Hochschule Wismar
Fakultät für Ingenieurwissenschaften
Bereich Elektrotechnik und Informatik

Master Thesis

Interpretation and interpolation of pixel values - Improvement of image scaling methods.

Supervised by:

Supervisor 1: Prof. Dr. rer. nat. Herbert Litschke

Supervisor 2: Prof. Dr.-Ing. Frank Krüger

Submitted By:

Mihir Pradip Patil
Date and place of Birth: 03.10.1994,
Vadodara, India

Submitted On: 11.10.2023

Disposition

Acknowledgement

I would like to express my gratitude to my supervisor, Professor Dr. rer. nat. Herbert Litschke, for his patience, advice, and support. I express my sincere gratitude to him for affording me the opportunity to do research and for his highly helpful feedback on this thesis. I would also like to thank Professor Dr.-Ing. Frank Krüge as the second supervisor of this thesis. I would like to sincerely acknowledge my deep appreciation to my family for their unwavering support and constant encouragement over the course of conducting research and composing this thesis. This achievement would not have been possible without their contributions. I express my gratitude for your assistance.

Author
Mihir Patil.

Abstract

This thesis presents an in-depth analysis of Image Scaling methodologies, with a primary emphasis on the interpretation of Pixel Values and the process of interpolation. Various traditional techniques such as Nearest Neighbour, Bilinear, and Bicubic, as well as advanced deep learning-based approaches, including SRCNN, ESPCN, Autoencoder Convolution, and SRGAN, have been widely used in the field of imaging. These approaches have been thoroughly assessed by measuring image quality using established metrics such as PSNR, SSIM, and perceptual quality. This survey conducts a comparison of the upscaling techniques. Additionally, it seeks to compare different learning-based methods with one another. Our findings provide insights into image scaling methods, aiding practitioners in selecting the most suitable technique for their applications. This research aims to advance image scaling by emphasizing accurate pixel interpretation and interpolation, benefiting fields like computer vision, image processing, and multimedia.

Kurzreferat

Diese Arbeit präsentiert eine eingehende Analyse von Bildskalierungsmethoden mit einem Schwerpunkt auf der Interpretation von Pixelwerten und dem Interpolationsprozess. Verschiedene traditionelle Techniken wie die Nächste-Nachbar-Methode, Bilinear- und Bicubic-Verfahren sowie fortschrittliche Deep-Learning-basierte Ansätze wie SRCNN, ESPCN, Autoencoder Convolution und SRGAN werden in der Bildverarbeitung weithin verwendet. Diese Ansätze wurden eingehend durch die Messung der Bildqualität anhand etablierter Metriken wie PSNR, SSIM und der wahrgenommenen Qualität untersucht. Diese Studie vergleicht die Hochskalierungstechniken miteinander und zielt darauf ab, verschiedene lernbasierte Methoden miteinander zu vergleichen. Unsere Ergebnisse liefern Einblicke in die Methoden der Bildskalierung und unterstützen Praktiker bei der Auswahl der am besten geeigneten Technik für ihre Anwendungen. Diese Forschung zielt darauf ab, die Bildskalierung durch die Betonung einer genauen Pixelinterpretation und Interpolation voranzutreiben, was Bereichen wie der Computer Vision, Bildverarbeitung und Multimedia zugutekommt.

Contents

1. Introduction

Recovering a High-Resolution(High Resolution (HR)) Image from a Low-Resolution Image(Low Resolution (LR)) is known as Image super-resolution, an essential class of image processing. Image super-resolution has applications in various domains, such as medical imaging, surveillance, security, and improving other computer vision tasks. However, it is a challenging and inherently debatable task as multiple HR images can correspond to a single LR image. To resolve this issue, various Super Resolution (SR) methods have been developed, including traditional approaches like interpolation and regularization, as well as modern techniques based on Convolutional Neural Networks (Convolutional Neural Network (CNN)). Image interpolation algorithms are utilized in digital image scaling to convert images from one resolution to another without losing visual content [1]. Image scaling is extensively applied in consumer electronics (e.g., digital cameras, mobile phones, tablets, display devices) and medical imaging, such as computer-assisted surgery and digital radiographs [2]. Enhancing the quality and processing performance of restructured images in hardware implementations is desirable in various applications, including consumer electronics and medical imaging [3]. Image up-scaling methods are implemented in a wide range of computer equipment, such as printers, digital television, media players, image processing systems, and graphics renders [4]. Conversely, high-resolution images may need to be down-scaled to fit smaller sizes [5]. Spatial or spectral correlation, or both, are exploited to restore more accurate and visually appealing results in image scaling.

The utilization of adaptive algorithms enables the detection of local spatial features within pixel neighborhoods, facilitating the selection of appropriate predictors. This process effectively mitigates the occurrence of undesirable artifacts such as zipper effects (Huang et al., 2012[6]). There are two basic areas in which interpolation methods can be classified: spatial domain and frequency domain. Spatial domain approaches are frequently employed in real-time applications because to their inherent simplicity. Frequency domain techniques utilize several transformations such as Discrete Cosine Transform (DCT), Discrete Fourier Transform (DFT), or wavelet transform. However, due to their increased computing complexity, these methods are not suited for real-time or low-cost applications (Huang et al., 2012[6]).

1.1 HR Image: Real-World Applications

1.1.1 Bio-metric information identification and Medical Dig-nose:

Low-resolution bio-metric photos are problematic for facial, fingerprint, and iris identification. Super Resolution methods have become an essential tool in overcoming these problems. The accuracy of feature extraction and matching procedures is considerably increased with the use of SR to increase picture resolution, which boosts recognition accuracy and the overall performance of biometric identification systems. SR technology has a lot of promise to advance security, identity verification, and access control in a variety of real-world applications. Furthermore, as technology develops, additional studies and improvements in SR methods are anticipated to result in even more significant improvements in bio-metric identification systems[7].

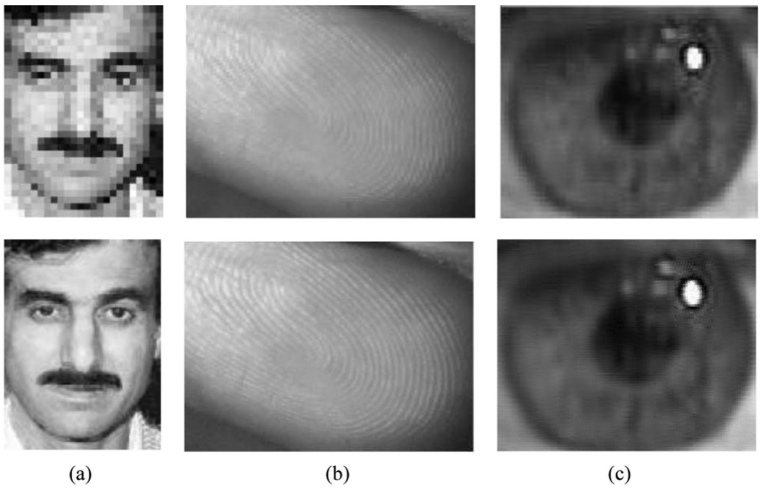


Figure 1.1: (a) Face hallucination,(b) fingerprint reconstruction, and(c) iris reconstruction [7]

As shown in Figure 1.1 enhancing the resolution of medical images is crucial for accurate diagnosis and effective treatment. It can greatly improve the ability to detect diseases and accurately segment images automatically.

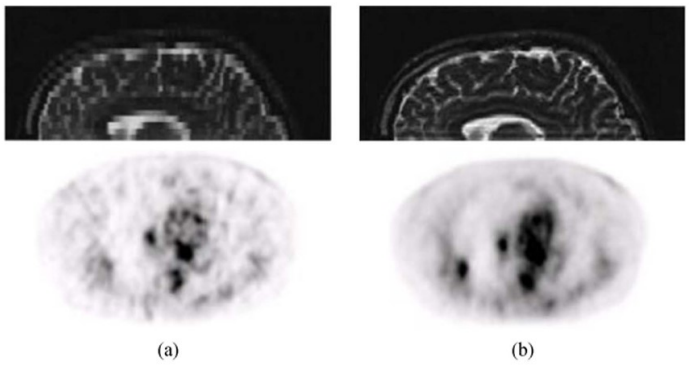


Figure 1.2: Medical Scan Image[7]

Figure 1.2 the first row is the LR image, while the second row is the constructed result. The introduction of digital medical imaging technologies like Computed Tomography (CT), Positron

Emission Tomography (PET), and Magnetic Resonance Imaging (MRI) has revolutionized medical science. However, despite advancements in imaging technology and reconstruction algorithms, obtaining images with the desired resolution is still challenging due to various factors such as imaging environments, limitations of physical systems, and quality issues like noise and blur.

1.1.2 Surveillance and Earth-Observation remote sensing:

Digital video recorder (DVR) devices are commonly utilized nowadays and have a big impact on applications like traffic monitoring and security monitoring. However, at this time, equipping large-scale HR devices is not viable. Therefore, it is essential to research image SR approaches. Two examples of SR are provided in Figure 1.3 for a Unmanned Aerial Vehicle (UAV) surveillance series and a walk sequence where (a) represents the LR image and (b) represents the reconstruction HR image. Although the methods have improved over time, using video SR in practice remains difficult. First of all, weather effects might have a negative influence on outdoor video equipment. Additionally, video data often contains a large quantity of information and intricate movements. While certain methods can handle motion outliers, their use is constrained by computational efficiency. An emphasis has also been placed on compressed video SR [7].

