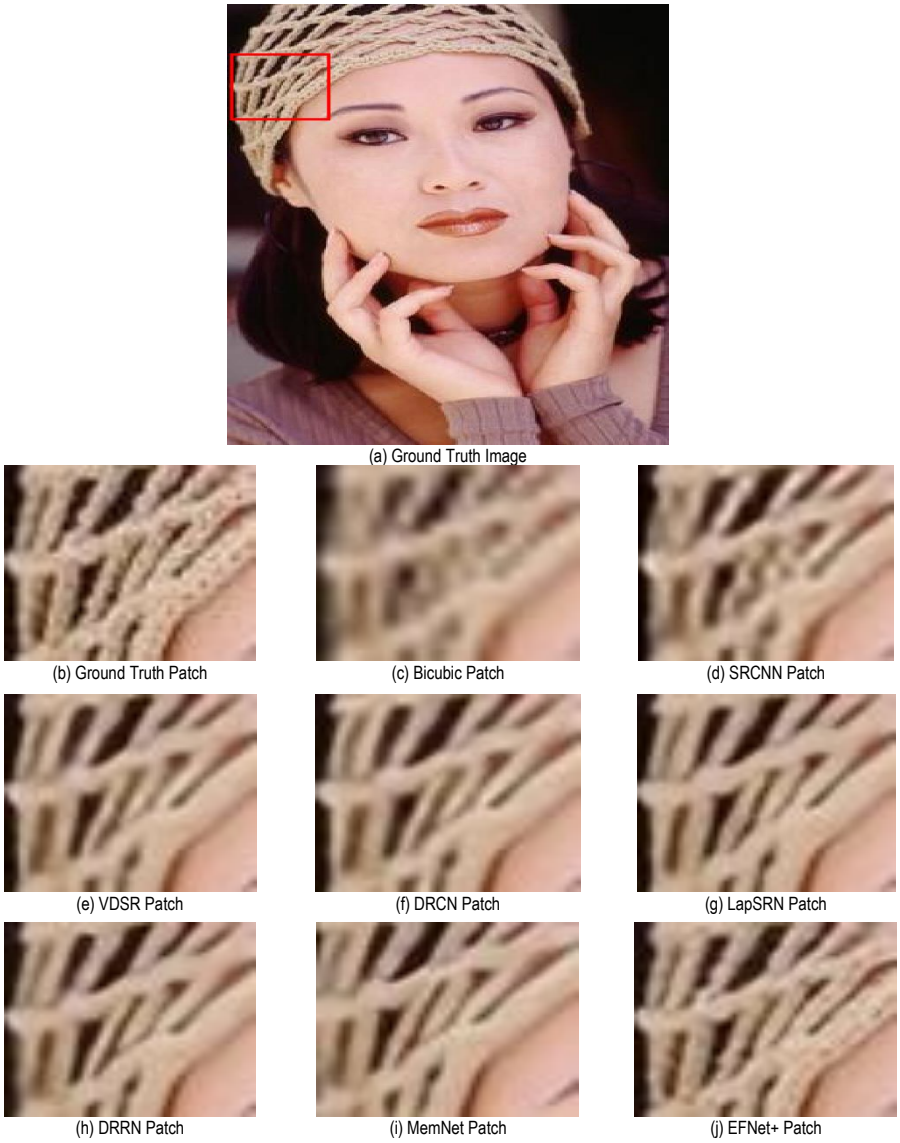


Figure 4.21: Visual comparison of the EFNet+ architecture with others on scale x3.

