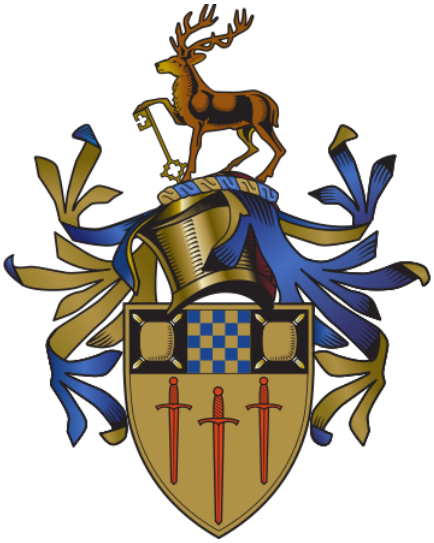
An In-depth Analysis on the Recent Development of Deep Image Super-Resolution

EEE3017 Year 3 Project

Herman Yau
6398723



A dissertation submitted in partial fulfilment
for the degree of
Bachelor of Engineering in Electronic Engineering
with Computer Systems

Department of Electrical and Electronic Engineering
University of Surrey
United Kingdom
15th May, 2018
Supervisor: Dr. Simon Hadfield

Abstract

Super-resolution is an ill-posed problem for decades in the field of both image processing and machine learning. Low resolution images prevents us from utilising images for machine-aware applications which requires subtle details often unavailable from it due to electronic limitations. This year 3 project discusses previous works which have been done on similar matters and important mathematical backgrounds on deep learning, as well as state-of-the-art developments and techniques on deep super-resolution approaches. This is followed by results from training nerual network models, in particular Residual Encoder-Decoder Network with Skip Connections (RED-Net) with the DIV2K training set. We combine a blend of techniques and evaluate the trained models on the DIV2K validation set, Set5 and Set14 datasets where the superior performance of skip connections and sub-pixel convolution are shown. Perceptual loss is also evaluated which shows the downside of the mean square error (MSE) loss function.

The code to this project is available from github: `https://github.com/hmhyau/Surrey-Year-3-Project/upload/master`.

Acknowledgement

I would like to thank my supervisor, Dr. Simon Hadfield for accepting my project proposal and providing continuous support during different phases of this project. This project would not have been possible without his guidance and explanation of abstract and unclear concepts.

I am also grateful to my family for affording the hefty tuition fees for me to attend university overseas, and their untold moral support throughout my journey to obtaining my Bachelor degree. I certinly would not be able to acquire my current achievements without them.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Super-resolution is one of the most fundamental operation yet ill-posed problem in computer vision, where the underlying goal is to enhance the resolution of pre-existing images and finds applications in image processing and compression. However, super-resolution problem is highly underdetermined, which means the missing information can either be interpolated or predicted only. This is especially pronounced for high upscaling factors, where edges and textures are typically absent from the input. As such, it heightens the difficulty to reproduce fine details with extremely limited prior available.

The task is also highly correlated to image denoising. Specifically, the input to such model is a low-resolution version of the image which is analogical to a dimensionality reduced, encoded image of autoencoders in image denoising applications. Therefore, it is crucial to study the problem formulation of super-resolution and investigate on the generation of missing or truncated information with an ensemble of techniques, in order to obtain the best original representation for further processing.

1.2 Aims & Objectives

This project aims to explore advanced deep learning techniques, in particular fully convolutional neural networks, sub-pixel convolution and perceptual loss, to enhance the resolution of existing image by training with pairs of ground truth and its downsampled version. The results will then be analysed accordingly to deduce the advantages and disadvantages of different methods, which is supported with comparisons between various network configurations.

This report details project work achieved throughout the academic year across Autumn semester and Spring semester. It is organised as follows:

- Chapter 2 composes of a background study of the development of artificial neural networks and image upsampling methods in deep learning. In addition, the chapter covers traditional image denoising and super-resolution algorithms in literature, as well as discussions on previous researches related to this project. Attempts and state-of-the-art developments on both problems are particularly highlighted in this chapter.
- Chapter 3 describes the fundamental theories and mathematical concepts of deep neural networks in general, including gradient descent, back-propagation, activation functions and recep-

tive field. These mathematical theories are essential to understand the basis of artificial neural networks which is the main focus of this project. In addition, it covers recent advancements and mathematics of the neural network model which is used in later experiments.

- Chapter 4 discusses and compares the empirical results obtained from the various neural network modesl for a variety of experiments with supporting figures and output results. They stress the importance of proper initial hyper-parameters for the network configurations, as well as a collection of insightful analysis on the impact of using skip connections, different upsampling schemes and loss functions.
- Chapter 5 concludes the project and provides an outline of possible future works if it were given more time to conduct further researches and practical works.

Chapter 2

Literature Review

2.1 Background

2.1.1 Artifical Neural Network

Artificial neural network (ANNs) is one of the major tool in the domain of machine learning revolving around areas such as natural language processing (NLP), digital signal processing, computer visions and robotics and statistics predictions. While the concept of perceptron dates back to the 1960s, there are multiple bottlenecks back then such as insufficient data inputs and mediocre computational power which stagnated the advancement of ANNs. However, by exploiting the computational power of modern graphical processing unit (GPU), parallel processing and cloud computing, the development of deep learning has broken the slow processing bottleneck and practical use is possible.

In general, an ANN is a computing system vaguely inspired by biological neural networks exist in animal neural systems. Instead of modelling the biological counterpart directly (which is impossible as even biologists are still struggling understanding the brain), ANN progressively learns by a combination of connecting mathematical models and training data without any a priori knowledge. This can be intuitively think as an infant (ANN) learning to perform a task (concept) over a long period of time. For brevity, we will now refer ANN simply as neural network.

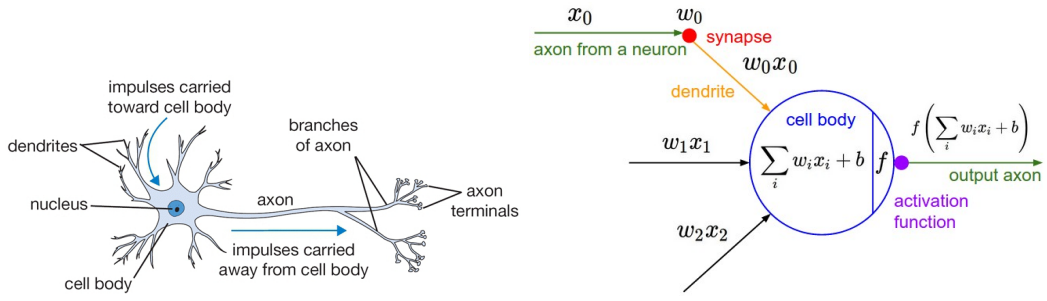


Figure 2.1: Figures depicting the difference between a real biological neuron and an artificial neuron in mathematical model. [1]

In traditional computer vision, image features are extracted with well-defined methods such as HOG [2], SIFT [3], SURF [4], etc. However, the downside is that all of them, despite machine learning, are hard-coded algorithms. This means that model tweaking is impossible as the system would rely on one or more of these feature extraction algorithms, trained to accommodate their respective data domain.

Another problem from it is that traditional computer vision defies the philosophy of modelling human perception. In deep learning context, we would like such system to produce sensible result by learning the necessary heuristic details and discriminates on its own, but the conventional techniques learn them based on what we have "instructed" to the system.

As opposed to traditional techniques, deep learning takes a heuristic approach where we allow neural networks to obtain outputs by feedforward computation and learn optimal parameters by itself for a specific task via gradient descent and back-propagation. In particular, we are interested in low-level, sub-symbolic details that are hard to comprehend and process by human cognition but can be learnt by a neural network with ample time. This information can then be used to construct high-level audio-visual concept. Neural networks has proved its powerful performance in classification tasks over previous statistical methods such as principal component analysis (PCA) [5] by self, unspecified data generalisation and non-linearity.

2.1.2 Convolutional Neural Network (CNN)

A more powerful powerful type of neural network is called convolutional neural network (CNN) pioneered by LeCun et al.'s LeNet [6]. A CNN typically consists of convolution, pooling and fully connected layers. Compared to fully connected networks, CNNs share weights within a layer such that the same set of weights are used for each receptive field. This reduces the computational time and number of parameters involved, thus granting CNNs superior performance. Since LeNet, much progress has been made in the development of CNNs, notably AlexNet by Krizhevsky et al. which exceeds human-level performance in the 2012 ILSVRC [7]. Since then, many novel architectures have been developed to propel the evolution of CNNs. Amongst them, the most signature works are VGGNet [8], GoogLeNet [9] and ResNet [10].

AlexNet

AlexNet was the earliest deep network to successfully push the limit of deep learning approaches, leading the ImageNet classification accuracy by a significant margin compared to traditional methods [7]. The network was the first to use the rectified linear unit activation function and trains at a much faster rate than previous activation functions, i.e. sigmoid and tanh.

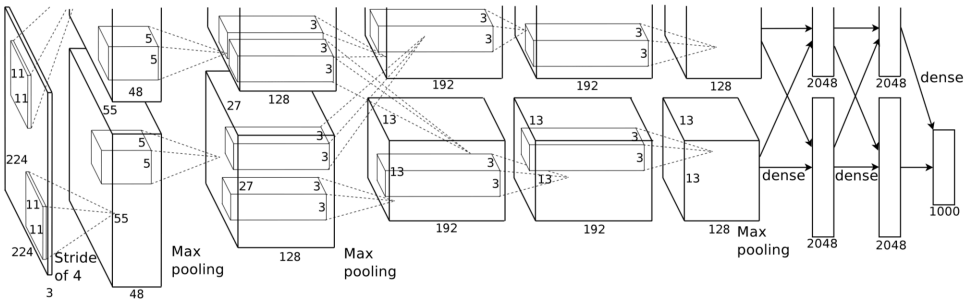


Figure 2.2: Neural network architecture of AlexNet. [7]

VGGNet

VGGNet is an improvement over AlexNet proposed by Simoyan et al. in VGG Group, University of Oxford [8]. Seeing AlexNet using large filters to create substantially large receptive field, the VGG network improves on the architecture by replacing single large filters into multiple stacked 3×3 filters instead. This allows the creation of similar size receptive fields while lowering computation complexity as well as increasing the depth, allowing the network to learn more complex and high-level features. Their work marks the concept of modules in deep learning to this day. It also introduced common practices in designing CNNs such as the use of 3×3 filters, max-pooling per 2 convolutions and doubling of filtes after each max-pooling layer.

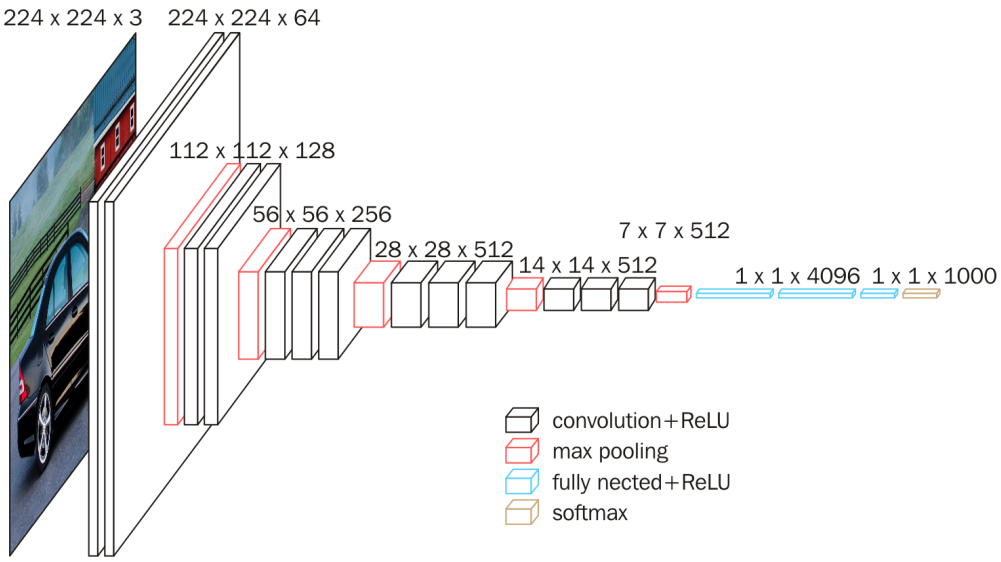


Figure 2.3: Neural network architecture of VGG16 network. [8]

GoogLeNet

Both VGGNet [8] and AlexNet [7] have phenomenal performance, but they suffer from the same problems: their model architectures contain fully-connected layers which creates great number of parameters; computation costs is high both in terms of memory and time which grows exponentially with the size and number of filters, and the depth of the network.

Observing most activations in deep networks are negligible or redundant, GoogLeNet resolves the above complications by introducing the inception module to approximate sparsity between neurons but able to capture details of various scales at the same time [9]. It also dramatically reduces the number of parameters by replacing all fully-connected layers and replace them with a global averaging pooling layer. This results in GoogLeNet succeeding in achieving comparable performance than former networks with significantly shorter and faster computations.

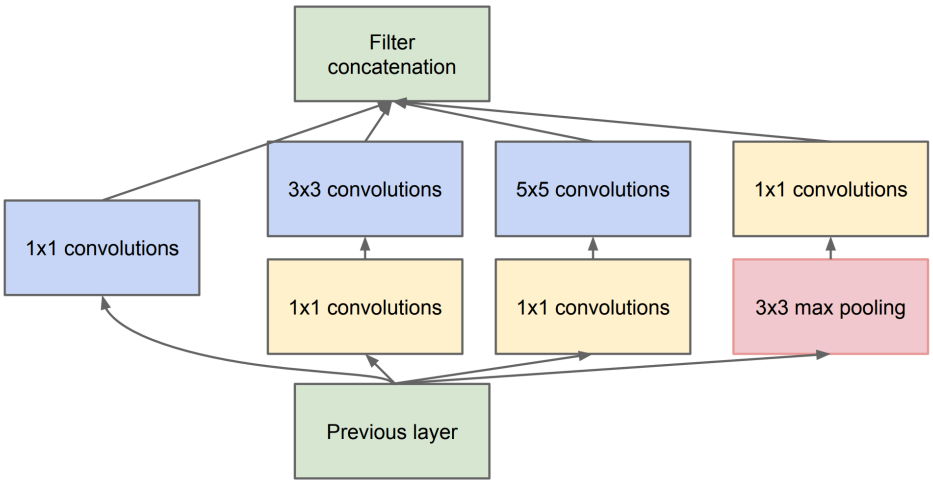


Figure 2.4: An inception module from GoogLeNet with dimensionality reduction. [8]

ResNet

It is intuitive that increasing the depth of a network improves its performance. However, one of the fundamental mechanism of deep neural network is back-propagation. The magnitude of a layer's output is proportional to the depth of the network and as such leads to the problem of vanishing gradient, meaning that the magnitude has become so small that learned features are negligible. Also, optimisation also has to be considered when designing a network with a naive network usually results in a higher training error; this is known as degradation.

ResNet introduced the residual module where the main idea is shortcut connections that skips layers as an direct input to latter parts of the network [10]. The advantage of using shortcuts is twofold. Firstly, the residual module output receives two outputs: a self-computed output and a previous version; this drives a particular hidden layer to learn the residual components and build up semantic information overtime. Secondly, the connections can handle the vanishing gradient problem; rather than back-propagating the gradients layer by layer, the network can pass them via the shortcuts and strengthen the magnitude of gradients. With these residual modules, they enabled the training of very deep neural networks with over 1000 hidden layers. Such feat is previously impossible due to vanishing gradient.

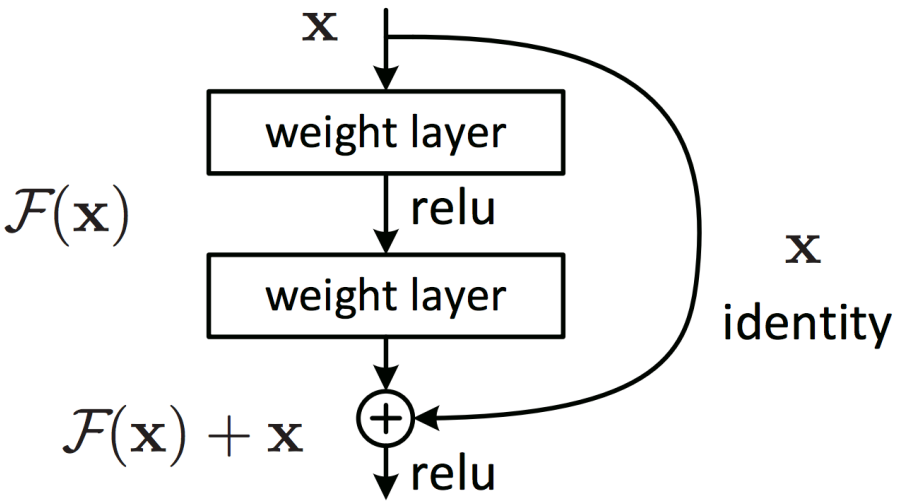


Figure 2.5: A residual module from ResNet. [10]

2.1.3 Perceptual Loss

The typical mean square error (MSE) loss suffers from fine details reconstruction as shown in Figure 2.6. Pixel offsets would result in huge changes per-pixel wise when perceptually I^{LR} and I^{HR} are essentially the same. It also produces overly smooth or blurry images due to inability to discriminate edges with surfaces. Therefore, MSE is not a robust loss function for super-resolution optimisation.