Figure 1.3: The SR reconstruction of the Walk sequence (top row) and a UAV surveillance sequence (bottom)[7]

It is well known that the concept of using SR methods for remote sensing imaging has been developed for decades. There have been a few successful, useful examples of actual data, even if it is difficult to find data that meets the demand for SR. Figure 1.4 (a) and (b) indicate the LR and HR images, respectively. The first row shows the test on multi-temporal MODIS images with a magnification factor of 2. The second row is the SR example for multi-angle World View-2 images with a magnification factor of 2 [7].

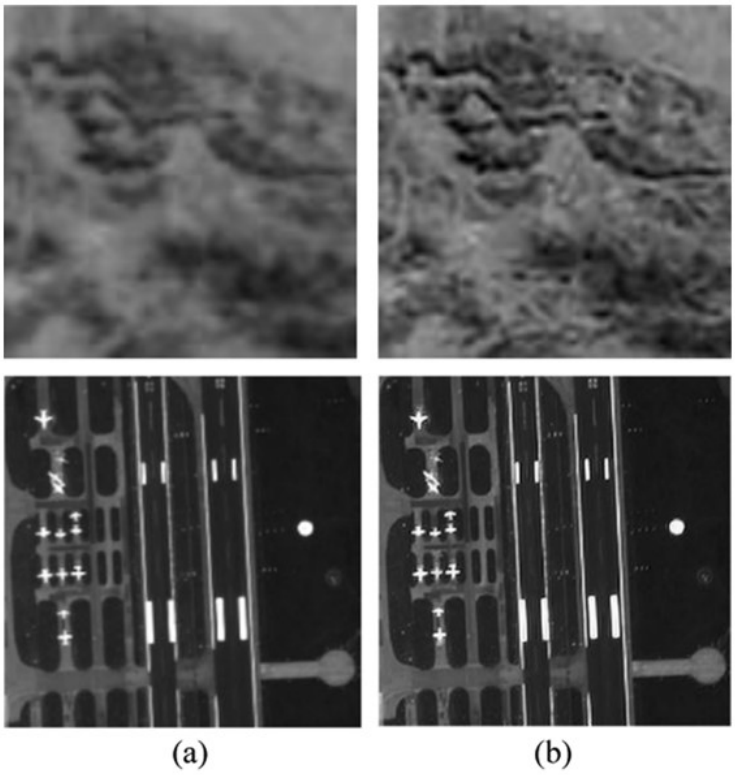


Figure 1.4: The SR reconstruction of remote sensing images[7]

1.1.3 Astronomical observation:

SR approaches can be used because the physical resolution of astronomical imaging systems is constrained by system parameters. Usually, astronomical systems can gather several photos for SR. SR can aid astronomers in their exploration of space by increasing the resolution of celestial photographs. Figure 1.5 provides (a) the original LR image and (b) the SR result of several star pictures. Additionally, satellites are currently being launched into space, such as the Mars Odyssey mission and the lunar exploration program. The SR can improve image resolution, which will make it easier to see minute items on the moon's surface. In addition, Hughes and Ramsey created an improved thermal infrared image of Mars' surface using Thermal Emission Imaging System (THEMIS) thermal infrared and visible datasets from various spectral areas [8].

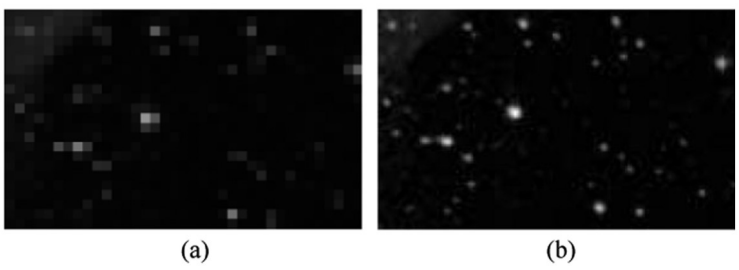


Figure 1.5: SR example of astronomical images[7]

1.2 Complication Observed

The challenge of single-image super-resolution (SR) can be likened to an inverse problem, where ambiguity arises due to multiple potential solutions for a given low-resolution image. Addressing this requires incorporating reliable prior knowledge to constrain the solution space. Furthermore, as the upscaling factor increases, so does the complexity of the problem. The growing number of variables makes the task of recovering lost scene details increasingly difficult, often leading to the introduction of inaccurate information. Additionally, assessing the quality of the output is not straightforward, as conventional quantitative metrics (e.g., Peak Signal to Noise Ratio (PSNR) and Structural Similarity Index Method (SSIM)) only provide a limited reflection of human perceptual experience.
Formally, considering the High-Resolution (HR) image as x and the Low-Resolution (LR) counterpart as y, the degradation process can be expressed as follows:

$$y = \Phi(x; \theta_\eta) \tag{1.1}$$

Here, Φ represents the degradation function with parameters θ_η, which encompass factors like scaling and noise. In real-world scenarios, only the LR image y is accessible, while the specifics of the degradation process and its parameters remain unknown. The aim of Super-Resolution is to mitigate the effects of degradation and reconstruct a faithful representation of the actual image x, as denoted by:

$$x = \Phi^{-1}(y, \theta_\zeta) \tag{1.2}$$

In this equation, θ_ζ signifies the parameters for the inverse function Φ^{-1}. The intricacies of the degradation process can be intricate and multifaceted, influenced by factors such as artifacts, compression, defocus blur, motion blur, and noise (sensor and speckle). As a result, a prevalent degradation model is often preferred, described by:

$$y = (x \otimes k) \downarrow_s + n \tag{1.3}$$

In this context, the variable "k" denotes the blurring kernel, "$x \otimes k$" represents the convolution between the HR image and the blur kernel, and "$\downarrow_s$" signifies the downsampling operation, reducing image resolution by a factor of "s". The variable "n" corresponds to additive white Gaussian noise (AWGN), characterized by a standard deviation of σ, which quantifies noise intensity [9].

1.3 Thesis Objective

This thesis's primary objective is to comprehensively examine Image Super Resolution (SR) techniques, with a specific emphasis on the comparative analysis of outcomes between conventional methods and learning-based approaches. The objective of this study is to comprehensively investigate the advantages and constraints associated with various techniques designed to enhance the understanding and use of image processing among professionals and scholars. This study aims to comprehensively analyse the benefits and various methodologies that may be employed to augment the knowledge and comprehension of image processing among practitioners and researchers. The objective of this study is to conduct a complete evaluation of the performance of seven distinct super-resolution (SR) techniques. The methods encompass a set of techniques, consisting of three conventional approaches, namely Nearest-neighbor, Bilinear, and Bicubic interpolation, alongside four learning-based methods, namely Super Resolution Convolutional Neural Network (SRCNN), Efficient Sub-Pixel Convolutional Network (ESPCN), Sub-pixel Upsampling, and Super Resolution Generative Adversarial Network (SRGAN). The evaluation will encompass quantitative measurements, such as Mean Square Error (MSE) and Peak Signal-to-Noise Ratio (PSNR), and qualitative assessments that consider perceptual quality and potential artefacts.

Thesis Goal This thesis seeks to enhance the existing knowledge in the field of image processing by conducting a comprehensive investigation into the various elements that influence super-resolution (SR) performance. The findings of this research endeavor will provide valuable insights for professionals, enabling them to make informed decisions when picking the most suitable SR approach that aligns with their individual use cases and requirements.

2. Key Concepts and Methods

This chapter provides a comprehensive understanding of Digital Images, Pixels, Reconstruction, and the topics mentioned in Fig 2.1. for Image Upscaling.

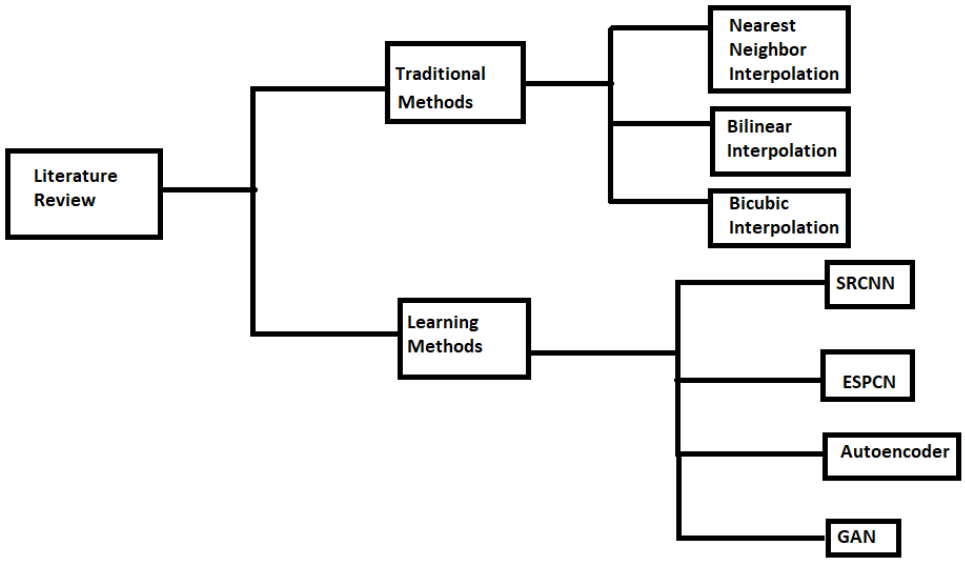


Figure 2.1: Literature Review

2.1 Image

Similar to traditional film cameras, digital cameras utilize optical lenses to capture light and concentrate it onto an image plane, resulting in the formation of an image. In contrast to traditional photographic film, digital cameras employ a solid-state sensor positioned in the image plane to capture and store visual information. Currently, two prevalent categories of solid-state sensors are frequently employed. The utilization of Charge-Coupled Device (CCD) arrays and complementary metal-oxide semiconductor Complementary Metal Oxide Semiconductor (CMOS) technology is prevalent in several academic and industrial applications. The ensuing discourse over the CCD case serves as a representative example for both scenarios.