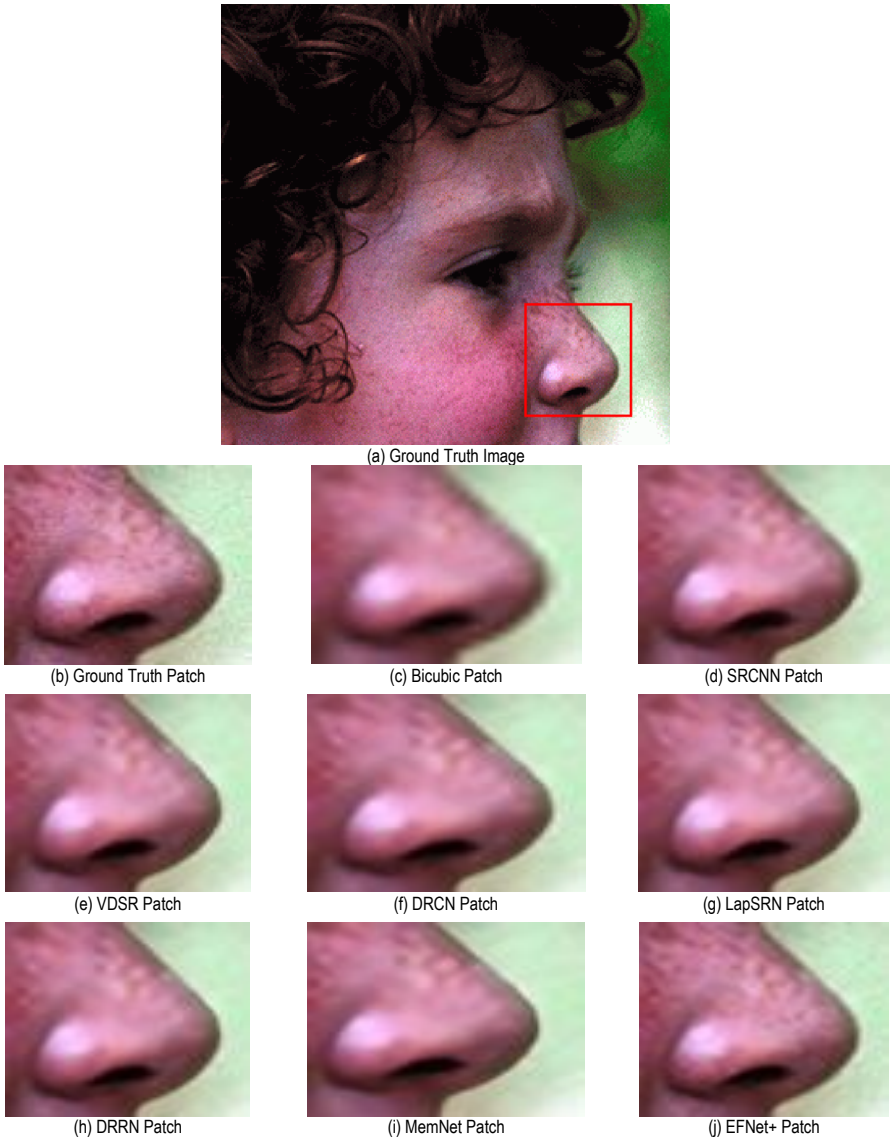
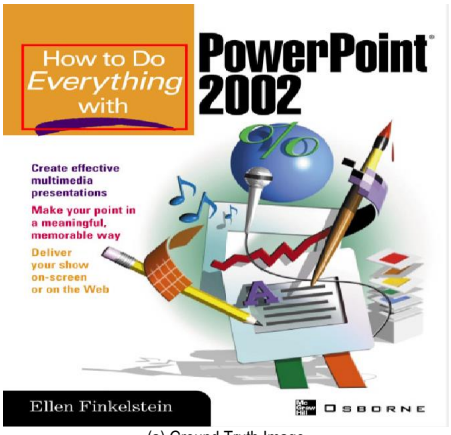
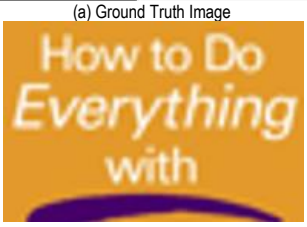


Figure 4.22: Visual comparison of the EFNet+ architecture with others on scale x3.



