Offen im Denken

Universitaet Duisburg-Essen
Fakultaet fuer Ingenieurwissenschaften
Lehrstuhl fuer Automatisierungstechnik und Komplexe Systeme

Feature Preserving Up- and Downsampling based on Neural Networks

Merkmalserhaltendes Up- und Downsampling von 2D-Skalarfeldern basierend auf neuronalen Netzwerken

Masterarbeit

Desaraju, Seeta Rama Aditya
Matr.-Nr.: 3027926

Erstpruefer	Prof. Dr.-Ing. Steven X. Ding
Zweitpruefer	Prof. Dr.-Ing. Dirk Soeffker
Betreuer	Prov.-Doz. Dr.-Ing. habil. Dirk Joachim Lehmann [IAV GmbH]

March 16, 2020

Abstract

Single Image Super Resolution (SISR) [11] is a topic widely sought for as it allows for upscaling of Low Resolution (LR) images to High Resolution (HR) images. This when successfully achieved can be applied to tasks ranging from but not exclusive to upscaling medical imagery for better detection of illness, improving on security footage and classification from LR images. Though Deep Learning (DL) has been researched for a while, SISR with its ample possible applications is still a relatively new area of research which was only delved into for a little over a decade. The availability of raw digital data and lack of computational power played were major factors at reducing its research. Now with the availability of high computational power and astounding amounts of raw unstructured data on the internet, the research in SISR rapidly exploded. Starting with the pioneering work of Dong et al. [11] with the model SRCNN, the quality of images produced by upscaling models shot up. And with the introduction of Generative Adversarial Networks (GAN) to the field of Super Resolution (SR), the perceptual quality leapt further forward.

Downscaling or downsampling of an image is often overlooked even with it being one of the most used image processing operation. This thesis is an extensive quantitative study of behavioural changes during the training and the changes in the effective performance of ERCA [19], SRGAN [32] and SRFeat [42] depending on the domain specificity of the datasets. Further the influence of adaptive downsampling approaches, like the perceptually based downscaling [41] and detail preserving image downscaling [57], on the performance of the upsampling models is studied and quantified. The results are quantitatively compared and by doing this draw upon the behavioural differences in the chosen models based on the training datasets and also show how each of the selected downsampling methods can potentially increase the performance of the current state-of-the-art models.

Acknowledgement

I would like to extend my sincere gratitude for the guidance of Prof. Dr.-Ing. Steven X. Ding [AKS], Prof. Dr.-Ing. Dirk Soeffker [SRS] and Prov.-Doz. Dr.-Ing.habil. Dirk Joachim Lehmann [IAV GmbH] and the opportunity to do my Master thesis at IAV GmbH in collaboration with the Institute of Automatic Control and Complex Systems, University of Duisburg Essen. It was because of them and the support from Dr.-Ing. Chris Louen[AKS] that I overcame all challenges in this work. I also am deeply grateful for the professors and asst. professors at the University of Duisburg Essen without whose teachings, I would not have achieved my current state. I am also very thankful for the valuable suggestions and support from the colleagues in the Vision Team, TM-R13.

I am thankful for everything I have learnt during the time that I worked at IAV GmbH and the experience I have gathered. Owing to all these pieces falling where they should, I have been able to successfully complete my thesis.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Task Description

Image re-sampling is the problem of altering the resolution of the image available and has two directions of resampling that could be achieved. Image upsampling commonly referred to as image super resolution (SR) is the problem of reconstructing an accurate high-resolution (HR) image from its low-resolution (LR) counterpart. Image downsampling on the other hand is the problem of reducing the available high-resolution (HR) image to a low-resolution (LR) image.

In this thesis, three super resolution models SRGAN [32], SRFeat [42] and ERCA [19] are trained under similar conditions on DIV2K [7], flickr1024 [54], oregon wildlife dataset [40] and stanford cars dataset [28]. The results are then compared against each other and critiqued. Three downsampling approaches namely the Content-adaptive downscaling [27], perceptually based downscaling [41] and detail preserving image downscaling [57] are studied and the influence of two of them on the performance of the upsampling models is quantified and evaluated.

1.2 Motivation

Since 2014 when the paper by Dong et al. [11] was published, there was a huge leap in the deep learning community to better the super resolution of the LR images using CNNs. There was a steady growth in the quality of the images produced till GAN [16] was published in 2017. The quality of images spiked as a result of the perceptual superiority of the images generated by these generative adversarial networks. In SRGAN [32], Christian et al. proposed a perceptual loss which consists of adversarial and content loss to attain a very perceptually pleasing SR image with a $4\times$ sampling factor. Many models were proposed from then and set new

standards with time. With an interest in the behavioural differences among upsampling models that were proposed there after, the work in this thesis was carried out. After an intensive review of the current literature available on super resolution, the models SRGAN [32], SRFeat [42] and ERCA [19] were chosen due to their part similarity in the neural network architecture. This work includes extensive study of the influence of domain-specific and multi-domain datasets and their choice on these models trained under similar conditions and similar training parameters all the while studying the training behaviour and model performance overall. The performance of the models on benchmark data sets for super resolution are quantified using PSNR, SSIM [56], RMSE and PI [8](Perceptual Index) . This thesis also contains a comparative study of two adaptive down sampling methods, the perceptually based down-scaling [41] and detail preserving image downscaling [57]. Further the influence of these downsampling methods if they were to be used to generate the LR images that are to be fed as inputs to any upsampling methods on the SR image quality is extensively studied. Any inference could potentially be directed to the training process as the change in the downsampling method for images while making the LR images during the pre-processing phase could also alter accordingly.

1.3 Structure of the Thesis

A brief introduction is given on the thesis in chapter 1. Chapter 2 explains the fundamentals necessary to understand the upsampling and downsampling problems further. Chapter 3 gives a brief overview of the conventional interpolation approaches followed by the deep learning based approaches for upsampling and adaptive downsampling methods. Chapter 4 describes my implementation, the training methods used, the datasets used to train the models and the method for testing the evaluation of downsampling methods. Chapter 5 is the evaluation of my implementation and a quantitative comparison based on their performance on the benchmark data sets. The results, the application, the achievement and drawbacks are discussed in chapter 6 and a descriptive conclusion is presented while mentioning potential future work in the field in the chapter 7.

CHAPTER 2

Fundamentals

Traditionally coding for a task involves the programmer or the user to feed the data and rules necessary to get the desired solution for the task. Here the data required is low and so is the computational expense to get the desired solution. In machine learning on the other hand the desired solution and data are fed to a model and the model learns the set of rules that would be needed to get the desired solution. This process is called training a model. A model as a norm in the machine learning community is used interchangeably with neural network which is a series of layers. Each layer is a cluster of nodes referred to as neurons and each of the neuron essentially is a function of the available data input to map it to the output. Each parameter in the function is initialised by a random value at the beginning of the training and through a series of iterative steps, the model converges on the values for every one of the parameters to successfully get to the desired solution. The number of parameters to train and the data required to train the model is higher depending on the task that the model needs to solve.

Deep learning is a subset of machine learning and involves more than one layer of neurons. With the increase in layers, the number of parameters that need to be trained increases and with it the computational expense. In general the number of iterations and the data required by the model to undergo successful training is directly proportional to the depth of the network or model. Any task that deals with images almost consistently need multiple layers in the model. Deep learning requires a large amount of data and has a high training time but it has also proven to show better results. For example, a color (RGB) image with a resolution of 64×64 comprises of 3 layers (Red, Green and Blue) which amounts to $12,288$ input pixel values. One neuron would have that many parameters to train and considering a few neurons in a layer and a few layers, the computational expense skyrockets. Though machine learning has existed for a really long time, researchers started showing

interest in it for only a little over a decade. The reasons among others were the lack of computational power, the lack of network speeds and availability of data. Now though, the computers have high computational power and there is ample amount of unstructured data available which shot up the use and the importance of deep learning.

2.1 Neural Networks

A simple neural network as in fig.2.1 consists of three layers, the input layer, a hidden layer and the output layer. Deep neural networks as in fig.2.1 refer to the increased number of hidden layers in a network. The connections indicated in fig.2.1 are the computations that are to be performed while training. If all the neurons of the current layer are connected to the previous layer and the following layer then, the layer is referred to as a fully connected or a dense layer.

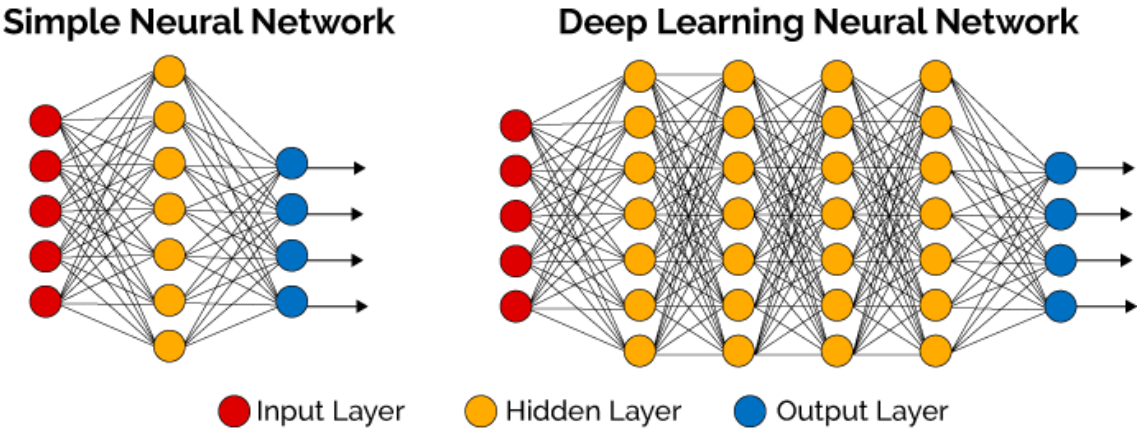

Figure 2.1: Basic neural network architecture [60]

Dense layer is among the core layers and one of the most used layer in the neural network architectures but the amount of computations are high as each of the parameters that are to be trained are independent of each other. For image data where the input data is high, the computations turns excessive. A better alternative would be to use Convolutional Neural Networks (CNN) explained in section 2.3

Training a neural network refers to the iterative process through which the machine or rather the model learns to predict the desired output. This learning is widely classified into 'Supervised Learning' and 'Unsupervised Learning.' In supervised learning the model is fed with structured data each input containing both the set of input features (the pixel values for images) and a label (desired output to be predicted). The model converges on predicting the desired output. In unsupervised learning the model is fed with unstructured data (for eg.: raw images, audio, text) and the model learns to extract features and pair them together for similarity. The idea of training thus far is by nature empirical. The following notation will be used

through chapter 2:
$\mathbf{x}$: input $feature\ vector$
y : $label\ /\ output$
n_w : $width\ of\ the\ input\ image$

n_h : $height\ of\ the\ input\ image$
n_c : $number\ of\ channels, n_c = 3\ for\ an\ RGB\ image$
n_x : $shape\ of\ the\ input\ \mathbf{x}$
m : $batch\ size\ /\ number\ of\ training\ examples$

For every training set (**x**, y) and considering a binary classification problem,

$\mathrm{n}_x : n_w \times n_h \times n_c$
$\mathbf{x} \in R^{n_x}$
$y \in \{0, 1\}$

For a training set of m training examples,

$\mathrm{m}\ \ training\ examples : \left\{\left(\mathbf{x}^{(1)}, y^{(1)}\right), \left(\mathbf{x}^{(2)}, y^{(2)}\right), ..., \left(\mathbf{x}^{(m)}, y^{(m)}\right)\right\}$
$\mathbf{X} \in R^{\,n_x \times m}$
$\mathbf{X}.shape = \ (n_x, m)$
$\mathbf{Y} \in R^{\,1 \times m}$
$\mathbf{Y}.shape = \ (1, m)$

2.1.1 Logistic Regression

The supervised learning in case of a binary classification is carried out by 'Logistic Regression'. $\hat{y}$ is the probability that y is 1 given the input features, **x**. The parameters that are to be learnt during the training process are **w** and b.

w : weights, $\mathbf{w} \in R^{\,n_x}$
b : $bias, b \in R$
$\hat{y}$: $\ prediction\ of\ y,\ 0 \leq \hat{y} \leq 1$
$\hat{y} = p\,(y = 1 \mid \mathbf{x})$

$$z = \mathbf{w}^T * \mathbf{x} + b \tag{2.1}$$
$$a = \hat{y} = \sigma\,(z) \tag{2.2}$$
$$\sigma\,(z) = \frac{1}{1 + e^{-z}} \tag{2.3}$$

if $z >> 0,\ \sigma(z) = 1,$
$if\ z << 0,\ \sigma(z) = 0$

On successful training, the model is to learn to predict y accurately, $\hat{y} \rightarrow y$. This is achieved through minimizing the cost function J which is the cumulative loss over the m training examples.

$$J(\mathbf{w}, b) = \frac{1}{m} * \sum_{i=1}^{m} L\left(\hat{y}^{(i)}, y^{(i)}\right) \tag{2.4}$$

where $L\left(\hat{y}^{(i)}, y^{(i)}\right)$ is the Loss function for the i-th training example. The parameters w and b are the values that generate the minimum cost function value $J(w, b)$.

2.1.2 Gradient Descent

Gradient descent is the process of reaching the point where the values of $\mathbf{w}$ and b for the minimum cost function J through progressive reduction. Fig.2.2 visualizes a simpler graph between the cost function $J(\mathbf{w})$ against $\mathbf{w}$ (b is ignored for this representation for simplicity). The weights are randomly initialized and are updated through iterations by a small value every iteration to close in on the global cost minima.

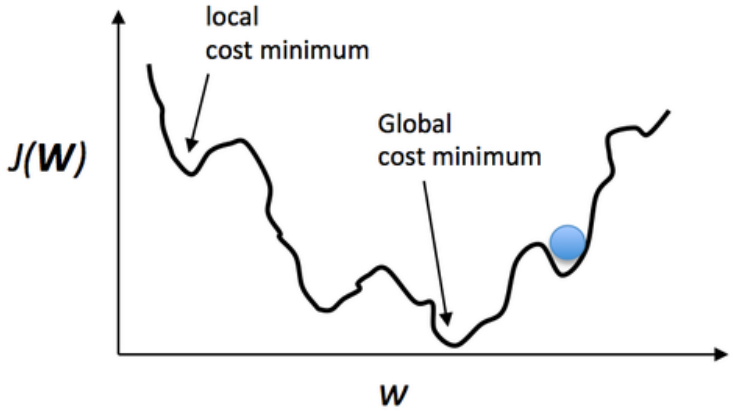

Figure 2.2: Gradient descent with local and global minima [44]

$\alpha:\ learning\ rate$
$dw:\ change\ in\ weights\ for\ update$

For every iteration, the w and b are updated as follows:

$$\mathbf{w} \quad : \mathbf{w} - \alpha\left(\frac{dJ(\mathbf{W}, b)}{dw}\right) \tag{2.5}$$

$$b \quad : b - \alpha \left(\frac{dJ(\mathbf{W},b)}{db} \right) \tag{2.6}$$

2.1.3 Forward Propagation

The forward pass through a computation graph is referred to as forward propagation. The value of J is calculated from the known initial input $\mathbf{w}$, weights $\mathbf{w}$ and bias b. Consider the simple equation for i-th[1] example:

$$z^{(i)} = \mathbf{x}_1^{(i)} * \mathbf{w}_1^{(i)} + \mathbf{x}_2^{(i)} * \mathbf{w}_2^{(i)} + b \tag{2.7}$$

$$a^{(i)} = \hat{y}^{(i)} = \sigma \left(z^{(i)} \right) \tag{2.8}$$

From the value of a ($\hat{y}$) obtained, the value of the loss function is calculated for all the m training examples per iteration followed by the computation of the cumulative cost function from 2.4.

2.1.4 Back Propagation

The backward pass through a computation graph is referred to as back propagation. The relative values of change in Cost function w.r.t.[2] the change in each of the input values and trainable parameters are computed. Considering a simple loss function for i-th example,

$$L \left(\hat{y}^{(i)}, y^{(i)} \right) = - \left(y^{(i)} * \log \hat{y}^{(i)} + \left(1 - y^{(i)} \right) * \log \left(1 - \hat{y}^{(i)} \right) \right) \tag{2.9}$$

If eqns. 2.7, 2.8 and 2.9 [3] are the steps in the forward propagation to attain the loss, the back propagation can be carried by computing the partial derivatives of the same in the opposite direction.

$$a^{(i)} = \hat{y}^{(i)} \tag{2.10}$$

$$da^{(i)} = \frac{\partial L^{(i)} \left(a^{(i)}, y^{(i)} \right)}{\partial a^{(i)}} = \frac{-y^{(i)}}{a^{(i)}} + \frac{\left(1 - y^{(i)} \right)}{1 - a^{(i)}} \tag{2.11}$$

$$dz^{(i)} = \frac{\partial L^{(i)}}{\partial z^{(i)}} = \frac{\partial L^{(i)}}{\partial a^{(i)}} * \frac{\partial a^{(i)}}{\partial z^{(i)}} = a^{(i)} - y^{(i)} \tag{2.12}$$

$$dw_1^{(i)} = \frac{\partial L^{(i)}}{\partial w_1^{(i)}} = \mathbf{x}_1^{(i)} * dz^{(i)} \tag{2.13}$$

[1] the number of the example is always represented by (i) inside parenthesis

[2] w.r.t.: with respect to

[3] Note: the loss function shown in 2.9 tends to optimize to a local minimum and is referred only for simplicity. Better loss functions are available for optimal gradient descent.

$$dw_2^{(i)} = \frac{\partial L^{(i)}}{\partial w_2^{(i)}} = \mathbf{x}_2^{(i)} * dz^{(i)} \tag{2.14}$$

$$db^{(i)} = dz^{(i)} \tag{2.15}$$

The back propagation is followed by updating the w and b values using eqn.2.5 and eqn.2.6 respectively taking the dw and db values. Each such iteration is known as an epoch. This process is repeated till the global minimum of the cost function is attained.

2.1.5 Vectorization

To iterate through larger data sets, explicit for loops are less efficient as they do not allow parallel computing. In deep learning where the amount of data is high, the for loops are avoided as much as possible to make the training faster. The method used to achieve this is called vectorization. The vectorized versions of eqn.2.1 and eqn.2.2 are as follows

$$\mathbf{Z} = \mathbf{W}^T + b \tag{2.16}$$

$$\mathbf{A} = \sigma(\mathbf{Z}) \tag{2.17}$$

where,
$\mathbf{Z} = \left[z^{(1)}, z^{(2)}, ..., z^{(m)}\right]$
$\mathbf{Z}.shape = (\ n_x,\ m)$
$\mathbf{A} = \left[a^{(1)}, a^{(2)}, ..., a^{(m)}\right]$
$\mathbf{A}.shape = (\ 1,\ m)$

2.1.6 Activation Functions

Till this point the activation function that was being used is sigmoid, $\sigma(z)$. But a more generalized representation of the activation function is $g(z)$ where g is a non-linear function. TanH [43] is one such activation, and is better at activation in most cases when compared to sigmoid except for binary classification where sigmoid would be the better choice. Note that with both the TanH [43] and Logistic/ sigmoid activation [43], the neurons with highest or lowest values also known as saturated neurons kill the gate gradients. In theory though ReLU [43] deactivates the neuron for $z < 0$, it has evidently been effective at activating the neurons with important data. An alternative would be to use Leaky ReLU [43] which allows a minor activation for z values at 0 or lower. A few of the activation functions are shown in [43].