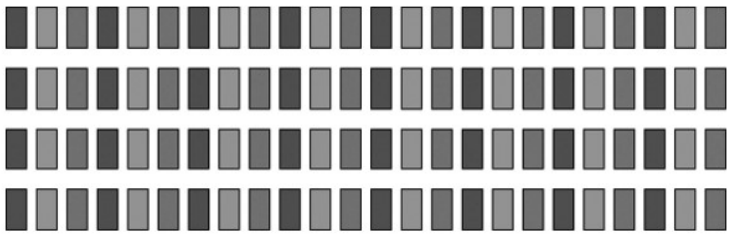


Figure 2.2: CCD array of a typical digital camera's sensor chip

Cameras equipped with charge-coupled device (CCD) arrays are known for their ability to provide images of superior clarity with minimal noise. In a charge-coupled device (CCD) array, the incident light on each individual region of the image, referred to as a photosite, results in the accumulation of electric charge. The quantity of charge is contingent upon the intensity of the illumination and the duration of the period over which the accumulation occurs. On average, the image consists of three photosites per pixel, specifically one for red, one for green, and one for blue (see Figure 2.2—plate). The photosites that are responsible for capturing red light are equipped with a filter layer that selectively permits the passage of red light while impeding the transmission of blue and green light. The green and blue photosites operate in a comparable fashion. After the image has been caught in the photosites, the electrical charge accumulated at each photosite is subsequently transferred to a nearby storage region. The storage region is shielded from light in order to minimize any fluctuations in the charge present. Subsequently, the accumulated charges originating from numerous storage locations can be transferred to an analog-to-digital (A/D) conversion circuit, commonly integrated on the same semiconductor chip as the array of photosites. The analog-to-digital (A/D) device generates numerical representations of the magnitudes of charges. Subsequently, the numerical data is transmitted from the sensor chip to a larger storage device, such as a Compact Flash memory card. Prior to being stored on a flash memory card, the image data has the potential to undergo digital enhancement within the camera and afterward undergo a reduction in size through the utilization of a technique such as JPEG compression, PNG, etc. [10].

2.1.1 Point Spread Function

The concept of image formation can be explained through the point spread function (PSF). The PSF characterizes how a point source of light generates a spread image in the spatial dimension. The present discussion pertains to the process of acquiring an image of a solitary point located at specific coordinates (x, y). In an ideal scenario characterized by a flawlessly focused imaging system devoid of stochastic disturbances, every photon originating from the point source will precisely converge onto the detector focal plane, so generating an image that is localized to a single point. Nevertheless, the representation of this particular source will not exhibit flawless replication, but rather a distorted rendition. The intensity at the central region is normally at its maximum, gradually decreasing as one moves away from the center, resulting in the formation of a Gaussian distribution. There are multiple causes that contribute to the phenomenon

of blurring, encompassing incorrect focusing, flaws in the lens, scattering of photons, and interactions with the detector array. The generated image is distinguished by its point spread function (PSF), which is defined as follows:

$$Ires(x, y) = Iid(x, y) * P(x, y) \tag{2.1}$$

Here, $*$ denotes the convolution operation, and I_{res} represents the resulting image when the input image I_{id} is convolved with the point spread function $P(x, y)$ at coordinates (x, y). The width of the PSF influences the nature of the resulting image.
With knowledge of the point spread function, image restoration through deconvolution becomes possible. Considering the analogy between convolution in the time domain and multiplication in the frequency domain, we have, in the Fourier Transform domain [11]:

$$F(I_{res}(x, y)) = F(I_{id}(x, y)) * F(P(x, y)) \tag{2.2}$$

The PSF need not be symmetrical, and its spread can differ along different directions.

2.1.2 Image Sampling and Quantization

Digital images are obtained in two fundamental ways: capture from the visual world and synthesis. The allocation of pixels to locations or regions in the original image is called sampling. After an image is digitized, each pixel will represent a sample of the image from a particular place. Sampling to create a digital image refers to the number and choice of locations in the scene or picture that correspond to pixels. Usually, the samples are arranged in a rectangular grid with even horizontal spacing and even vertical spacing. Both the spacing and the number of locations used are important. Also, the precise placement of the points can make a difference in some scenes.
The sampling period, according to Nyquist criterion, should be smaller than or at the most equal to half of the period of the finest detail present within an image. equal to half of the period of the finest detail present within an image. This implies that the sampling frequency along the x-axis $Wxs \geq [2w_x^L]$ and along the y-axis $Wys \geq [2w_y^L]$, where w_x^L and w_x^L are the limiting factors of sampling along x and y directions. Since we have chosen a sampling of $x\Delta$ along the X-axis and $y\Delta$ along the Y-axis, $x\Delta \leq \frac{\pi}{w_x^l}$ and $y\Delta \leq [\frac{\pi}{w_y^l}]$.The values of $x\Delta$ and $y\Delta$should be chosen in such a way that the image is sampled at Nyquist frequency. If $x\Delta$ and $y\Delta$ values are smaller, the image is called over-sampled, while if we choose large values of $x\Delta$ and $y\Delta$ the image will be under-sampled. If the image is over-sampled or exactly sampled, it is possible to reconstruct the band-limited image. If the image is under-sampled, then there will be spectral overlapping, which results in an aliasing effect.
If the original image is band-limited in Fourier domain, and if the sampling is performed at the Nyquist rate, then it is possible to reconstruct the original image using appropriate sample interpolation [11].
Conversion of the sampled analog pixel intensities to discrete-valued integer numbers is the

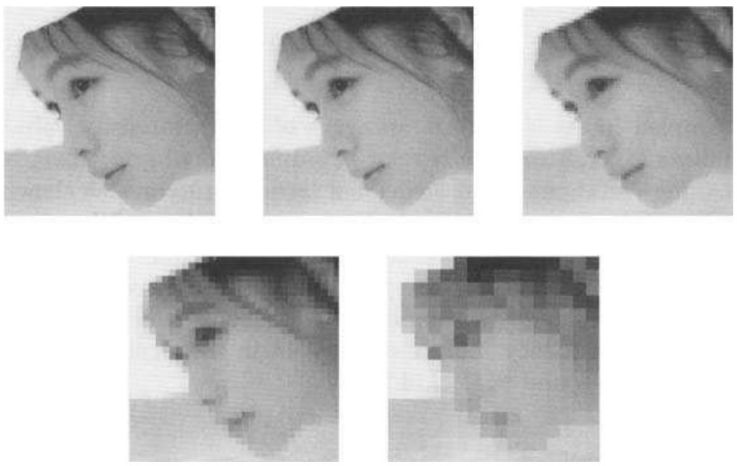


Figure 2.3: Images sampled at (256 x 256), (128 x 128) , (64 x 64), (32 x 32), and (16 x 16) rectangular sampling grids.

process of quantization. Quantization involves assigning a single value to each sample in such a way that the image reconstructed from the quantized sample values is of good quality and the error introduced because of quantization is small. The dynamic range of values that the samples of the image can assume is divided into a finite number of intervals, and each interval is assigned a single level. When the quantization levels are chosen equally spaced at equal intervals, it is known as uniform quantization. In many situations, however, the image samples assume values in a small range quite frequently and other values infrequently. In such a situation, it is preferable to use nonuniform quantization. The process of nonuniform quantization is implemented by the process of companding, in which each sample is first processed by a nonlinear compressor, then quantized uniformly and finally again processed by an expander before reconstruction of the original image.

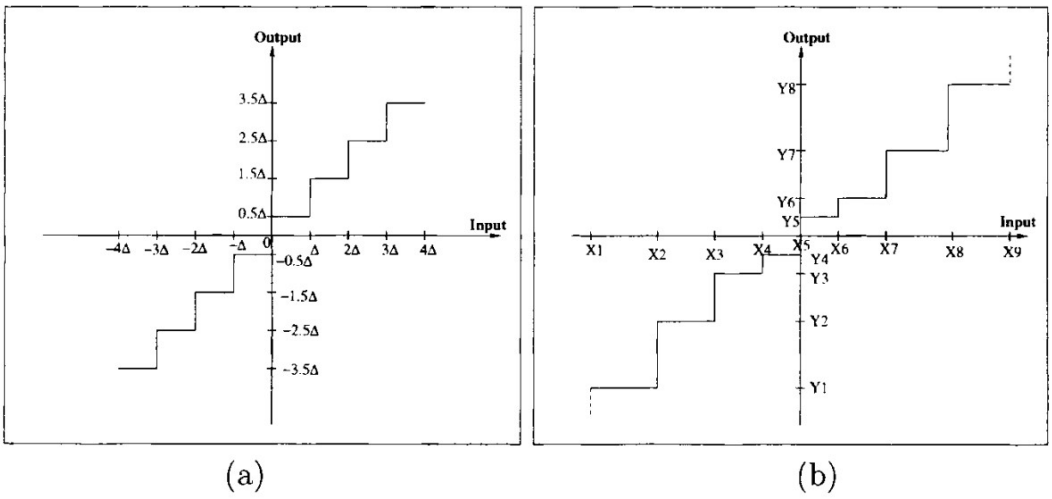


Figure 2.4: Two-dimensional (a) uniform quantization (b) nonuniform quantization

A color image, represented by red, green, and blue components, is quantized in individual color bands. When each color component is linearly quantized over a maximum range into 2n levels,

then each color sampled pixel is quantized in 3n bits, because it requires n bits for each color component.