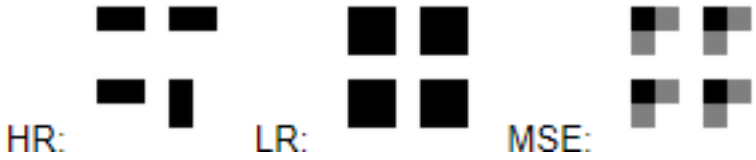


Figure 2.6: Problem induced by MSE loss. Here we can see that the reconstruction by MSE from LR produces same results in each grid, which is not the case looking back in the HR version. [11]

Instead of considering per-pixel loss between the output and the ground truth image as in MSE loss, Johnson et al. [12] proposed the perceptual loss. The loss function considers the perceptual differences between ground truth and output images. In other words, it compares and optimises extracted high-level features of images from hidden layers of pretrained networks. The intuition behind using a pretrained network like VGG [8] is that such network is trained for image classification tasks in large datasets, thus it already stores feature maps of common objects and is able to capture perceptual contents of other input images. While numerically perceptual loss does not excel compared to traditional loss functions and generates significant artifacts, it is still visually more pleasing by preserving edge details. It is also proven useful for artistic style transfer which has no single correct solution.

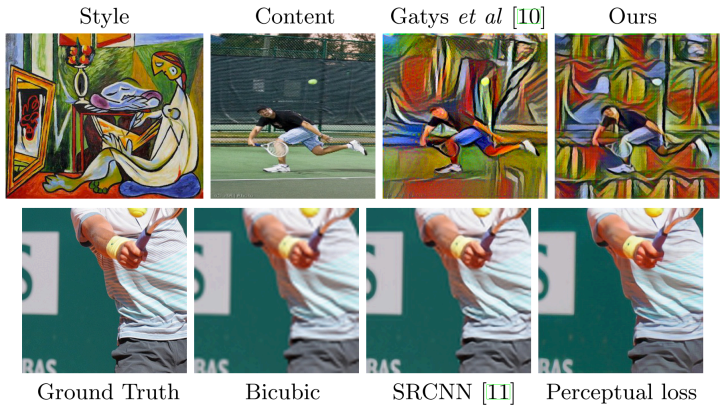


Figure 2.7: Visualisation of style transfer and super-resolved image using perceptual loss. [12]

2.1.4 Upsampling Methods

Very often in image denoising and image super-resolution, we require the neural network to build up a high resolution image from its latent representation's high level descriptors. This is achieved by transposed convolution. This allows us to impaint non-existent pixels of the reconstruction with every single pixel of the low resolution image.

Apart from the problem of choosing the right loss function, we also have to dampen artifacts and preserve perceptual content of the ground truth image I^{HR}. Below we illustrate two major developments in this area.

Resize Convolution

The naive implementation of transposed convolution create a serious problem: it generates checkerboard artifacts. This is particularly prominent in image with strong colours.

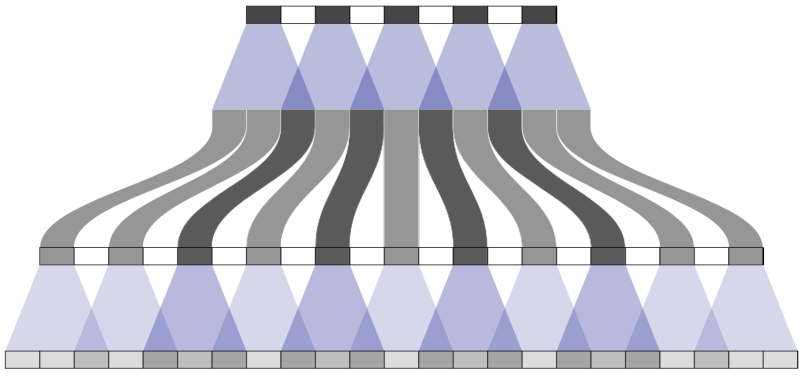


Figure 2.8: An illustration of how checkerboard artifacts are formed. The overlapping area from upper layers caused an non-uniform addition of elements, thus causing the phenomenon.

From Figure 2.8, we can see that the output values from transposed convolution is stacked and added together. This creates an unevenness where some pixels are in fact overlaps of multiple pixels.

Such problem can also be stacked across multiple layers, creating checkerboard artifacts of various severity.

One approach to remove these checkerboard artifacts is the resize convolution proposed by Odena et al [13]. Instead of doing the operation with one single transposed convolution, resize convolution separate out the upsampling operation from convolution. Resize happens first by with nearest-neighbour interpolation or bilinear interpolation, followed by the actual convolution operation. Thus, an injective mapping from I^{LR} to I^{SR} is now possible without checkerboard artifacts.

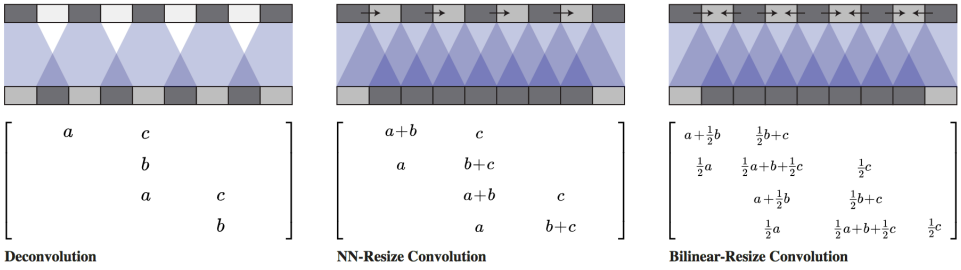


Figure 2.9: Matrices of different convolutional upscaling schemes. We can see that both NN-resize convolution and Bilinear-resize convolution results in addition of multiple values into a single matrix element, thus rectify the prevalence of checkerboard artifacts.

Sub-Pixel Convolution

From Equation 2.2 and 2.3, we know that I^{LR} inevitably must be super-resolved at some point such that the shape of I^{SR} matches I^{HR}. This creates the following problems:

- **Computation Cost**: Direct computations on I^{LR} is both extremely computationally and memory demanding and expensive especially for large images or super-resolution factor. An easy way to ensure I^{SR} and I^{HR} have the same shape is to interpolate I^{LR} during preprocessing, however no extra information is brought upon and may conversely introduce artifacts. Other solutions involve upscaling of image progressively through the neural network [14], but that does not solve the efficiency issue entirely.
- **Zero-padding**: Very often in deep learning, we perform upsampling by transposed convolution which does the same operation as normal convolution except zero-padding is added while fractionally strided to guarantee correctly shaped outputs. However, numerically it is impossible to be the same since a proper deconvolution requires all values generated from standard convolution operation which does not exist. These meaningless zeroes yield no symbolic meaning in super-resolving an image.

To cope with the aforementioned complications, Shi et al. [15] proposed Sub-pixel convolution. Sub-pixel convolution works by aggregating filter outputs in a manner such that the final image can be nicely concatenated into I^{SR} with shape of I^{HR}. The benefits of sub-pixel convolution is twofold: The upscaling operation can be done at the end of the neural network, saving unnecessary intermediate computational cost; it also gets rid of all zero-padding operations so that we only have to operate on the original, unmodified version of the image without any loss or artificial addition of information.

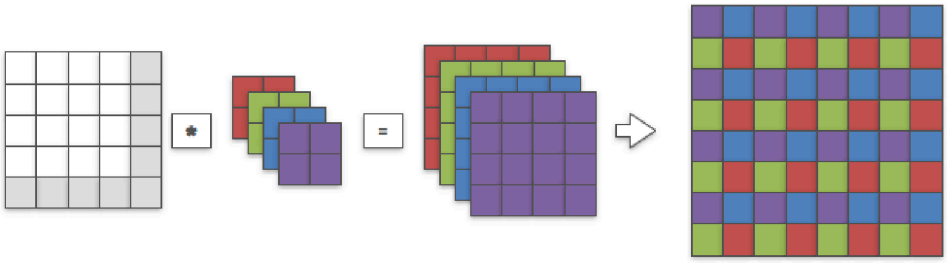


Figure 2.10: The intuition behind sub-pixel convolution. Convolution is directly applied to I^{LR} without any preprocessing. [16]

2.2 Problem Formulation: Denoising

Prior to the use of deep learning in image processing, there have been related researches on noise reduction techniques for image denoising. Image denoising is a long-lasting problem in computer vision and image processing since decades as corruption of data is ubiquitous in the digital domain. The stochasticity in noise nature and time variance as well as application dependence particularly complicated the development of a versatile noise reduction algorithm. It is associated with critical importance in image pre-processing and more recently, pre-training for unlabelled data in deep learning. Mathematically, one can formulate the image degration and restoration problem by

$$y = D(x) + n \tag{2.1}$$

where y is the output image, D is certain image degration function depending on noise models and n is some additive noise.

We should note that detection of sparse and spatially localised features, such as edges and singularities, are particularly compelling for machine evaluation. Thus, we should preserve them to the greatest extent with minimal trade-offs. An edge in an image, by definition, is a significant local change in image intensity associated with discontinuities in either image intensity or its gradient [17]. The literature survey can be split into two sections: a brief survey on traditional denoising algorithms and discussion on development of deep learning denoising methods.

2.3 Classical Methods in Image Denoising

Traditionally, image denoising relies on prior information of image formulation as well as noise nature to apply a suitable algorithm of choice, hence passive. Classical noise removal techniques can be broadly classified into spatial filtering and transform domain filtering. Below briefly discusses the novelty of the three filtering methods.

2.3.1 Spatial Filtering

Spatial filtering is usually employed to deal with additive noise. By assumption of an image having high spatial frequency components, these filters operate by evaluating each pixel and its neighbours and picking a value based on the kind of filter implemented. It can be thought as convolution of an image with filter in the spatial domain. Notable spatial filtering methods include linear filters such

as mean filter and Wiener filter [18], and non-linear filters such as median filter [19] and bilateral filter [20]. While these filters successfully achieve noise reduction via smoothing, they generally fail to preserve edges as the aim is to remove spatial abnormalities.

2.3.2 Transform Domain Filtering

Transform domain filtering, generally speaking, achieve image denoising by adapting frequency domain filters as opposed to spatial domain operations. It can be sub-divided into two transformation techniques: spatial frequency filtering and wavelet domain filtering.

- **Spatial Frequency Filtering**: Noise usually realises as decorrelated portion of an image i.e. high frequency component. We can easily separate them by implementing a low-pass filter by fast Fourier transform (FFT) with a cut-off frequency f_c, followed by an inverse transformation. This thus has the same effect as smoothing. However as edges co-exists with noise in the frequency spectrum, it fails at edge preservation as spatial filtering does, as well as ringing effects and oscillations due to overshooting during transform. A separate edge detection filter can be implemented though [21].
- **Wavelet Domain Filtering**: Fourier transform performs well on stationary signals which rarely occurs in image processing due to its fundamental basis function and signal localisation in frequency domain only. Wavelet transform localise signal in both time and frequency domain [22], and it can characterise the original waveform with shape irregularity. This gives wavelets a better ability to distinguish between noise and image content. Although with less severe outcome, it still suffers from the same side-effects as FFT [23].

As per discussion, all methods suffer from some kind of drawbacks during image denoising due to limitations in algorithms or on the mathematical basis. Currently the benchmark algorithm is block-matching in 3D transform domain (BM3D) [24], which has top performance compared to other general denoising methods.

2.4 Image Denoising with Deep Learning

One of the earlier works on utilising DNNs on image denoising is the denoising autoencoder (DAE) introduced by Vincent et al. [25], albeit the original purpose is to extract features from unlabelled data. Based on the original autoencoder(AE) proposed by Hinton et al. [26], the model accepts a destroyed, corrupted input from the encoder to obtain a latent representation. A decoder with a reverse mapping of the encoder is then used to reconstruct the original representation. This modification has proved to create AEs with robustness to small, irrelevant inputs for feature extraction pre-processing. The same authors also stacked the DAEs together to create stacked denoising autoencoder (SDAE) [27]. SDAE yields better classification and reconstruction results than stacked autoencoders (SAE).

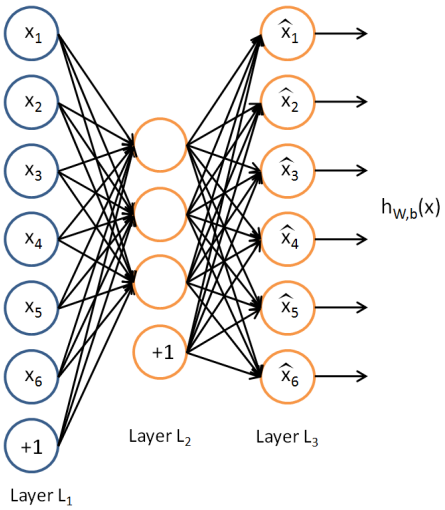


Figure 2.11: An Illustration of a Naive AE Architecture [28]

To further improve the AE's ability for good feature extraction, Makhzani et al. [29] introduced sparse autoencoder (SPAE). It penalises over-active neurons in the AE by introducing a sparsity term using Kullback-Leibler divergence, thus giving stronger gradients and limiting the number of significantly active neurons in each layer. Xie et al. introduced the concept into the regular DAE architecture and stacked DAEs together, which is called stacked sparse denoising autoencoder (SSDAE) [30]. SSDAE has proven to be effective in image denoising as well as impainting removal, especially when hidden layers have more units than the input does.

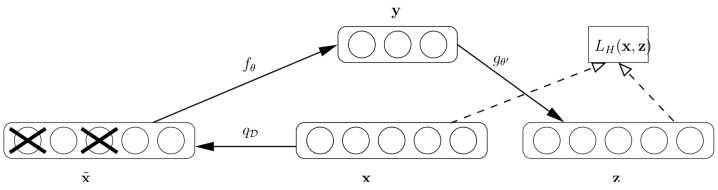


Figure 2.12: Illustration of a DAE Architecture[25]

2.4.1 Related Works

Jain et al. first proposed the use of convolutional neural network for image denoising and has shown on par or occasionally superior performances to back then state-of-the-art wavelet and Markov random field (MRF) methods [31]. Masci et al. later improved the AE by incorporating convolution into the model design as well as stacking to create a deep hierarchy, termed stacked convolutional autoencoder (SCAE) [32]. This greatly enhances AE's capability to identify and preserve localised spatial features by expanding into multiple input channels and encompassing them afterwards at the output layer.

Mousavi et al., instead of applying statistical methods and greedy algorithms as in compressive sensing (CS) approach, employed the stacked denoising autoencoder (SDA) and set a back-then new framework for sensing and recovery of structured signal [33]. They showed that SDA is capable of learning linear or slightly non-linear structured representation, and obtain an estimation without finding statistical dependencies mathematically.

Gondara built a CAE to achieve medical imaging denoising [34]. The paper has shown even a simple 2-layer encoder-decoder CAE model trained with just a net total of 722 images outperforms traditional well-defined methods such as median filtering and non-local means. However, validation error fails to converge at high Gaussian noise levels even after 100 epochs.

In [35], instead of applying an autoencoder for noise removal, Quijas et al. has attempted to remove JPEG blocking artifacts by constructing a framework to predict DCT coefficients and pixel values, then combine both predictions and replace the affected pixels with the two parameters. This method has shown great success in removing many JPEG artifacts.

Yu et al. improves further on compression artifact removal based on the topology of Super-Resolution Convolutional Neural Network (SRCNN) [36] with "feature-enhancement" layers in between, namely Artifacts Removal Convolutional Neural Network (AR-CNN) [37]. It demonstrates extremely effective performance in dealing with different forms of compression artifacts with only four layers.

Mao et al. proposed a very deep fully convolutional autoencoder network termed "RED-Net", i.e. Residual Encoder-Decoder Network with skip connections [38]. The skip connections allow forward propagation from convolution layer to corresponding deconvolution layer and avoid vanishing gradient problem manifests in very deep neural networks. It displayed a very strong performance in different areas including image denoising, image super-resolution, JPEG deblocking, deblurring and impainting removal, suggesting performance exceeding state-of-the-art Weighted Nuclear Norm Minimisation (WNNM) method [39].

2.5 Problem Formulation: Super-Resolution

The primary aim of this task is to estimate a high-resolution, super-resolved image I^{SR} from a low resolution input image I^{LR}. I^{LR} is the low-resolution version of its high-resolution counterpart I^{HR}, which can only be provided in training phase as such data would not be available in practice. Similar to image denoising, it finds direct applications to areas such as medical imaging, surveillance, astrophotography, compression and so on. Image super-resolution is closely related to denoising application as it usually assumes I^{LR} not only is downsampled, but also noisy and low-pass filtered with different degrees of information loss. Super-resolution is also a much more difficult problem to solve as it contains one-to-many mapping, in other words theoretically infinity solutions. As a result, searching for a suitable solution for the SR problem formulation is inherently ill-posed. It is easy to notice that a $4\times$ SR problem implies a tremendous 93.75% generation of non-existent pixels in the LR space. Consequently, it indicates 16 times greater raw memory consumption for I^{LR} to I^{SR} mapping. The ambiguity is exponentially proportional to the super-resolution factor as fine-details will show little to no traces in I^{LR}, thus the network has no possible way to learn or interpolate them correctly.

In general, the problem formulation of single image super-resolution can be trivially described as

$$I^{HR} = A(I^{LR}) \tag{2.2}$$

where A is a downscale operator typically composes of decimation and low-pass filtering. Then, the estimated super-resolved image would be the inverse of downscaling:

$$I^{SR} = A^{-1}(I^{LR}) \tag{2.3}$$

Traditionally, upsampling is achieved by interpolation and resampling, including nearest-neighbour, bilinear, bicubic interpolation, Lanczos resampling, etc. The majority of SR algorithms focus on either

single-channel or multi-channel image super-resolution. In the latter case, the image is transformed to a different colour space first (YCbCr or YUV) and perform SR on the illumination channel only.

2.5.1 Related Works

Numerous researches have been dedicated into this area to produce both efficient and high-quality SR images with great success, especially after the explosion of deep learning.