The following is the representation that will be followed through the following chapters for the derivative of the non linear activation or the slope of g wrt z. Consider the sigmoid function from eqn.2.3,

$$g\left(z\right) = a = \sigma\left(z\right) \tag{2.18}$$

$$g'(z) = \frac{d\, g\left(z\right)}{dz} \tag{2.19}$$

$$g'(z) = g(z) * (1 - g(z)) \Rightarrow a * (1 - a) \tag{2.20}$$

2.2 Deep L-Layer Neural Network

Neural networks with more than one hidden layers are known as deep neural networks. Here we consider the neural network to have L layers. The additional notations for deep neural networks are,

L : *total number of layers excluding the input layer*
l : *number of the current layer*[4]
$n^{[l]}$: *number of units / neurons in the layer* l

The generalized vectorized equations for forward and back projection from 2.1.3 and 2.1.4 are rewritten as follows,

Vectorized Forward Propagation for Layer l

$$\mathbf{Z}^{[l]} = \mathbf{W}^{[l]} * \mathbf{A}^{[l-1]} + b^{[l]} \tag{2.21}$$

$$\mathbf{A}^{[l]} = g^{[l]}\left(\mathbf{Z}^{[l]}\right) \tag{2.22}$$

$$\text{at } \; l = 0, \; \mathbf{A}^{[0]} = \mathbf{X}$$

Vectorized Back Propagation for Layer l

$$d\mathbf{Z}^{[l]} = d\mathbf{A}^{[l]} * g^{[l]'}\left(\mathbf{Z}^{[l]}\right) \tag{2.23}$$

$$d\mathbf{W}^{[l]} = \frac{1}{m} * d\mathbf{Z}^{[l]} * \mathbf{A}^{[l-1]^T} \tag{2.24}$$

$$db^{[l]} = \frac{1}{m} * \sum_{l=1}^{l=L} d\mathbf{Z}^{[l]} \tag{2.25}$$

$$d\mathbf{A}^{[l-1]} = \mathbf{W}^{[l]^T} * d\mathbf{Z}^{[l]} \tag{2.26}$$

Update Weights and Bias

[4] $n^{[l]}$: The number in the square brackets as a superscript indicates the current layer

$$\mathbf{W}^{[l]} = \mathbf{W}^{[l]} - \alpha * d\mathbf{W}^{[l]} \tag{2.27}$$
$$b^{[l]} = b^{[l]} - \alpha * db^{[l]} \tag{2.28}$$

Matrix Dimensions

$$\mathbf{W}^{[l]}, d\mathbf{W}^{[l]} = \left(n^{[l]}, n^{[l-1]}\right)$$
$$\mathbf{X} = \left(n^{[0]}, m\right)$$
$$\mathrm{b}^{[l]} = \left(n^{[l]}, m\right)$$
$$\mathbf{Z}^{[l]}, \mathbf{A}^{[l]}, d\mathbf{Z}^{[l]}, d\mathbf{A}^{[l]} = \left(n^{[l]}, m\right)$$

2.3 Convolutional Neural Networks

The process of overlapping a kernel in steps over the range of input data to extract low dimensional features is convolution. A kernel or a filter is an array of values and is one of the base components of a convolution operation. Like shown in fig. 2.3, a dot product of the section of the input data and the kernel is the resultant value in the associated block in the convoluted feature or the feature map. Conventionally, the size of the filter, f is odd and almost never even. Some of the classic CNN architechtures are LeNet-5 [31], AlexNet [29] and VGG-16 [47].

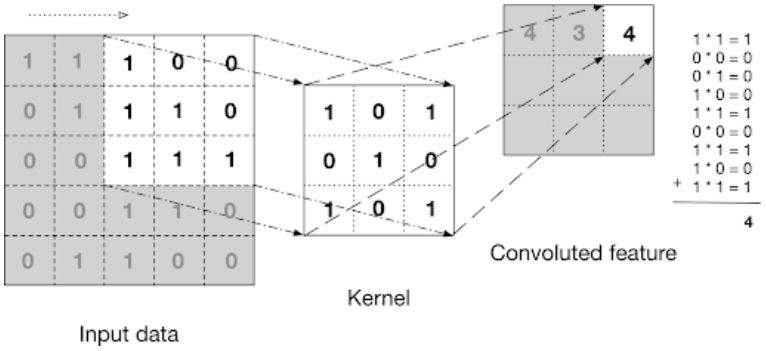

Figure 2.3: Basic convolution operation [52]

The notations to be followed through this section are,

n : *number of pixels in input image*
n_w : *width of the input image*
n_h : *height of the input image*
f : *filter / kernelsize*
p : *padding*
s : *stride*
n_c : *number of channels*

$*$: *convolution operation*

2.3.1 General Convolution

The change in dimension on performing a convolution operation between the input data and the feature map.

$$[n \times n] * [f \times f] \rightarrow [(n - f + 1) \times (n - f + 1)] \tag{2.29}$$

2.3.2 Padded Convolution

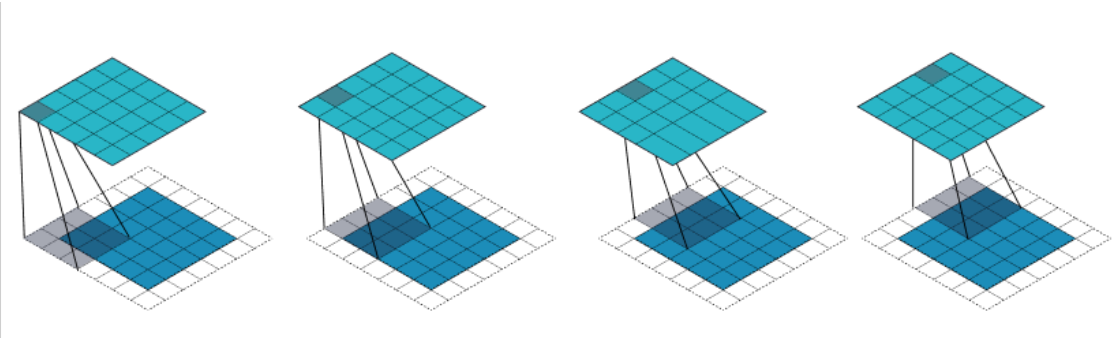

Figure 2.4: Padded convolution with $p = 1$, no stride [6]

Fig. 2.4 shows the steps of a padded convolution with no stride. The change in dimension on performing a convolution operation with a padding p between the input data and the feature map.

$$[(n + (2 * p)) \times (n + (2 * p))] * [f \times f] \rightarrow [(n + (2 * p) - f + 1) \times (n + (2 * p) - f + 1)] \tag{2.30}$$

There are two types of padding methods used, 'valid' and 'same'. In 'valid', the padding is set to 0 and is similar to 2.29. in 'same', the padding is set so that the output size is same as the input size.

$$p = \frac{f - 1}{2} \tag{2.31}$$

2.3.3 Strided Convolution

The change in dimension on performing a convolution operation with a padding p and stride s between the input data and the feature map.

$$[(n + (2 * p)) \times (n + (2 * p))] * [f \times f] \rightarrow \left[\left\lfloor \frac{n - (2 * p) + f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n - (2 * p) + f}{s} + 1 \right\rfloor\right] \tag{2.32}$$

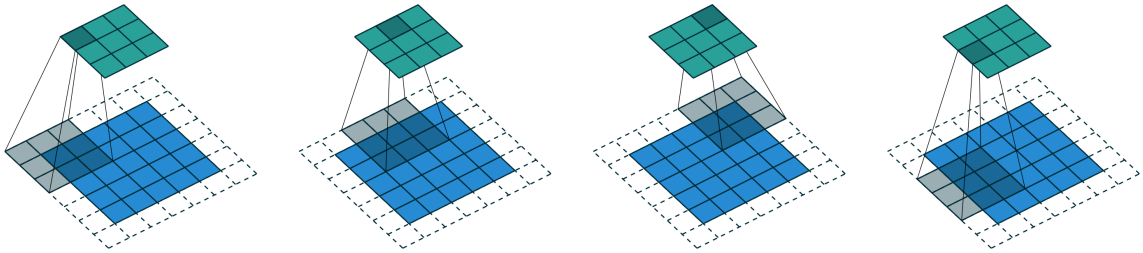

Figure 2.5: Strided convolution with $p = 1,\ s = 2$ [4]

Fig. 2.5 shows the steps of a strided convolution with 1 pixel padding and 2 pixel stride. Note that if the value of $\frac{n-(2*p)+f}{s}$ is not an integer then it is rounded to its closest integer.

2.3.4 Pooling Layers

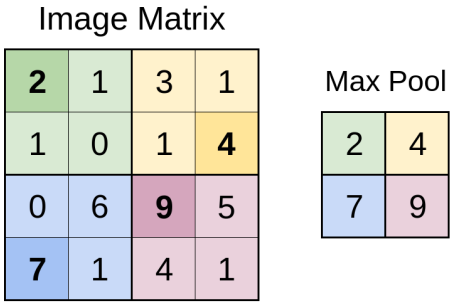

Figure 2.6: A simple pooling [45] operation with $s = 2, f = 2$

If the 4×4 image matrix in fig. 2.6 is the representation of features detected from a previous layer and the size of the filter is 2×2 then the output as shown on the right of fig. 2.6 is by eqn.2.32 of size 2×2. The max pooling [45] operation preserves the strongest feature that was detected. The pooling operation is performed over each channel independently. Padding is commonly set to 0. While there is loss of data, pooling helps in condensing the data while preserving important features. There are no additional training parameters that are there in a pooling layer. The filter size,

stride, padding and the type of pooling (eg. Max Pooling, Average Pooling etc.) are variable and are set before training. These are referred to as hyper parameters.

2.3.5 Convolution over 3D Volumes

In the case of image data, the convolutions are to be carried in 3 dimensions. The width, the height and the number of channels. In 2.7, an image with RGB channels is represented on the left, the size of the input data shown in fig.2.7 is $6 \times 6 \times 3$, the three-dimensional kernel is represented by 3 layers of 2-dimensional filters and the size of each of the 3D kernel shown in fig.2.7 is $3 \times 3 \times 3$. The convolutional operation is done to all three layers in the input data at the same time to compute one value in the output feature map. The RGB dots on the feature map on the right denote a value from the corresponding layer and not the color itself. Note that, for an RGB image, the number of channels in the input data and the number of channels of the filter must be the same. One filter is used to detect one feature from the image like an edge, a corner or a pattern. If multiple features are to be detected, multiple filters need to be used. the size of the filter stays the same in a layer. As shown in fig.2.7, the data convolved using each filter forms one layer or channel of feature map. The number of filters in the layer is equal to the number of channels in the output feature map.

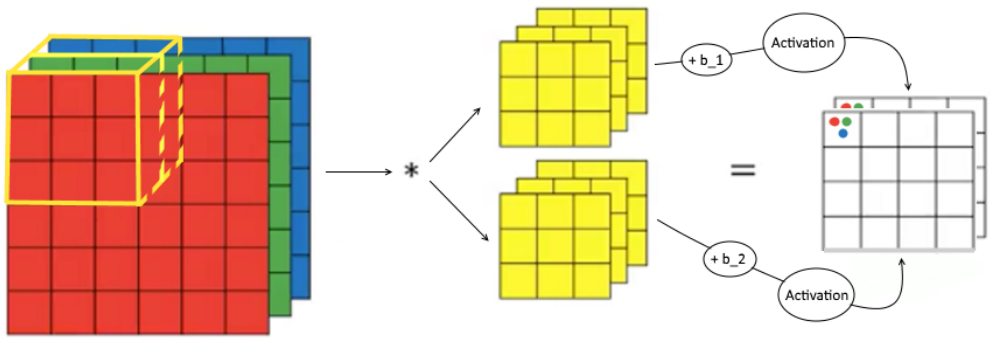

Figure 2.7: Convolution over RGB data using a 3D kernel [1]

2.3.5.1 General 3D Convolution

The general notation and the representation of the dimensions in a convolutional layer with multiple filters is as follows

n_c' : *number of channels in the output feature map*

$$[n \times n \times n_c] * [f \times f \times n_c] \rightarrow [(n - f + 1) \times (n - f + 1) \times n_c{}'] \tag{2.33}$$

The trainable parameters in a convolutional layer are the weights in each filter and one bias per filter. In fig.2.7 the number of parameters would be $3 \times 3 \times 3$ per filter, that is 54 and 2 biases, one per filter which amounts to 56 trainable parameters.

2.3.5.2 Multi-Layer 3D Convolution

Following the same notation earlier where super-scripted $[l]$ represents the number of the current layer, the general notation and dimensions for a multi layered convolutional layer would be as follows

$\mathrm{f}^{[l]} : filter\ size$
$p^{[l]} : padding$
$s^{[l]} : stride$
$n_c^{[l]} : number\ of\ filters\ in\ layer\ l$
$n_c^{[l-1]} : number\ of\ channels\ in\ layer\ l-1$
$\mathbf{w}^{[l]} : weights\ for\ layer\ l$
$a^{[l]} : activation$
$m : batch\ size$
$\mathbf{A}^{[l]} : a^{[l]}\ vectorized\ over\ m$

$n_h^{[l]} = \left\lfloor \frac{n_h^{l-1} + \left(2 * p^{[l]}\right) - f^{[l]}}{s^{[l]}} + 1 \right\rfloor$
$n_w^{[l]} = \left\lfloor \frac{n_w^{l-1} + \left(2 * p^{[l]}\right) - f^{[l]}}{s^{[l]}} + 1 \right\rfloor$
$input, a^{[l-1]} = n_h^{[l-1]} \times n_w^{[l-1]} \times n_c^{[l-1]}$
$size\ of\ each\ filter = f^{[l]} \times f^{[l]} \times n_c^{[l-1]}$
$\mathbf{w}^{[l]} = f^{[l]} \times f^{[l]} \times n_c^{[l-1]} \times n_c^{[l]}$
$b^{[l]} = 1 \times 1 \times 1 \times n_c^{[l]}$
$\mathbf{A}^{[l]} = m \times n_h^{[l]} \times n_w^{[l]} \times n_c^{[l]}$
$output, a^{[l]} = n_h^{[l-1]} \times n_w^{[l-1]} \times n_c^{[l-1]}$

2.3.6 Advantages of a CNN

The following are some of the advantages of a CNN

- One of the main advantages of using a Convolutional Neural Network in comparison to traditional fully connected network is parameter sharing. A feature detector or a filter that is useful in one part of the image, can also detect similar features in the other parts of the image.

- The sparcity of connections between the input and output feature map. Each value in output feature is connected or depends on a small set of values from the input matrix depending on the size of the filter used.
- CNNs are good at tracing translation and or variance in an image. The object of interest positioned at the left half of the pixels or right would be treated the same way as a collective features is what the convolutional layer tries to learn. For eg., A cat at the left of an image or the right of an image would still be recognised as a cat by an image classifier.

2.4 Residual Networks

The networks were growing deeper through time since the deep learning took off but the deeper the network was, the tougher it was to achieve network identity. The Neural Network structures all being black box architectures in which nothing that is computed during training is available but the output itself. There was a subsequent loss of data. In 2016 He et al. in [17] introduced the residual blocks, example shown in fig.2.8 where in they trained a 152 layer neural network with skip connections.

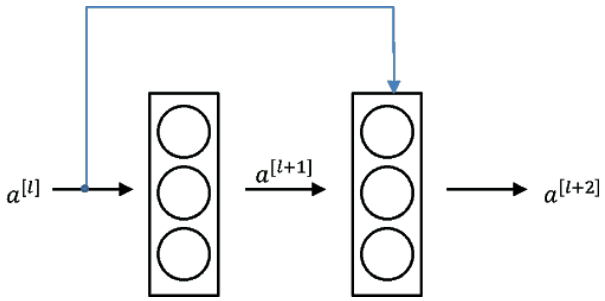

Figure 2.8: A residual block with a skip connection from layer l to layer $l+2$ [17]

These skip connections carried the activation information from a few layers prior to the current layer to the current layer. The padding across the network was maintained as 'same'(see section 2.3.4) which would facilitate easier matrix addition owing to retained dimensions.

2.5 Introduction to Generative Adversarial Networks

A neural network is capable of learning, given the right answer as label during training. The network on successful training becomes capable of producing the answer, given input data in the same domain. Creating new data was not something the networks were capable of doing. In 2014, Ian Goodfellow et al. proposed a framework to train a network to generate new data (in the same domain as the training set) called Generative Adversarial Networks [16]. The framework as seen

in fig.2.9 consists of two separate networks namely the generator and discriminator acting as adversaries during the training process.

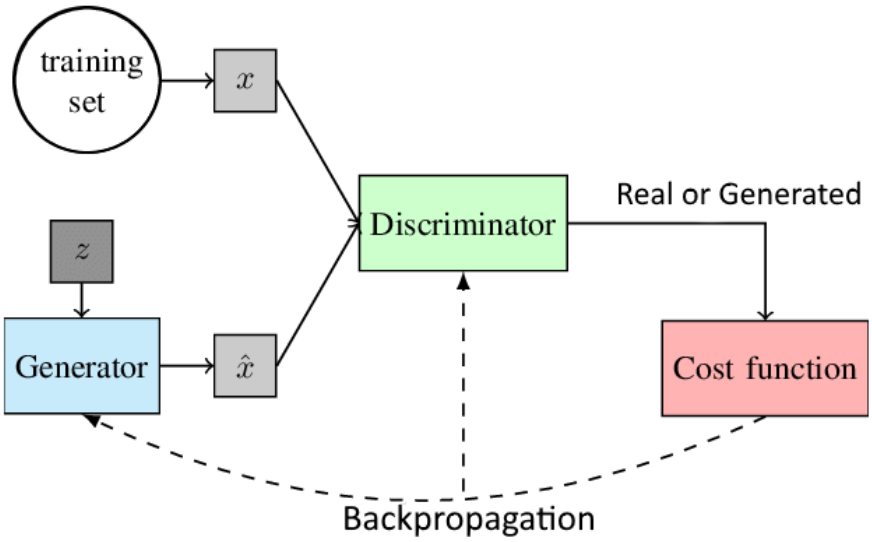

Figure 2.9: Generative Adversarial Networks framework [16]

Given a sample from real data x and a sample from generated data $\hat{x}$, the task of the discriminator is to check if the generated data is from the same domain as the real data. In other words, the discriminator checks if the generated data is real or fake. Given random noise z generated in a latent space, the task of the generator is to map z to $\hat{x}$. Discriminator, typically a classifier [16], outputs the error between the real and generated data. The cost function associated to that error is minimized using gradient descent during the back propagation step. As the training progresses, the generator learns to generate data closer to the domain of the real data and the discriminator learns to better distinguish between real and generated data. Theoretically, the ideal that can be achieved would be for the generator to learn to generate data that can not be distinguished by the discriminator from the real data. But, in practice, the discriminator usually trumps the generator and the training is stopped at a point when the data generated resembles the real data. Once the training is completed, the discriminator is discarded and the generator is used to generate data that resembles the real data, given random noise in the latent space. Though GAN improved the perceptual quality, the training itself is fairly unstable owing to issues including but not exclusive to vanishing gradient and mode collapse.