2.2 Image Transformation

The study of two-dimensional image transforms holds significant importance within the field of image processing. The resulting image in the transformed space can be subjected to analysis, interpretation, and subsequent processing to facilitate the execution of various image processing tasks. The utilization of these transformations is increasing due to their ability to represent an image as a linear combination of fundamental signals, referred to as basis functions.

2.2.1 Fourier transform

When performing a Fourier transform on an image, the basis signals utilized are sinusoidal signals with varying durations that serve to represent the spatial frequencies present within the image. This suggests that a picture can be broken down into its individual sinusoidal components by utilizing the Fourier transform. The magnitudes of different frequencies within the image create the frequency spectrum. The inverse Fourier transform procedure entails the synthesis of an image by aggregating its constituent frequencies. The concept of frequency, particularly spatial frequency, is not solely a mathematical abstraction.
Fourier transform is one of the most important tools which have been extensively used not only for understanding the nature of an image and its formation but also for processing the image. The two-dimensional Fourier transform of a continuous function f(x, y) is denoted by

$$F(\omega,\psi)) = \oint \oint F(x,y)exp[-j2\pi(x\omega + y\psi)]dydx. \tag{2.3}$$

Using Euler's formula the exponential function can be decomposed as

$$exp[-j2\pi(x\omega + y\psi)] = cos(2\pi(x\omega + y\psi)) - jsin(2\pi(x\omega + y\psi)) \tag{2.4}$$

This implies for the function f (x, y) is essentially multiplied by the terms $cos(2\pi\omega x)cos(2\pi\psi y)$, $sin(2\pi\omega x)$ $sin(2\pi\psi y)$, $sin(2\pi\omega x)cos(2\pi\psi y)$, and $cos(2\pi\omega x)sin(2\pi\psi y)$. If the function f(x, y) is a doubly symmetric function along both the X and Y directions, then the Fourier transform of f(x,y) involves only the multiplication of the $cos(2\pi\omega x)cos(2\pi\psi y)$ term. The integral F(ω, ψ) thus yields the results of limit summation of an infinite number of sin and cos terms. The variable ω in indicates the frequency, i.e., the number of waves per unit length in X direction, and ψ indicates the number of waves along the Y direction [11].
The corresponding inverse two-dimensional Fourier transform is,

$$F(x,y)) = \oint \oint F(\omega,\psi)exp[j2\pi(x\omega + y\psi)]d\omega d\psi. \tag{2.5}$$

2.2.2 Discrete Fourier Transform

When the function or signal is represented in discrete form using a sequence of discrete samples such as f(z) = f(0), f(l), . . . , f(N- l), the corresponding Fourier Transform of the discrete signal is the Discrete Fourier Transform (DFT). Since the signal is discretized, the operation of integration in continuous Fourier transform (CFT) is replaced by summation operations in DFT.

The two-dimensional discrete Fourier transform of a two-dimensional signal f(z,y) of dimension M x N with integer indices x and y running from 0 to M-1 and 0 to N-1, is represented by

$$F(u,v)) = \frac{1}{MN}\sum\sum F(x,y)exp[-j2\pi(\frac{ux}{M}+\frac{vy}{N})] \tag{2.6}$$

The equivalent two-dimensional inverse DFT is

$$f(x,y)) = \sum\sum F(u,v)exp[j2\pi(\frac{ux}{M}+\frac{vy}{N})] \tag{2.7}$$

2.2.3 Matrix Form Representation

In case the kernel is observed to be symmetric and separable it may be expressed as

$$F = Af(x,y)A \tag{2.8}$$

where f(x,y) is an image matrix of dimension N x N and A is a symmetric transformation matrix of dimension N x N with elements = g(i,j). Multiplying both sides of Eq. 2.8 by a matrix B, we get

$$BFB = B[Af(x,y)A]B \tag{2.9}$$

To recover the original image f(z,y), we need to choose $B = A^{-1} = g^{-1}(z,j)$. In that case, Eq. 2.9 reduces to

$$BFB = B[Af(x,y)A]B = A^{-1}Af(z,y)AA^{-1} = f(x,y). \tag{2.10}$$

Thus the original image f(x, y) can be reconstructed.

2.2.4 Image Reconstruction

Both input and output Digital Images are restricted to lie on integer coordinates. In inverse mapping, the output pixels are passed through a mapping function that generates a sampling grid which is used to resample the input. Unfortunately, each point in the new resampling grid may take any continuous value. Therefore, it is necessary to interpolate a continuous surface through discrete input samples. This process is known as image reconstruction. The quality of the output image depends on the accuracy of the interpolated image which in turn depends

on the interpolation function [12]. This thesis focuses on a deep review on Image Interpolation using traditional methods and Deep Learning based methods.

Properties:

Following are the Image Transformation Properties.
Translation: The translation of a Fourier transform pair is

$$f(x, y) \exp(j2\pi(px + qy)) \rightleftharpoons F(u - p, v - q) \tag{2.11}$$

The above implication indicates the correspondence between a two-dimensional image function and its Fourier transform.

Rotation: Assuming that the function f(x, y) undergoes a rotation of a, the corresponding function f(x, y) in polar coordinates will then be represented as f(r,α), where z = rcosα and y = rsinα. The corresponding DFT F(u,w) in polar coordinates will be represented as F(β, γ), where u = $\beta cos\gamma$ and v = $\beta sin\gamma$. The above implies that if f(x, y) is rotated by α0, then F(u, v) will be rotated by the same angle α0 and hence we can imply that f(r, α + (α0) corresponds to F($\beta, \alpha + \alpha 0$) in the DFT domain and vice versa.

Separability: The separability property of a two-dimensional transform and its inverse ensures that such computations can be performed by decomposing the two-dimensional transforms into two one-dimensional transforms.

Scaling property: The DFT of a function f(z,y) multiplied by a scalar(k) is identical to the multiplication of the scalar with the DFT of the function f(z, y), i.e.,

$$fkf(x, y) = kF(u, v). \tag{2.12}$$

Convolution: The DFT of convolution of two functions is equal to the product of the DFT of these two functions, i.e.,

$$Ffl(x, y) * fl(x, y) = F1(u, v).F2(u, v). \tag{2.13}$$

Correlation: The DFT of the correlation of two functions fl (x, y) and f2(x, y) is the product of the complex conjugate of the DFT of the first function, and the DFT of the second function, i.e.,

$$F[f1(x, y) \cdot f1(x, y)] = F_1^*(u, v).F2(u.v). \tag{2.14}$$

Periodicity: The DFT of a two-dimensional function f(x,y) and its inverse are both periodic with period γ, i.e.,

$$F(u, v) = F(u + \gamma, v) = F(u, v + \gamma) = F(u + \gamma, v + -\gamma). \tag{2.15}$$

2.3 Traditional Methods:

Interpolation is the process of estimating approximate values from known points. Interpolation algorithms are divided into two types such as adaptive and non-adaptive techniques. Non-adaptive algorithms apply fixed pattern to each pixel without considering its other parameter as features, or edges of images. This type of interpolation technique may cause some artifacts such as the zipper effect in the interpolated image. These techniques are mainly classified into 3 categories as nearest neighborhood, bilinear, and bicubic image interpolation [13].
Adaptive algorithms employ spectral and spatial features available in the neighboring pixels in order to interpolate the unknown pixel as close to the original as possible. These algorithms offer better quality than low-complexity methods but it requires high silicon cost to implement hardware, which is undesirable [13].

2.3.1 Nearest Neighborhood Interpolation

It is the basic and simplest technique which decides the intensity value from the closest pixel to the specified input coordinates and assigns that value to the output coordinates [14]. For 1-D and 2-D, the number of grid points needed to evaluate the interpolation function is two and four respectively. It is a simple method of linear interpolation. It gives a poor quality image. The interpolation kernel of it for each direction is

$$u(x) = \begin{cases} 0, & \text{if } x > 0.5 \\ 1, & \text{if } x > 0.5 \end{cases} \tag{2.16}$$

Where x = distance between the interpolating point and grid point.

2.3.2 Bilinear image interpolation

Bilinear interpolation uses a weighted average of the 4 neighborhood pixels to calculate its final interpolated value. The result is a much smoother image than the original image. When all known pixel distances are equal, then the interpolated value is simply their sum divided by four. This technique gives better result than nearest-neighbor interpolation and takes less computation time compare to bicubic interpolation. Recently to reduce the artifacts generated by bilinear, S. L. Chen et al [15] proposed a system. It is consist of a clamp filter and a sharpening filter with bilinear interpolation. The interpolation kernel for bilinear interpolation is,

$$u(x) = \begin{cases} 0, & \text{if } |x| > 1 \\ 1 - |x|, & \text{if } |x| < 1 \end{cases} \tag{2.17}$$

Where x = distance between point to be interpolated and grid point.