One of the earliest, practically and quantitatively viable work on using deep learning to solve a SR problem is introduced by Dong et al. in 2014 [40]. The authors constructed a simple 9-1-5 architecture super-resolution convolutional neural network (SRCNN) model to solve the problem. It demonstrates that a CNN can effectively learn and establish an end-to-end mapping between LR and HR space and that a relatively shallow CNN can achieve or is superior to then state-of-the-art performance compared to traditional algorithms with sufficient training.

The same authors developed Fast Super-Resolution Convolution Neural Network (FSRCNN) in 2015 [41]. The network reported more than 40x speed-up while still maintaining on-par or superior performance compared to SRCNN. The major difference between the two network is that FSRCNN does not do any interpolation preprocessing but applies a transposed convolution layer for upsampling. This way FSRCNN gains the ability to learn optimal upsampling filters. In addition, fine-tuning the final upscaling layer is sufficient for adapting different upscaling factors; this is not the case in SRCNN as the authors reported huge differences instead.

Shi et al. first put the idea of sub-pixel convolution into practice in their proposed Efficient Sub-pixel Convolution Neural Network (ESPCN) [15]. It is claimed to be $\log_2 r^2$ times faster than transposed convolution in training time and r^2 times compared to other forms of upscaling operations. Not only are the portrayed numerical results are better, runtime evaluations also shown a significant speed improvement of > 10 times compared to the aforementioned CNNs. This can be considered as a milestone that researchers are looking for both runtime and visual improvements.

Lim et al. formulated the Enhanced Deep Super-Resolution Network (EDSR) [42]. The authors are aware of the sensitivity of reconstruction performance to minor architectural changes of the neural network. Also, previous works did not consider mutual relationships between different super-resolution scales but treated them as individual problems. So, by simply removing unnecessary batch normalisation layers in each of the residual building blocks, the network is able to show substantial improvement over its baseline model both in memory usage and performance. They also proposed a multi-scale deep super-resolution system (MDSR) capable of constructing super-resolved images of different scale within a single network architecture.

With the introduction of generative adversarial network (GAN) by Goodfellow et al. [43], the machine learning community has also explored into self-generation of super-resolved images rather than pure reconstruction with a standard loss function. Ledig et al. proposed Super-Resolution Generative Adversarial Network (SRGAN) and its non-generative counterpart SRResNet with the use of VGG19 perceptual loss [44]. The network is capable of producing granular details by leveraging the power of GAN, producing more photo-realistic and convincing images compared to then-existing models with reported mean opinion scores (MOS) leading previous works by a significant margin. SRResNet, while using residual blocks instead, produces statistically better results under different image evaluation metrics.

Chapter 3

Methodology

3.1 Noise Models & Compression Schemes

Noise can exist in digital data in different forms, while various compression schemes introduce certain fixed artifacts. Below we describe a number of commonly found noise models and artifacts.

3.1.1 Additive White Gaussian Noise (AWGN)

AWGN noise mimics random processes that occurs in nature. The name denotes the following characteristics:

- **Additive**: The noise is added to the original signal, i.e. $y(t) = x(t) + n(t)$, where $x(t)$ is the original signal, $n(t)$ is noise and $y(t)$ is the output signal.
- **White**: The noise has uniform power across all frequency. In other words, noise level is flat. White is taken from the analogy of the colour where it composes of colours from all frequencies.
- **Guassian**: The noise is random in nature and not deterministic with a Gaussian distribution. A Gaussian distribution is zero mean and variance by definition.

Thus, the noise is generally modelled as

$$n(t) = \frac{1}{\sqrt{2\pi}\sigma^2} e^{\frac{(n-\mu)^2}{2\sigma^2}} \tag{3.1}$$

where μ is mean of the distribution, σ is standard deviation, σ^2 is variance.

(a) Uncorrupted image

(b) Image corrupted with Gaussian noise

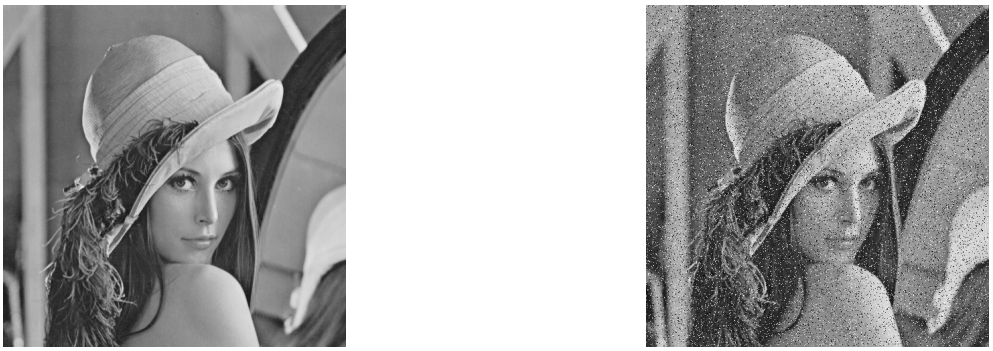
Figure 3.1: Example of AWGN Noise

3.1.2 Salt-and-Pepper Noise

Salt-and-Pepper is also commonly known as impulsive noise, and is usually caused by transmission errors during analogue-to-digital conversion. It exhibits a characteristic of sparsely distributed extreme value pixels. For instance, a salt-and-pepper noise corrupted image may have white pixels in dark regions and black pixels in white regions. It can be modelled as

$$n(t) = \begin{cases} 0 & n < \lambda \\ 1 & n \geq \lambda \end{cases} \tag{3.2}$$

where λ is a self-defined threshold for setting pixel values to minimum or maximum.



(a) Uncorrupted image

(b) SaP noise corrupted image

Figure 3.2: Example of Salt-and-Pepper Noise

3.1.3 JPEG Compression Artifacts

JPEG standard[1] [45] is one of the most widely used lossy compression of images, and belongs to the class of transform coding techniques and makes use of discrete cosine transform (DCT). The steps for JPEG encoding is as follows:

1. Divide the image of $N \times N$ array into blocks of 8×8 sub-arrays.
2. Compute the DCT of each 8×8 array, which is given by

$$G_{u,v} = \frac{1}{4}\alpha_u \alpha_v \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} \cos\left[\frac{(2k+1)u\pi}{16}\right] \cos\left[\frac{(2l+1)v\pi}{16}\right] \tag{3.3}$$

   where $G_{u,v}$ is DCT of coordinate u, v of 8×8 sub-array,
   $\alpha_u, \alpha_v = \begin{cases} \frac{1}{\sqrt{2}} & u, v = 0 \\ 1 & \text{otherwise} \end{cases}$ are normalising scale factor to make transformation orthonormal
3. DCT coefficients are quantised to obtain $B_{u,v}$ using quantisation matrix $Q_{u,v}$ specified by JPEG standard as per below equation, with samples arranged in a vector by zigzag sampling:

$$B_{u,v} = \text{round}\left(\frac{G_{u,v}}{Q_{u,v}}\right) \tag{3.4}$$

[1]The discussion is based on the 1992 DCT-based JPEG standard. More modern JPEG standards such as wavelet-based JPEG 2000 and JPEG XR are not covered.

(a) Uncorrupted image

(b) JPEG encoded image

Figure 3.3: JPEG Block Artifacts

As the coding procedures divide the image into sub-blocks, intuitively there will be blocking artifacts composed of the size of the sub-blocks as we zoom in. Additionally, since JPEG operates in the frequency domain in which image are represented as sum of oscillating waves, there also exists blurring and ringing artifacts.

3.2 Artificial Neural Network: A Primer

This section covers the fundamental mathematics in order to understand the underlying principles of an artificial neural networks.

3.2.1 Artificial Neuron

An artificial neuron is a set of mathematical functions to model biological neurons.

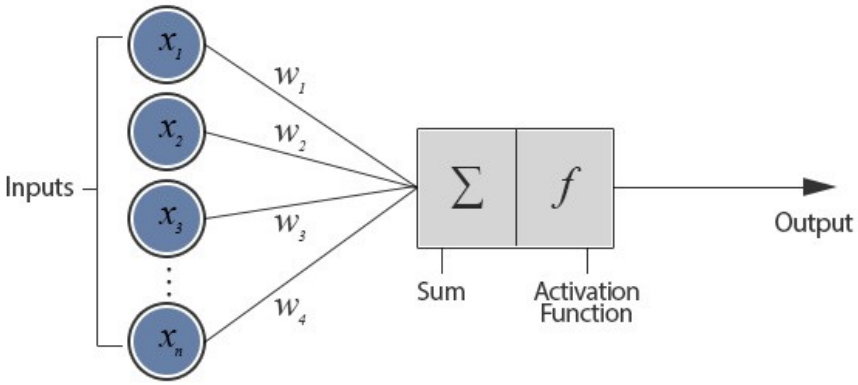


Figure 3.4: An illustration of an artificial neuron. This is how the neuron can be modelled to its biological counterpart mathematically.

For a given artificial neuron, it accepts a set of input W, b. They are summed together and are multiplied by a non-linear function commonly known as activation function. Below is a discussion of individual components and mathematics of an artificial neuron.

3.2.2 Gradient Descent

Gradient descent is a first-order iterative optimization algorithm used broadly in machine learning and deep learning to minimize functions in the direction of steepest descent(negative gradient). In the context of deep learning, this is the optimisation of parameters in neural network layers. In other words, gradient descent is an algorithm to minimise the objective loss function.

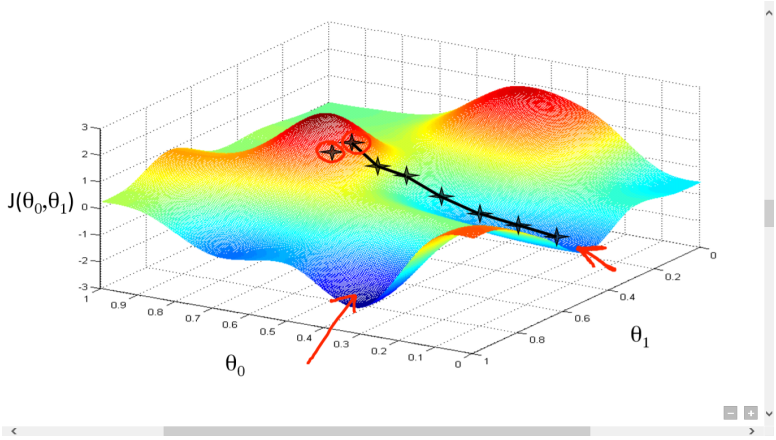


Figure 3.5: A 3-D plot. Starting from the top of the hill (red area), we would like to find an optimal approach that leads to minima (red arrows).

Given a loss function $J(W, b)$ with N observations which uses MSE,

$$J(W, b) = \frac{1}{2N} \sum_{i=1}^{N} (h_{W,b}(x^{(i)} - y^{(i)})^2 \tag{3.5}$$

The new updated parameters W, b is the partial derivatives of $J(W, b)$:

$$W_{new} = W_{old} - \alpha \frac{\partial J(W_{old}, b_{old})}{\partial W_{old}} \tag{3.6}$$

$$b_{new} = b_{old} - \alpha \frac{\partial J(W_{old}, b_{old})}{\partial b_{old}} \tag{3.7}$$

where α is the learning rate of gradient descent algorithm which decides the speed of the algorithm and rate of convergence. The process is repeated until the $J(W, b)$ is converged to a desired minima. Note that α should be set to an appropriate value; large α can overshoot the minima and never converges whereas a low α can significantly increase the number of iterations before the algorithm is able to find the minima. In vectorised form, this can be written as Jacobian matrix of N elements.

3.2.3 Back-propagation

To use gradient descent, we must have an efficient way to compute analytical, exact gradients at each neuron where W, b are involved. Back-propagation is a form of automatic differentiation in reverse mode. It is also sometimes referred as backward propagation of errors as the error calculated at the output of neural network is propagated via layers and distributed back to the input.

The algorithm can be interpreted as a computational graph as shown in Figure.

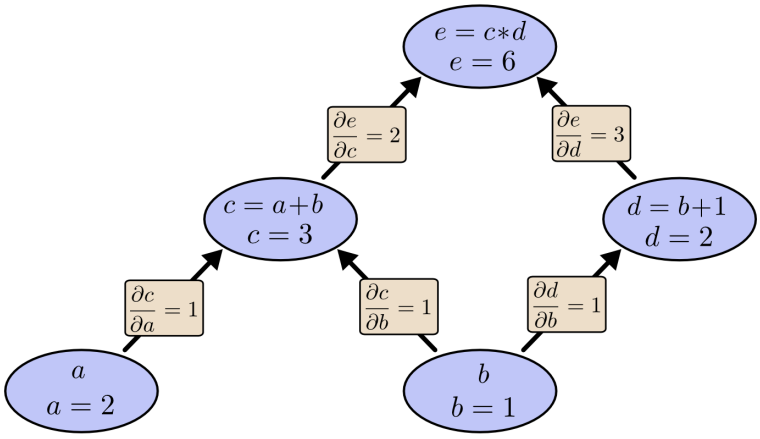


Figure 3.6: A computation graph depicting the process of back-propogation. Here we have the outputs from each node, and they are used during backward pass from the ouput to calculate gradients.

Since we would have obtained all the computational solutions at each point of the graph and they affect one another either directly or indirectly, intuitively we can apply chain rule to get the gradients.

Combining gradient descent and back-propagation algorithms, we can then establish the optimisation algorithm used in neural networks:

- **Stage 1: Propagation**
  For each propagation, forward propagate through the network to obtain output values for each individual node and and calculate the loss. Then the loss is backpropagated to calculate the gradients (differences) between the targeted and actual output values of all output and hidden neurons.
- **Stage 2: Weight Update (Gradient Descent)**
  For each weight and bias W, b, gradients are calculated and multiplied by the learning rate α. The obtained gradient is then subtracted from the original value to get the updated values. Repeat Stage 1 until stopping criterion is satisfied.

3.2.4 Activation Functions

Although we have established an optimisation algorithm, this does not grant a neuron the representative power to differentiate or model curve with respect to data variance. The use of multiple hidden layers in neural network is useless since the overall output is still a linear transformation. As such, its problem solving capacity is highly constrained with a single function regardless of modifications. To actually generate the complexity we need to use non-linear transformations.

For each neuron with input x, output y and non-linear activation function σ, y will be

$$y = \sigma\left(\sum_{i=1}^{N}(W^{(i)}x + b^{(i)})\right) \tag{3.8}$$

Without non-linearities, back-propagation would not have been possible and deep neural network models would be non-existent. Below is some of the widely used activation functions in deep learning.

1. **Logistic Sigmoid**

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.9}$$

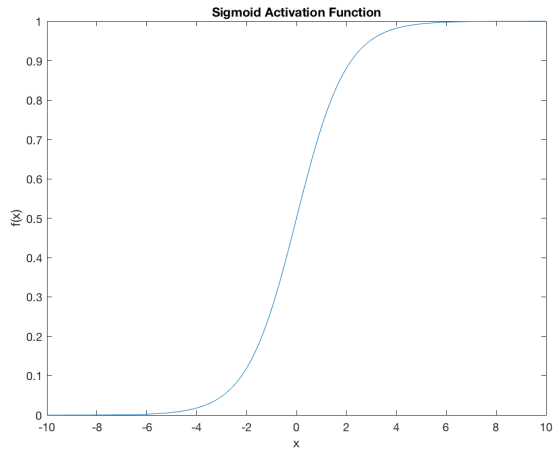


Figure 3.7: Plot of sigmoid activation function.

Sigmoid has several benefits. Firstly, it is non-linear in nature, which means stacks of sigmoid is also non-linear. Secondly, the range of sigmoid function is bounded to $[0, 1]$, so activations will not "explode", i.e. computed infinity values. This is known as the exploding gradient problem. However it can give rise to problem of vanishing gradient as the gradient response can become saturated towards negative or positive extremes. This would lead to problems training deeper neural network models. In other words, it is only possible to create shallow networks with sigmoid.

2. **Hyperbolic Tangent (tanh)**

$$f(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} = \frac{2}{1 + e^{-2x}} - 1 \tag{3.10}$$

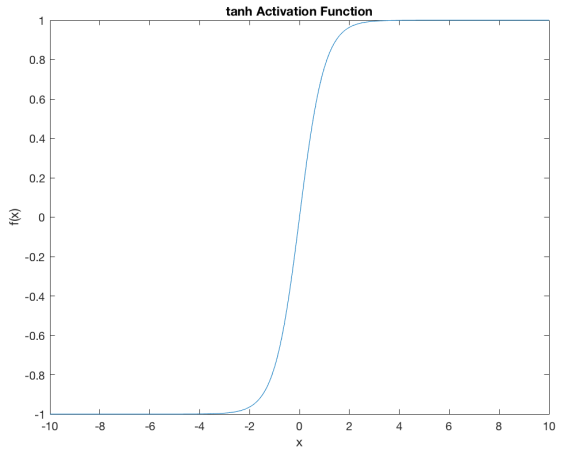


Figure 3.8: Plot of tanh Activation Function.

tanh is in fact a scaled sigmoid:

$$tanh(x) = 2sigmoid(2x) - 1 \tag{3.11}$$

Compared to sigmoid, the range is bounded to $[-1, 1]$ so we do not have to concern about exploding gradient. It has a stronger gradient than sigmoid with deeper derivatives. The vanishing gradient problem still exist though. Together with the sigmoid activation function, they were the most popular and widely used activation functions prior to 2011.