2.6 Transfer Learning

Transfer learning is the process with which a trained network model can be trained further to better fit the domain of the task. In general the process involves downloading the architecture and its pre-trained weights from an open source implemen-

tation. Because some of the available models have been trained on large data sets, they would work better than a model trained from the scratch on a small data set. It is almost always recommended to check if the task can be optimally handled by pre-existing trained networks by transfer learning to save time and computational expense. Weiss et al. in [58] go through a survey of transfer learning approaches, brief description of which is given hereafter.

- Based on how different the task at hand is from what the model was previously trained for [58], the number of layers that need to be re-implemented is decided. For eg., if the model was trained to classify 47 objects and the user only needs 4 classes, then just the last softmax layer needs to be re-implemented and if the model was trained to classify objects and the user needs to localize the object in the image, then a few layers from the top including the softmax layer need to be re-implemented. The layers that can use the weights from the pre-trained models are set to non-trainable (frozen), which allows for the network to only train the layers following.

- Depending on how much data is available, the number of layers that can be set to non-trainable [58] can be chosen. For lower amounts of data, the model might learn little to nothing if too many layers are trainable. The number of layers that can learn successfully is directly proportional to the amount of data fed to the network during training.

- Depending on how much computational power one has at their disposal, the amount of data needs to be set, so as to manage the time the model would need to train [58]. Theoretically if there is a high amount of data, ample time and a lot of computational power available, then the model can be trained from the scratch. If the computational power available is pretty high and the data set is pretty large, then the weights of the network can also be initialized with the pre-trained weights instead of randomly initializing them (this is the standard process) and weights of all the layers can be set as trainable parameters.

2.7 Resampling / Rescaling

An image is represented as a grid of fixed size block elements. Each such element is called a pixel. An image is represented as an array of pixel intensity values. Image re-sampling is the process of altering the image resolution with the help of available sampling methods which are categorized by the sampling factor value being greater or lesser than 1. 2.10 shows a simple representation of up- and downsampling of a grid of pixels.

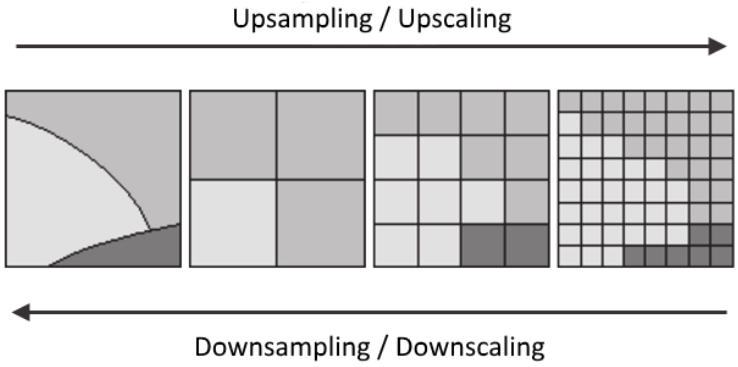

Figure 2.10: A simple representation of up- and down-sampling [37]

2.7.1 Image Resolution

There are three broad representations of image resolutions.

- **Pixel Count Resolution**: It is a simple representation of the amount of pixels a digital image. The number of pixels in the width of the image times the number of pixels in the height of the image, commonly referred to as 'image resolution'. These values remain constant unless the image is re-sampled or cropped.
- **Spatial Resolution**: This is a representation of pixels per inch on a display. An image of $n_w \times n_h$ can be displayed on a screen bigger than the image (A high definition monitor or a TV) or a screen smaller than the image (a mobile or a digital camera screen). The term "dpi" (dots per squared inch) is also highly used to represent the pixels per inch. Spatial resolution is not a fixed property till the image takes a permanent physical form, for example, in Printing or on screens.
- **Temporal Resolution**: This is the representation of an image in terms of a time. For example, if a scene is captured everyday then the temporal resolution of the scene would be one day. Note that this topic is out of scope for this thesis and for information regarding the temporal resolution, refer to the section 6.07.2.1.3 in [33]

2.7.2 Upsampling

If the sampling factor is greater than 1, then the process is called upsampling or super-sampling. The intuition of upsampling is to insert additional blank pixel rows and columns next to the existing ones and filling these pixels with values computed using the existing values surrounding them. For image data, upsampling is also commonly referred to as upscaling. This computation can be done using various interpolation approaches and using neural networks which will be discussed in chapter 3. Since the added pixel values do not originally exist and are computed, there

can be a large number of possible solutions and the output of the said approaches is one such possible solution. The problem to solve here is to produce an upsampled approximate of the original image while preserving or improving the sharpness and details.

2.7.3 Downsampling

If the sampling factor is lesser than 1, then the process is called downsampling or sub-sampling. The intuition of downsampling is to reduce the number of pixels while trying to preserve as much feature information as possible. For image data, downsampling is commonly referred to as downscaling [27]. The non-adaptive and adaptive interpolation approaches for downsampling will be discussed further in chapter 3. Since the output is the data after strategically eliminating data that the approach deems less important. There is a definite loss of data and the problem to solve here is to produce a downsampled approximate of the original image while preserving how the image is perceived by a human eye or the perceptual quality of the image by reducing the loss of important features as much as possible. As a result, the output image has lesser storage size while retaining most of its defining features.

CHAPTER 3

Related Work / State of the Art

In this chapter, a brief description of the available upsampling and down-sampling approaches is given. The standard approaches common to both up- and down-sampling are introduced in section 3.1 followed by more advanced ML (Machine Learning) based upsampling approaches in section 3.2 and the adaptive image down-sampling approaches in section 3.4.

3.1 Interpolation Approaches

A comprehensive study of the interpolation methods and the kernels used to achieve them are given in [14]. This section contains a brief introduction to the standard interpolation approaches from [14].

Figure 3.1: Example upsampling using interpolation approaches, scaling factor = 4, source image from [2]

Figure 3.2: Example downsampling using interpolation approaches, scaling factor = 2, source image from [15]
[1]

Interpolation is the problem of inferring a continuous function from a discrete set of values and there are many techniques that surfaced over the years. Meijering et al. in [38] gives a brief chronological history of interpolation techniques. In fig.3.1 and fig.3.2 is up-sampled by a factor of 4 and down-sampled by a factor of 2 respectively. Following are the standard interpolation methods used to date.

3.1.1 Nearest Neighbour

This is a non-adaptive interpolation method where the blank pixels added for up-sampling are filled with the neighbouring pixel values. Since the each pixel value in this interpolation approach is independent of the pixels around it, the output up-sampled images are extremely pixelated. This kind of interpolation is good to preserve sharp edges but in a general application, the edges are sharper than they need to be as it fails to preserve the softer curves. In fig.3.3 the image is to be up-sampled by a factor of 2 and the values of each of the pixels is replicated in its neighbouring pixel positions.

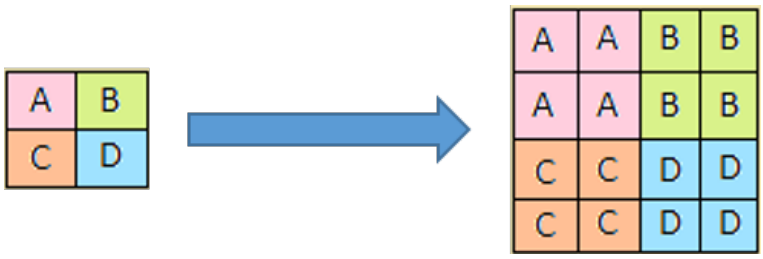

Figure 3.3: Nearest neighbour interpolation with a scaling factor of 2 [3]

[1] Note that the spatial resolution of the images has been altered to fit the size of the paper without using any interpolation

3.1.2 Bi-linear

In a linear interpolation, the weighted sum of the source pixel values is used to fill a new pixel. The weight in linear interpolation is dependent on the respective distance between the source and the new pixel. The closer the source is to the new value, the higher the weight associated with that value. A 1D linear interpolation is the weighted sum of 2 pixel values. A 2D linear interpolation by extension is the weighted sum of 4 pixel values.

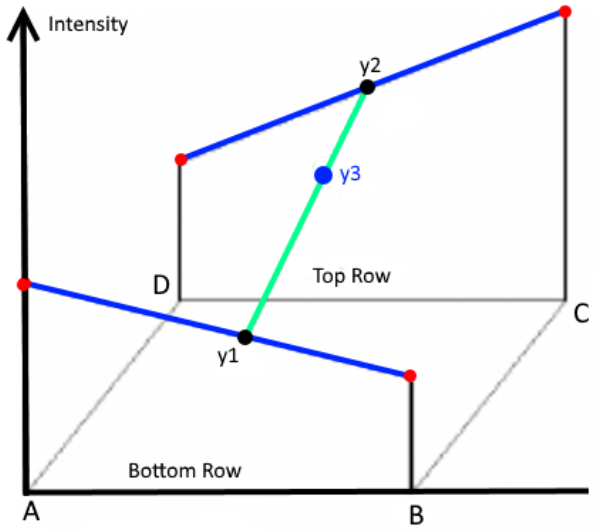

Figure 3.4: Bilinear interpolation represented in a 2D pixel plane ABCD

In fig.3.4 the red dots are the available intensity values of pixels A, B, C and D. The blue lines between A, B and C, D represent the linear interpolation in x-direction to estimate the intermediate pixel values y_1 and y_2. The pixel value y_3 between these points can be estimated along the green line y_1y_2. If y_1 and y_2 are the pixel values, length of $y_1y_2 = 1$ and y_1y_3 and y_3y_2 are the distances from y_1 and y_2 to y_3 respectively, then

$$y_3 = y_1\,(1 - y_1y_3) + y_2\,(1 - y_3y_2) \tag{3.1}$$

where $(1 - y_1y_3)$ and $(1 - y_3y_2)$ are the weights associated with y_1 and y_3 respectively. In a general representation where x is the pixel distance between the source and the resultant pixel, for bilinear interpolation, the weight is given by $1 - |x|$.

Downsampling using bilinear interpolation uses the same method. For intuition, the resulting new pixels interpolated from 4 source pixels are concatenated together and the source pixels are discarded to create the new image per step, thereby reducing 1 row and 1 column of pixels.

3.1.3 Bi-cubic

While a 1D linear interpolation is dependent on 2 pixel values and determining the values in between those values along the line. A 1D cubic interpolation is dependent on 4 pixel values. The intensity gradients between source pixel values are considered and hence, to draw a gradient at an intensity value, the two source pixels following the current would be needed. Considering p_0, p_1, p_2, p_3 to be the source pixel intensity values, the continuous function drawn between them, a cubic spline can be represented by y in eqn.3.2 where a_0, a_1, a_2, a_3 are the weights associated to the pixel values.

$$y = a_0.p_0 + a_1.p_1 + a_2.p_2 + a_3.p_3 \tag{3.2}$$

A general representation of eqn.3.2 would be

$$f(x) = \sum_{i=0}^{i=3} a_i.x_i \tag{3.3}$$

For Bicubic interpolation (2D) [23], a window of 16 pixel values is used. In fig.3.5, the intensity values are represented by the black dots, the 1D and 2D cubic interpolations are represented by bold black and red splines respectively.

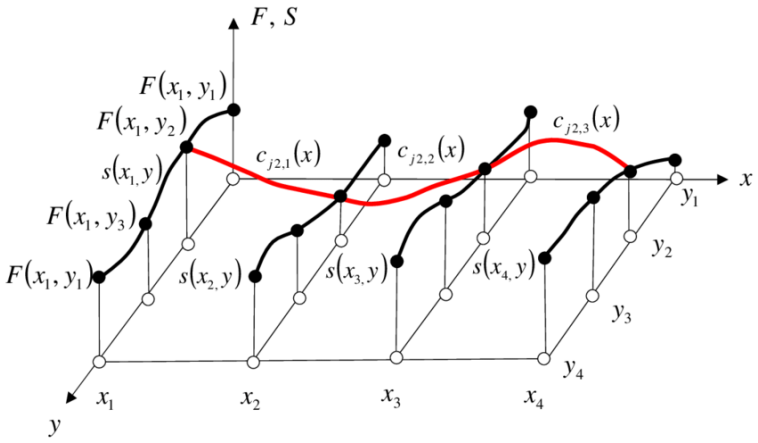

Figure 3.5: Bicubic interpolation represented in a 2D pixel xy-plane [35]

By extension eqn.3.3 for 2D space, the general representation for bicubic can be written as

$$f(x, y) = \sum_{i=0,j=0}^{i=3,j=3} a_{i,j}.x_i.y_j \tag{3.4}$$

3.1.4 Lanczos

Lanczos interpolation is done using a windowed sinc function. A normalized sinc function is given in [13] by

$$sinc\,(x) = \frac{sin\,(\pi x)}{\pi x},\ x \neq 0 \tag{3.5}$$

If the considered filter size parameter is a positive integer, a then the size of the the Lanczos reconstruction kernel or the Lanczos kernel is $2a-1$ is given from [13] by

$$L\,(x) = \begin{cases} 1 & if\ x = 0 \\ \frac{a.sin(\pi.x).sin\left(\frac{\pi.x}{a}\right)}{\pi^2.x^2} & if\ -a \leq x < a\ and\ x \neq 0 \\ 0 & otherwise \end{cases}$$

This kernel determines the weights assigned to each of the source pixel intensity values. In the python package "open cv" (cv2) the Lanczos kernel available is with $a = 4$ which implies that the kernel size is 8×8.

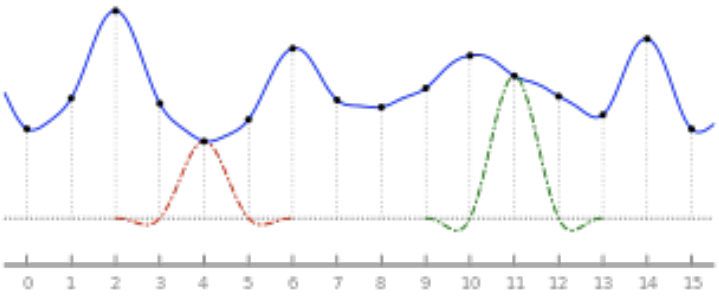

Figure 3.6: Lanczos interpolation represented in 1D with two individual Lanczos kernels with $a = 2$ shown at $x = 4, 11$ [5, 13]

Fig.3.6 shows the Lanczos interpolation shown by the blue spline formed from the discrete set of pixel values represented by the black points. For 2D, at $a = 2$, 4 splines as in fig.3.6 would be needed across the second dimension. Lanczos kernel in 2D [13] can by extension of 3.6 be given by

$$L\,(x, y) = L\,(x)\,L\,(y) \tag{3.6}$$

3.2 Deep Learning based Upsampling Approaches

[2] As back-propagation algorithm was introduced to the computer vision field in [30] and showed promise, the super resolution researchers took interest in its merits in this field. The evolution of the super resolution algorithms over the next few decades is noteworthy and an integral part of this study. The improvement shown in the computational power of the hardware developed during this time was exponential and the training of the models for super resolution was hence made possible. In the next few sections, a comprehensive overview of the said literature is given. An extensive analysis on different models could be found in [15]. As this thesis involves an extensive study of GANs, the chapters following would show experimental evaluation of selected models. The sec 3.3 will describe the architecture, loss functions and assessment criterion used in respective papers. Figures 3.9 and 3.10 show the architectures of the models that will be discussed in the sections 3.2.1 and 3.2.2.

3.2.1 Convolutional Network Approaches

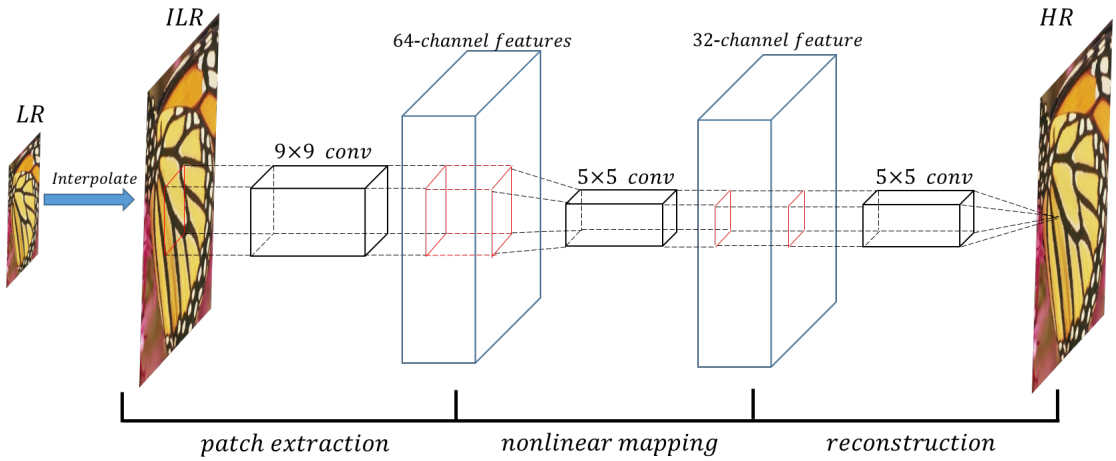

Figure 3.7: Deep super resolution architectures [11]

In 2014, the first image super resolution CNN, the SRCNN [11] for the task of SISR (Single Image Super Resolution) was proposed which was a three layered network as shown in fig.3.7 consisting of three convolutional layers with 64 filters of 9×9, 32 filters of 5×5 and 32 filters of 5×5 kernel sizes respectively. It was optimized with MSE (mean square error) as its loss function. Though relatively simple in its architecture, with its non-linear mapping and reconstruction proved superior to the methods existing for SISR up untill then. The input given to the SRCNN model was a bicubically interpolated HR image and not the LR counterpart. FSRCNN

[2]Note: This section gives a brief overview of the relevant literature and for further details on the models the readers may refer to the respective citations.