(a) Ground Truth Image

(b) Ground Truth Patch



(c) Bicubic Patch

(d) SRCNN Patch

(e) VDSR Patch

(f) DRCN Patch

(g) LapSRN Patch

(h) DRRN Patch

(i) EFNet+ Patch

Figure 4.23: Visual comparison of the EFNet+ architecture with others on scale x3.

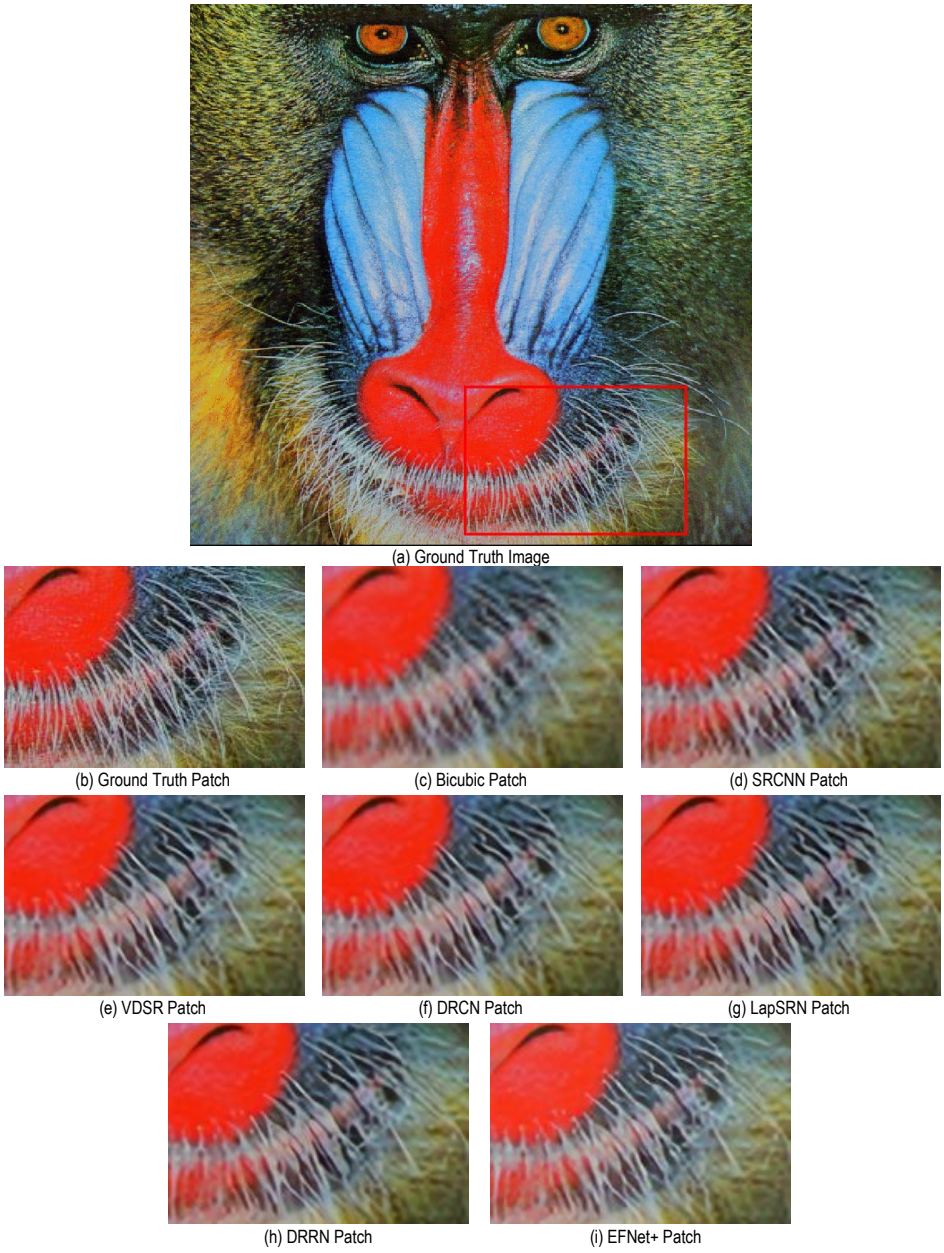


Figure 4.24: Visual comparison of the EFNet+ architecture with others on scale x3.