3. **Rectified Linear Unit (ReLU)**

$$f(x) = \begin{cases} 0 & x < 0 \\ x & x \geq 0 \end{cases}$$
$$= max(0, x) \tag{3.12}$$

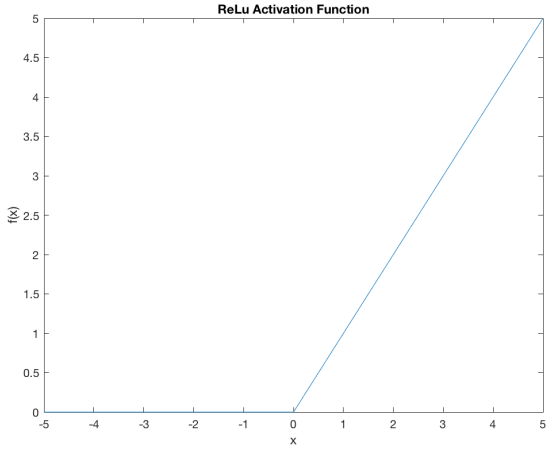


Figure 3.9: Plot of ReLU Activation Function.

The ReLU activation function is first reported by Hahnloser et al. with biological coherence and mathematical justifications [46]. AlexNet first used ReLU to demostrate its ability to provide better, effective trainings and loss convergence [7]. It is by far the most widely used activation function in neural network training.

Several benefits are provided by the ReLU activation function over logistic functions. Firstly, it provides innate sparsity with only 50% of hidden units activation. This is comparable to the use of dropout and forces each artificial neuron to generalise features. Secondly, it produces better gradients; there is no problem of saturation compared to sigmoidal activation functions.

It also comes with some potential problems. ReLU is prone to explosive gradients since positive outputs from the activation function is unbounded, i.e. output ranges from $[0, +\inf)$. It also has the problem of "dying ReLU"; that is, a gradient of 0 is obtained from the neuron output. Subsequently, there will be no updating of parameters following Equations 3.6 and 3.7.

3.2.5 Receptive Field

In context of biological visual system, receptive field refers to a particular region of sensory space that fires a neuron upon stimulus. The term is adopted reminiscently in artificial neural networks, CNNs in particular. In this sense, it is defined as a particular region of the input space of which a convolution kernel is affecting. We can easily visualise CNN feature maps, but it is extremely difficult

to instinctively obtain the size of receptive field and relative positions of each feature in the map. For the latter parameters, it is better to compute the values with convolution arithmetic [47, 48].

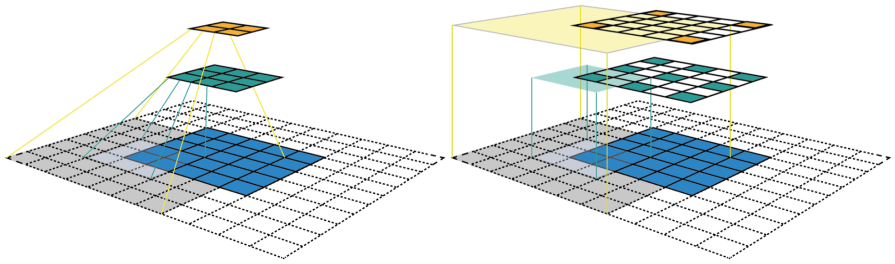


Figure 3.10: Two ways to visualise a CNN's output feature map with respect to input and the canonical volume of receptive field.

Assuming we have a squared input image into a CNN with zero padding and non-unit stride, the equations for receptive field arithmetic o can be inferred as

$$o = \left\lfloor \frac{i + 2p - k}{s} \right\rfloor + 1 \tag{3.13}$$

$$j_o = j_i * s \tag{3.14}$$

$$r_o = r_i + (k - 1) * j_i \tag{3.15}$$

$$start_o = start_i + \left(\frac{k - 1}{2} - p \right) * j_i \tag{3.16}$$

Equation 3.13 calculates the number of output features. Here, i is the number of input features, p is the size of convolution padding, k is the size of convolution kernel, s is the stride of kernels;
Equation 3.14 calculates the jump j from one feature to the next in the output feature map;
Equation 3.15 calculates the size of receptive field r of the convolution kernel;
Equation 3.16 calculates the starting position $start$ of output feature in the output feature map.

3.2.6 Hyper-parameters

It should be clear now that an artificial neural network contains millions of parameters, and it is impossible to manually alter each value. Fortunately, we can somewhat gain a degree of control by hyper-parameters. By definition, a hyper-parameter is parameter that is set before training commences. In fact, the behavioural patterns of parameters within a neural network is attributed to and extremely sensitive to the choice of hyper-parameters. This is analogical to environmental influence to a person.

Below is some of the most crucial hyper-parameters one would like to consider:

1. Size of mini-batches: The number of samples that is forward propagated per training iteration. Some works advocate the use of thousands samples per iteration, but Masters et al. suggests that the optimal number should be between 2 and 32 [49].

2. Learning rate: The rate of change of gradients adjustment with respect to computed loss. It is arguably the most critical hyper-parameter as an inappropriate value can lead to consequences discussed in section 3.2.2. The choice of learning rate is also dependent on optimisation algorithm.

Inappropriate hyper-parameter settings can cause problems such as overfitting (prediction corresponds to training data too closely and fail to generalise data) or underfitting (prediction fails to learn and cannot distinguish between data).

3.3 Neural Networks

Before going into the details of the model we are going to use in this work, it is critical to understand the basic concepts of autoencoders and convolutional autoencoders since the model is based on them.

3.3.1 Autoencoders

Autoencoders (AE) is a an unsupervised learning method, meaning that it infers functions to describe features from unlabelled data, i.e. it does not involve classification and clustering tasks. It is also a special type of unsupervised learning where accuracy can be used as a training metric, since we are attempting to map our obtained output vectors to input. The architecture of a trivial AE is simply a stack of restricted Boltzmann Machine (RBM) [26]. Let us denote $x_i \in [0,1]^n$ as the ground truth and $y_i \in [0,1]^n$ as the output for $i = 1, 2, 3, ..., n$ where n is the total number of units representation in each layer. Since we are attempting reconstruction, $y = \hat{x}$ where $\hat{x}$ is our reconstruction. Then we define function $\sigma(\cdot) : \mathbb{R}^n \to \mathbb{R}^m$, $m < n$,

$$h_i(x_i) = \sigma\left(Wx_i + b\right) \tag{3.17}$$

$$y_i = \sigma\left(W' h_i(x_i) + b'\right) \tag{3.18}$$

where σ is non-linearity activation function which is applied to each hidden layer representation, h_i is hidden layer activation, $W \in \mathbb{R}^{n \times m}$ is non-zero weight matrix, and $b \in \mathbb{R}^n$ is bias vector. We can optionally impose constraint $W' = W^T$ for tied weights in order to reduce degrees of freedom by magnitudes, speed up training and avoid over-fitting. We should note that $W \in \mathbb{R}_{\neq 0}$ as any 0 in the matrix will cause the particular neuron to produce zero gradient back-propagation, resulting in parameter update failure and death of neuron. If we parametrise in a manner such that $\theta = \{W, b\}, \theta' = \{W', b'\}$, then the parameters can be minimised by Euclidean distance l_2 in each training step:

$$\begin{aligned} \theta, \theta' &= \frac{1}{n} \underset{\theta,\theta'}{\operatorname{argmin}} \sum_{i=1}^{n} \left(l_2(x_i, y_i)\right) \\ &= \frac{1}{n} \underset{\theta,\theta'}{\operatorname{argmin}} \sum_{i=1}^{n} ||x_i - y_i||_2^2 \end{aligned} \tag{3.19}$$

Another feasible loss function that can be used is reconstruction cross-entropy denoted by l_p, and in this case Equation (3.19) becomes

$$\begin{aligned} \theta, \theta' &= \frac{1}{n} \underset{\theta,\theta'}{\operatorname{argmin}} \sum_{i=1}^{n} \left(l_p(x_i, y_i)\right) \\ &= -\frac{1}{n} \underset{\theta,\theta'}{\operatorname{argmin}} \sum_{i=1}^{n} \left((x_i \log y_i + (1 - x_i) \log(1 - y_i)\right) \end{aligned} \tag{3.20}$$

A DAE takes and corrupts x_i to obtain a noisy version $\tilde{x_i}$ then follows the same equations given as above:

$$h_i(\tilde{x_i}) = \sigma\left(W\hat{x_i} + b\right) \tag{3.21}$$

$$y_i(\tilde{x_i}) = \sigma\left(W' h_i(\tilde{x_i}) + b'\right) \tag{3.22}$$

However, we compare our obtained reconstruction with the original input instead of the corrupted version. We want the model to reconstruct the encoded representation base on the uncorrupted image instead of replicating the corruption. Then, Equation (3.19) becomes

$$\theta, \theta' = \frac{1}{n} \underset{\theta,\theta'}{\operatorname{argmin}} \sum_{i=1}^{n} ||y_i(\tilde{x_i}) - x_i||_2^2 \tag{3.23}$$

Alternatively, DAE can also be implemented using a technique named dropout [50]. Instead of modifying the source image beforehand, we introduce a probabilistic model on the activation level to randomly set neuron outputs to 0. This results in forcing the DAE to learn more significant features instead of redundant ones.

Another variation of DAE is the sparse denoising autoencoder (SPDAE). SPDAE employs a sparsity loss using Kullback-Leibler divergence which is given as

$$\mathrm{KL}(\rho \,||\, \hat{\rho}) = \sum_{j}^{|\hat{\rho}|} \rho \log \frac{\rho}{\hat{\rho}} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}}, \quad \hat{\rho} = \frac{1}{N} \sum_{i}^{N} h(x_i) \tag{3.24}$$

where $\hat{\rho}$ is average activation of a hidden layer with N neurons, ρ is sparsity parameter which typically is a value close to zero. Then, we can formulate our regularised loss function:

$$l_{sparse}(\mathbf{x_i}, \mathbf{y_i}) = l(\mathbf{x_i}, \mathbf{y_i}) + \beta \mathrm{KL}\left(\rho \,||\, \hat{\rho}\right) \tag{3.25}$$

where β is a hyper-parameter controlling the weight of our sparsity penalty term. We should note that $\sigma(\mathbf{h(x_i)}) \in (0, 1)$, otherwise $\mathrm{KL}(\rho \,||\, \hat{\rho})$ will be undefined. A simple solution would be to use a sigmoid activation function.

3.3.2 Convolutional Autoencoders

A convolutional neural network (CNN) works by discrete convolution and pooling operations. We learn features from our instantiated kernel, and 'slide' it across the input feature map to create individual receptive fields with each representing certain feature. Our convolved feature map is then sub-sampled using mean or max-pooling to reduce in dimension, serving as concatenation of spatial features and encoding. These operations are repeated until the encoded image reaches our desired size. The output is finally flattened and reshaped into a 1-D fully-connected layer and used for classification with a softmax layer.

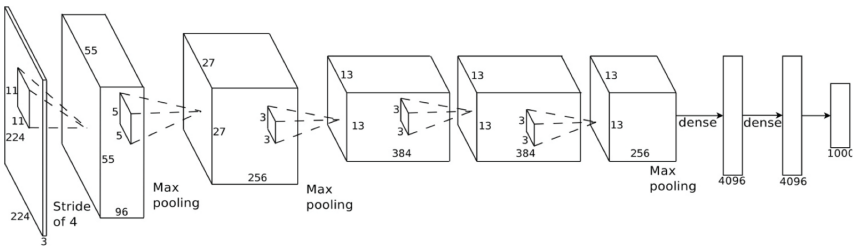


Figure 3.11: Depiction of a CNN classifier architecture [51]

In 2D discrete space of an mono-channel input image with finite discrete signals, the convolution operation is defined as

$$(I * f)(x, y) = \sum_{u=-2k-1}^{2k+1} \sum_{v=-2k-1}^{2k+1} I(x-u, y-v) f(u, v) \tag{3.26}$$

where $*$ denotes 2D convolution operation, $f(x, y)$ is the convolutional kernel, $I(x, y)$ is the input image, both in coordinates (x, y) and $2k+1$ is the side of $f(x, y)_{odd}$. A rule of thumb for the kernel $f(x, y)$ size choice is 3×3.

In the case of a convolutional autoencoder (CAE), we follow the encoder-decoder structure of a typical AE. However, we perform deconvolution and mean or max-unpooling in order to restore the dimension of our encoded latent representation and obtain our reconstruction. It also differs from conventional AEs in the sense that it removes redundancy by sharing weights and hence preserve spacial locality, while AEs forces each feature to be global [32]. In other words, AEs and DAEs alike are constrained by the number of inputs based on the image while CAEs are independent of it.

The architecture of CAEs is loosely similar to AEs, except we apply convolutions. For mono-channel input x, representation of i^{th} feature map h is given by

$$h_i = \sigma(x * W + b) \tag{3.27}$$

where bias b is broadcasted to the whole map. Single bias is used to reduce the degrees of freedom, and the reconstruction y is obtained by

$$y = \sigma\left(\sum_{i \in H} h_i * \tilde{W}_i + b\right) \tag{3.28}$$

As per above, b is the bias per input channel, H is the set of latent feature maps, $\tilde{W}$ is the flip operation over both weights' dimension. We can still use MSE as the minimisation cost function. As for CDAE, we similarly replace the input image with its corrupted version.

3.4 Perceptual Loss

Typical loss function used in image reconstruction, image denoising and super-resolution alike, is the l_2 norm loss function as seen from Equation 3.19. As such, it returns a per-pixel loss and is back-propagated to the entire network. This effectively tells the network to create a reconstruction image that is exactly the same as the ground-truth. This is not optimal as it would not allow the network to preserve spatial and perceptual information, which is crucial to defining shape of objects.

To achieve this, perceptual loss does not encourage such behaviour. Rather, the loss function encourages the reconstruction to retain similar feature representations as precomputed by certain deep learning network ϕ. Let ϕ_j be the activation of layer j of a pretrained network for image classification; if ϕ_j is a convolutional network then the shape of the activation will be a feature map tensor of size $C_j \times W_j \times H_j$. Let y be the input to the network, we can now define the perceptual feature reconstruction loss as the Euclidean distance between the feature representations of network output and input:

$$l_{feat}^{\phi,j} = \frac{1}{C_j W_j H_j} ||\phi_j(\hat{y}) - \phi(y)||_2^2 \tag{3.29}$$

The pretrained network ϕ is now our loss network. It remains fixed and the parameters must not be modified when backpropogating to the main network.

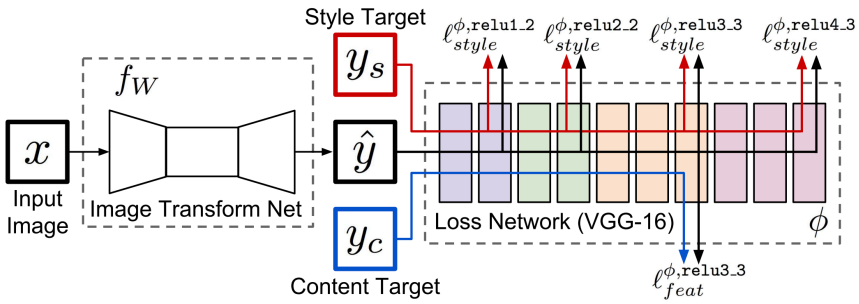


Figure 3.12: An illustration of how the feature reconstruction loss of a particular layer ϕ_j is extracted from a pretrained network.

It should be noted that the extent of retained attributes of the image is dominated by the chosen output of network ϕ. The deeper layers of ϕ is used during network training, the less likely actual contents will be preserved. This leaves only the abstractions available with hallucinated contents. Therefore, despite not forcing the two to match exactly, striking a balance between them is nevertheless crucial for a visually pleasing outcome.

3.5 Sub-pixel Convolution

Sub-pixel convolution is an alternative way to upsample an image as discussed in Section 2.1.4 [15]. As mentioned in the paper, convolution with stride of $\frac{1}{r}$ with filter W_s of size k_s with weight spacing $\frac{1}{r}$ has a total number of activation patterns of exactly r^2. The reason behind that is we do not have to calculate irrelevant weights which falls between pixels. With at most $\lceil \frac{k_s}{r} \rceil^2$ weights per activation pattern, they are periodically activated depending on sub-pixel location $\text{mod}(x, r)$, $\text{mod}(y, r)$ where x, y are the output pixel coordinates in HR space. I^{SR} calculation is then

$$I^{SR} = f^N(I^{LR}) = \mathcal{PS}(W_N * f^{N-1}(I^{LR}) + b_N) \tag{3.30}$$

where $\mathcal{PS}$ is a periodic shuffling operator to rearrange image tensor of shape $H \times W \times C \cdot r^2$ into a tensor of shape $rH \times rW \times C$. $\mathcal{PS}$ can be mathematically written as

$$\mathcal{PS}(T)_{x,y,c} = T_{\lfloor \frac{x}{r} \rfloor, \lfloor \frac{y}{r} \rfloor, C \cdot r \cdot mod(x,r) + C \cdot r \cdot mod(y,r) + c} \tag{3.31}$$

Equation 3.31 may look confusing but it is simply reshaping of the periodic shuffling operator where the subscript denotes the shape of tensor.

So, the MSE loss function using the sub-pixel convolution can now be written as

$$l(W_{1:N}, b_{1:N}) = \frac{1}{r^2 HW} \sum_{x=1}^{rH} \sum_{y=1}^{rW} \left(I^{HR}_{x,y} - f^N_{x,y}(I^{LR}) \right)^2 \tag{3.32}$$

Since sub-pixel convolution avoids computation in space I^{SR} until upsampling at the end, it consumes much less memory and computational power. This results in $\log_2 r^2$ times faster and r^2 times faster compared to regular transposed convolution layers and other implementations of upscaling before convolution.

3.6 Residual Encoder-Decoder Network with Skip Connections

The neural network architecture of this project is based on Residual Encoder-Decoder Network with Skip Connections (RED-Net) proposed by Mao et al. [38].