[12] implemented a deconvolutional layer [62] for adaptive upsampling and could reconstruct HR images from an LR input. The deconvolutional layer required less computation as this computation was done at the end of the network to reconstruct the HR image but the layer itself used the nearest neighbour interpolation convolved with a filter to achieve the HR feature map which had effect on the final performance as the upsampled features are repeated in all directions as described in the section 3.1.1. Shi et al. proposed a method to deal with the problem caused by using a deconvolutional layer with efficient sub-pixel convolution neural network (ESPCN). In ESPCN [46], rather than explicitly upsampling the feature maps like in nearest neighbour, the layer employs the expansion of channels to store the extra data and rearranges them to form the HR feature map using a specific mapping method as seen in fig.3.8

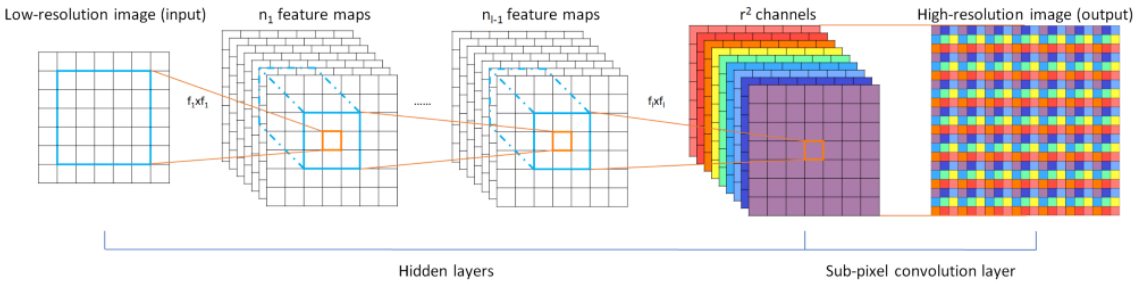

Figure 3.8: ESPCN - Sub-pixel convolution layer [46]

While ESPCN tries to solve the fact that SRCNN takes in an interpolated approximation as an input, VDSR [25] improved the performance using a deeper VGG-net [47]. VGG sets all convolutional kernels to the size 3×3. Kim et al. while following the same input as SRCNN, in VDSR use a 20-layer VGG network as shown in fig. 3.9(a) and train it using learning rate decay[3] for faster convergence. Further Kim et. al proposed DRCN [26] where in the inference or the non-linear mapping part which was recursive for 19 times in VDSR was reduced to 16 as shown in fig.3.10(b) to lower the trainable parameters. Also, a multi-supervised strategy is applied which creates shorter paths for the gradient to flow smoothly.

3.2.2 Residual Network Approaches

VGG-net is a plain architecture with no skip connections and thus to train a deeper version of similar network would be harder. As such the residual networks which employed skip connections to efficiently reduce the loss of data, most represented in ResNet [17] were used by the authors of [32][4] to propose SRResNet. Extensive anal-

[3]learning rate decay: Using a higher learning rate initially and using gradient clipping, the reduction of the learning rate at specific epochs as a function of the epoch to accelerate the convergence initially and prevent gradient explosion at the end of the training.

[4]Further discussion on GANs would be in 3

ysis of why the residual networks function are given in [17,18]. SRResNet employs 16 residual blocks with each block using batch normalization [21] as seen in fig.3.9(c) to stabilize the training. Tai et. al proposed DRRN [49] which rearranged the topology of the residual blocks to form recursive blocks as shown in fig.3.10(d) and reduced the parameters further by reusing the parameters in each block recursively.

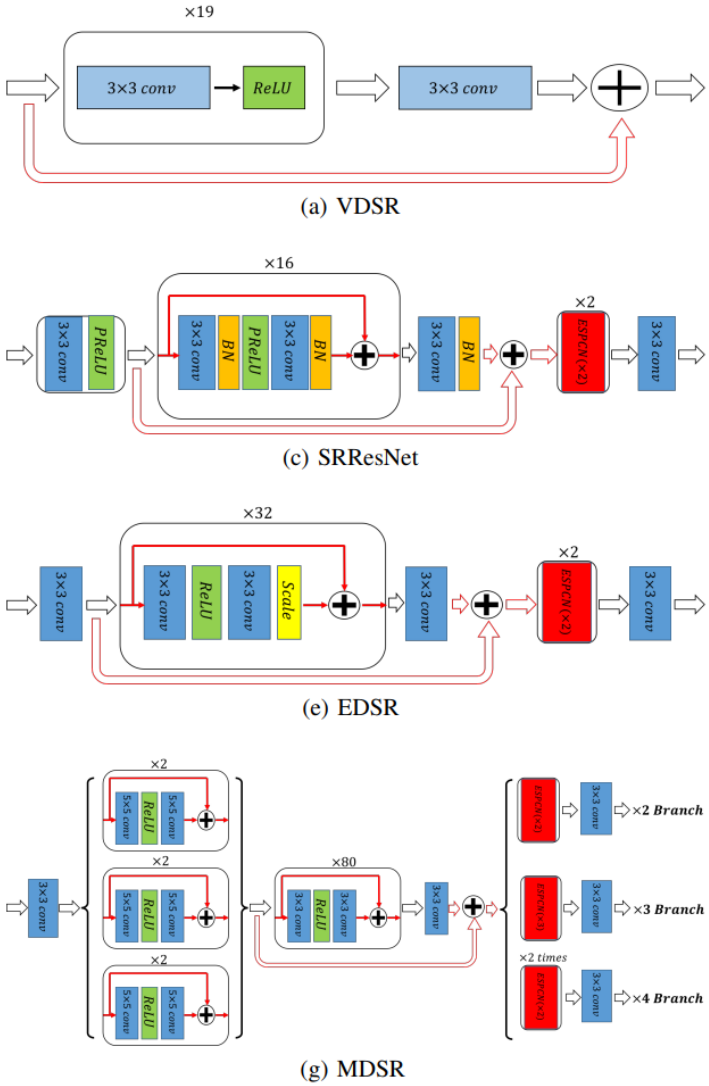

Figure 3.9: Deep super resolution architectures [61], part 1

EDSR [34] proposed by Lee et al. showed a very steep improvement over the models proposed before then. The change made most notably to SRResNet is the removal of batch normalization as seen in fig.3.9(e) from the residual blocks as that was part of the networks developed for classification problem and since it caused for the evident unpleasant artifacts in upsampled images and reduces the generalisation ability [34, 53]. The residual scaling method proposed in [48] is further used to

reduce the difficulties that arise during the training of the employed wider ResNet in EDSR.

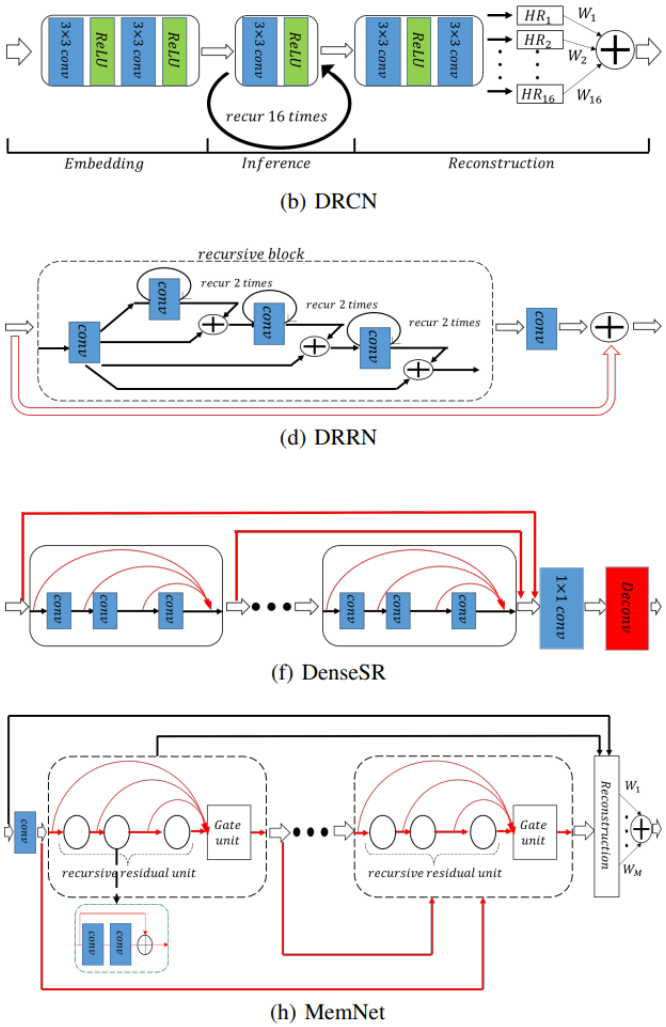

Figure 3.10: Deep super resolution architectures [61], part 2

Inspired by the fact that SISR with different sampling factors are strongly related, the authors of EDSR, when training the higher sampling factors (for example: $\times 3, \times 4$) initialized the weights from a pre-trained $\times 2$ network. This accelerated the training process by a huge factor while improving the final performance of the model. This invoked the implication that the models trained for different sampling factors shared many intermediate transformations. This idea was further explored by the authors by building a multi-scale model architecture. The resultant model MDSR is also proposed in [34]. The non-linear mapping section of the architecture is shared among different sampling factors as shown in fig.3.9(g). Only the feature extraction section and the sub-pixel convolution section at the end of the architecture were different. Each update was performed by randomly choosing mini-batches for $\times 2$, $\times 3$ and $\times 4$ and updating the corresponding parts in the sub-pixel convolution

section of the model. In [61], the authors pointed out that, While ResNet focuses on reusing the features to save computation, DenseNet [20] enables new feature exploration by skip connections connecting each layer with each of its previous layers and before each block of layers the information is cached and passed on to the final reconstruction part of the network. Inspired by DenseNet, SRDenseNet [51] was proposed by Tong et al., the architecture of which can be seen in fig.3.10(f). MemNet [50] proposed by Tai et al. uses recursive units instead of convolutional layers inside each residual block and dense connections are added among different blocks. The architecture can be visualized in fig.3.10(h). The authors of [50] explained that the local connections in the same block resemble short-term memory and the connections with previous blocks resemble long-term memory. In [63], Zhang et al. introduce a residual in residual (RIR) structure with channel attention which allowed learning of high frequency information making it capable of producing higher quality SR images as quantitatively shown in the paper.

The first line of research which involves most of the aforementioned models are trained while minimising pixel-wise loss functions such as MSE and works on maximising PSNR values which fundamentally disagree with the subjective evaluation of human observers [32]. There is another line of research focused on attaining higher perceptual quality of an image and this tends towards GAN based SISR which will be discussed in the next section.

3.3 GAN Based Super Resolution Approaches

GAN based approaches have evidently generated images with better perceptual quality. This section consists of a study of four GAN models, the SRGAN [32], ESRGAN [53], SRFeat [42] and ERCA [19]. The readers may refer to the respective papers for a detailed description and evaluations done by respective authors.

3.3.1 SRGAN

Christian et al. in 2017 proposed a GAN based Super Resolution network architecture, the SRGAN [32] which beat the existing quality of images at the point by a large margin thereby setting a new state-of-the-art benchmark. The authors proposed SRResNet and SRGAN network architectures and a new perceptual loss function. SRResNet with the architecture as shown in fig.3.11 is the generator part of the network. The discriminator consists of 8 convolutional layers with same size kernels (3×3) and incremental channels as seen in fig.3.11.

The perceptual loss function proposed by the authors is a weighted sum of a content loss (l_X^{SR}) and an adversarial loss (generative loss) component. The over all loss l^{SR} was given by

$$l^{SR} = l_X^{SR} + 10^{-3} l_{Gen}^{SR} \tag{3.7}$$

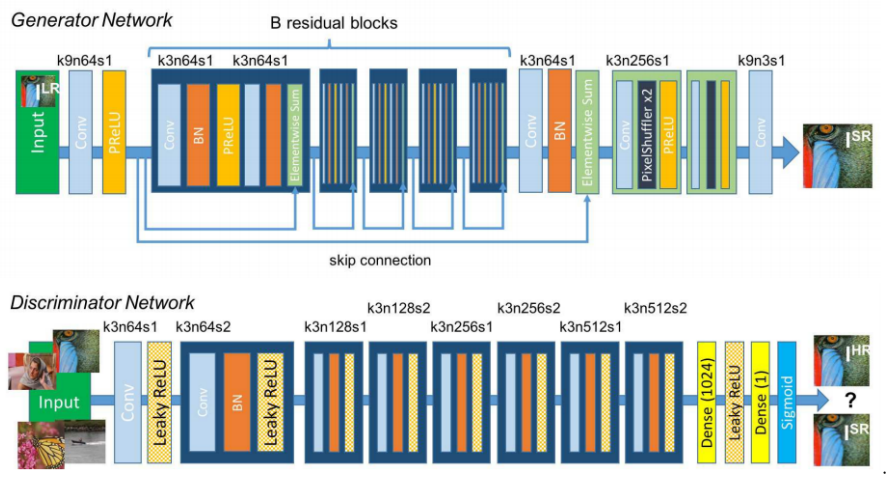

Figure 3.11: [32] SRGAN - Generator and Discriminator architectures [k9n64s1 for example stands for kernel size = 9, number of channels = 64 and stride = 1]

where

$$l_{Gen}^{SR} = \sum_{n=1}^{N} -log\left[D_{\Theta_D}\left(G_{\Theta_G}\left(I^{LR}\right)\right)\right] \tag{3.8}$$

X in content loss was a choice between MSE or $VGG_{i,j}$. The content loss defined in the paper was a VGG loss based on a pre-trained 19-layer VGG network. The i, j represent the choice of feature map obtained by the j-th convolution after activation before the i-th max-pooling layer. The observation drawn by the authors [32] in this aspect was that a better texture detail was attained on choosing a higher level feature map ($VGG_{5,4}$ when compared to $VGG_{2,2}$). $D_{\Theta_D}\left(G_{\Theta_G}\left(I^{LR}\right)\right)$ is the probability that the reconstructed image $G_{\Theta_G}\left(I^{LR}\right)$, where I^{LR} is the LR image, is a HR image.

3.3.2 ESRGAN

Wang et al. in 2018 proposed an enhanced SRGAN termed ESRGAN improving on the SRGAN architecture. The changes made are removal of BN[5] from each residual blocks as in [34] which showed reduction in artifacts and computational expense. Though the higher architecture of the generator derived from SRResNet, the residual blocks are replaced by RRDBs[6] which is a multi level residual network as seen in fig.3.12 and using residual scaling [34,48][7]. In addition a smaller initialization is used to better train the deeper residual network [53].

The standard discriminator used in SRGAN is replaced by a Relativistic Discriminator [22, 53] which allows for the distinguishing factor to be a more comparative 'more realistic than fake data or vice versa' when compared to a direct 'real or fake'.

[5]BN: Batch Normalization
[6]RRDB: Residual in Residual Dense Block
[7]Residual Scaling: multiplying a constant between 0 and 1 to the residuals

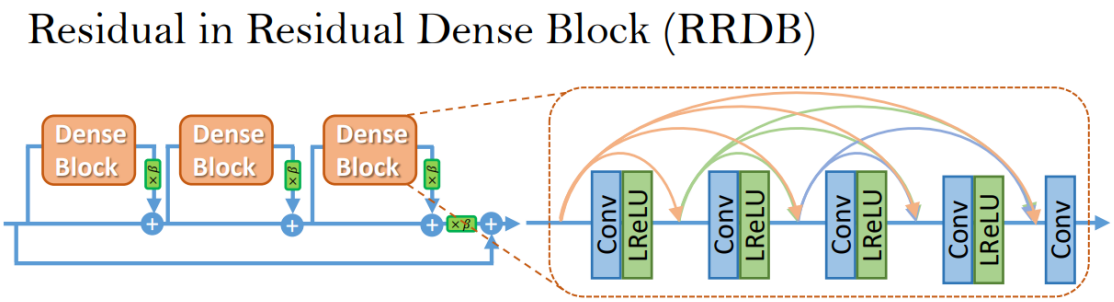

Figure 3.12: [53] RRDB architecture

This takes into the account the gradients of the generated and real data into the loss function as opposed to only generated data in SRGAN [53]. This change of discriminator allows for the network to learn generating better texture detail and sharper edges. Further, a change in perceptual loss introduced in [32] is proposed by the authors. The perceptual loss measured in [32] was measured from the feature map extracted after the activation. The authors explain that, in deeper networks the activation features are sparse and thus provide weak supervision resulting in lower performance. And that the features after activation also caused inconsistent brightness during the reconstruction phase. Also, the authors empirically found that training a deeper network benefited from the greater receptive field offered by a larger patch size, 128×128 HR sub-images in [53] as opposed to 96×96 HR sub-images during training in [32], as it helped capture more semantic information. This benefit though came at the cost of computational expense and longer training durations.

3.3.3 SRFeat

Like ESRGAN [53], the higher architecture of SRResNet was retained by the authors of SRFeat [42] in their generator model with the 16 residual blocks for non linear mapping and 2 sub-pixel convolution [46] blocks.

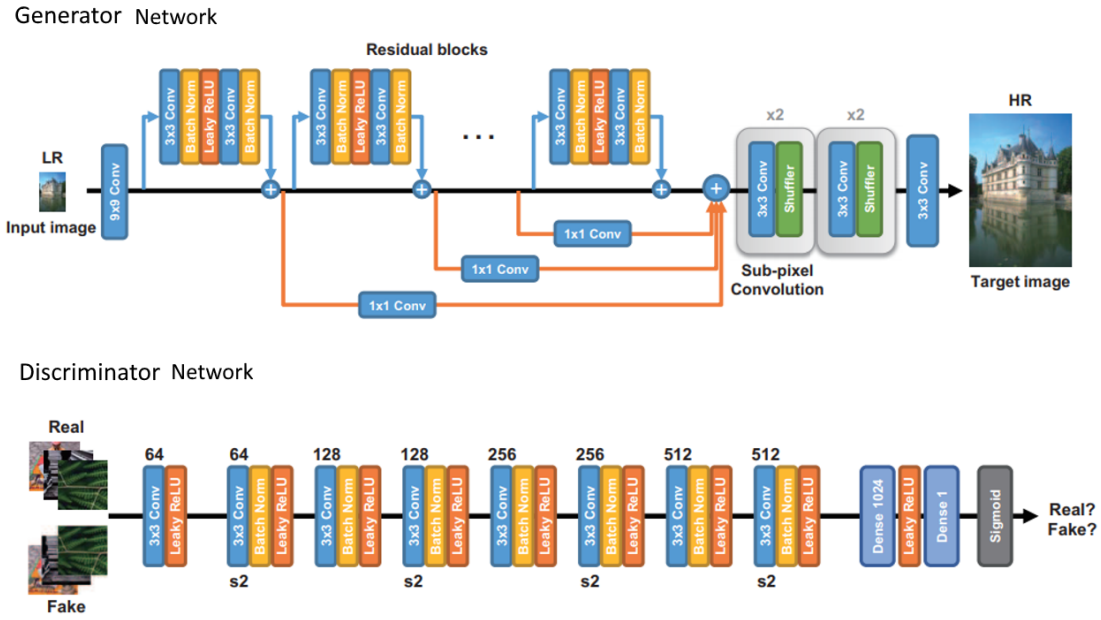

Figure 3.13: [42] SRFeat architecture with the short and long range skip connections represented in the Generator network

The changes made to the generator in [32] are the addition of a 9×9 convolutional layer at the beginning of the network for extracting a feature map from a wider initial receptive field, the addition of the long skip connections with 1×1 convolutions to retain information from the low level features extracted and as a means for the network and short skip connections among the residual blockss for identity mapping to learn the residual values better [42]. This also provided another path for the gradient to flow during the back-propagation. Unlike [53], the network retains the BN layers. The major difference though lies in the adversarial part of the training where two discriminators are employed namely the image discriminator to measure the losses in pixel values and a feature discriminator to measure the losses in the feature maps of the images. The architectures of the generator and the discriminator can be seen in fig.3.13. Unlike the network architectures in [32, 53], the number of channels maintained through the nonlinear mapping part of the network is constant. Two variants are experimented with 68 channels and 128 channels respectively [42]. Quantitative evaluation of the same can be found in [42]. The generator network proposed by Park et al. with 128 channels set a new state-of-the-art benchmark for PSNR and SSIM values computed on the Y-channel.