2.3.3 Bicubic image interpolation

Bicubic interpolation is always chosen among all non-adaptive techniques. Bicubic interpolation uses a weighted average of 16 pixels to calculate its final interpolated value. These pixels are at various distances from the unknown pixel. Closer pixels are given a higher weighting in the calculation [14]. Bicubic gives sharper images than above mentioned two methods. This technique gives a better result but takes more computational time than the other two non-adaptive methods. The interpolation kernel for bicubic interpolation is [16],

$$u(x) = \begin{cases} (a+2)|x|^3 - (a+3)|x|^2 + 1, & \text{if } 0 <= |x| < 1 \\ a|x|^3 - 5a|x|^2 + 8a|x| - 4a, & \text{if } 1 <= |x| < 2 \\ 0, \end{cases} \tag{2.18}$$

Where x = distance between the interpolated point and grid point and a = free variable(for good effect (-1, 1/2))

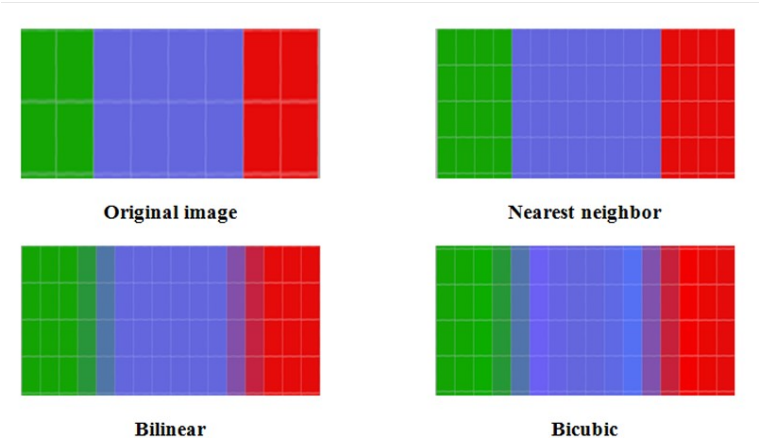


Figure 2.5: Comparison of non adaptive methods

Figure 2.5 compares the 3 algorithm's output compared to an original image. This figure gives a complete idea that the Bicubic Algorithm has introduced more details than the other 2 algorithms.

2.4 Deep Learning Based:

Inventors have long dreamed of creating machines that think. Ancient Greek myths tell of intelligent objects, such as animated statues of human beings and tables that arrive full of food and drink when called. The difficulties faced by systems relying on hard-coded knowledge suggest that AI systems need the ability to acquire their own knowledge, by extracting patterns from raw data. This capability is known as machine learning. A type of machine learning, a technique that allows computer systems to improve with experience and data is called deep learning [17].

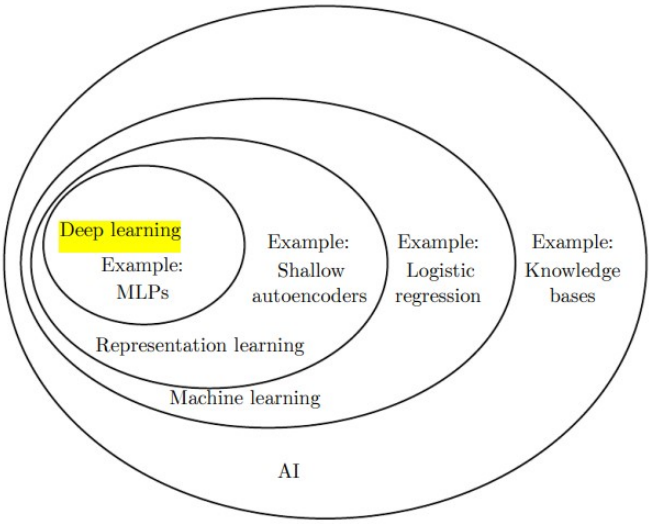


Figure 2.6: A Venn diagram showing how deep learning is a kind of representation learning [17]

Deep learning is advantageous if input data is in the form of images of pixel data, text data documents, or audio data files. Traditional machine learning has the supervised learning method, where the programmer has to specify the attributes of a target object. The machine learning model's performance is determined by how a developer effectively describes the features. This method is known as feature extraction. The feature extraction method in deep learning is automated and unsupervised.

Artificial Neural Networks (ANNs)

Artificial Neural Networks (ANNs) are computational processing systems that are heavily inspired by the way biological nervous systems (such as the human brain) operate. ANNs are mainly comprised of a high number of interconnected computational nodes (referred to as neurons), which work entwined in a distributed fashion to collectively learn from the input in order to optimize its final output. The basic structure of an ANN can be modeled as shown in Figure 2.7. We would load the input, usually in the form of a multidimensional vector to the input layer which will distribute it to the hidden layers. The hidden layers will then make decisions from the previous layer and weigh up how a stochastic change within itself detriments or improves the final output, and this is referred to as the process of learning. Having multiple hidden layers stacked upon each other is commonly called deep learning [18].

2.4.1 Components of Artificial Neural Network

Input Layers, Neurons, and Weights

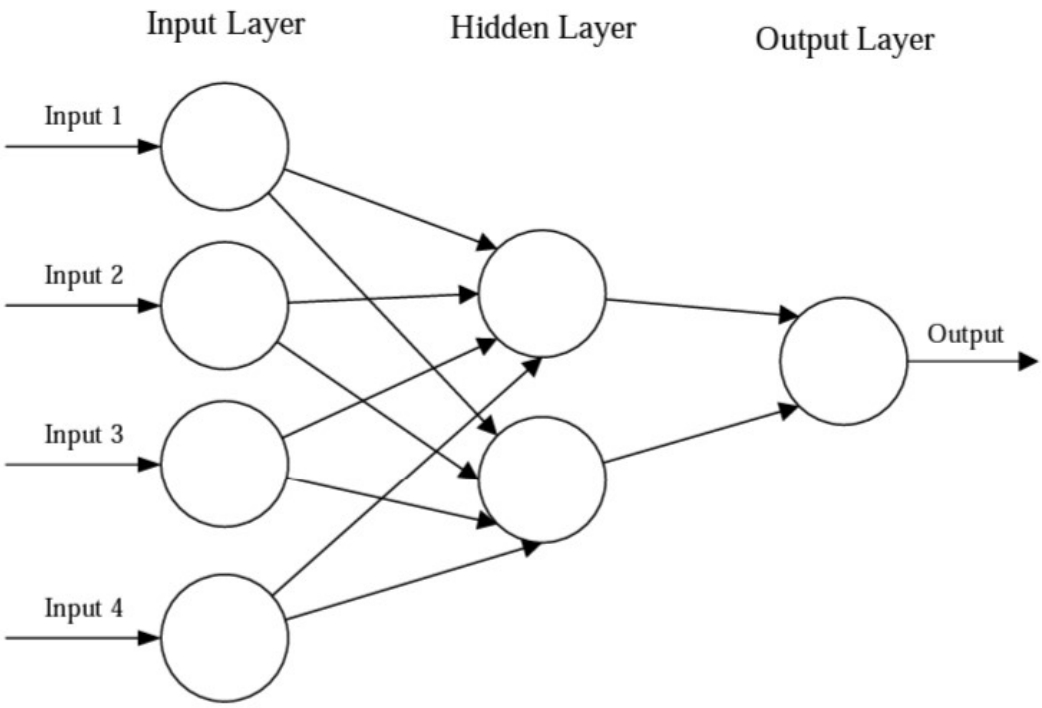


Figure 2.7: Neural Network[18]

In Figure 2.7 given above, the outermost layer is the input layer. A neuron is the basic unit of a neural network. They receive input from an external source or other nodes. Each node is connected with another node from the next layer, and each such connection has a particular weight. Weights are assigned to a neuron based on its relative importance against other inputs. When all the node values from the yellow layer are multiplied (along with their weight) and summarized, it generates a value for the first hidden layer. Based on the summarized value, the blue layer has a predefined "activation" function that determines whether or not this node will be "activated" and how "active" it will be [19].

Hidden Layers and Output Layer

The layer or layers hidden between the input and output layer is known as the hidden layer. It is called the hidden layer since it is always hidden from the external world. The main computation of a Neural Network takes place in the hidden layers. So, the hidden layer takes all the inputs from the input layer and performs the necessary calculation to generate a result. This result is then forwarded to the output layer so that the user can view the result of the computation.

Functions

Neural networks feature three primary functions: the Activation function, the Error function, and the summing function. The summation function ties the weights and neurons together to obtain their sum. Error function decreases error by modifying the weights of neurons because they are allocated randomly at the beginning of a neural network. Finally, an activation function is also known as a Transfer function. It is responsible for obtaining output from a neuron. The activation function might be linear or non-linear.

Flow of information in network

There are 2 ways information flows in the Neural Networks as given below.

Feedforward Networks : In this model, the signals only travel in one direction, towards the output layer. Feedforward Networks have an input layer and a single output layer with zero or multiple hidden layers. They are widely used in pattern recognition.

Feedback Networks : In this model, the recurrent or interactive networks use their internal state (memory) to process the sequence of inputs. In them, signals can travel in both directions through the loops (hidden layer/s) in the network. They are typically used in time-series and sequential tasks.

Dataset

In a Neural Network, the learning (or training) process is initiated by dividing the data into three different sets:

Training dataset : This dataset allows the Neural Network to understand the weights between nodes.

Validation dataset : This dataset is used for fine-tuning the performance of the Neural Network.

Test dataset : This dataset is used to determine the accuracy and margin of error of the Neural Network.

Once the data is segmented into these three parts, Neural Network algorithms are applied to them for training the Neural Network. The procedure used for facilitating the training process in a Neural Network is known as the optimization, and the algorithm used is called the optimizer. There are different types of optimization algorithms, each with their unique characteristics and aspects such as memory requirements, numerical precision, and processing speed.