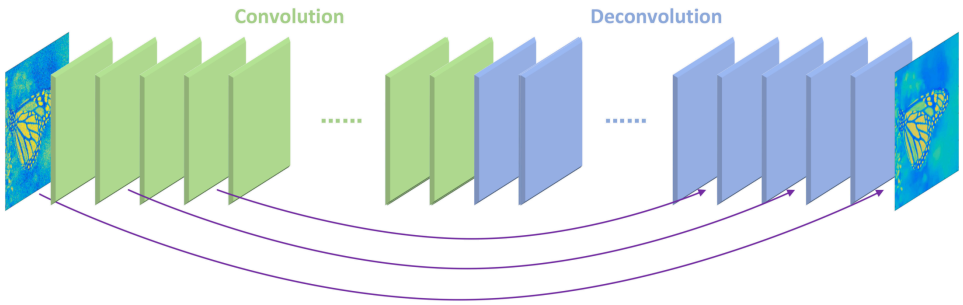


Figure 3.13: The overall architecture of RED-Net. As we can see from the figure, it consists of a convolution-based encoder and transpose convolution-based decoder with interconnected skip connections to help gradient propagations.

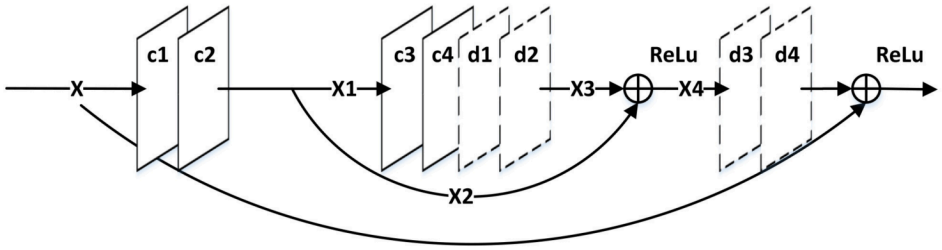


Figure 3.14: An example building block of RED-Net. $\oplus$ denotes element-wise sum operation.

RED-Net consists of three types of layers: convolution, transposed convolution and element-wise sum. Given an input X with kernel weight and bias W_k, b_k, the output of X can be expressed as

$$f(X) = \max(0, W_k * X + b_k) \tag{3.33}$$

where $*$ is convolution or transposed convolution operation. For skip connection, it can be rewritten as the element-wise sum of two same-size input with a ReLU activation:

$$f(X_1, X_2) = \max(0, X_1 + X_2) \tag{3.34}$$

and MSE loss function follows Equation 3.19. It is then optimised with Adam [52] which has been found with faster convergence than stochastic gradient descent (SGD). Gradient g with respect to loss function l, layer i and timestep t is calculated as

$$g = \nabla_{\theta_{i,t}} l(\theta_{i,t}) \tag{3.35}$$

Two moment vector estimates m, v are computed as

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_{i,t} \tag{3.36}$$

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_{i,t} \tag{3.37}$$

and Adam's update rule is

$$\alpha_t = \alpha \cdot \sqrt{1 - \beta_2^t}/(1 - \beta_1^t) \tag{3.38}$$

$$\theta_{i,t} = \theta_{i,t-1} - \alpha_t \cdot /(\sqrt{v_t} + \hat{\epsilon}) \tag{3.39}$$

where α is learning rate, $\beta_1, \beta_2 \in [0, 1)$ are exponential decay rates for moment vector estimates m, v, ϵ is a very small number to prevent division by 0 (Computers cannot handle divisions by 0). They are all configurable hyper-parameters. The suggested initialisation scheme for the aforementioned parameters are $\alpha = 10^{-4}, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$.

The usefulness of skip connection in RED-Net can be justified mathematically. Consider Figure 3.14 and denote output as Y. Let the gradient of c2 be θ_{c2}, then the backpropagation of θ_{c2} will be

$$\nabla_{\theta_{c2}} Y(\theta_{c2}) = \frac{\partial Y}{\partial X_4}\frac{\partial X_4}{\partial X_3}\frac{\partial X_3}{\partial X_1}\frac{\partial X_1}{\partial \theta_{c2}} + \frac{\partial Y}{\partial X_4}\frac{\partial X_4}{\partial X_2}\frac{\partial X_2}{\partial \theta_{c2}} \tag{3.40}$$

From Equation 3.40, it is straightforward to see that the latter term has less derivatives. As a rule of thumb in deep learning, gradients gradually diminish as the number of derivatives progresses. Thus, skip connection can reduce the involvement of derivatives and help rectify any potential vanishing gradient problem. This can be extended to include further encoder-decoder pairs.

Chapter 4

Experiments

This chapter discusses the findings from experiments with different neural network configurations and hyper-parameters.

4.1 Datasets

All neural networks in the experiments are trained by the training set of DIV2K dataset [53], and are evaluated with benchmark datasets Set5 [54], Set14 [55] and DIV2K validation set.

The DIV2K dataset is a relatively new dataset proposed in New Trends in Image Restoration and Enhancement workshop on super-resolution (NTIRE) in conjunction with CVPR 2017. The dataset is composed of 1000 images are RGB coloured at 2K resolution. It is divided into 800 training images, 100 validation images and 100 test images. Test images are not made available by the authors, therefore test results are evaluated on the validation set instead.

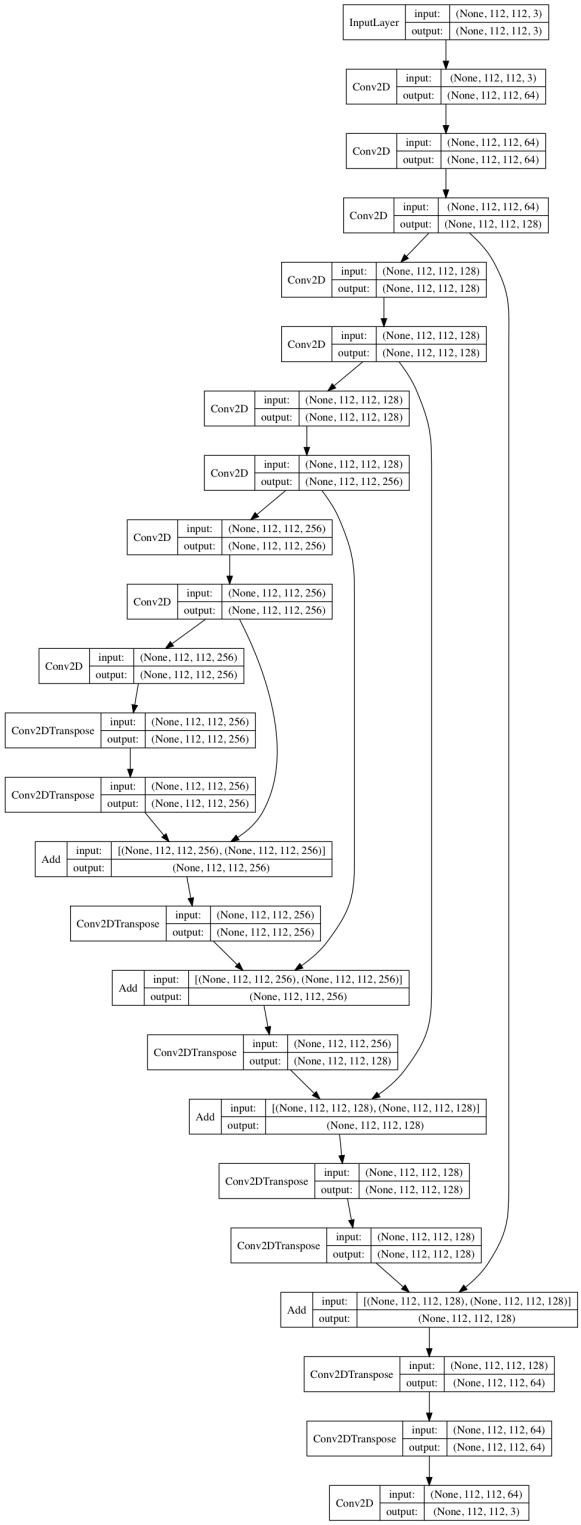
4.2 Training Details

All results are trained and evaluated with GPU servers located in Centre for Vision, Speech and Signal Processing, University of Surrey. System specification is Intel® Xeon® Gold 5122 CPU 3.6GHz and NVIDIA® GTX 1080 Ti 12GB GPU installed installed with Ubuntu 16.04.4 LTS (Xenial Xerus), CUDA 9.1 and CuDnn 7.0.5.

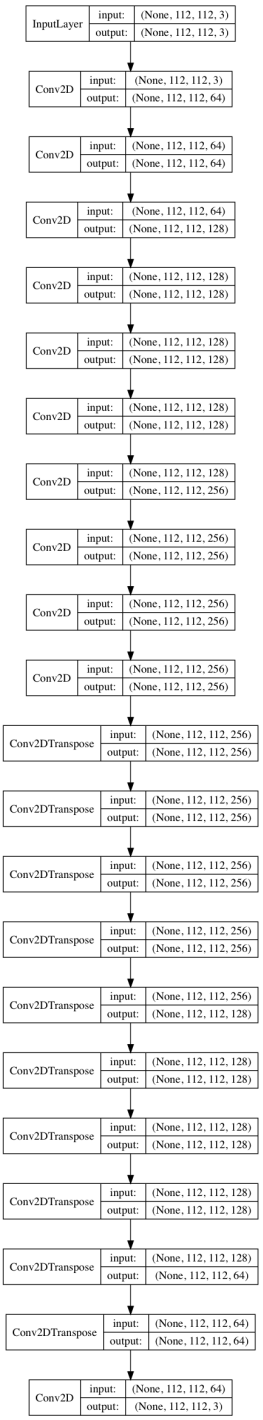
Python codes for this project is written in Python under Tensorflow 1.4.1 framework [56]. All neural network models are iterated for 10^6 times using ADAM optimizer with $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$ as recommended by the authors from the original paper [52]. Weights are initialised by Glorot initialisation (also known as Xavier) [57]. Mini-batch size is set to 16. Learning rate α is initialised to 10^{-4}. By default, all trained models uses the ReLU activation function unless otherwise specified. RGB input patches of 112×112 are extracted from ground truth image and resampled into 28×28 of corresponding LR patches. All downsampling and upsampling operations are performed using Tensorflow's built-in nearest neighbour interpolation. No data augmentation is done prior to training. l_2 loss is used for all models except for experiments involving perceptual information. Due to time constraint and hard drive limitations, only $4\times$ scaling factor is trained and evaluated. For nearest neighbour-resize convolution RED-Net models, trainings usually take 16 hours whereas trainings for

sub-pixel variants take 3 hours. Figure 4.1 and Figure 4.2 shows the architectures of RED-Net for the experiments conducted.

We will look at the outputs of loss functions, peak signal-to-noise ratio (PSNR) and output visualisations for the evaluation. Super-resolved image outputs from the trained networks are provided at the end of this chapter.



(a) Model architecture of standard RED-Net.



(b) Model architecture of RED-Net without skip connections.

Figure 4.1: The two RED-Net models used in evaluations for the impact of hyper-parameters. The left one is with skip connections, the right one without.

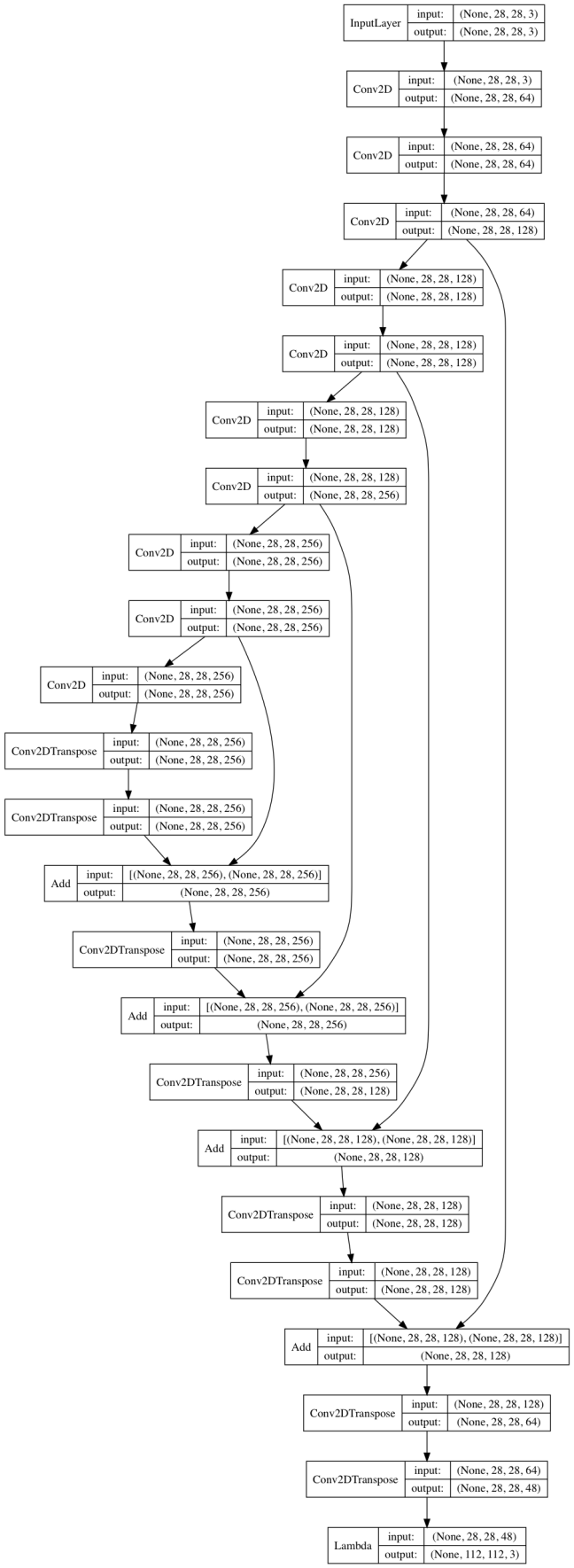


Figure 4.2: Model architecture of RED-Net with sub-pixel convolution implemented.

4.2.1 Peak Signal-to-Noise Ratio (PSNR)

In image processing, PSNR is one of the most commonly used quality estimation metrics to measure the quality of reconstruction image when compared to its ground truth. In this context, noise is the error introduced by the reconstruction process.

In order to compute the PSNR, one has to first calculate the mean-squared error (MSE). Given a ground truth image $f(m, n)$ and its reconstruction $g(m, n)$, the MSE is given by:

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} (f(i, j) - g(i, j))^2 \tag{4.1}$$

Then the PSNR can be calculated by

$$PSNR = 20 \log_{10} \left(\frac{MAX_f}{\sqrt{MSE}} \right) \tag{4.2}$$

where MAX_f represents the maximum possible pixel value of $f(m, n)$.

Numerical metrics are not the best option for evaluation though, which we will see in the results of perceptual loss experiment.

4.3 Activation Functions

The effects of the sigmoid and ReLU activation functions are studied. Figure 4.3 and Figure 4.4 shows the loss and PSNR plots for sigmoid and ReLU activation functions trained with a vanilla RED-Net.

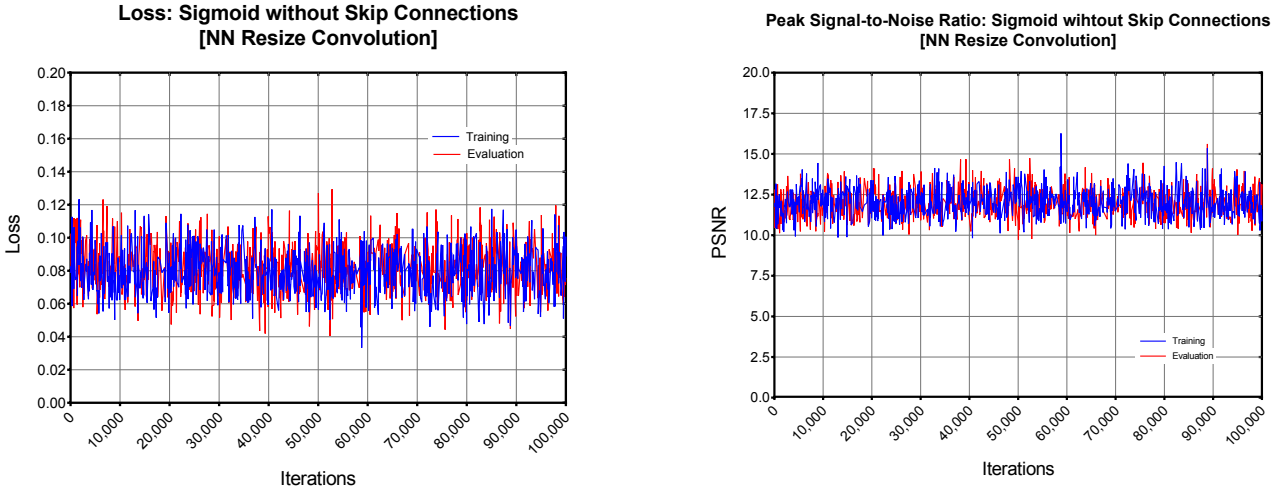
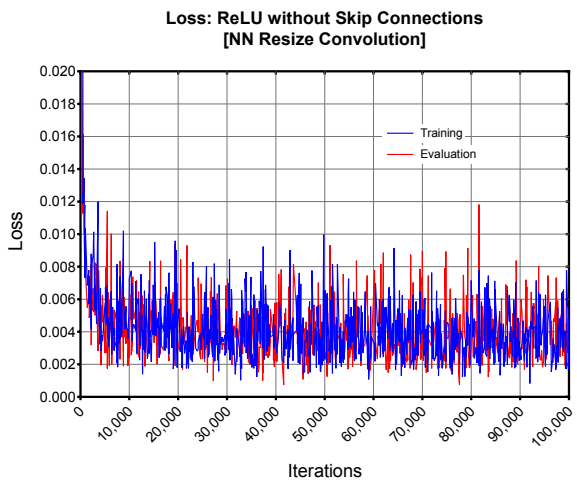


Figure 4.3: Training and evaluation loss and PSNR plots of sigmoid activation function without skip connections.