3.3.4 ERCA

In 2019, Hoang et al. proposed the generator network architecture ERCA [19] Like, the authors of [53], the batch normalization was removed from all the residual blocks of the generator network architecture of [32] as the increase in performance and reduction in computational expense was proven in papers like [53]. Further, the

residual channel attention proposed by Zhang et al. in [63] was employed in each of the residual blocks to adaptively rescale each channel-wise feature by modelling the inter-dependencies across feature channels and follwing the authors of [42] a weighted long skip connection for facilitating ease of back-propagation during training is added. The network architecture can be seen in fig.3.14 taken from the paper.

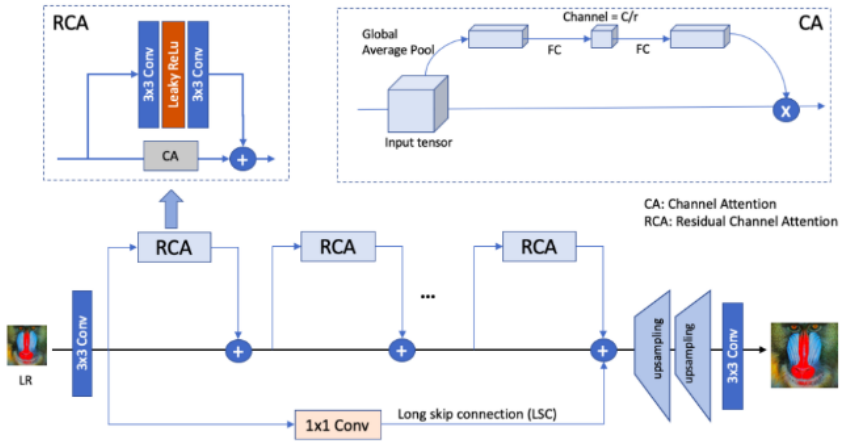

Figure 3.14: ERCA [19] generator network architecture with residual channel attention

The discriminator architecture is similar to [42] as seen in fig.3.13. The architecture stays the same for both image discriminator and the feature discriminator but with each having their own input. The usage of the two discriminators is inspired by [42].

3.4 Advanced Downsampling Approaches

Down-sampling is one of the most used operation in image processing be it to reduce the size of an image to view it on a screen or to reduce the size of an image to reduce the computational expense during training. This is constantly performed using non-adaptive interpolation approaches mentioned in sec.3.1. This section studies some of the more advanced, adaptive approaches to downsampling images.

3.4.1 Content-Adaptive Image Downscaling

While the interpolation methods mentioned in sec.3.1 have achieved fairly good results, there is no correlation between the output pixels. Two pixels generated have no inter-dependency which would lead to discontinued or overly smoothed images, the latter in most cases. In 2013, Kopf et al. proposed a novel method that had the ability to interpolate source pixels while adapting to the features of the surrounding

pixels. It follows the Expectation-Maximization algorithm [10] proposed by Dempster et al. in 1977 which is used to solve a maximum likelihood problem. The interpolation approaches in sec.3.1 have uniform kernel sizes which are distributed spatially in a regular grid [27]. The authors solved this reconstruction problem of the input image from a smaller set of kernel functions localized spatially. Each pixel is considered a samplr randomly drawn from one such local kernels with uniform probability distribution [27].

Given an input image X, the proposed algorithm searches for the set of parameters $\theta = \{\mu_k, \Sigma_k, \nu_k, \sigma_k\}$ specific to kernel k with the maximum likelihood. The parameters μ_k, Σ_k are the mean and covariance matrix of the spatial gaussian and ν_k, σ_k are the mean and variance of the color space gaussian [27]. Since the EM optimization algorithm processes each kernel independently and this does not allow for interaction between the said kernels. The authors introduce a three step method consisting of expectation, maximization and correction steps following the initialization of kernels. While, the expectation and maximization steps compute probabilities of pixels with respect to the kernels and using these probabilities in a weighted maximum-likelihood fit to estimate θ, the correction step introduces three types of constraints to obtain a better representation of the down-sampled image.

- **Spatial constraint**: to constraint any drastic drift in the kernel's location in reference to the reconstructed downsampled image. This is achieved by limiting the extent the spatial mean can move by constraining μ_k in a box centered around the center of the output pixel. Further improving the smoothness by shifting μ_k halfway between its estimated location and the mean of its four neighbours [27].

- **Locality constraint**: to prevent the kernels to become too large or to vanish as the output pixels are all of the same size. This is achieved by clamping the elements of the diagonal eigenvalue matrix to the interval $[0.005, 0.1]$ providing a hard constraint on the spatial component. Further Σ_k instead of being estimated is controlled explicitly to avoid overly smooth configurations of color. Further details can be found in [27].

- **Edge Orientation constraint**: to remove false edges so as to retain the orientation of boundary drawn between two neighbouring pixels. This is achieved by computing the strong edges between neighbouring kernels with strong gradient change and increasing the smoothness of both such kernels if the orientation deviates by more than 25 degrees from the orientation of the pixel edge. Mathematical representation can be found in [27].

3.4.2 Perceptually Based Downscaling of Images

Though content-adaptive downscaling proposed by Kopf et al. performs better than the interpolation approaches previously mentioned, the approach by itself does not manage to capture the high frequency information. Also, the approach itself focuses

on maximizing the PSNR value which as mentioned in [32], fails to capture the perception of a human observer. SSIM [56] however is a metric that takes into account the structure of the image and the checks if an image managed to preserve the structure of an image while processing. In 2015, Oeztireli et al. in [41] proposed an optimization problem based on the SSIM.

SSIM is a local measure of patches of images of the same spatial resolution. Rather than losing the information by downsampling the HR image, the authors upsampled the downscaled image to compute SSIM value [41]. The downscaling proposed by the authors was carried out in two steps. Downscaling the image using a regular interpolation and then adding a filter to adaptively sharpen the downscaled image to recover important features. Considering a HR image, H and the corresponding downsampled image as D, the upsampled version of D to be X. The dissimilarity of H and X by $d(H, X)$. The ideal downscaled image D^* corresponding to the X^* resulting in a minimum dissimilarity $d(H, X)$ is solved for. The optimum patch $P^*(X)$ is given by

$$P^*(X) = argmin_{P(X)}\, d(P(H), P(X)) \tag{3.9}$$

The authors of [41] thereby define a non-linear filter as follows

$$d_i^* = \frac{1}{n_p} \sum_{P_k} \mu_h^k + \frac{\sigma_h^k}{\sigma_l^k} \left(l_i - \mu_h^k\right) \tag{3.10}$$

where d_i^* gives the weight assigned to the i^{th} pixel of image D, P_k denotes the n_p sized patches that overlap this pixel. The ratio of the standard deviations of the input image patch h and the filtered version l as σ_h^k/σ_l^k. The standard patch size chosen by the authors is $n_p = 4$ corresponding to a 2×2 patch. The authors also pointed out that the increase in the patch size leads to a loss of small scale features.

The resulting images can be found in [41]. Though the Oeztireli et al. succeeded in reducing jagged edge artifacts and flicker in real time, the ringing artifacts still exist and aliasing still occurs in continuous regular structures. As mentioned in [41], the approach is indifferent to scene semantics. Like other adaptive approaches, noise if present in the HR image is carried to the downsampled image. Any intentional blur in the image is not recognized or preserved as the method involves solving for optimum local structures. The filter developed through optimization in [41], for simplicity consider the input image consisting only of one channel. For the downsampling operation, the optimized filter is applied to each channel of a given image individually which could cause the colors of the image to vary [57].

3.4.3 Rapid, Detail-Preserving Image Downscaling

While the Kopf et al. and Oeztireli et al. proposed approaches for adaptive downscaling, the runtime on both the approaches was very high for any real time task. In 2016, Weber et al. in [57] proposed rapid, detail-preserving image downscaling

(DPID) to preserve the high-frequency details even at high downscaling factors while making it less computationally expensive and time efficient.

The algorithm is a two step convolution operation. Since the task involves downsampling by large factors, for a faster approximation, a box filter is used to rapidly downscale an input HR image I to obtain I_D of dimension similar to the desired output. I_D is then convolved the first time to obtain a guidance image $\tilde{I}$. Each pixel in $\tilde{I}$ is referred to as the local neighbourhood of the corresponding patch of pixels in I. In [57], the patch of pixels in I that are mapped to a pixel p in output image O is denoted by $\Omega_I(p)$. This guidance image $\tilde{I}$ is an integral part of assembling the final image output O from I. For each patch of pixels $\Omega_I(p)$, the pixel values which deviate from the local neighbourhood p are assigned higher weights for the final convolution. The authors of [57] refer to the difference of pixels from the local neighbourhood as the distinctness and is computed by $\|I(q) - I(p)\|$ where $q \epsilon \Omega_I(p)$. Each output pixel $O(p)$ is computed as shown in [57] as

$$\tilde{I} = I_D \otimes \frac{1}{16}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

$$O(p) = \frac{1}{k_p} \sum_{q \epsilon \Omega_I(p)} I(q) \left(\frac{\left\| I(q) - \tilde{I}(p) \right\|}{V_{max}} \right)^{\lambda} \tag{3.11}$$

Where $\left(\left\| I(q) - \tilde{I}(p) \right\| / V_{max} \right)^{\lambda}$ is the range kernel and k_p is the sum of the range kernel across $\Omega_I(p)$. λ is the only variable in the whole equation, that the user can change. V_{max} is the norm of the color space.

The adaptive question contrary to the previous approaches favors the values with higher differences from the local neighbourhood. Experimentally and through a thorough user study, Weber et al. figured that the values of λ around or less than 1 produced more pleasing images than higher. Extensive study can be found in [57]. This approach, due to its neglecting the low-pass filter to remove frequencies above the Nyquist limit, is prone to aliasing and thickening of thin lines with its emphasis on distinctness. The user study in [57] still shows consistently high approval to the approach and the low run time make this one of the most viable approaches.

3.4.4 Comparing Downsampling Approaches

Fig.3.15 shows the different approaches discussed in this section applied to three scenarios. As can be seen in the topmost row, DPID handles the noise in the input image better than the other two approaches especially at $\lambda = 0.5$. Aliasing can be noticed in the second row, in both Kopf et al. and Oeztireli et al.'s approaches owing to the repeated structure. They also fail to capture a lot of the information

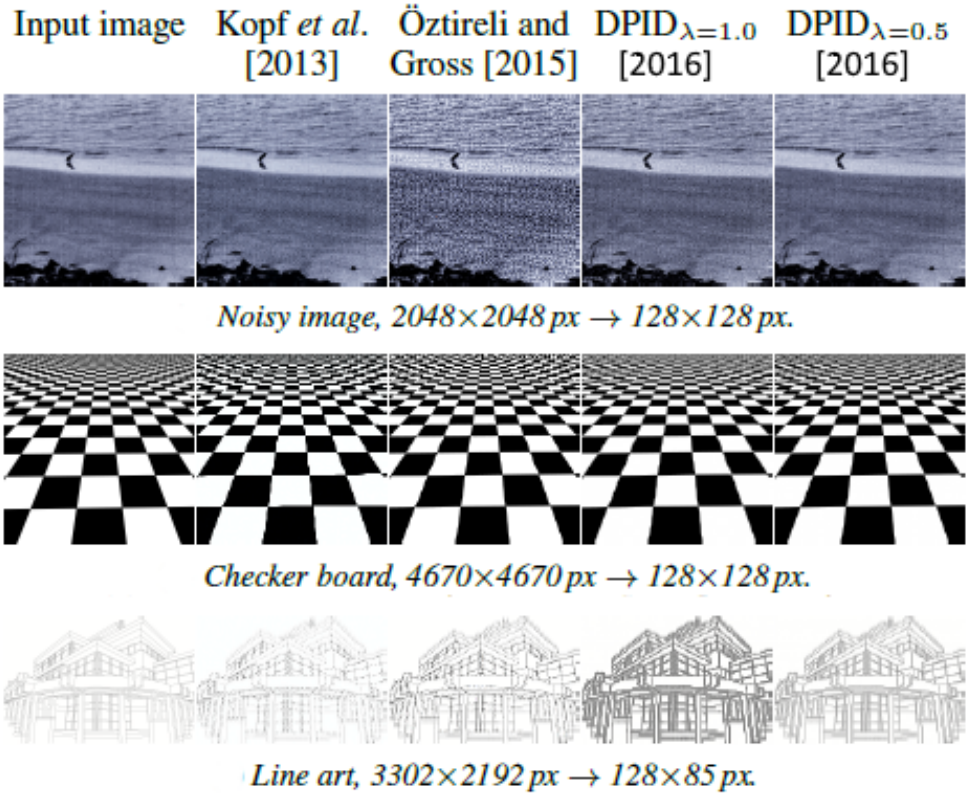

Figure 3.15: [57] Examples of Downsampled Images

in the line art. While DPID at $\lambda = 1.0$ overemphasizes on the lines and thickens it, it still manages to stay within bounds to produce a very clean downscaled image. This tendency to overemphasize, fails it on pixel art, text and images with strong boundaries where DPID tends to make them bolder than would be necessary.

Task Aware Image Downscaling [24] proposed by Kim et al. in 2018 is highly accurate and practical at high sampling factors specific to the super resolution task. The authors use a deep convolutional auto-encoder trained jointly for the upsampling and downsampling processes so as to preserve features that better facilitate the upsampling image to closely approximate the HR image. The results of the same in [24] show great promise as they can aid the pre-existing SR models to perform better.

CHAPTER 4

Implementation

The work done in this thesis includes training selected upsampling models on specific datasets and studying the influence of the downsampling approaches discussed in sec.3.4 on these upsampling model versions.

4.1 Upsampling

The criterion for choosing models for the study in this thesis were the similarities in the approaches. The models that best produced superior results as discussed in sec.3.3 are GAN based and will be the point of focus in the study. The models thus chosen are SRGAN [32], SRFeat [42] and ERCA [19]. The study is to verify the influence of the training datasets on each said model. For fairness during the evaluation, the models were trained as similarly as possible for all the trainnig datasets from the scratch. The generator networks from each of the models mentioned above are unaltered but the discriminator networks used during the GAN part of the training inspired from [42]. Following [42] and [19], two discriminators were used during the GAN part of the training, namely the image discriminator and the feature discriminator. The training was conducted in two phases. The generator networks from [32], [42] and ERCA [19] were first pre-trained on a augmented DIV2K dataset. Following this, each of the pre-trained models were trained using the GAN framework on flickr1024, stanford cars dataset and oregon wildlife dataset. All the images in the datasets are augmented as mentioned in sec.4.1.2. The models were all trained to upsample by a factor of 4 with the HR images of size 296×296 and LR images of size 74×74[1]. The resultant trained models are not restricted

[1]Note: In every instance where a number of images in an augmented dataset is mentioned in sec.4.1.2, it corresponds to that many 296×296 HR and corresponding bicubic downsampled 74×74 LR images.

by these specific resolutions but will be operating with a sampling factor of 4. The training process used will be further discussed in sec. 4.1.3 and sec.4.1.4 respectively.

4.1.1 Hardware

All the models were trained in the thesis were carried on a NVIDIA Quadro P5200 GPU on board a HP ZBook 17 G5. The workbook runs a Intel(R) Core(TM) i7-8850H processor with a 32 GB RAM and a CPU memory clock speed of 2.60 GHz. The on board NVIDIA GPU has a dedicated memory of 16 GB, contains 2560 cores for multiple threads, a bus width of 256 bits and a band width of 230 GB/s. The GPU has a base clock speed of 1.556 GHz and can reach boosted clock speed of 1.746 GHz. The effective memory clock speed stands at 7.200 GHz. The code was also tested on a machine running a Intel(R) core(TM) i5-6300U processor with a memory clock speed of 2.40 GHz, 8 GB Ram and no on board GPU but the run-time of the code subsequently increases if run on a CPU.

4.1.2 Training Datasets

- The dataset chosen for the pre-training of the generator networks is DIV2K [7] which contains 800 images for training and 100 images for testing each with image resolutions around 2000×1500. These are individually augmented using the pre processing similar to [42] to a cumulative $104,000$ training images and $13,000$ test images.

- The first dataset chosen for the gan-training is flickr1024 dataset [54] which contains 800 training images each with resolution above 800×800. This dataset is then augmented similarly to a cumulative $14,400$ images. This dataset was chosen as it has multiple-domains including (but not only) landscapes, humans, vehicles.

- The second dataset chosen for the gan-training is the oregon wildlife dataset [40] with $11,969$ training images. It only contains images of wildlife and is also domain-specific. This dataset is augmented similar to the stanford cars dataset to a cumulative of $10,350$ training images.

- The third dataset chosen for the gan-training is the stanford cars dataset [28] with $8,144$ training images. They comprise of images of variety of cars and is domain-specific unlike flickr1024. This dataset was not augmented similarly as to its wide range of resolutions. All the images above 296×296 were center cropped to match the aspect ratio $1:1$ and resized to 296×296. This amounts to a cumulative of $6,082$ training images.

4.1.3 Pre-Training Phase for selected Generators

The generator networks that were trained on the augmented DIV2K dataset are

- SRResNet [32] as seen in the generator network in fig.3.11.
- SRFeat_{128} [42] with 128 channels as seen in the generator network in fig. 3.13.
- ERCA [19] generator network as seen in fig.3.14.