Epochs

When an entire dataset is passed forward and backward through the neural network for one time is known as 1 epoch [20].

Batch Size

Since one epoch is too big for feeding the computer we divide them into batches or sets. A certain number of data will be provided with each epoch to train the network. This number is known as batch size.

Iterations

Iterations is the number of batches needed to complete one epoch.

Learning Rate

The rate of learning or speed at which the model learns is controlled by the hyperparameter. It regulates the amount of allocated error with which the model's weights are updated each time they are updated, such as at the end of each batch of training instances.

Underfit

Underfit neural network creates substantial losses in training and testing. Underfit model is inappropriate for any application and may be found by studying significant losses with the training dataset. An increase in training data and training time may assist prevent underfitting [20].

Balanced

The ideal model is somewhere between overfit and underfit. It can generalize the new unseen data. Optimum models relate to real-world situations [20].

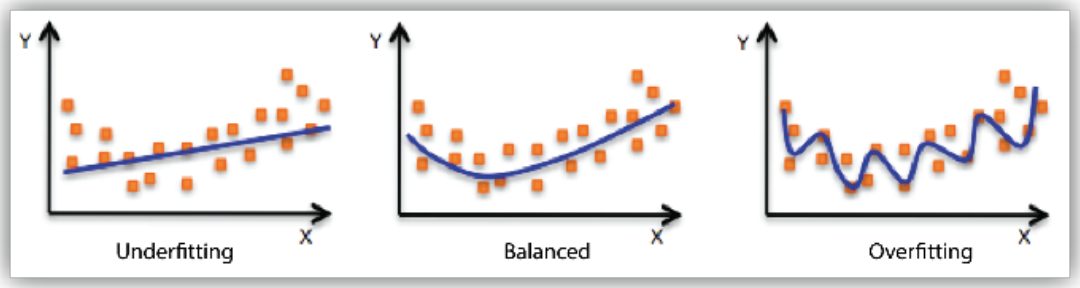


Figure 2.8: Overfit, Underfit, Optimum[21]

Overfit

Overfit refers to a model that mimics the training data quite well. Overfitting occurs when a model learns the detail and noise in the training data to the degree that it unfavorably impacts the model's performance on new unseen data. Overfitting can be avoided using techniques such as Data augmentation, increasing train data, Early stopping, feature selection, Regularization, and ensemble methods [20].

Dropout

The dropout approach overcomes the issue of overfitting in a deep neural network by randomly deleting units and their connections from the network during training. Consequently, this strategy may increase the performance of a neural network.

2.4.2 Convolutional Neural Network(CNNs)

Convolutional Neural Networks (CNNs) are analogous to traditional ANNs in that they are comprised of neurons that self-optimise through learning. Each neuron will still receive an input and perform an operation (such as a scalar product followed by a non-linear function) - the basis of countless ANNs. From the input raw image vectors to the final output of the class score, the entire of network will still express a single perceptive score function (the weight). The last layer will contain loss functions associated with the classes, and all of the regular tips and tricks developed for traditional ANNs still apply.
The only notable difference between CNNs and traditional ANNs is that CNNs are primarily used in the field of pattern recognition within images. This allows us to encode image-specific features into the architecture, making the network more suited for image-focused tasks - whilst further reducing the parameters required to set up the model. One of the largest limitations of traditional forms of ANN is that they tend to struggle with the computational complexity required to compute image data.

CNNs Architecture

CNNs are comprised of three types of layers. These are convolutional layers, pooling layers and fully-connected layers. When these layers are stacked, a CNN architecture has been formed [18].

Input layer

The Input layer holds the Pixel Values of the Image. It is an ANN layer.

Convolutional layer

The convolutional layer will determine the output of neurons of which are connected to local regions of the input through the calculation of the scalar product between their weights and

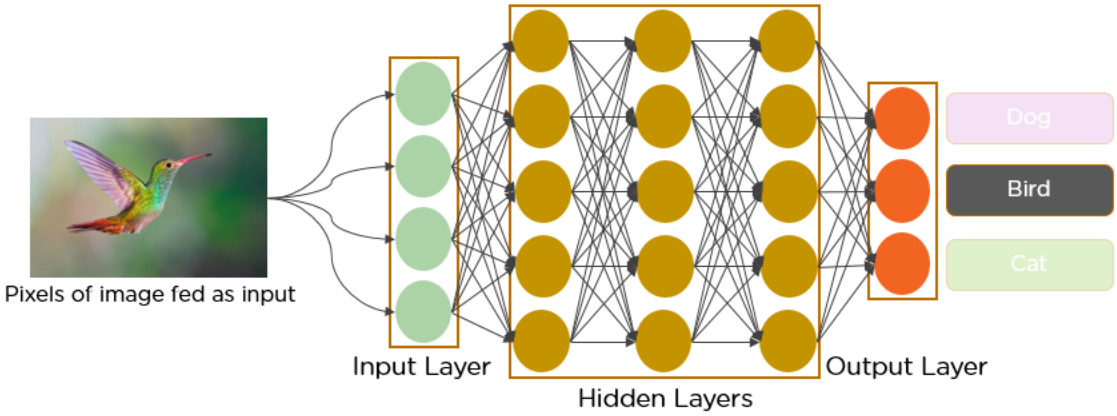


Figure 2.9: Convolutional Neural Network [22]

the region connected to the input volume. The rectified linear unit (commonly shortened to ReLu) aims to apply an 'elementwise' activation function such as sigmoid to the output of the activation produced by the previous layer.

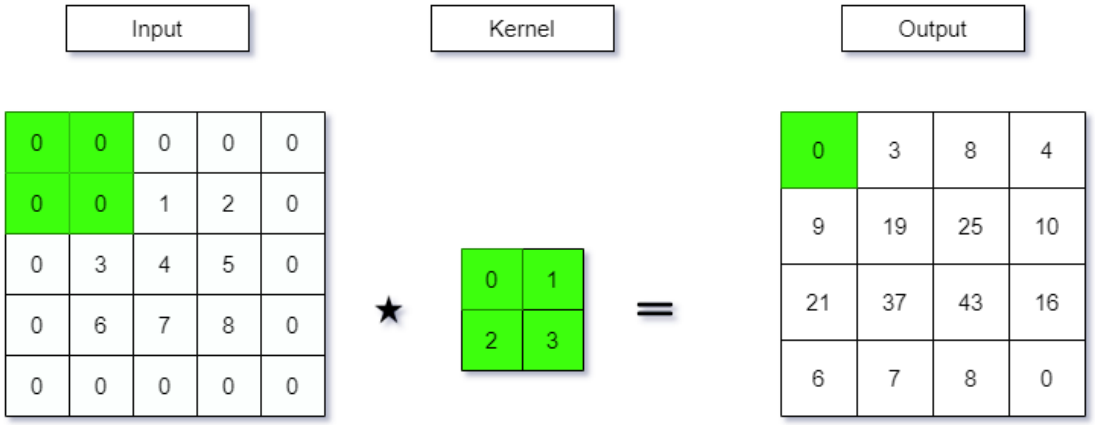


Figure 2.10: Convolution

Pooling Layer

Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data through dimensionality reduction. Furthermore, it is useful for extracting dominant features which are rotational and positional invariant, thus maintaining the process of effectively training of the model. There are two types of Pooling, Max Pooling as in fig 2.12 and Average Pooling. Pooling returns the maximum value from the portion of the image covered by the Kernel. On the other hand, Average Pooling returns the average of all the values from the portion of the image covered by the Kernel.

Max Pooling also performs as a Noise Suppressant. It discards the noisy activations altogether and also performs de-noising along with dimensionality reduction. On the other hand, the Average Pooling simply performs dimensionality reduction as a noise-suppressing mechanism. Hence, we can say that Max Pooling performs a lot better than Average Pooling.

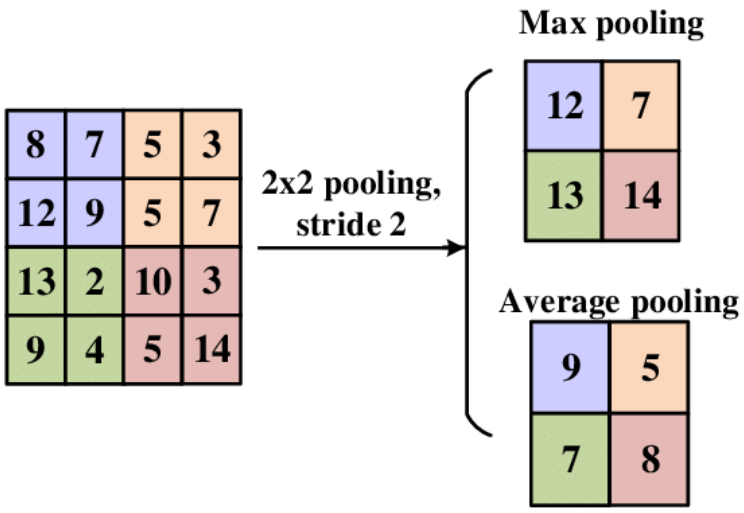


Figure 2.11: Pooling[23]

Fully-connected layers

The fully-connected layers will then perform the same duties found in standard ANNs and attempt to produce class scores from the activations, to be used for classification. It is also suggested that ReLu may be used between these layers, as to improve performance.