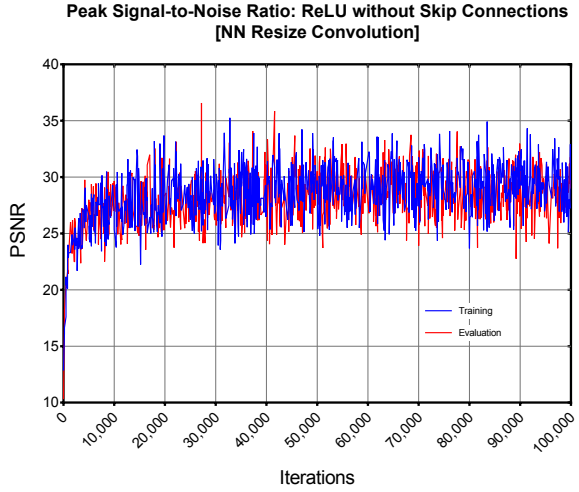
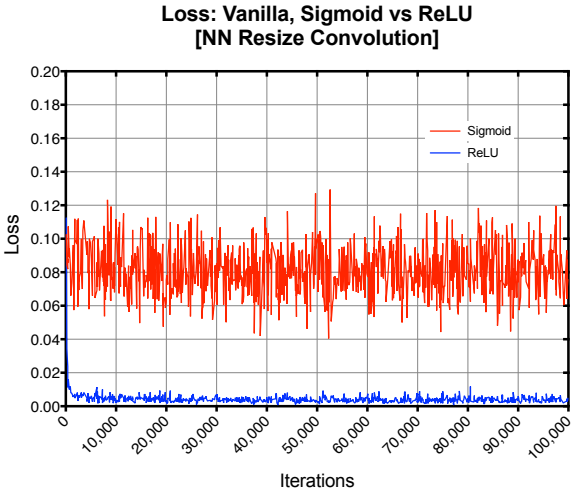


Figure 4.4: Training and evaluation loss and PSNR plots of ReLU activation function without skip connections.



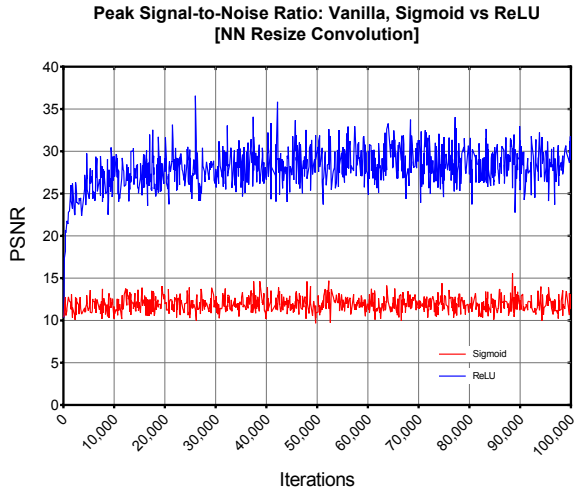


Figure 4.5: Evaluation loss and PSNR plots of sigmoid and ReLU activation function with vanilla RED-Net.

From Figure 4.5, we can observe clearly that the vanilla model is heavily impacted by the deficiency of sigmoid activation function and fails to yield any trend of convergence. A similar phenomenon also prevailed on the PSNR. This is due to the prominence of vanishing gradient and the effect is amplified by the large number of layers. Therefore, the magnitude from the output layer is negligible by the time it reaches the input, rendering the learning to be ineffective. This is further supported by figure 4.6. ReLU activation function, however, is able to display a trend of convergence early in the training progress even though the training curve is very noisy. The observation is also reflected on the PSNR. This is in line with Krizhevsky et al. where it states that ReLU activation function yields 6× better performance than the tanh variant [7] and previous discussions.

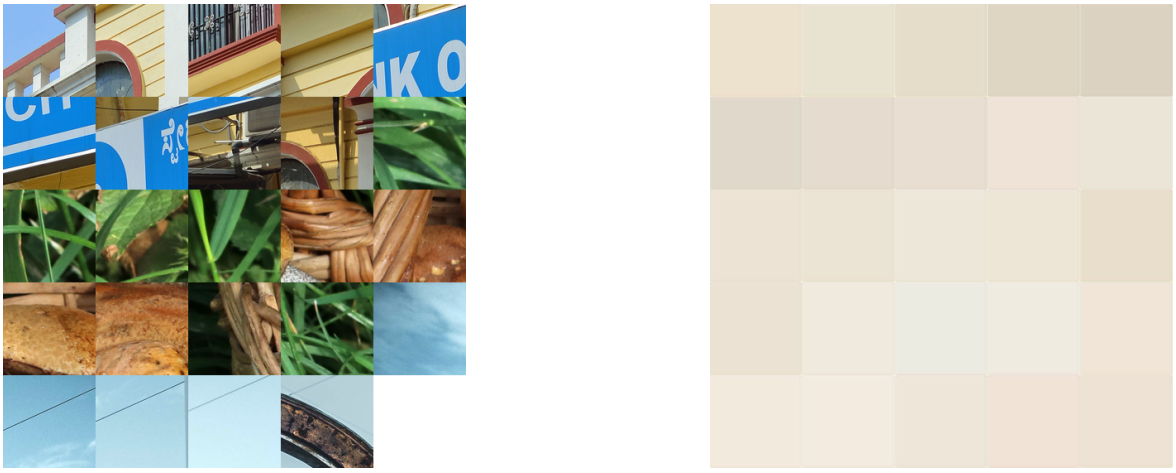


Figure 4.6: Selected ground truth and output image patches from the last 2500 iterations of vanilla sigmoid RED-Net. The network showed its inability to update any parameters, thus only close-to-white outputs were produced.

4.4 Skip Connections

We examine the viability of skip-connections in training models with multitudes of convolutional layer. Figure 4.7 shows the loss and PSNR plots for RED-net with skip connections. Figure 4.8 shows comparison between the evaluation loss and PSNR of the ReLU RED-Net models without and with skip connections.

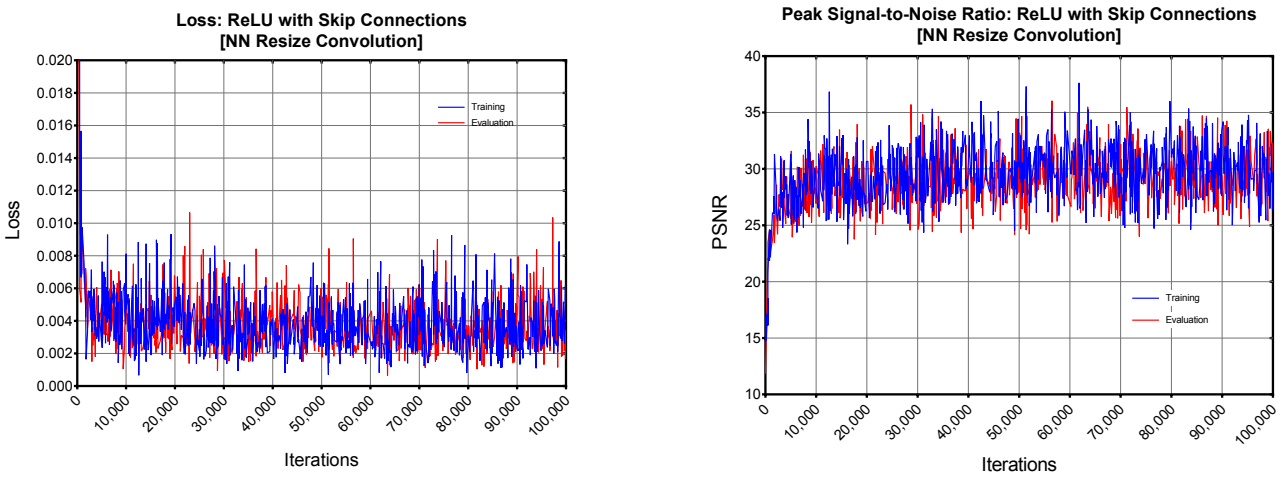
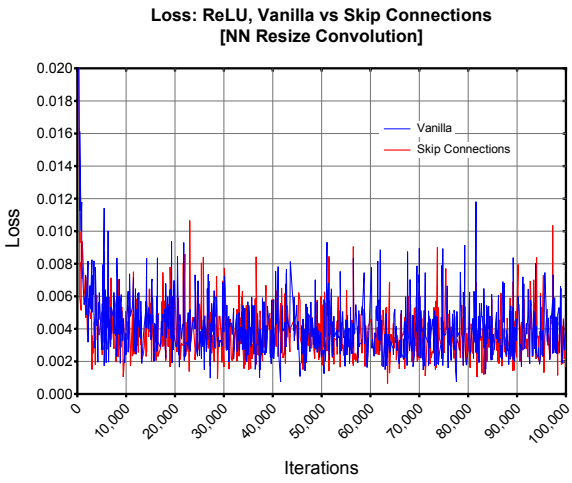


Figure 4.7: Training and evaluation loss and PSNR plots of ReLU activation function with RED-Net with skip connections.



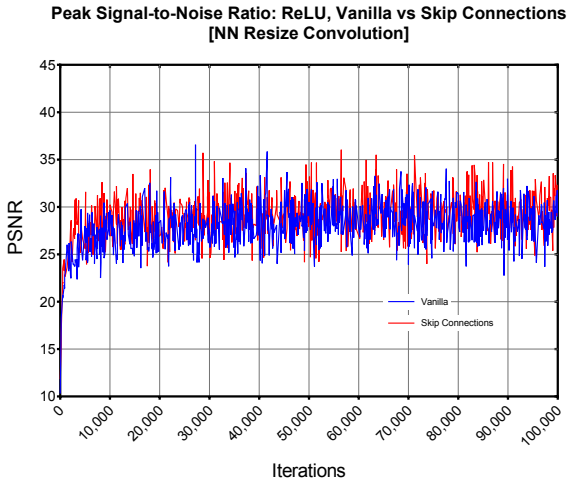
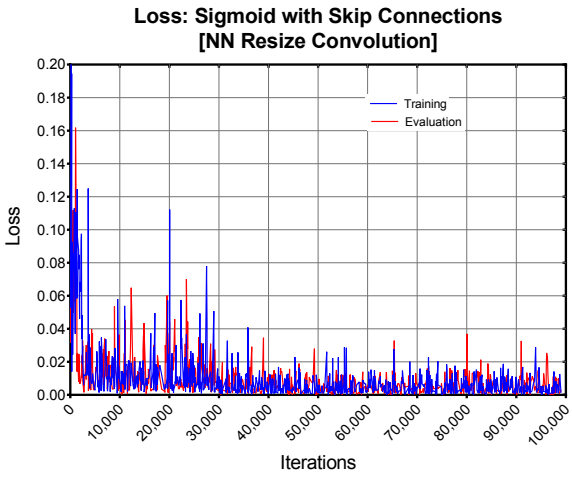


Figure 4.8: Evaluation loss and PSNR plots of ReLU activation function RED-Net with and without skip connections.

By direct comparison with the vanilla version as depicted in Figure 4.8, we can observe that the same model with skip connection can obtain a slightly more optimal solution with a faster rate during training. This is proved by the lower starting point and speedily converged training loss. Apart from it, there seemed to be no difference at all numerically so it may worth repeating the experiment with the sigmoid activation function.



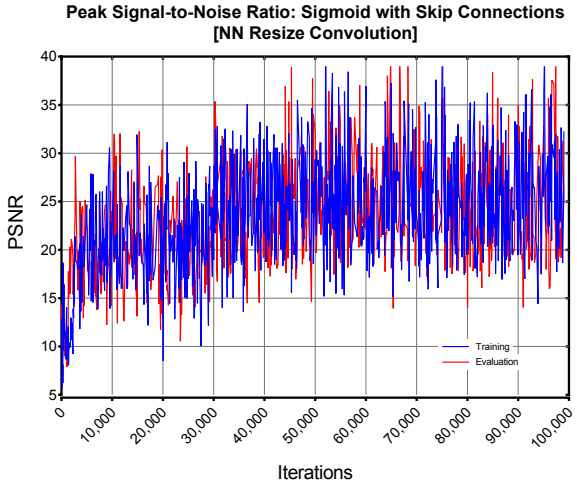


Figure 4.9: Training and evaluation loss and PSNR plots of sigmoid activation function with skip connections configured RED-Net.

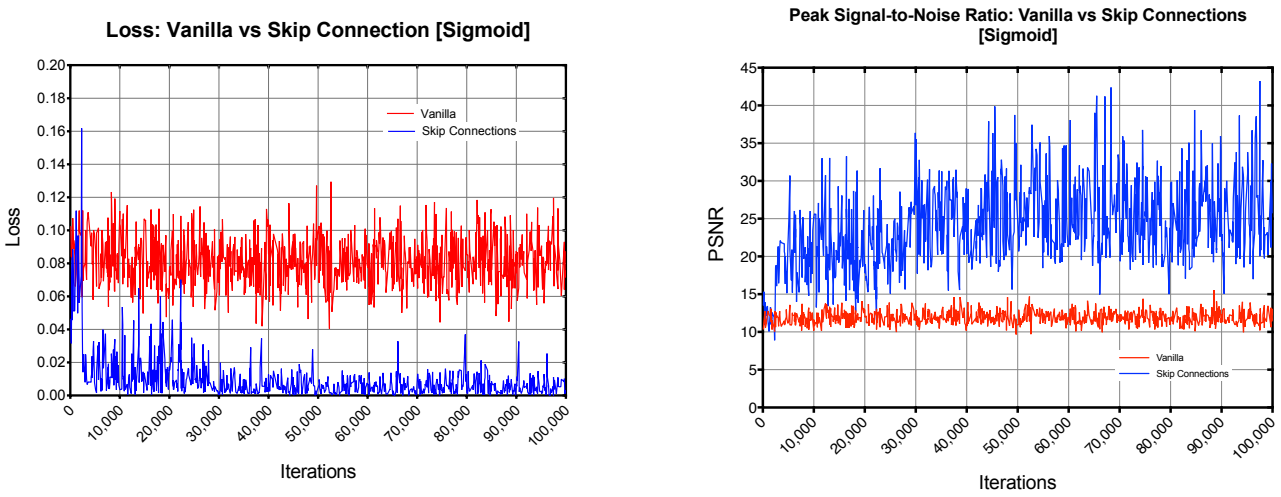


Figure 4.10: Evaluation loss and PSNR plots of sigmoid activation function RED-Net with and without skip connections.

With the skip connections, sigmoid is able to be trained to a certain degree which is demonstrated by the loss and PSNR plots. Looking into the outputs in Figure 4.11, however, the behaviour is not as nice as one would expect. Despite learning, it exhibits great difficulties in recovering colours even after $100,000$ iterations. This can be explained by the properties of sigmoid as per previous discussions. In fact, the learning is probably only learning the residuals of images which is analogical to object slicing in programming context.

Empirically it is found that the training time for each batch is about 0.512 seconds. This will be used to compare with the training time when using sub-pixel convolution.

In sum, skip connections do indeed help back-propagating output gradients especially in deeper models, while the use of nearest neighbour-resize convolution removes any potential checkerboard artifacts [13, 58].

Figure 4.11: Selected ground truth and output image patches from the last 2500 iterations of skip connection configured RED-Net with sigmoid activation function.

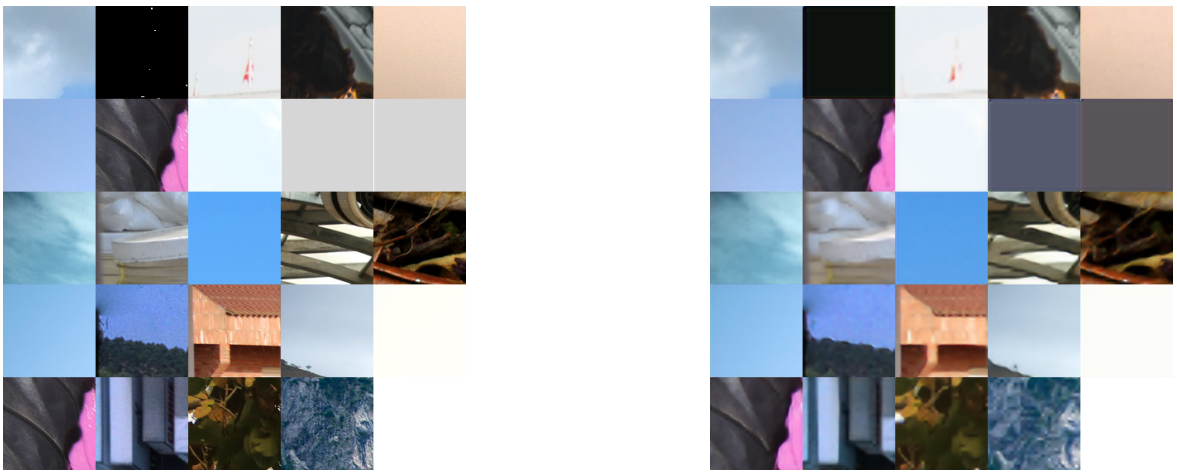


Figure 4.12: Selected ground truth and output image patches from the last 2500 iterations of skip connection configured RED-Net with ReLU activation function.

4.5 Sub-pixel Convolution

We now evaluate the effect of sub-pixel convolution in contrast to resize convolution. Figure 4.13 shows the loss and PSNR plots for the sub-pixel convolution variant at $\alpha = 10^{-4}$, while Figure 4.14 shows an example of sub-pixel convolution output in its infancy.