Each of the networks was trained for 13×10^4 update iterations over $104,000$ pairs of 296×296 HR and 74×74 LR images[2]. Each iteration is done with a batch size of 16 or 16 image pairs and 6500 batches per epoch resulting in a total of 20 epochs. The loss function used for SRResNet and SRFeat is 'mean squared error' and the loss function used for ERCA is 'mean absolute error'. The optimizer used for convergence in all the models during training is the adam optimizer with momentums, $\beta_1 = 0.9$, $\beta_2 = 0.99$ and a learning rate of 10^{-4} with a decay= 0.5 scheduled at 10^{th} and 15^{th} epochs.

4.1.4 Adversarial Training Phase

In this thesis, each of the above generators is trained using the GAN [16] framework and two discriminators simultaneously. The feedback during the update is dependent on the values of both the discriminators. The learning rate selected for the GAN/Adversarial training is 10^{-4} and is scheduled to decay with a rate of 0.1 at 3^{rd} and 5^{th} epochs. The GAN framework [16] solves a min-max problem defined similar to [16, 32, 42, 53] as

$$min_g\ max_d \Big(\Sigma_{y \sim p_{data}(y)} [log(d\,(y)\,)] + \Sigma_{x \sim p_x(x)} [log(1 - d\,(g\,(x))\,)] \Big) \tag{4.1}$$

where d is the discriminator, y is a sample from the real data and $g\,(x)$ is the generated output for the given random noise x from the latent space.

The Loss function followed during the adversarial training of each of the models is

$$L_g = L_p + \lambda \left(L_a^i + L_a^f \right) \tag{4.2}$$

Where L_p is the perceptual similarity loss which is computed in the feature domain between the feature maps of the real image and the generated image obtained by using the VGG-Network [47]. Considering d^i and d^f as the image and feature discriminators as used in [42], the corresponding pixel domain image GAN loss and the feature domain feature GAN loss are given by L_a^i and L_a^f respectively. The equations for the GAN losses and Discriminator losses can be found in [42].

The graphs representing the behaviour of the networks during the gan trainings performed are shown in the next three sub-sections.

[2] The HR images are downsampled by a factor of $\times 4$ using the bicubic interpolation to obtain the LR images

4.1.4.1 SRGAN

The generator Network from [32] pre-trained on an augmented DIV2K dataset and the discriminator network from [42] are trained with the GAN framework on the training datasets mentioned in sec. 4.1.2. The total loss graphs for the generator and discriminator for SRGAN (SRResNet + Discriminator) are given in fig.4.1.

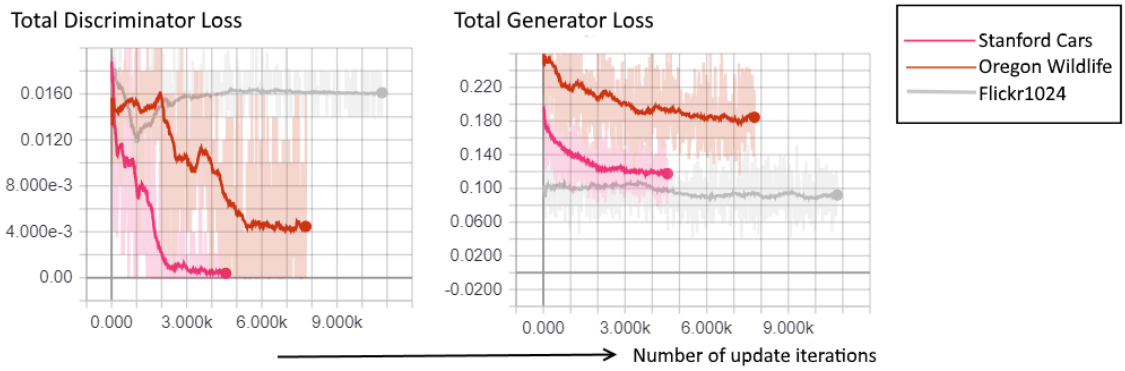

Figure 4.1: Losses during training of SRGAN on stanford cars, oregon wildlife and flickr1024 datasets

4.1.4.2 SRFeat

The generator Network from [42] pre-trained on an augmented DIV2K dataset and the discriminator network from [42] are trained with the GAN framework on the training datasets mentioned in sec. 4.1.2. The loss graphs for the generator and discriminator for SRFeat are given in fig.4.2. The graph shows higher number of the update iterations when compared to the other two models. This was done to verify if higher update iterations varied the performance. The evaluation in the next chapter of this model was only done with the model trained for 6 epochs for fairness.

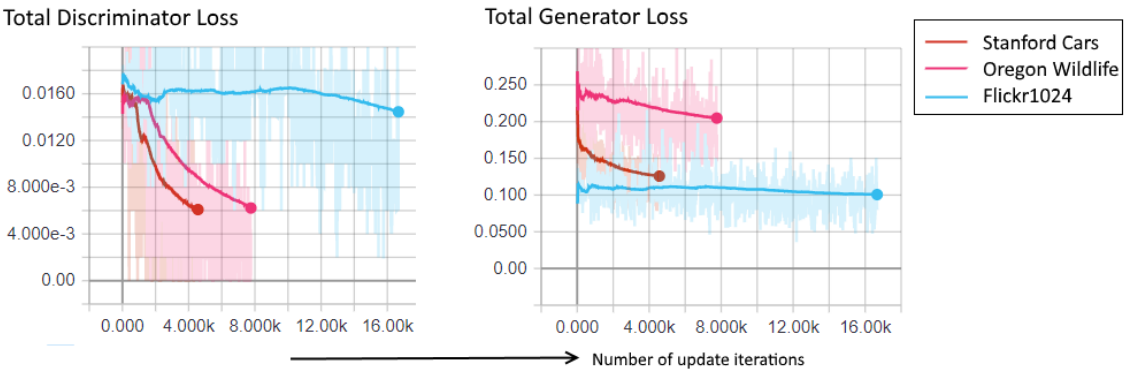

Figure 4.2: Losses during training of SRFeat on stanford cars, oregon wildlife and flickr1024 datasets

4.1.4.3 ERCA

The generator Network from [19] pre-trained on an augmented DIV2K dataset and the discriminator network from [42] are trained with the GAN framework on the training datasets mentioned in sec. 4.1.2. The loss graphs for the generator and discriminator for ERCA are given in fig.4.3

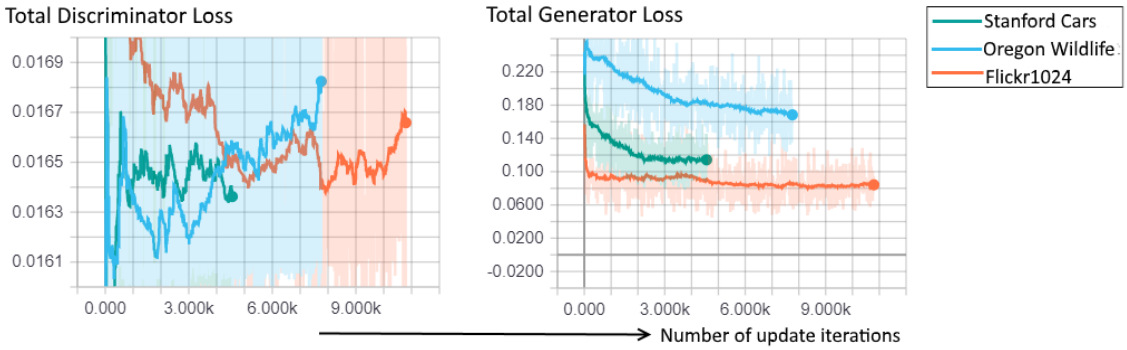

Figure 4.3: Losses during training of ERCA on stanford cars, oregon wildlife and flickr1024 datasets

4.2 Downsampling

Figure 4.4: Downsampled and rescaled crops of the comic image from set14

Downsampling models chosen for this thesis are two adaptive interpolation approaches. The task here is to study the influence of available downsampling models on the quality of the images generated during upsampling. For each available HR image, a downscaled version (LR) of the image is generated using bicubic [23], Lanczos [13], Perceptually based downscaling [41] and DPID($\lambda = 0.5$) [57] approaches.

These LR images are then upsampled using each of the model from section 4.1. The quantitative study of which can be found in the next chapter. Fig. 4.4 shows the downscaling operation done on a sample image from Set14 [55] dataset using each approach. Content adaptive image downscaling [27] is also studied initially but the processing time for downscaling using this method is extremely high and is not viable for any real-time application and hence is excluded from the extensive evaluation in chapter 5. During the implementation of the content adaptive downscaling, it was noted that an image of 62×90 takes about 45 min on an Intel Core i5 processor at 2.40Ghz speed.

CHAPTER 5

Evaluation

This chapter contains the quantitative evaluation of the up- and downsampling performed by the models discussed in sections 4.1 and 4.2 respectively. The methods and metrics used are described in the following sections.

5.1 Benchmark Test Sets

The benchmark test sets chosen for the evaluation of the upsampling models are

- **Set5** [55]: This dataset consists of 5 images of varying resolutions and is commonly used a test set in the field of super resolution.
- **Set14** [55]: This dataset consists of 14 images of varying resolutions and is commonly used a test set in the field of super resolution alongside set5. This dataset is further used to evaluate the downsampling approaches and is discussed further in section 5.4.
- **Subset of oregon wildlife dataset** [40]: This dataset contains a subset of 11 test images from the oregon wildlife dataset excluded from the training process. This is to test the performance of the models on unknown images in the same domain.
- **Subset of stanford cars dataset** [28]: This dataset contains a subset of 10 test images from the stanford cars dataset excluded from the training process. This is to test the performance of the models on unknown images in the same domain.

5.2 Metrics

The metrics chosen to evaluate the performance of all the trained upsampling models and the influence of the downsampling approaches on each of them are

- **PSNR**: Peak Signal to Noise Ratio is the ratio of the maximum possible value ($MAX_{[}I] = 255$ for images) to the mean squared error (MSE) usually expressed in logarithmic decibels and is a pixel-wise metric. Higher the PSNR value, higher the pixel-wise accuracy of the image. It is given by

$$PSNR = 10.log_{10}\Big(\frac{MAX_I^2}{MSE}\Big) \tag{5.1}$$

- **SSIM** [56]: A full reference metric made as an improvement to PSNR which compares the uncompressed reference of the image. High SSIM indicates better structural similarity. It compares two windows x, y of common size $N \times N$ with averages across the values μ_x, μ_y, variances σ_x^2, σ_y^2 respectively and a covariance σ_{xy}. It is given by

$$SSIM\,(x, y) = \frac{(2\mu_x\mu_y + c_1)\,(2\sigma_{xy} + c_2)}{\Big(\mu_x^2 + \mu_y^2 + c_1\Big)\,\Big(\sigma_x^2 + \sigma_y^2 + c_2\Big)} \tag{5.2}$$

  where c_1 and c_2 are variables to stabilize the fraction.

- **RMSE** [8]: In [8], the authors use RMSE (Root Mean Squared Error) in tandem with the PI (Perceptual Index) to evaluate it while comparing it with a pixel-wise metric. Low values of RMSE implies better pixel-wise accuracy of the image. RMSE is given by

$$RMSE\,(x, y) = \sqrt{\sum_{i=1}^{i=n} \frac{\Big\|(x_i - y_i)^2\Big\|}{n}} \tag{5.3}$$

  where i is the current pixel and x_i and y_i are the two pixels each in the real and generated image to be compared and n is the total number of pixels.

- **PI** [8]: The authors of [8] introduced this as a no reference metric that quantitatively represents the perceptive quality of a given image independent of any reference image. This metric has been used in PIRM-SR 2018 challenge. Lower the value of PI, better the quality of the image. PI is computed from the two no reference values namely the Ma [36] value and the NIQE [39](Naturalness Image Quality Evaluator). The Perceptual Index, PI is given by

$$PerceptualIndex\,(PI) = \frac{1}{2}(\,(10 - Ma) + NIQE) \tag{5.4}$$

5.3 Evaluation of Upsampling Models

To evaluate upsampling models, the HR images from benchmark test datasets mentioned in section 5.1 are first downsampled using bicubic interpolation and then upsampled using versions of SRGAN [32], SRFeat [42] and ERCA [19]. Each of the models is trained individually on DIV2K dataset, flickr1024 dataset, oregon wildlife dataset and stanford cars dataset. 3 pre-trained generators trained on DIV2K and each trained on the rest of the 3 datasets resulting in 9 fully trained generators (after training them using the GAN framework). This results in a cumulative of 12 models which are evaluated on the 4 test sets from section 5.1. The following sections contain extensive quantitative study using the metrics from section 5.2 represented in tabular form. Each HR, SR image pair is evaluated using the metrics and the mean of the whole set is shown in the corresponding tables. Note that DIV2K in training dataset column represents the 3 pre-trained generators and is not fully trained using the GAN [16] framework. Flickr1024, oregon wildlife and stanford cars in the same column represent a fully trained generator post adversarial training. Also, there are 48 possible test dataset-US[1] pairs and 4 metrics per pair. For readability, each pair of table only shows the values for one test set resulting in a total of 8 tables. All observations from the evaluation of the upsampling models can be found in sec. 5.3.6.

5.3.1 Set5

Table 5.1 and 5.2 contain the PSNR, SSIM and RMSE, PI values respectively which were obtained from the images from Set5 [55] upsampled using all 12 models. The model architecture is given in the top row and the training dataset used for each model in the left column.

Training Dataset	**SRGAN**		**SRFeat**		**ERCA**	
	PSNR (db)	*SSIM*	*PSNR (db)*	*SSIM*	*PSNR (db)*	*SSIM*
DIV2K	31.75	0.88	31.73	0.89	32.19	0.89
Flickr1024	29.44	0.74	29.12	0.74	28.01	0.78
Oregon Wildlife	27.38	0.71	28.01	0.76	26.45	0.68
Stanford Cars	26.43	0.67	27.82	0.71	24.27	0.70

Table 5.1: Table of PSNR and SSIM values for upsampled Set5 images

[1] test datasets (4) - Upsampling Models(12)

Training Dataset	SRGAN		SRFeat		ERCA	
	RMSE	*PI*	*RMSE*	*PI*	*RMSE*	*PI*
DIV2K	7.14	6.14	7.13	6.08	6.73	5.97
Flickr1024	10.06	5.68	9.26	5.59	10.46	5.56
Oregon Wildlife	11.34	6.53	10.03	6.42	12.72	6.05
Stanford Cars	10.90	5.17	9.54	5.02	13.69	4.94

Table 5.2: Table of RMSE and PI values for upsampled Set5 images

5.3.2 Set14

Table 5.3 and 5.4 contain the PSNR, SSIM and RMSE, PI values respectively which were obtained from the images from Set14 [55] upsampled using all 12 models. The model architecture is given in the top row and the training dataset used for each model in the left column.

Training Dataset	SRGAN		SRFeat		ERCA	
	PSNR (db)	*SSIM*	*PSNR (db)*	*SSIM*	*PSNR (db)*	*SSIM*
DIV2K	28.80	0.79	28.74	0.79	29.07	0.80
Flickr1024	27.63	0.69	25.12	0.71	25.77	0.68
Oregon Wildlife	23.30	0.68	25.65	0.67	25.16	0.61
Stanford Cars	26.64	0.65	26.41	0.69	24.12	0.62

Table 5.3: Table of PSNR and SSIM values for upsampled Set14 images

Training Dataset	SRGAN		SRFeat		ERCA	
	RMSE	*PI*	*RMSE*	*PI*	*RMSE*	*PI*
DIV2K	10.77	5.52	10.82	5.40	10.49	5.34
Flickr1024	14.61	5.15	10.83	5.03	14.40	5.10
Oregon Wildlife	18.56	5.97	12.19	5.45	15.04	7.05
Stanford Cars	16.29	5.44	11.17	5.26	17.14	4.86

Table 5.4: Table of RMSE and PI values for upsampled Set14 images

5.3.3 Oregon Wildlife Dataset

Table 5.5 and 5.6 contain the PSNR, SSIM and RMSE, PI values respectively which were obtained from the images from the subset of oregon wildlife test set [40] upsampled using all 12 models. Note that the images used to test were excluded from the training process and the models processed the images for the first time. In the tables the model architecture is given in the top row and the training dataset used for each model in the left column.

Training Dataset	SRGAN		SRFeat		ERCA	
	PSNR (db)	*SSIM*	*PSNR (db)*	*SSIM*	*PSNR (db)*	*SSIM*
DIV2K	28.42	0.78	28.39	0.77	28.51	0.78
Flickr1024	26.68	0.69	25.54	0.77	26.10	0.68
Oregon Wildlife	27.67	0.71	28.33	0.68	25.45	0.61
Stanford Cars	26.76	0.64	27.53	0.64	24.05	0.64

Table 5.5: Table of PSNR and SSIM values for upsampled images from subset of oregon wildlife dataset

Training Dataset	SRGAN		SRFeat		ERCA	
	RMSE	*PI*	*RMSE*	*PI*	*RMSE*	*PI*
DIV2K	10.52	6.18	10.55	6.03	10.44	6.12
Flickr1024	11.47	5.98	11.98	5.29	13.46	5.34
Oregon Wildlife	12.50	6.01	12.15	5.73	14.21	5.93
Stanford Cars	10.31	5.83	10.36	5.22	12.14	5.46

Table 5.6: Table of RMSE and PI values for upsampled images from subset of oregon wildlife dataset

5.3.4 Subsets of Stanford Cars Dataset

Table 5.7 and 5.8 contain the PSNR, SSIM and RMSE, PI values respectively which were obtained from the images from the subset of stanford cars test set [28] upsampled using all 12 models. Note that the images used to test were excluded from the training process and the models processed the images for the first time. In the tables the model architecture is given in the top row and the training dataset used for each model in the left column.

Training Dataset	SRGAN		SRFeat		ERCA	
	PSNR (db)	*SSIM*	*PSNR (db)*	*SSIM*	*PSNR (db)*	*SSIM*
DIV2K	27.27	0.85	27.22	0.84	27.54	0.86
Flickr1024	28.32	0.62	27.18	0.71	24.85	0.75
Oregon Wildlife	26.78	0.62	26.82	0.69	25.96	0.59
Stanford Cars	29.88	0.64	24.93	0.65	24.84	0.65

Table 5.7: Table of PSNR and SSIM values for upsampled images from subset of stanford cars dataset

Training Dataset	SRGAN		SRFeat		ERCA	
	RMSE	*PI*	*RMSE*	*PI*	*RMSE*	*PI*
DIV2K	11.33	5.44	11.42	5.38	11.00	5.15
Flickr1024	11.24	5.74	12.66	5.42	14.75	5.90
Oregon Wildlife	14.75	5.53	12.75	5.63	16.38	6.21
Stanford Cars	12.42	5.04	11.64	4.99	18.72	5.03

Table 5.8: Table of RMSE and PI values for upsampled images from subset of stanford cars dataset

5.3.5 Sample Image Sets from Upsampling

This section contains some sample resultant image sets. Fig.5.1 consists of the baby image from Set5 test set upsampled using the 3 pre-trained generators. The values clearly represent the close similarity in performance among the 3 models before adversarial training. Fig. 5.2 consists of the bird image from Set5 test set upsampled using the SRGAN model trained on each of the training datasets. Fig.5.3 consists of the lenna image from Set14 test set upsampled using the ERCA model trained on each of the training datasets. Though the fully trained models in fig.5.2 show some checkerboard artifacts, the models trained on stanford cars in both fig.5.2 adn 5.3 outperformed the others perceptually by a certain margin possibly owing to the training dataset consisting of monochromatic cars which led to the flattening of colours which are perceptually more pleasing than jagged details but that at the cost of texture detail as can easily be seen in the blue feathers in the fig.5.3.