2.4.3 Autoencoder

Autoencoder is a type of neural network where the output layer has the same dimensionality as the input layer. In simpler words, the number of output units in the output layer is equal to the number of input units in the input layer. An autoencoder replicates the data from the input to the output in an unsupervised manner and is therefore sometimes referred to as a replicator neural network.[17].

Architecture of autoencoders

An autoencoder consists of three components shown in Figure 2.12:
1. Input Layer: The input layer consists of a feedforward, fully connected neural network that compresses the input into a latent space representation and encodes the input image as a compressed representation in a reduced dimension. The compressed image is the distorted version of the original image.
2. Hidden Layer: This part of the network contains the reduced representation of the input that is fed into the decoder.
3. Output Layer: The output layer is also a feedforward network like the encoder and has a similar structure to the encoder. This network is responsible for decoding and reconstructing the input back to the original dimensions from the hidden layer.
The main objective of the autoencoder is to get an output identical to the input. Note that the decoder architecture is the mirror image of the encoder. This is not a requirement but it's

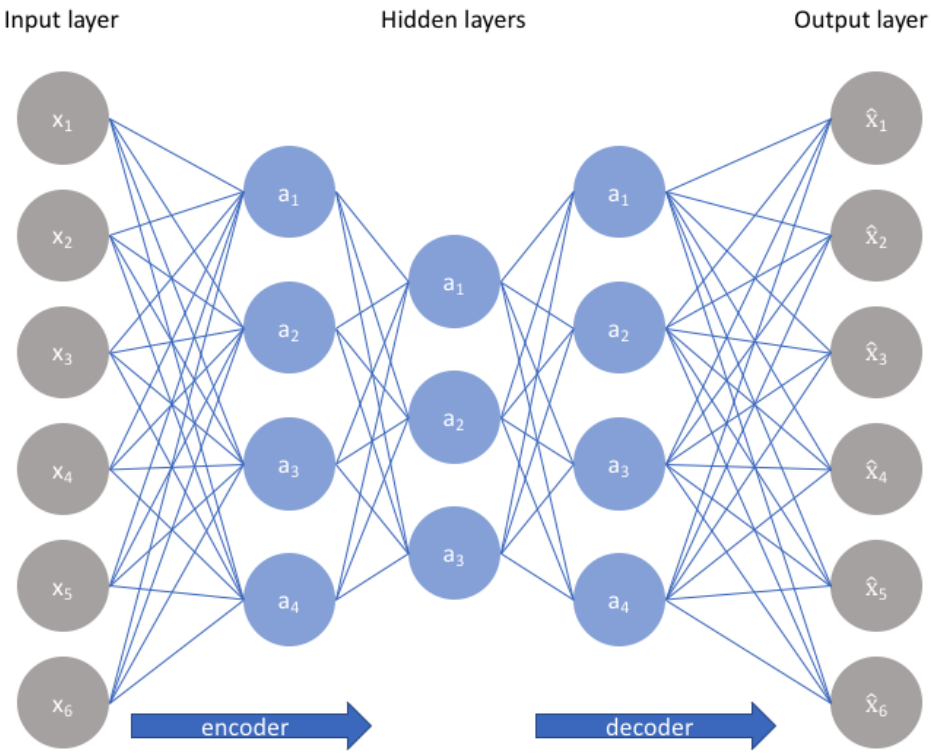


Figure 2.12: Autoencoder Architecture [24]

typically the case. The only requirement is the dimensionality of the input and output must be the same.

2.4.4 Generative Adversarial Network

Generative adversarial networks (GAN) represent a shift in architecture design for deep neural networks. This new architecture pits two or more neural networks against each other in 32 adversarial training to produce generative models. Also, it is an approach to generative modeling using deep learning methods, such as convolutional neural networks[25]. Generative modeling is an unsupervised learning task in machine learning that involves automatically discovering and learning the regularities or patterns in input data in such a way that the model can be used to generate or output new examples that plausibly could have been drawn from the original dataset. GANs are a clever way of training a generative model by framing the problem as a supervised learning problem with two sub-models: the generator model that we train to generate new examples, and the discriminator model that tries to classify examples as either real (from the domain) or fake (generated) as seen in Figure 2.13. The two models are trained together in a zero-sum game, adversarial, until the discriminator model is fooled about half the time, meaning the generator model is generating plausible examples. It can also be considered as a secondary field of ML algorithms inspired by the brain structure and functionality. In the applications of image identification, speech synthesis, and text mining applications by receiving a distinct kind of data hierarchical models can be built by representing probability distributions. Deep learning is dependent on an end-to-end wireless communication

system with conditional GANs using Deep Neural Networks (DNNs) to do functions of message passing like encoding, decoding, modulation, and demodulation. For this, the right judgment of the immediate channel transfer state is required to transfer DNN.

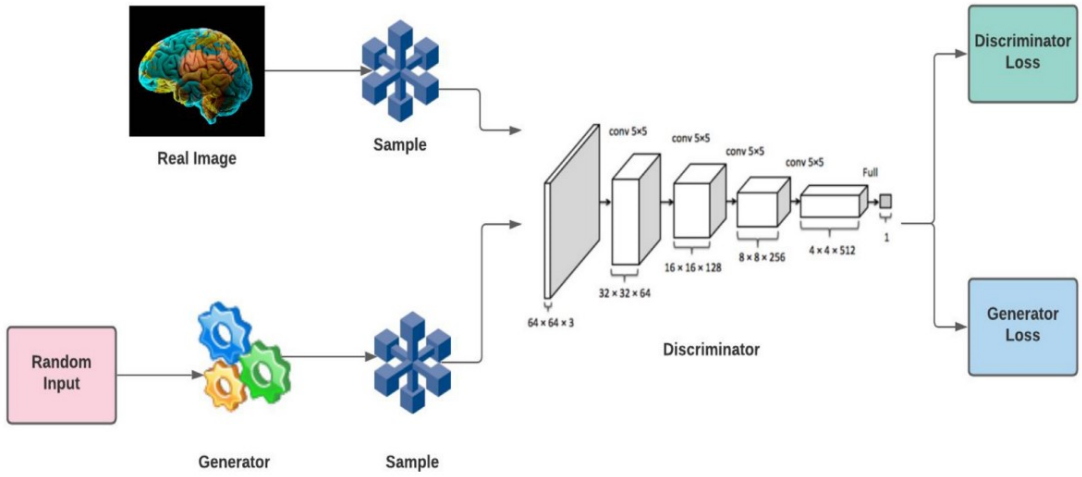


Figure 2.13: Block Diagram of GAN[25]

Generative and discriminative models Machine learning (ML) and deep learning can be described by two terms: generative and discriminative modeling. When discussing the machine learning techniques that most people are familiar with, the thinking of a discriminative modeling technique, such as classification. The GAN model architecture involves two sub-models: a generator model for generating new examples and a discriminator model for classifying whether generated examples are real, from the domain, or fake, generated by the generator model.

Generator:

Model that is used to generate new plausible examples from the problem domain.

Discriminator:

The model that is used to classify examples as real (from the domain) or fake (generated)

Benefits of Deep Learning

1. Features are automatically extracted, thus eliminating the need for domain expertise.
2. Robustness to natural variations in the data is automatically learned.
3. The same neural network-based approach can be applied to many different applications and data types.
4. The deep learning architecture is flexible to adapt to new real-world problems.
5. Works well with unstructured data.
6. It can execute many complex operations simultaneously.

Challenges and Limitations of Deep Learning

1. Requires a very large amount of labeled data for better performance.
2. Deep learning requires expensive GPUs and hundreds of machines to train deep learning models.
3. There is no standard theory in selecting the right deep learning tools as it requires knowledge of topology, training methods, and other parameters.
4. If a deep learning model trains on data that contains biases, the model will reproduce those biases in its predictions.

Transfer Learning & Pretrained Models

"Transfer learning is crucial in training deep neural networks on new target tasks. Current transfer learning methods always assume at least one of (i) source and target task label spaces overlap, (ii) source datasets are available, and (iii) target network architectures are consistent with source ones [26]." Transfer learning gives a tremendous edge versus real-world issues that do not have millions of labeled data points to train such complicated algorithms. The goal of transfer learning is to employ a model previously trained on an extensive and general enough dataset that can serve as the generic model of the visual world. That offers an advantage when a dataset is not vast, and it is feasible to employ learned feature maps without constructing the method.

In Machine Learning, a pre-trained model falls under the category of transfer learning. Pre Trained models are machine learning models that are trained, developed and made available by other developers. They are generally used to solve problems based on deep learning and are always trained on a very large dataset.

2.5 Methods of Assessment

2.5.1 Subjective analysis

In subjective testing a group of people are asked to give their opinion about the quality of each image. There are several types of subjective methods like the following [27]:

Single stimulus categorical rating

In this method, test images are displayed on a screen for a fixed amount of time, after that, they will disappear from the screen and observers will be asked to rate the quality of them on an abstract scale containing one of the five categories: excellent, good, fair, poor, or bad.

Double stimulus categorical rating

This method is similar to single stimulus method. However, in this method both the test and reference images are being displayed for a fixed amount of time. After that, images will disappear from the screen and observers will be asked to rate the quality of the test image according to the abstract scale described earlier.

Ordering by force-choice pair-wise comparison

two images of the same scene are being displayed for observers. Afterward, they are asked to choose the image with higher quality. Observers are always required to choose one image even if both images possess no difference. There is no time limit for observers to make the decision. The drawback of this approach is that it requires more trials to compare each pair of conditions

Pair-wise similarity judgment

In pair-wise similarity judgment observers are asked not only to choose the image with higher quality but also to indicate the level of difference between them on a continuous scale. One might be tempted to use the raw rating results such as: excellent, good, fair, etc. for quality scores. However, these rating results are unreliable. One reason for this is that observers are likely to assign different quality scales to each scene and even distortion types.