Chapter 5

Conclusions

5.1 Brief History of Thesis

In this thesis, three novel methods are proposed to improve image SR reconstruction in an efficient manner. In the beginning, a baseline model (EFNet-B) is shown, which can expand the receptive field with a very low number of parameters. In addition to that, a shallow multiplication segment is incorporated (in EENet), which tends to emphasize more on the edge and texture details of the baseline model. And finally, two common edge operators-Sobel and Laplacian are infused (in EFNet+) to more accurately sharpen the image with fine textures and details. All these models can increase the efficiency with a quite substantial performance boosting both qualitatively and quantitatively.

In the beginning, different convolution arrangements are analyzed to find the best fit for the baseline, and then, evaluation of baseline performance with different recursions are applied to achieve the optimal trade-off. Through the design space exploration, it is shown, how the separable convolution technique reaches the peak performance with a careful design of the network such that, there is a relaxed trade-off among the memory consumption, runtime and accuracy. In the next phase, 1-D convolution kernel is introduced in the shallow masking sub-part of the network to expand the receptive field with very few parameters. It appears in the experiment that, using 1-D kernel helps to utilize bigger filters with less computation, hence, extracts the global features economically. At the end, the importance of edge fusion for higher scaling factors through learned upscaling process is demonstrated. The improvements are quite substantial with a very low memory footprint.

The impact of three different networks is clearly shown with higher scaling factors gradually. In

general, proposed designs show how to achieve better reconstruction accuracy, and increase depth with less parameters and without major sacrificing the speed. Recursion, 1-D kernel and depth wise separable convolution are applied in that regard. Also, a novel architecture is proposed to better fit all those techniques. Unquestionably all proposed models achieve significant improvement over the state-of-the-art methods, especially with the higher scaling factors.

5.2 Expansion of Work

Although, a performance improvement is achieved, there is still scope for further improvement. Further design space exploration may include squeezing the architecture more efficiently. Recursions are used here to increase the depth, while after a certain point recursions are detrimental. Perhaps, some other network architecture may enable the network to apply more recursions without affecting the performance. In addition, analyzing flattened convolution technique proposed in [67] can also be done for more efficiency. In the shallow part of EFNet, only a few 9×1 & 1×9 size kernels are applied. Different architecture can also be used here to more accurately extract the edges for the Attention part. In EFNet+, only Sobel and Laplacian operators are used. However, these operators are highly sensitive to noise. So, a noise reduction layer may be included in the architecture to better super resolve the images. The canny edge detector can also be introduced for superior edge feature extractions. Since, proposed architecture shows better results, this can be used as the generator in a GAN model for performance upgrade.

Another extension may be proposed by upgrading the loss function, i.e., incorporating the structural information while optimizing the network. Here in the experiments, only MAE is used, which is basically pixel base loss function. This is not effective for human visual system for higher scaling factors. So introducing a feature based loss function like [46, 68] in the Attention part of proposed network will help to generate more high-level features. A pitfall of this technique is, it tends to corrupt original structure while sharpening the images. Since, this loss function will be applied in the shallow Attention part of the network besides the pixel base loss in the main part of the network, this duo will generate higher PSNR, SSIM and more accurate image for human visual system. A careful combination of these losses can be a better solution to address classic SISR problem.

Bibliography

[1] R. C. Gonzalez, R. E. Woods *et al.*, "Digital image processing," 2017.