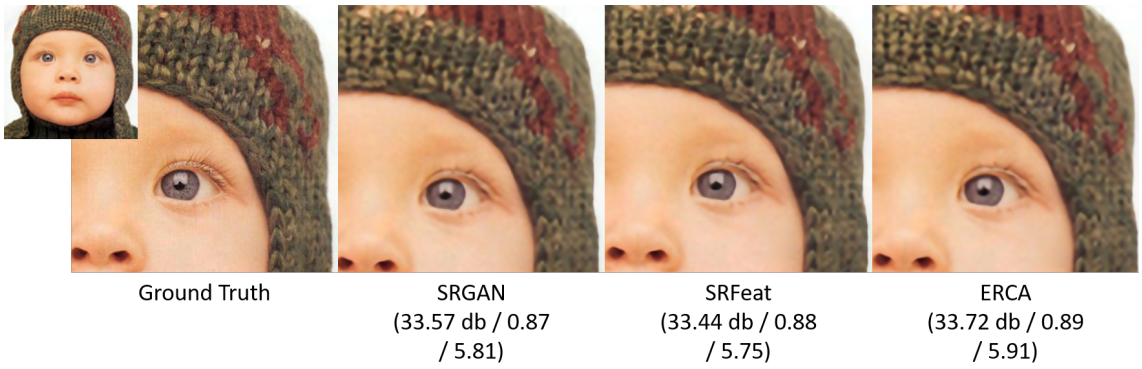

Figure 5.1: Sample image from Set5 [55] upsampled using the pre-trained generators trained on DIV2K with values represented as PSNR/SSIM/PI

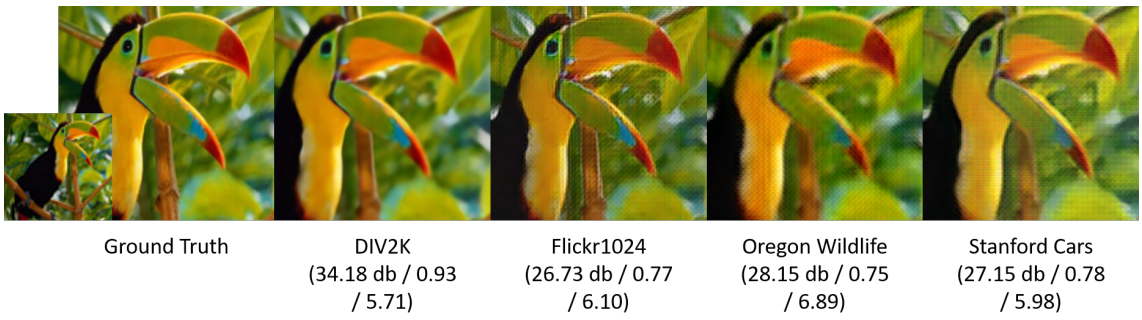

Figure 5.2: Sample image from Set5 [55] upsampled using SRGAN trained on all 4 datasets with values represented as PSNR/SSIM/PI

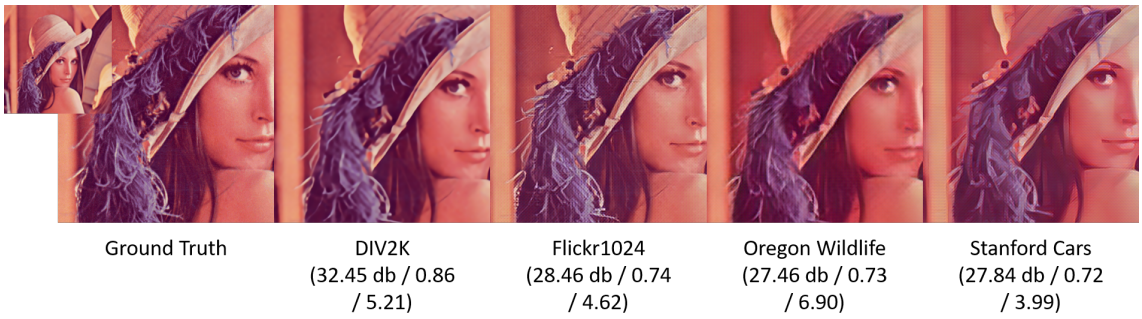

Figure 5.3: Sample image from Set14 [55] upsampled using ERCA trained on all 4 datasets with values represented as PSNR/SSIM/PI

5.3.6 Observations from the Study of Upsampling Models

This observations made on this study and are given by dataset that the models SRGAN, SRFeat and ERCA were trained on. Note that, DIV2K was used to pre-train the generators and the rest of the datasets were used to train the models further using the GAN framework. All three models were trained under similar conditions with the same training parameters to produce fairly similar results to draw on the differences caused by the dataset they were trained on.

- The performance of pre-trained generators post training on DIV2K were comparable to the state-of-the-art results from [32, 42] as can be seen from the tables in the section 5.3. The images produced by the pre-trained generators were highly consistent and will make for a pretty efficient choice on their own because training with the GAN framework is unstable and it is possible to run into issues like the vanishing gradient and mode collapse [9].

- In all the models trained on flickr1024, the models showed minor improvement in performance as the models seem to have learnt very less. This can be observed from the loss curves in figs. 4.1, 4.2 and 4.3. Though the images from flickr1024 were of fairly high definition, the background and the foreground

are simultaneously in focus and most of the images are highly saturated colors which could be the reason for the overshot colors as can be seen in fig.5.2.

- In all the models trained on Oregon wildlife dataset, the model deteriorated in performance and showed checkerboard artifacts for some and it can be noticed in figs 5.2 and 5.3. Ideally, a model trained on domain specific data should perform better than the models trained on multiple domains but as can be seen in table 5.6 even on the subset of Oregon wildlife dataset, the models trained on multiple domains performed better. What one can infer from this is the specificity of the dataset could be a factor. The dataset though specific to wildlife in Oregon has 30 different animal species and contained images that were sketches or taken from a footage as well.

- In all the models trained on Stanford cars dataset, the models performed fairly well for the metrics chosen as can be seen in tables 5.7 and 5.8. But the images itself had flat tones of color. The images though pleasant to the look at, any texture detail from the HR image are not replicated. This can be observed clearly in the feathers from fig. 5.3. This can be from the fact that the dataset contains monochromatic car images with fairly flat tones of color. Some resultant images also showed reduced saturation of colors as can be seen in the last row of fig. 5.4.

5.4 Upsampling specific Evaluation of Downsampling Methods

To evaluate the influence of the downsampling method used before the upsampling on the performance of the upsampling models, the images from Set14 are downsampled using bicubic [23], lanczos [13], perceptually based downscaling [41] and rapid detail preserving downscaling [57] methods. These are each then upsampled using the 12 model variations (3 pre-trained generators and 9 fully trained generators). Content adaptive downsampling is another method studied but the processing time of the algorithm was extremely slow for it having any real-time applications. On an Intel core i5 processor running at 2.40GHz, an image of size 32×90 took 45 mins to downscale. Hence it was excluded from the extensive study using the test sets. The PSNR, SSIM and RMSE, PI are calculated for each of the HR, SR image pair. The means for each SR version of Set14 are then represented in a tabular form in the following sections. Note that this amounts to 48 possible DS-US[2] pairs and 4 metrics per pair. For readability, each pair of tables contain the values for the models trained on one particular training dataset. The resulting 8 tables can be found in the following sections.

[2]Downsampling Method (4) - Upsampling Model (12)

5.4.1 Generators Pre-trained on DIV2K

Table 5.9 and 5.10 contain the PSNR, SSIM and RMSE, PI values respectively which were obtained by first downsampling the images from Set14 [55] using the 4 downsampling methods and then upsampling each of them using the 3 generator models obtained post pre-training SRGAN, SRFeat and ERCA on DIV2K dataset [7]. The model architecture is given in the top row and the downsampling methods used in the left column. Example set of resultant images from running the ERCA model trained on DIV2K dataset can be found in the DIV2K (2^{nd}) row of images in fig. 5.4.

DS Method	**SRGAN**		**SRFeat**		**ERCA**	
	PSNR (db)	*SSIM*	*PSNR (db)*	*SSIM*	*PSNR (db)*	*SSIM*
Bicubic	28.80	0.79	28.74	0.79	29.07	0.80
Lanczos	28.45	0.79	28.41	0.79	28.55	0.80
Perceptually Based	21.45	0.57	21.57	0.57	20.35	0.54
DPID (λ=0.5)	26.73	0.74	26.90	0.74	25.85	0.72

Table 5.9: Table of PSNR and SSIM values for Set14 upsampled from various LR image versions by models trained on DIV2K dataset

DS Method	**SRGAN**		**SRFeat**		**ERCA**	
	RMSE	*PI*	*RMSE*	*PI*	*RMSE*	*PI*
Bicubic	10.77	5.52	10.82	5.40	10.49	5.34
Lanczos	11.09	5.42	11.12	5.17	11.00	5.19
Perceptually Based	22.61	4.12	22.35	4.04	25.75	3.78
DPID (λ=0.5)	13.25	4.47	13.07	4.39	14.26	4.26

Table 5.10: Table of RMSE and PI values for Set14 upsampled from various LR image versions by models trained on DIV2K dataset

5.4.2 Generators Trained on Flickr1024

Table 5.11 and 5.12 contain the PSNR, SSIM and RMSE, PI values respectively which were obtained by first downsampling the images from Set14 [55] using the 4 downsampling methods and then upsampling each of them using the 3 generator models obtained post the adversarial training of SRGAN, SRFeat and ERCA on flickr1024 dataset [54]. The model architecture is given in the top row and the downsampling methods used in the left column. Example set of resultant images from running the ERCA model trained on flickr1024 dataset can be found in the flickr1024 (3^{rd}) row of images in fig. 5.4.

DS Method	SRGAN		SRFeat		ERCA	
	PSNR (db)	*SSIM*	*PSNR (db)*	*SSIM*	*PSNR (db)*	*SSIM*
Bicubic	27.63	0.69	25.12	0.71	25.77	0.68
Lanczos	26.88	0.68	25.10	0.68	25.93	0.64
Perceptually Based	20.53	0.55	21.12	0.60	18.37	0.28
DPID (λ=0.5)	24.90	0.61	24.93	0.64	22.20	0.48

Table 5.11: Table of PSNR and SSIM values for for Set14 upsampled from various LR image versions by models trained on flickr1024 dataset

DS Method	SRGAN		SRFeat		ERCA	
	RMSE	*PI*	*RMSE*	*PI*	*RMSE*	*PI*
Bicubic	14.61	5.15	10.83	5.03	14.40	5.10
Lanczos	15.28	4.93	12.55	4.98	14.22	5.02
Perceptually Based	18.09	4.69	18.54	4.50	16.76	4.36
DPID (λ=0.5)	14.44	4.72	11.78	4.32	13.78	4.57

Table 5.12: Table of RMSE and PI values for Set14 upsampled from various LR image versions by models trained on flickr1024 dataset

5.4.3 Generators Trained on Oregon Wildlife Dataset

Table 5.13 and 5.14 contain the PSNR, SSIM and RMSE, PI values respectively which were obtained by first downsampling the images from Set14 [55] using the 4 downsampling methods and then upsampling each of them using the 3 generator models obtained post the adversarial training of SRGAN, SRFeat and ERCA on oregon wildlife dataset [40]. The model architecture is given in the top row and the downsampling methods used in the left column. Example set of resultant images from running the ERCA model trained on Oregon wildlife dataset can be found in the Oregon wildlife (4^{th}) row of images in fig. 5.4.

DS Method	SRGAN		SRFeat		ERCA	
	PSNR (db)	*SSIM*	*PSNR (db)*	*SSIM*	*PSNR (db)*	*SSIM*
Bicubic	23.30	0.68	25.65	0.67	25.16	0.61
Lanczos	23.79	0.58	26.05	0.62	25.96	0.63
Perceptually Based	21.15	0.51	23.15	0.59	25.44	0.66
DPID (λ=0.5)	23.47	0.58	25.93	0.62	24.56	0.64

Table 5.13: Table of PSNR and SSIM values for for Set14 upsampled from various LR image versions by models trained on oregon wildlife dataset

DS Method	SRGAN		SRFeat		ERCA	
	RMSE	*PI*	*RMSE*	*PI*	*RMSE*	*PI*
Bicubic	18.56	5.97	12.19	5.45	15.04	7.05
Lanczos	16.65	5.24	13.58	5.26	13.80	6.85
Perceptually Based	23.04	4.95	16.22	5.01	15.24	5.79
DPID (λ=0.5)	17.50	5.10	12.45	5.09	16.89	5.97

Table 5.14: Table of RMSE and PI values for Set14 upsampled from various LR image versions by models trained on oregon wildlife dataset

5.4.4 Generators Trained on Stanford Cars Dataset

Table 5.15 and 5.16 contain the PSNR, SSIM and RMSE, PI values respectively which were obtained by first downsampling the images from Set14 [55] using the 4 downsampling methods and then upsampling each of them using the 3 generator models obtained post the adversarial training of SRGAN, SRFeat and ERCA on stanford cars dataset [28]. The model architecture is given in the top row and the downsampling methods used in the left column. Example set of resultant images from running the ERCA model trained on Stanford cars dataset can be found in the stanford cars (last) row of images in fig. 5.4.

DS Method	SRGAN		SRFeat		ERCA	
	PSNR (db)	*SSIM*	*PSNR (db)*	*SSIM*	*PSNR (db)*	*SSIM*
Bicubic	26.64	0.65	26.41	0.69	24.12	0.62
Lanczos	27.63	0.67	27.15	0.68	24.67	0.64
Perceptually Based	24.88	0.59	22.89	0.61	20.80	0.53
DPID (λ=0.5)	27.47	0.66	27.34	0.64	23.88	0.62

Table 5.15: Table of PSNR and SSIM values for for Set14 upsampled from various LR image versions by models trained on stanford cars dataset

DS Method	SRGAN		SRFeat		ERCA	
	RMSE	*PI*	*RMSE*	*PI*	*RMSE*	*PI*
Bicubic	16.29	5.44	11.17	5.26	17.14	4.86
Lanczos	16.51	4.79	11.26	4.73	16.32	3.66
Perceptually Based	17.57	4.01	18.13	4.17	19.56	3.57
DPID (λ=0.5)	16.33	4.30	12.62	4.23	20.02	3.39

Table 5.16: Table of RMSE and PI values for Set14 upsampled from various LR image versions by models trained on stanford cars dataset

5.4.5 Sample Image Set for Upsampling specific Downsampling

For simplicity, the baboon image from Set14 is downsampled using bicubic, lanczos, perceptually based downscaling and DPID and each of the 4 images is upsampled using the 4 versions of ERCA. The downsampling methods used to generate the LR images are shown at the top and the generator's training dataset is shown at the left of each corresponding row. The first row contains the versions of LR images and the right column contains the same HR image for reference. Note that $\lambda = 0.5$ which represents how much weight is assigned to the pixels that are different from their local neighbourhood (see sec. 3.4.3) as at $\lambda = 1$ the emphasis is much higher leading to a noisy image, see [57] for further examples.

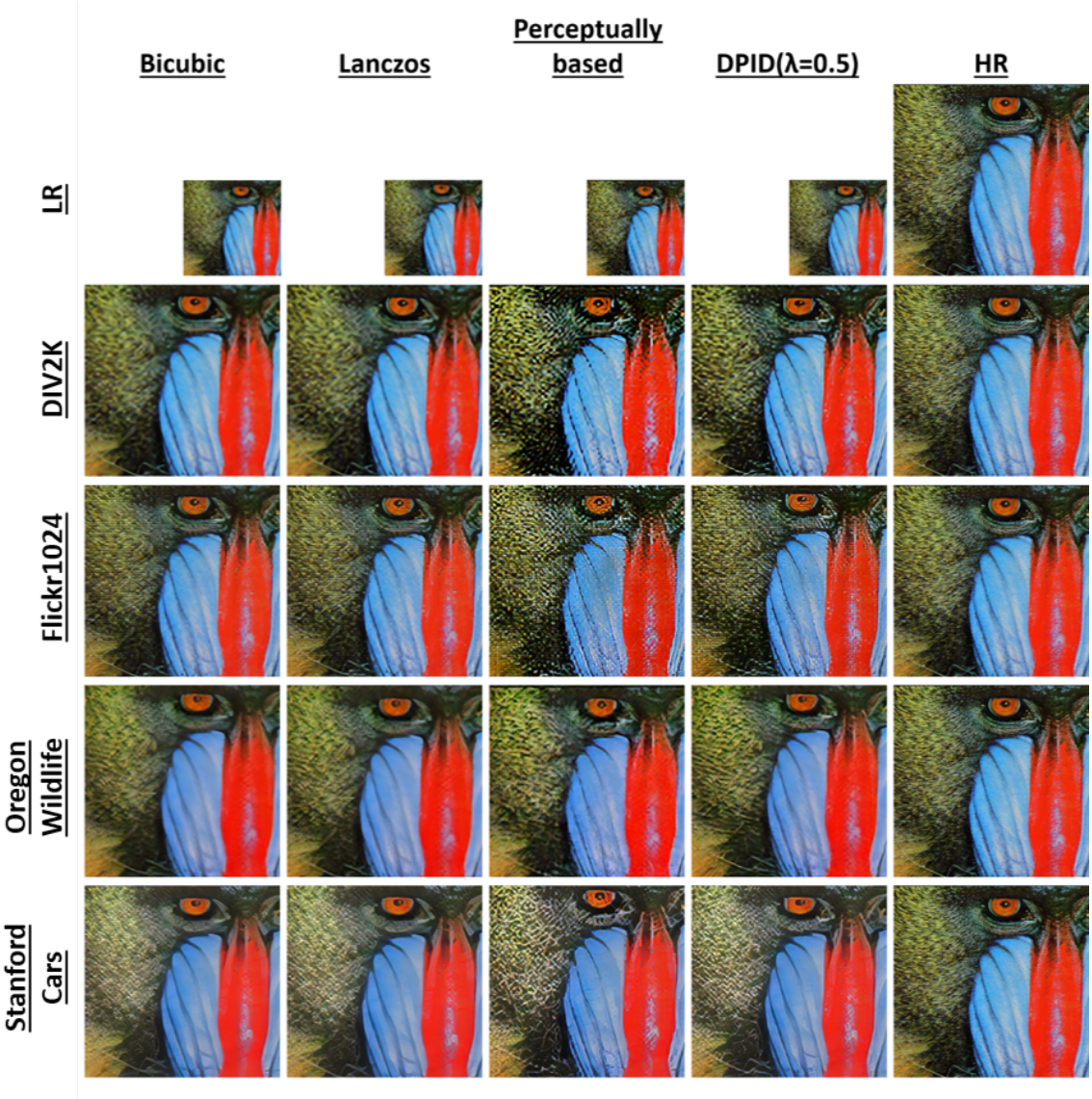

Figure 5.4: Sample image from Set14 [55] upsampled from each LR version using ERCA trained on all 4 datasets

5.4.6 Observations from the Evaluation of Downsampling Methods

Other than the fact that the models trained on Oregon wildliefe dataset's improved perception and replication of fur like patterns as seen in the 4^{th} row of fig. 5.4, the observations made from the study of the influence of downsampling methods on the performance of upsampling models are given by downsampling method hereafter. It is noteworthy that the performance of all the upsampling model versions improved when downsampling methods other than the bicubic interpolation were chosen.