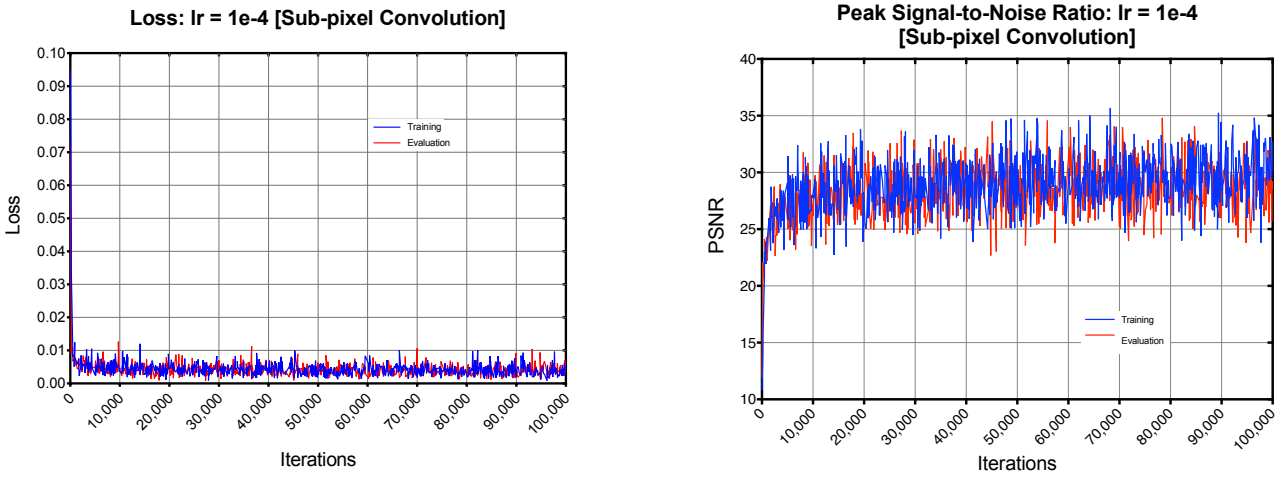


Figure 4.13: Evaluation loss and PSNR plots of ReLU activation function with RED-Net with skip connections.

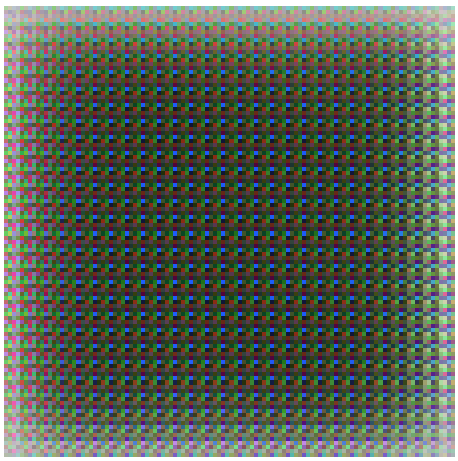


Figure 4.14: Manifestation of pixel reshuffling artifacts during sub-pixel convolution.

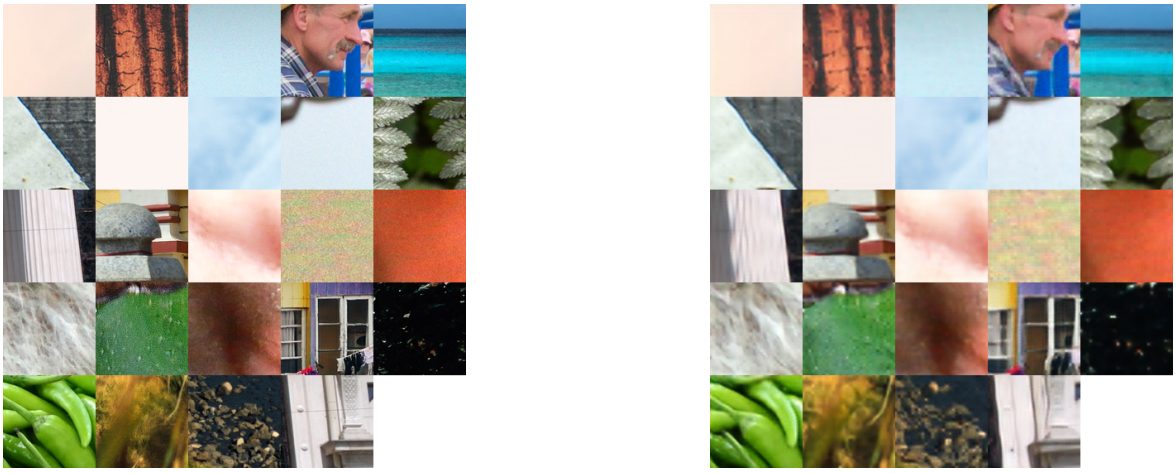


Figure 4.15: Selected ground truth and output image patches from the last 2500 iterations of the sub-pixel convolution variant of RED-Net at $\alpha = 10^{-4}$.

We can notice from Figure 4.14 that 4×4 pixel blocks from different channels are arranged in order. This is the characteristic behaviour of the pixel shuffling operator in the sub-pixel convolution, which involves reshaping output channels to obtain super-resolved output. Figure 4.15 shows the final output from the model where each image patch is nicely super-resolved accordingly.

It is found that the training time per 28×28 patch with sub-pixel convolution is 0.0145s, in other words it is $34\times$ faster than interpolating image from the LR space to HR space during training time. These results agrees with the conclusion drawn by [15] about sub-pixel convolution.

Combining the two interpretations, it is evident that sub-pixel convolution is able to achieve a similar, if not superior performance when compared to resize convolution with much faster runtime than nearest neighbour-resize convolution. In other words, at the same computation complexity, sub-pixel convolution is be able to hold more parameters, thus better modelling power. It does, unlike resize convolution, suffer from the checkerboard artifacts from random initialisation as mentioned from [58]. A potential solution is initialise parameters to convolution resize instead, but further experiments from the machine learning community is required.

4.6 Learning Rate

We also look into the impact to RED-Net training with different magnitudes of learning rates. Experiments were conducted with sub-pixel convolution, with $\alpha = \{10^{-4}, 10^{-5}, 10^{-6}\}$. No such experiments are conducted with the resize convolution due to the long training time which violates GPU fair use policy. Below shows the loss and PSNR plots for different α.

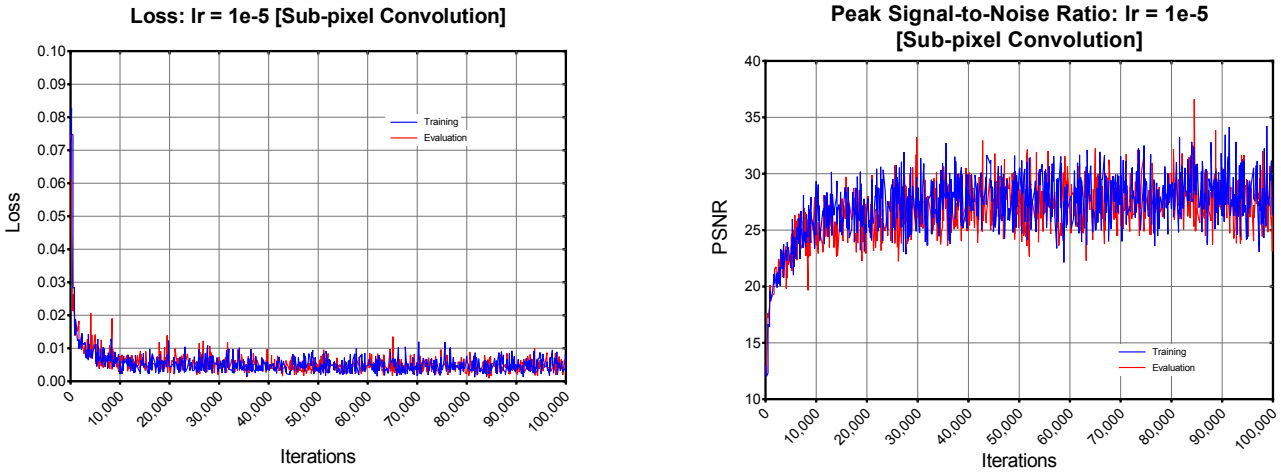


Figure 4.16: Training and evaluation loss and PSNR plots of RED-Net with skip connections and sub-pixel convolution at $\alpha = 10^{-5}$.

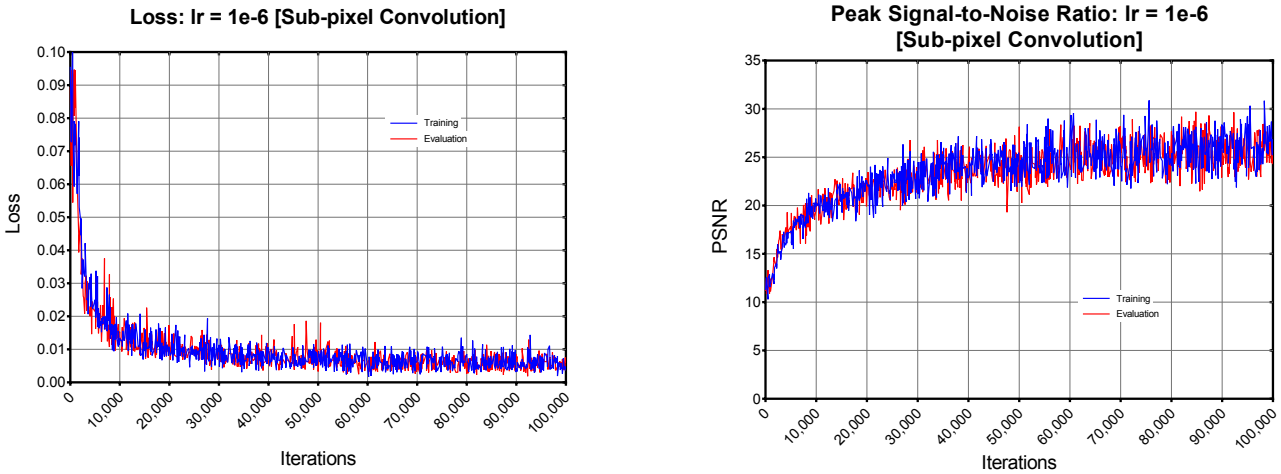


Figure 4.17: Training and evaluation loss and PSNR plots of RED-Net with skip connections and sub-pixel convolution at $\alpha = 10^{-6}$.

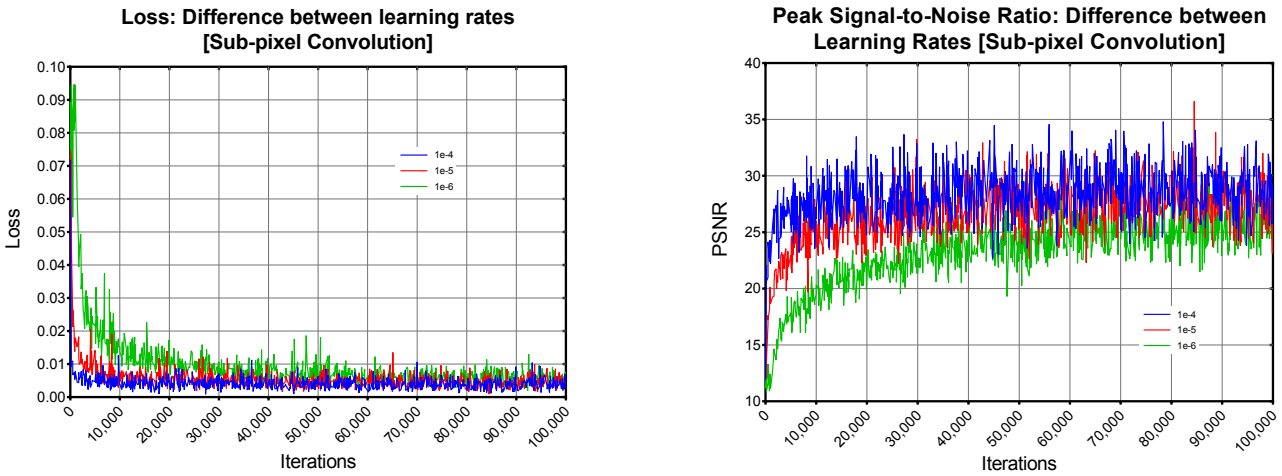


Figure 4.18: Evaluation loss and PSNR plots of RED-Net with skip connections and sub-pixel convolution at different α.

From Figure 4.18, we can notice a trend of decreasing rate of convergence both in loss and psnr plots. In particular, at $\alpha = 10^{-6}$, the network showed symptoms of failure with persisting pixel shuffling artifacts at the outputs. These results emphasise the fact that a proper learning rate setting is imperative to train a neural network model effectively and efficiently.

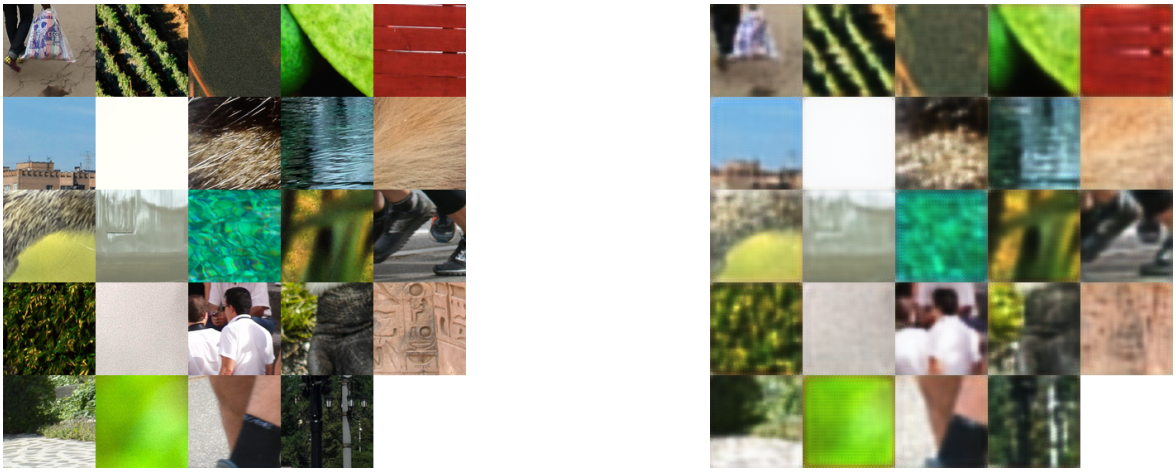


Figure 4.19: Selected ground truth and output image patches from the last 2500 iterations of sub-pixel convolution variant of RED-Net at $\alpha = 10^{-6}$. From the outputs, we can clearly notice the characteristic pixel reshuffling patterns.

4.7 Perceptual Loss

Perceptual loss yields some intriguing results in that it has prerequisites of specific configurations. An attempt is also made to combine the sub-pixel convolution and perceptual loss. The model in Figure 4.1 is modified such that a VGG16 network is being fed from the output of RED-Net (not visualised due to size), to obtain the desired perceptual loss value.

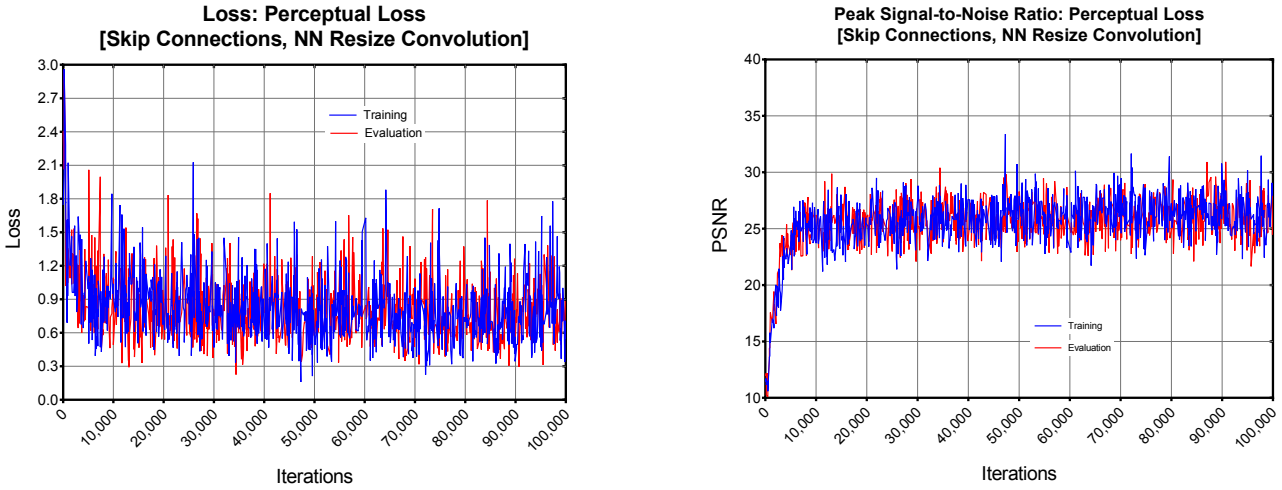


Figure 4.20: Training and evaluation loss and PSNR plots of RED-Net with skip connections and perceptual loss.

From Figure 4.20, the loss and psnr plots are rather noisy compared to previous sections. In truth, [12] does not recommend evaluating numerical metrics either due to its inaccuracy in measuring perceptual content reconstruction. Thus, the plots are best to be interpreted only to verify the learning is indeed taking place.

Figure 4.21: Selected ground truth and output image patches from the last 2500 iterations of RED-Net with skip connections and perceptual loss.