[2] R. Szeliski, "Computer vision: algorithms and application," 2010.

[3] R. Tsai, "Multiframe image restoration and registration," *Advance Computer Visual and Image Processing*, vol. 1, pp. 317–339, 1984.

[4] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *IEEE transactions on image processing*, vol. 19, no. 11, pp. 2861–2873, 2010.

[5] S. Dai, M. Han, W. Xu, Y. Wu, and Y. Gong, "Soft edge smoothness prior for alpha channel super resolution." in *CVPR*, vol. 7. Citeseer, 2007, pp. 1–8.

[6] J. Sun, Z. Xu, and H.-Y. Shum, "Image super-resolution using gradient profile prior," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2008, pp. 1–8.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[9] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[11] ——, "Identity mappings in deep residual networks," in *European conference on computer vision*. Springer, 2016, pp. 630–645.

[12] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[13] Y. Tai, J. Yang, and X. Liu, "Image super-resolution via deep recursive residual network," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, no. 4, 2017.

[14] J. Kim, J. Kwon Lee, and K. Mu Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1646–1654.

[15] ——, "Deeply-recursive convolutional network for image super-resolution," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1637–1645.

[16] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu, "Image super-resolution using very deep residual channel attention networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 286–301.

[17] S. Kim, N. K. Bose, and H. M. Valenzuela, "Recursive reconstruction of high resolution image from noisy undersampled multiframes," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 38, no. 6, pp. 1013–1027, 1990.

[18] N. Bose, H. Kim, and H. M. Valenzuela, "Recursive total least squares algorithm for image reconstruction from noisy, undersampled frames," *Multidimensional Systems and Signal Processing*, vol. 4, no. 3, pp. 253–268, 1993.

[19] R. R. Makwana and N. D. Mehta, "Single image super-resolution via iterative back projection based canny edge detection and a gabor filter prior," *International Journal of Soft Computing & Engineering*, vol. 3, no. 1, pp. 379–384, 2013.

[20] X. Li and M. T. Orchard, "New edge-directed interpolation," *IEEE transactions on image processing*, vol. 10, no. 10, pp. 1521–1527, 2001.

[21] L. Zhang and X. Wu, "An edge-guided image interpolation algorithm via directional filtering and data fusion," *IEEE transactions on Image Processing*, vol. 15, no. 8, pp. 2226–2238, 2006.

[22] J. Yang, J. Wright, T. Huang, and Y. Ma, "Image super-resolution as sparse representation of raw image patches," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*. Citeseer, 2008, pp. 1–8.

[23] R. Timofte, V. De Smet, and L. Van Gool, "Anchored neighborhood regression for fast example-based super-resolution," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 1920–1927.

[24] ——, "A+: Adjusted anchored neighborhood regression for fast super-resolution," in *Asian conference on computer vision*. Springer, 2014, pp. 111–126.

[25] S. Wang, L. Zhang, Y. Liang, and Q. Pan, "Semi-coupled dictionary learning with applications to image super-resolution and photo-sketch synthesis," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 2216–2223.

[26] C. Francois, "Deep learning with python," 2017.

[27] I. Goodfellow, Y. Bengio, and A. Courville, "Deep learning. book in preparation for mit press," *URLi http://www. deeplearningbook. org*, 2016.

[28] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[29] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[31] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[32] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[33] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[34] D. Frossard. (2016) Vgg in tensorflow. [Online]. Available: https://www.cs.toronto.edu/~frossard/post/vgg16/

[35] A. Veit, M. J. Wilber, and S. Belongie, "Residual networks behave like ensembles of relatively shallow networks," in *Advances in neural information processing systems*, 2016, pp. 550–558.

[36] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 2, pp. 295–307, 2016.

[37] C. Dong, C. C. Loy, and X. Tang, "Accelerating the super-resolution convolutional neural network," in *European conference on computer vision*. Springer, 2016, pp. 391–407.