- Upsampling operation in studies are commonly run on an image downsampled using bicubic and is a common practice among the researchers in the field of super resolution. The values obtained from upsampling images downsampled using bicubic interpolation are set as benchmarks and all following methods will be studied in comparison to them.

- Upsampling the images downsampled by Lanczos [13] interpolation performed slightly better than bicubic as can be seen in the tables from 5.4 and the lanczos (2^{nd}) column in fig. 5.4. Considering that the run-time of downscaling an image using Lanczos is not that much slower than using bicubic and also that it is available with most image processing frameworks including and not exclusive to OpenCV and Pillow, its a better alternative owing to its accessibility on par with Lanczos.

- Perceptually based downscaling proposed by the authors of [41] quantitatively show the best performance according to the perceptual index metric and lowest according to all the pixel wise metrics as seen from the tables in section 5.4. That said, as seen in the perceptually based downscaling (3^{rd}) column of the fig. 5.4, the downsampled image shows that the method preserves better features during downsampling operations and has better edges than the two non adaptive interpolation methods above. But, when upsampled, the images have a high emphasis on all the edges making them look more noisy and jagged than the rest of the methods.

- DPID proposed by the author of [57] performs extremely well second only to the perceptually based downscaling according to the perceptual index metric as seen from the tables in section 5.4. The images in the DPID (4^{th}) column represents this pretty well. It has a good balance of the emphasis on the pixels outside the local neighbourhood (high frequency information) and overall image perceptual quality. DPID ($\lambda = 0.5$) being the best among the selected approaches to better the results of upsampling approaches significantly. Also, since λ in the method is a variable and can alter how much weight is given to the high frequency information, this makes a very viable option for futher improving the performance. As given in [57], the values of λ greater than 1 generate noisy images, so the values in the range of 0.5 and 1 are better suited.

CHAPTER 6

Discussion

This chapter contains the discussion of the results, the subsequent achievements and potential drawbacks of the work done in this thesis.

6.1 Results

- An extensive study and evaluation has been performed on the training behaviour of SRGAN [32], SRFeat [42] and ERCA [19] with respect to each being trained on the datasets DIV2K [7], Flickr1024 [54], Oregon wildlife [40] and Stanford cars [28] datasets. The training conditions and parameters were kept similar to evaluate the differences caused by the training datasets under a better scope. It was found that the models trained on domain specific datasets have a high potential to outperform on the specific domain given the dataset is specific enough. All the quantified results have been represented in a tabular form in section 5.3 and the detailed inference from the evaluation is given in section 5.3.6.

- An extensive study and evaluation has been performed on the influence of adaptive downsampling methods namely "perceptually based image downscaling" [41] and "rapid, detail preserving image downscaling" [57] on the performance of each of the models trained on different datasets. It was found that the varying the downscaling method highly influences the performance of even the current state-of-the-art models in a positive way. All the quantified results have been represented in a tabular form in section 5.4 and the detailed inference from the evaluation is given in section 5.4.6.

6.2 Achievements

- The training behaviour of the upsampling models from [19, 32, 42], their performance on benchmark datasets and the influence of adaptive downsampling methods from [41, 57] on the performance of said upsampling models were quantitatively studied and evaluated.
- In tandem with the thesis, a Graphical User Interface was developed in collaboration with IAV GmbH to train the integrated models on available datasets and run the models to up- or downsample images with the evaluated upsampling models and downsampling methods respectively.

6.3 Drawbacks

- Though to narrow the differences caused by the model network itself on the training behaviour, the performance and to specifically test the influence of the chosen training datasets, the discriminator network from [42] was used during the adversarial training for all three models. This also implies that the quantified performance might not be true to the model proposed by respective authors and they could produce results better or otherwise if the study was performed true to the network architectures.
- Since the process of tweaking the training parameters is empirical in nature, the initial experiments lasted for a long duration and were all fairly expensive computationally. Each tweaked set of parameters took anywhere from 5 to 19 hours per epoch depending on the combination of training dataset and model architecture. All quantified results shown in this work are from the training parameters stated and not from the experiments.

CHAPTER 7

Conclusion and Future Work

Depending on how specific the domain of the chosen training dataset is, the performance of any given model architecture can be improved. The downsampling methods have a substantial influence on the performance even on the current state-of-the-art models in the field of Super Resolution (SR). In this work, only the influence of downsampling methods on the performance of fully trained generators is evaluated. In the future, advanced downsampling methods may be used to create the HR LR sets needed for the training of a SR model which should theoretically better the performance of the generators used in this study overall since they would have better LR maps for the HR images. Also, this study strictly quantifies the quality of generated images using available metrics and does not involve any surveys or quantification on classification problems to reinforce the inference drawn which could also potentially be worked on in the future.

Bibliography

[1] *Convolution on RGB Images.* http://datahacker.rs/convolution-rgb-image/

[2] *HD Image of Goomba from Paper Mario.* https://www.pinterest.de/pin/821414419507948343/

[3] *Image Scaling Methods And Matlab Implementations.* https://aditharajakaruna.wordpress.com/2013/07/12/image-scaling-methods-and-matlab-implementations/#more-3

[4] *An Introduction to Convolutions.* https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d

[5] *Lanczos Interpolation Image.* https://en.wikipedia.org/wiki/Lanczos_resampling#/media/File:Lanczos-r02-filtering.svg

[6] *Practical Guide To Tensorflow Conv2D.* https://missinglink.ai/guides/tensorflow/tensorflow-conv2d-layers-practical-guide/

[7] Agustsson, Eirikur ; Timofte, Radu: Ntire 2017 challenge on single image super-resolution: Dataset and study. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, S. 126–135

[8] Blau, Yochai ; Michaeli, Tomer: The perception-distortion tradeoff. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, S. 6228–6237

[9] Borji, Ali: Pros and cons of gan evaluation measures. In: *Computer Vision and Image Understanding* 179 (2019), S. 41–65

[10] Dempster, Arthur P. ; Laird, Nan M. ; Rubin, Donald B.: Maximum likelihood from incomplete data via the EM algorithm. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 39 (1977), Nr. 1, S. 1–22

[11] Dong, Chao ; Loy, Chen C. ; He, Kaiming ; Tang, Xiaoou: Image super-resolution using deep convolutional networks. In: *IEEE transactions on pattern analysis and machine intelligence* 38 (2015), Nr. 2, S. 295–307

[12] Dong, Chao ; Loy, Chen C. ; Tang, Xiaoou: Accelerating the Super-Resolution Convolutional Neural Network: Supplementary File. In: *Computer Vision-ECCV 2016* (2016)

[13] Duchon, Claude E.: Lanczos filtering in one and two dimensions. In: *Journal of applied meteorology* 18 (1979), Nr. 8, S. 1016–1022

[14] Dumitrescu, DragoÈ™ ; Boiangiu, Costin-Anton: A Study of Image Upsampling and Downsampling Filters. In: *Computers* 8 (2019), Nr. 2, S. 30

[15] Goodfellow, Ian ; Bengio, Yoshua ; Courville, Aaron: *Deep learning*. MIT press, 2016

[16] Goodfellow, Ian J. ; Pouget-Abadie, Jean ; Mirza, Mehdi ; Xu, Bing ; Warde-Farley, David ; Ozair, Sherjil ; Courville, Aaron ; Bengio, Yoshua: Generative adversarial networks (2014). In: *arXiv preprint arXiv:1406.2661* (2019)

[17] He, Kaiming ; Zhang, Xiangyu ; Ren, Shaoqing ; Sun, Jian: Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, S. 770–778

[18] He, Kaiming ; Zhang, Xiangyu ; Ren, Shaoqing ; Sun, Jian: Identity mappings in deep residual networks. In: *European conference on computer vision* Springer, 2016, S. 630–645

[19] Hoang T Hieu, Nguzen Viet A. Nguzen Xuan Thanh T. Nguzen Xuan Thanh: ERCA: SISR with Dual Discriminator and Efficient Residual Channel Attention. `https://drive.google.com/file/d/1GFEMT8rCR\7SovhudMWFP_lvP_DrtHoTP/view`

[20] Huang, Gao ; Liu, Zhuang ; Van Der Maaten, Laurens ; Weinberger, Kilian Q.: Densely connected convolutional networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, S. 4700–4708

[21] Ioffe, Sergey ; Szegedy, Christian: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: *arXiv preprint arXiv:1502.03167* (2015)

[22] Jolicoeur-Martineau, Alexia: The relativistic discriminator: a key element missing from standard GAN. In: *arXiv preprint arXiv:1807.00734* (2018)

[23] Keys, Robert: Cubic convolution interpolation for digital image processing. In: *IEEE transactions on acoustics, speech, and signal processing* 29 (1981), Nr. 6, S. 1153–1160

[24] Kim, Heewon ; Choi, Myungsub ; Lim, Bee ; Mu Lee, Kyoung: Task-aware image downscaling. In: *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, S. 399–414

[25] Kim, Jiwon ; Kwon Lee, Jung ; Mu Lee, Kyoung: Accurate image super-resolution using very deep convolutional networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, S. 1646–1654

[26] Kim, Jiwon ; Kwon Lee, Jung ; Mu Lee, Kyoung: Deeply-recursive convolutional network for image super-resolution. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, S. 1637–1645

[27] Kopf, Johannes ; Shamir, Ariel ; Peers, Pieter: Content-adaptive image downscaling. In: *ACM Transactions on Graphics (TOG)* 32 (2013), Nr. 6, S. 1–8

[28] Krause, Jonathan ; Stark, Michael ; Deng, Jia ; Fei-Fei, Li: 3D Object Representations for Fine-Grained Categorization. In: *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*. Sydney, Australia, 2013

[29] Krizhevsky, Alex ; Sutskever, Ilya ; Hinton, Geoffrey E.: Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems*, 2012, S. 1097–1105

[30] LeCun, Yann ; Boser, Bernhard ; Denker, John S. ; Henderson, Donnie ; Howard, Richard E. ; Hubbard, Wayne ; Jackel, Lawrence D.: Backpropagation applied to handwritten zip code recognition. In: *Neural computation* 1 (1989), Nr. 4, S. 541–551

[31] LeCun, Yann ; Bottou, Léon ; Bengio, Yoshua ; Haffner, Patrick: Gradient-based learning applied to document recognition. In: *Proceedings of the IEEE* 86 (1998), Nr. 11, S. 2278–2324

[32] Ledig, Christian ; Theis, Lucas ; Huszár, Ferenc ; Caballero, Jose ; Cunningham, Andrew ; Acosta, Alejandro ; Aitken, Andrew ; Tejani, Alykhan ; Totz, Johannes ; Wang, Zehan u. a.: Photo-realistic single image super-resolution using a generative adversarial network. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, S. 4681–4690

[33] Liang, Shunlin: *Comprehensive Remote Sensing*. Elsevier, 2017

[34] Lim, Bee ; Son, Sanghyun ; Kim, Heewon ; Nah, Seungjun ; Mu Lee, Kyoung: Enhanced deep residual networks for single image super-resolution. In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2017, S. 136–144

[35] Limongelli, Maria P. ; Carvelli, Valter: Damage localization in a glass fiber reinforced composite plate via the surface interpolation method. In: *Journal of Physics: Conference Series* 628 (2015), 07. `http://dx.doi.org/10.1088/1742-6596/628/1/012095`. – DOI 10.1088/1742–6596/628/1/012095

[36] Ma, Chao ; Yang, Chih-Yuan ; Yang, Xiaokang ; Yang, Ming-Hsuan: Learning a No-Reference Quality Metric for Single-Image Super-Rolution. In: *Computer Vision and Image Understanding* (2017), S. 1–16

[37] Mahour, Milad: *Scaling of Remote Sensing Information for Orchard Management*, Diss., 12 2018. `http://dx.doi.org/10.3990/1.9789036546898`. – DOI 10.3990/1.9789036546898

[38] Meijering, Erik: A chronology of interpolation: from ancient astronomy to modern signal and image processing. In: *Proceedings of the IEEE* 90 (2002), Nr. 3, S. 319–342

[39] Mittal, Anish ; Soundararajan, Rajiv ; Bovik, Alan C.: Making a â€œcompletely blindâ€ image quality analyzer. In: *IEEE Signal Processing Letters* 20 (2012), Nr. 3, S. 209–212

[40] Molina, David: *Oregon Wildlife Dataset*. `https://www.kaggle.com/virtualdvid/oregon-wildlife`. – [Online; accessed 28-Oct-2019]

[41] Öztireli, A C. ; Gross, Markus: Perceptually based downscaling of images. In: *ACM Transactions on Graphics (TOG)* 34 (2015), Nr. 4, S. 1–10

[42] Park, Seong-Jin ; Son, Hyeongseok ; Cho, Sunghyun ; Hong, Ki-Sang ; Lee, Seungyong: Srfeat: Single image super-resolution with feature discrimination. In: *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, S. 439–455

[43] Ramachandran, Prajit ; Zoph, Barret ; Le, Quoc V.: Searching for activation functions. In: *arXiv preprint arXiv:1710.05941* (2017)

[44] Raschka, Sebastian: *Visual Explanation on Back Propagation*. `https://sebastianraschka.com/faq/docs/visual-backpropagation.html`

[45] Sharma, Shallu ; Mehra, Rajesh: Implications of Pooling Strategies in Convolutional Neural Networks: A Deep Insight. In: *Foundations of Computing and Decision Sciences* 44 (2019), Nr. 3, S. 303–330

[46] Shi, Wenzhe ; Caballero, Jose ; Huszár, Ferenc ; Totz, Johannes ; Aitken, Andrew P. ; Bishop, Rob ; Rueckert, Daniel ; Wang, Zehan: Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, S. 1874–1883

[47] Simonyan, Karen ; Zisserman, Andrew: Very deep convolutional networks for large-scale image recognition. In: *arXiv preprint arXiv:1409.1556* (2014)

[48] Szegedy, Christian ; Ioffe, Sergey ; Vanhoucke, Vincent ; Alemi, Alexander A.: Inception-v4, inception-resnet and the impact of residual connections on learning. In: *Thirty-first AAAI conference on artificial intelligence*, 2017

[49] Tai, Ying ; Yang, Jian ; Liu, Xiaoming: Image super-resolution via deep recursive residual network. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, S. 3147–3155

[50] Tai, Ying ; Yang, Jian ; Liu, Xiaoming ; Xu, Chunyan: Memnet: A persistent memory network for image restoration. In: *Proceedings of the IEEE international conference on computer vision*, 2017, S. 4539–4547

[51] Tong, Tong ; Li, Gen ; Liu, Xiejie ; Gao, Qinquan: Image super-resolution using dense skip connections. In: *Proceedings of the IEEE International Conference on Computer Vision*, 2017, S. 4799–4807

[52] Wadhawan, Ankita ; Kumar, Parteek: Deep learning-based sign language recognition system for static signs. In: *Neural Computing and Applications* (2020), S. 1–12

[53] Wang, Xintao ; Yu, Ke ; Wu, Shixiang ; Gu, Jinjin ; Liu, Yihao ; Dong, Chao ; Qiao, Yu ; Change Loy, Chen: Esrgan: Enhanced super-resolution generative adversarial networks. In: *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, S. 0–0

[54] Wang, Yingqian ; Wang, Longguang ; Yang, Jungang ; An, Wei ; Guo, Yulan: Flickr1024: A Large-Scale Dataset for Stereo Image Super-Resolution. In: *The IEEE International Conference on Computer Vision (ICCV) Workshops*, 2019

[55] Wang, Zhangyang ; Yang, Yingzhen ; Wang, Zhaowen ; Chang, Shiyu ; Han, Wei ; Yang, Jianchao ; Huang, Thomas: Self-tuned deep super resolution. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2015, S. 1–8

[56] Wang, Zhou ; Bovik, Alan C. ; Sheikh, Hamid R. ; Simoncelli, Eero P.: Image quality assessment: from error visibility to structural similarity. In: *IEEE transactions on image processing* 13 (2004), Nr. 4, S. 600–612

[57] Weber, Nicolas ; Waechter, Michael ; Amend, Sandra C. ; Guthe, Stefan ; Goesele, Michael: Rapid, detail-preserving image downscaling. In: *ACM Transactions on Graphics (TOG)* 35 (2016), Nr. 6, S. 1–6

[58] Weiss, Karl ; Khoshgoftaar, Taghi M. ; Wang, DingDing: A survey of transfer learning. In: *Journal of Big data* 3 (2016), Nr. 1, S. 9

[59] Wolberg, George: *Digital image warping*. Bd. 10662. IEEE computer society press Los Alamitos, CA, 1990

[60] Xing, Wanli ; Du, Dongping: Dropout Prediction in MOOCs: Using Deep Learning for Personalized Intervention. In: *Journal of Educational Computing Research* (2018), 03, S. 073563311875701. http://dx.doi.org/10.1177/0735633118757015. – DOI 10.1177/0735633118757015

[61] Yang, Wenming ; Zhang, Xuechen ; Tian, Yapeng ; Wang, Wei ; Xue, Jing-Hao ; Liao, Qingmin: Deep learning for single image super-resolution: A brief review. In: *IEEE Transactions on Multimedia* 21 (2019), Nr. 12, S. 3106–3121

[62] Zeiler, Matthew D. ; Taylor, Graham W. ; Fergus, Rob: Adaptive deconvolutional networks for mid and high level feature learning In: Computer Vision. In: *IEEE International Conference On* Bd. 2025, 2018

[63] Zhang, Yulun ; Li, Kunpeng ; Li, Kai ; Wang, Lichen ; Zhong, Bineng ; Fu, Yun: Image super-resolution using very deep residual channel attention networks. In: *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, S. 286–301