We can observe from Figure 4.21 that the general structure of objects are preserved at the outputs, while the actual contents are not smoothed but rather interpolated. This is typical due to our network now doing predictions on the missing pixels based on the VGG16 ImageNet-pretrained network.

An attempt was made to combine both the perceptual and MSE losses such that pixel contents were preserved to a certain degree. The results however are highly similar to Figure 4.20 due to the difference in magnitude of the two losses. One way to rectify it is to scale either loss by a constant, i.e. γl_{MSE}, but due to time constraint this is not done.

This experiment is also repeated on the sub-pixel convolution network given the merits of the method. The results are problematic though as we can observe from Figure 4.22 that the sub-pixel convolution reshuffling operator artifacts in Figure 4.14 resurfaced.

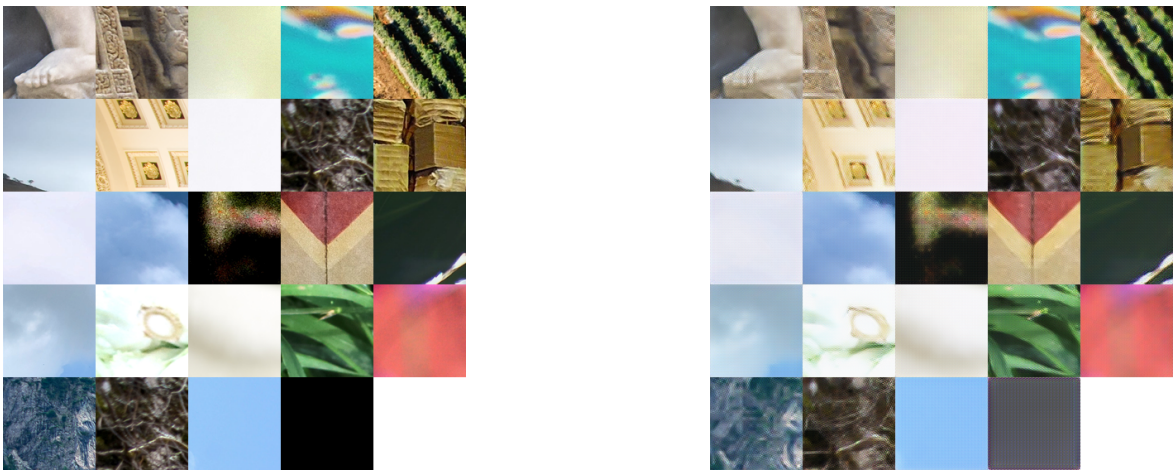


Figure 4.22: Selected ground truth and output image patches from the last 2500 iterations of RED-Net with sub-pixel convolution and perceptual loss.

4.8 Super-resolved Outputs

This section covers super-resolved images from different configurations of RED-Net. Specifically, we look at the outputs under ReLU activation function with and without skip connections, sub-pixel convolution and perceptual loss settings.

Figure 4.23: A list of images that are used to generate super-resolved outputs in this section.

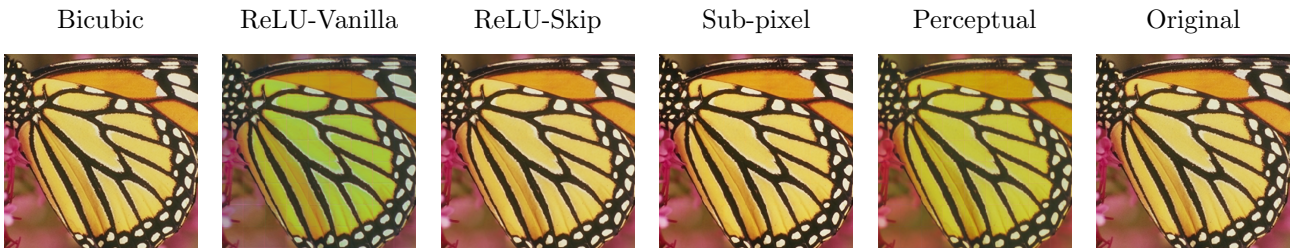


Figure 4.24: Bicubic, ReLU-Vanilla, ReLU-Skip, Sub-pixel, Perceptual reconstruction results and corresponding original reference image of Set5 Butterfly.

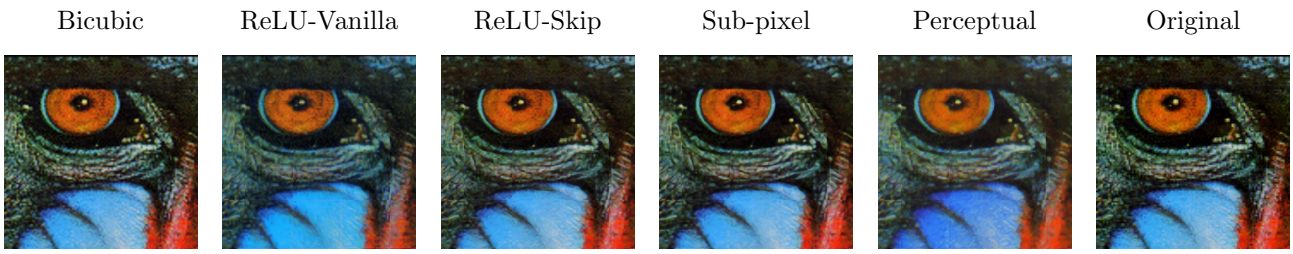


Figure 4.25: Bicubic, ReLU-Vanilla, ReLU-Skip, Sub-pixel, Perceptual reconstruction results and corresponding original reference image of Set14 Baboon.

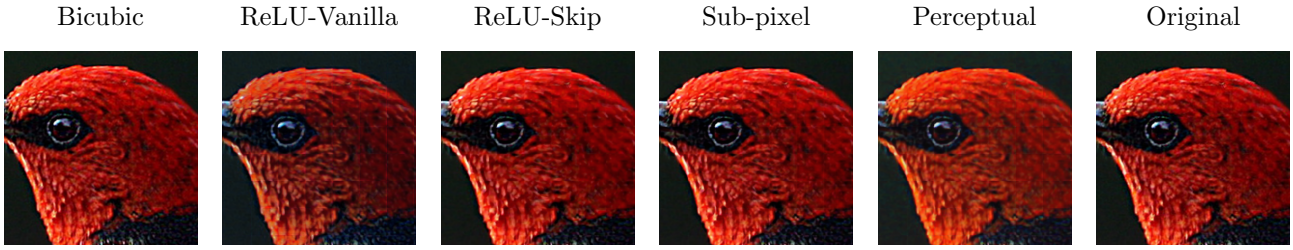


Figure 4.26: Bicubic, ReLU-Vanilla, ReLU-Skip, Sub-pixel, Perceptual reconstruction results and corresponding original reference image of DIV2K 0853.

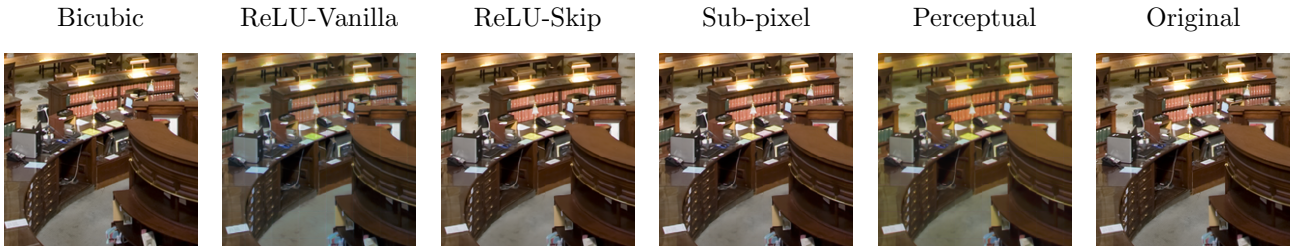


Figure 4.27: Bicubic, ReLU-Vanilla, ReLU-Skip, Sub-pixel, Perceptual reconstruction results and corresponding original reference image of DIV2K 0884.

One observation that is found in the inferenced super-resolved images is the presence of grid artifacts which resembles JPEG block artifacts. This is due to the lack of spatial coherence between each image patch and pixel predictions, such that the network is confined to information within itself and unable to predict these areas where stitching is vital. This is especially prevalent at the output of perceptual loss without per-pixel information. A possible solution is to do inference with half stride and selectively replace the overlapping part.

Another one is the discolouration at the outputs of vanilla RED-Net ReLU model. While numerically there are limited differential between RED-Net ReLU model without and with skip connections, visually we can notice the ambiguities. Comparing this with the same model with skip connections, it further supports the use of skip connections which helps back-propagation of information to a great extent.

Figure 4.28: An example of grid artifacts formed during inference.

We find that the inference time of the sub-pixel convolution variant of RED-Net is magnitudes faster than the resize convolution. By comparison, the resize convolution RED-Net models takes 45.5 and 45.7 seconds to complete inferences and patch concatenations on DIV2K 0853 and DIV2K 0884, whereas the sub-pixel convolution variant uses only 5.96 and 6.06 seconds. In other words, the sub-pixel convolution is about 7.5× faster without much sacrificed quality. This further encourages the community to use the sub-pixel convolution despite the downfall of checkerboard artifacts and non-trainable upsampling layer.

Chapter 5

Conclusion

This project covered the theoretical and practical aspects of super-resolution and explores state-of-the-arts methods in this area. For the basics, we have covered the fundamentals of deep learning and emphasised the importance of proper hyper-parameter configurations. In image processing and computer vision context, we have covered the use of MSE loss function and perceptual loss, resize convolution and sub-pixel convolution and explored the merits and drawbacks of using or combing these methods.

In summary, the contributions of this project includes:

1. Unifies the theoretical foundation of machine learning and various deep learning state-of-the-art technologies and apply them to super-resolution.
2. Demonstrates in terms of hyper-parameters that sigmoid is a poor activation function to be applied to very deep models and learning rate exhibits certain impact to the convergence rate and learning process; in terms of network configurations that the lack of skip connections give discolouration and strong grid artifacts, sub-pixel convolution provides a much faster training and inference time but cannot be applied to perceptual loss, and perceptual exhibits some discolouration despite better reconstruction of fine details.

5.1 Future Works

There is an excellent scope to expand the work of this project. In particular, some of the techniques covered in the project are recently developed and deserved to be explored further for optimisations and applications in other similar image processing problems.

Many tasks that are planned for this project cannot be accomplished due to the competitiveness of GPU usage in the university servers and sudden change of project direction from image denoising to image super-resolution. For example, GANs are not experimented within the project and it would be interesting to measure the difference in performance compared to regular CNNs. Another possible experiment is to test the perceptual loss with other pre-trained networks or other layer outputs for possible improvements.

Bibliography

[1] S. University, "Stanford cs231n convolutional neural networks for visual recognition." `http://cs231n.github.io/`.

[2] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, (Washington, DC, USA), pp. 886–893, IEEE Computer Society, 2005.

[3] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, pp. 91–110, Nov. 2004.

[4] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded-up robust features (surf)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346 – 359, 2008. Similarity Matching in Computer Vision and Multimedia.

[5] I. Jolliffe, *Principal Component Analysis*. Springer Verlag, 1986.

[6] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," in *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, 1998.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[8] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," pp. 1–14, 2014.

[9] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, C. Hill, and A. Arbor, "Going Deeper with Convolutions," pp. 1–9, 2014.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[11] K. Kaska and P. Goliski, "Using deep learning for single image super resolution." `https://blog.deepsense.ai/using-deep-learning-for-single-image-super-resolution/`.

[12] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9906 LNCS, pp. 694–711, 2016.

[13] A. Odena, V. Dumoulin, and C. Olah, "Deconvolution and checkerboard artifacts," *Distill*, 2016.

[14] C. Osendorfer, H. Soyer, and P. V. D. Smagt, "Image Super-Resolution with Fast Approximate Convolutional Sparse Coding," *International Conference on Neural Information Processing*, no. Iconip, pp. 250–257, 2014.

[15] 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," 2016.

[16] 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?," 2016.

[17] R. Jain, R. Kasturi, and B. G. Schunck, *Machine Vision*. New York, NY, USA: McGraw-Hill, Inc., 1995.

[18] J. S. Lee, "Digital Image Enhancement and Noise Filtering by Use of Local Statistics," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-2, pp. 165–168, mar 1980.

[19] T. Huang, G. Yang, and G. Tang, "A fast two-dimensional median filtering algorithm," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 27, pp. 13–18, feb 1979.

[20] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, pp. 839–846, jan 1998.

[21] K. S. Shanmugam, F. M. Dickey, and J. A. Green, "An Optimal Frequency Domain Filter for Edge Detection in Digital Pictures," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, pp. 37–49, jan 1979.

[22] I. Daubechies, "The wavelet transform, time-frequency localization and signal analysis," *IEEE Transactions on Information Theory*, vol. 36, pp. 961–1005, sep 1990.

[23] A. S.-D. SERŠIĆ, "Signal Decomposition Methods for Reducing Drawbacks of the DWT," *Engineering Review*, vol. 32, no. 2, pp. 70–77, 2012.

[24] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image Denoising by Sparse 3-D Transform-Domain Collaborative Filtering," *IEEE Transactions on Image Processing*, vol. 16, pp. 2080–2095, aug 2007.

[25] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," *Proceedings of the 25th international conference on Machine learning - ICML '08*, pp. 1096–1103, 2008.

[26] G. E. Hinton and R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[27] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[28] M. LLC, "Stanford UFLDL Tutorial, Autoencoders." `http://ufldl.stanford.edu/tutorial/unsupervised/Autoencoders/`.

[29] A. Makhzani and B. J. Frey, "k-Sparse Autoencoders," *CoRR*, vol. abs/1312.5, 2013.

[30] J. Xie, L. Xu, and E. Chen, "Image Denoising and Inpainting with Deep Neural Networks," *Nips*, pp. 1–9, 2012.

[31] V. Jain and S. Seung, "Natural Image Denoising with Convolutional Networks," in *Advances in Neural Information Processing Systems 21* (D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, eds.), pp. 769–776, Curran Associates, Inc., 2009.

[32] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction," in *Artificial Neural Networks and Machine Learning – ICANN 2011: 21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part I* (T. Honkela, W. Duch, M. Girolami, and S. Kaski, eds.), pp. 52–59, Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[33] A. Mousavi, A. B. Patel, and R. G. Baraniuk, "A Deep Learning Approach to Structured Signal Recovery," *CoRR*, vol. abs/1508.0, 2015.

[34] L. Gondara, "Medical image denoising using convolutional denoising autoencoders," *CoRR*, vol. abs/1608.0, 2016.

[35] O. Quijas, J; Fuentes, "Removing JPEG blocking artifacts using machine learning," in *2014 Southwest Symposium on Image Analysis and Interpretation*, pp. 77–80, apr 2014.

[36] C. Dong, C. C. Loy, K. He, and X. Tang, "Image Super-Resolution Using Deep Convolutional Networks," *CoRR*, vol. abs/1501.0, 2015.

[37] K. Yu, C. Dong, C. C. Loy, and X. Tang, "Deep Convolution Networks for Compression Artifacts Reduction," pp. 1–13, 2016.

[38] X.-J. Mao, C. Shen, and Y.-B. Yang, "Image Restoration Using Convolutional Auto-encoders with Symmetric Skip Connections," *CoRR*, vol. abs/1606.0, 2016.

[39] S. Gu, L. Zhang, W. Zuo, and X. Feng, "Weighted Nuclear Norm Minimization with Application to Image Denoising," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2862–2869, jun 2014.

[40] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *CoRR*, vol. abs/1501.00092, 2015.

[41] C. Dong, C. C. Loy, and X. Tang, "Accelerating the Super-Resolution Convolutional Neural Network,"

[42] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced Deep Residual Networks for Single Image Super-Resolution," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, vol. 2017-July, pp. 1132–1140, 2017.

[43] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27* (Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds.), pp. 2672–2680, Curran Associates, Inc., 2014.

[44] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network," 2016.

[45] G. K. Wallace, "The JPEG still picture compression standard," *Communications of the ACM*, pp. 30–44, 1991.

[46] R. Hahnloser, R. Sarpeshkar, M. Mahowald, R. Douglas, and H. S. Seung, "Digital Selection and Analogue Amplification Coexist in a Cortex-inspired Silicon Circuit," *NATURE*, vol. 405, no. 31, pp. 517–567, 2000.

[47] T. Dang-Ha, "A guide to receptive field arithmetic for convolutional neural networks." `https://medium.com/mlreview/a-guide-to-receptive-field-arithmetic-for-convolutional-neural-networks-e0f514068807`.

[48] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *ArXiv e-prints*, Mar. 2016.

[49] D. Masters and C. Luschi, "Revisiting Small Batch Training for Deep Neural Networks," *ArXiv e-prints*, Apr. 2018.

[50] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[51] A. Geitgey, "Machine Learning is Fun! Part 3: Deep Learning and Convolutional Neural Networks." `https://medium.com/@ageitgey/machine-learning-is-fun-part-3-deep-learning-and-convolutional-neural-networks-f40359318721`.

[52] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[53] 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.

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

[55] R. Zeyde, M. Elad, and M. Protter, "On single image scale-up using sparse-representations," in *Proceedings of the 7th International Conference on Curves and Surfaces*, (Berlin, Heidelberg), pp. 711–730, Springer-Verlag, 2012.

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

[57] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (Y. W. Teh and M. Titterington, eds.), vol. 9 of *Proceedings of Machine Learning Research*, (Chia Laguna Resort, Sardinia, Italy), pp. 249–256, PMLR, 13–15 May 2010.

[58] A. Aitken, C. Ledig, L. Theis, J. Caballero, Z. Wang, and W. Shi, "Checkerboard artifact free sub-pixel convolution: A note on sub-pixel convolution, resize convolution and convolution resize," 2017.