[38] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.

[39] W. Shi, J. Caballero, F. Huszár, 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1874–1883.

[40] W. Shi, J. Caballero, L. Theis, F. Huszar, A. Aitken, C. Ledig, and Z. Wang, "Is the deconvolution layer the same as a convolutional layer?" *arXiv preprint arXiv:1609.07009*, 2016.

[41] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[42] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang, "Deep laplacian pyramid networks for fast and accurate super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 624–632.

[43] A. Bruhn, J. Weickert, and C. Schnörr, "Lucas/kanade meets horn/schunck: Combining local and global optic flow methods," *International journal of computer vision*, vol. 61, no. 3, pp. 211–231, 2005.

[44] W. Yang, J. Feng, J. Yang, F. Zhao, J. Liu, Z. Guo, and S. Yan, "Deep edge guided recurrent residual learning for image super-resolution," *IEEE Transactions on Image Processing*, vol. 26, no. 12, pp. 5895–5907, 2017.

[45] Y. Liu, Y. Wang, N. Li, X. Cheng, Y. Zhang, Y. Huang, and G. Lu, "An attention-based approach for single image super resolution," in *2018 24th International Conference on Pattern Recognition (ICPR)*. IEEE, 2018, pp. 2777–2784.

[46] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4681–4690.

[47] T. Tong, G. Li, X. Liu, and Q. Gao, "Image super-resolution using dense skip connections," in *Computer Vision (ICCV), 2017 IEEE International Conference on*. IEEE, 2017, pp. 4809–4817.

[48] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," in *The IEEE conference on computer vision and pattern recognition (CVPR) workshops*, vol. 1, no. 2, 2017, p. 4.

[49] Y. Tai, J. Yang, X. Liu, and C. Xu, "Memnet: A persistent memory network for image restoration," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4539–4547.

[50] 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)*, 2018.

[51] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[52] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[53] C. Peng, X. Zhang, G. Yu, G. Luo, and J. Sun, "Large kernel mattersimprove semantic segmentation by global convolutional network," in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE, 2017, pp. 1743–1751.

[54] G. Seif and D. Androutsos, "Large receptive field networks for high-scale image super-resolution," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 763–772.

[55] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[56] C.-F. Wang. (2018) A basic introduction to separable convolutions. [Online]. Available: https://towardsdatascience.com/a-basic-introduction-to-separable-convolutions-b99ec3102728

[57] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, vol. 2. IEEE, 2001, pp. 416–423.

[58] 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.

[59] R. Timofte, E. Agustsson, L. Van Gool, M.-H. Yang, L. Zhang, B. Lim, S. Son, H. Kim, S. Nah, K. M. Lee *et al.*, "Ntire 2017 challenge on single image super-resolution: Methods and results," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*. IEEE, 2017, pp. 1110–1121.

[60] R. Timofte, S. Gu, J. Wu, L. Van Gool, L. Zhang, M.-H. Yang, M. Haris *et al.*, "Ntire 2018 challenge on single image super-resolution: Methods and results," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

[61] M. Bevilacqua, A. Roumy, C. Guillemot, and M. L. Alberi-Morel, "Low-complexity single-image super-resolution based on nonnegative neighbor embedding," 2012.

[62] R. Zeyde, M. Elad, and M. Protter, "On single image scale-up using sparse-representations," in *International conference on curves and surfaces*. Springer, 2010, pp. 711–730.

[63] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 5, pp. 898–916, 2010.

[64] J.-B. Huang, A. Singh, and N. Ahuja, "Single image super-resolution from transformed self-exemplars," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5197–5206.

[65] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli *et al.*, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[66] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[67] J. Jin, A. Dundar, and E. Culurciello, "Flattened convolutional neural networks for feedforward acceleration," *arXiv preprint arXiv:1412.5474*, 2014.

[68] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European conference on computer vision*. Springer, 2016, pp. 694–711.