Mean opinion score (MOS)

MOS is obtained by asking observers to rate images for their quality on a particular scale. such as a scale from 1 to 5 where 1 is bad and 5 is excellent.

2.5.2 Objective methods

Objective methods depend on mathematical model to measure not human testers. There are several types of objective methods like the following [27]:

Mean Square Error (MSE)

The difference between the Pixel value of one image and the corresponding Pixel value of the other image. The MSE measures the average of the square of the errors

$$MSE = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}[I(i,j) - K(i,j)]^2 \tag{2.19}$$

Peak Signal to Noise Ratio(PSNR)

PSNR is used to calculate the ratio between the maximum possible signal power and the power of the distorting noise this ratio is used as a quality measurement between the original and

a compressed image. The higher the PSNR, the better the quality of the compressed, or reconstructed image.

$$PSNR = 10 \cdot \log_{10} \left(\frac{MAX^2}{MSE} \right) \tag{2.20}$$

Structure Similarity Index Method (SSIM)

It measures the similarity between two images in 3 aspects (luminance, structure, contrast) SSIM can be expressed through these three terms as:
L is the luminance (used to compare the brightness between two images)
C is the contrast (used to differ the ranges between the brightest and darkest region of two images)
S is the structure (used to compare the local luminance pattern between two images to find the similarity of the images), and α, β, and γ are the positive constants.

$$SSIM(x, y) = [l(x.y)]^{\alpha}.[c(x.y)^{\beta}.s[(x.y)]^{\gamma}. \tag{2.21}$$

Universal Image Quality(UQI)

It is a reference-based image comparison technique. This method quantifies the image quality by comparing it to a reference image(in our case original image). This method is known to be Universal Image Quality Index (UQI).

$$Q = \frac{\sigma_x \cdot \sigma_y}{\sigma_x \cdot \sigma_y} \cdot \frac{2(\overline{x} \cdot \overline{y})}{\overline{x}^2 + \overline{y}^2} \cdot \frac{2\sigma_x \cdot \sigma_y}{\sigma_x^2 + \sigma_y^2} \tag{2.22}$$

"The first component is the correlation coefficient between x and y, which measures the degree of linear correlation between x and y., and its dynamic range is [-1,1]. The best value 1 is obtained when y_i = ax_i+b for all i = 1,2,..., N, where a and b are constants and a $>$ 0. Even if x and y are linearly related, there still might be relative distortions between them, which are evaluated in the second and third components [28]."
"The second component, with a value range of $[0, 1]$, measures how close the mean luminance is between x and y. It equals 1 if and only if $\overline{x} = \overline{y}$. [$\sigma_x$ and σ_y] can be viewed as estimates of the contrast of x and y [28]."
"So the third component measures how similar the contrasts of the images are. Its range of values is also[0,1], where the best value of 1 is achieved if and only if σ_x=σ_y [28]."
If we use a slider approach using a sliding matrix of A x A over the image with M number of steps then the quality index is,

$$Q = \frac{1}{m} \sum_{j=1}^{m} Q_j \tag{2.23}$$

2.6 Model Description

This ´section will discuss the models given in Figure 2.14

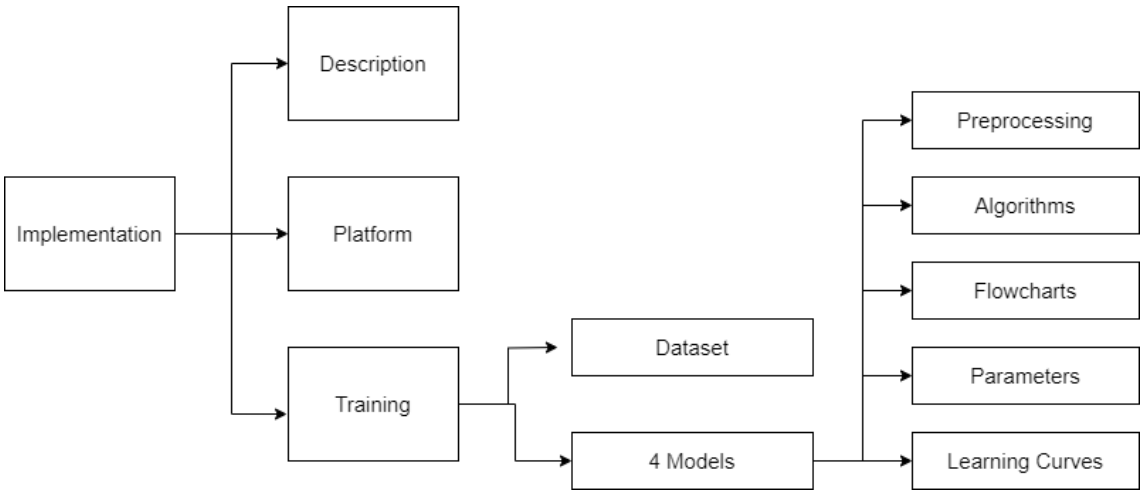


Figure 2.14: Implementation Block Diagram

2.6.1 Datasets, Training and Testing

In deep learning data in our case image data is divided into 3 parts, as given below:

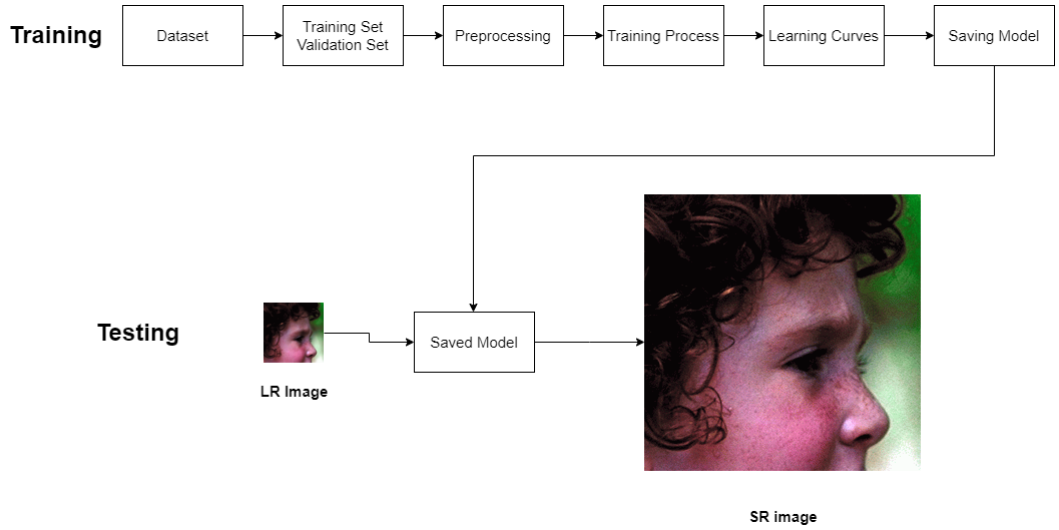


Figure 2.15: Training and Testing Block Diagram

Training Dataset:

This dataset is used to train the model. Normally for super-resolution tasks, we use a pair of low-resolution(LR) and high-resolution(HR) images. The network is fed such data and trained to learn the relation between LR to HR.

The dataset utilized is DIV2K [29], a widely recognized single-image super-resolution dataset. This dataset comprises 1000 high-quality RGB images, each featuring various scenes and sizes

in PNG format. Originally created for the NTIRE2017 and NTIRE2018 Super-Resolution Challenges, its primary aim was to stimulate research in the field of image super-resolution, particularly focusing on more authentic degradations.

Within this dataset, you can find low-resolution images that have been subjected to various types of degradation. In addition to the standard bicubic downsampling, several degradation methods were employed to generate low-resolution images, tailored to the specific challenges' tracks.

For instance, in the NTIRE 2017 Track 1, low-resolution images were created with an unknown x4 downscaling factor. In the NTIRE 2018, both Track 2 and Track 4 represent scenarios with realistic mild $\times 4$ and realistic wild $\times 4$ adverse conditions, respectively. Under the realistic mild $\times 4$ setting, the low-resolution images exhibit issues like motion blur, Poisson noise, and pixel shifting. Meanwhile, the realistic wild $\times 4$ setting introduces varying levels of degradation from one image to another.

Validation Dataset:

When fine-tuning the performance of the Neural Network, we deviated slightly from the approach taken by Agustsson and Timofte in their work on the NTIRE 2017 Challenge on single-image super-resolution. In their study, Timofte et al.[29], they employed a classical setup of using 800 images for training, 100 images for validation, and the remaining 100 images for testing. In contrast, our approach involved using 800 images for training and 200 images for validation during the training process.

Testing Dataset:

Once the model is trained and saved, then it is tested using LR images. We call the model and try to predict the super-resolution(SR) image of the LR image. We compare this SR image with the corresponding HR image and observe the loss and accuracy of the model. We used the well known Set5 [30] and Set14 [31] image datasets.

Set5 and Set14 consist of five different dataset variations, including GTmod12, LRbicx2, LRbicx3, LRbicx4, and the original dataset. These datasets come with pre-downscaled images using a well-defined downscale operator, which simplifies our processing task."The bicubic downscaling(imresize from Matlab)is the most used degradation operator to simulate the HR to LR transformation[29]."

2.6.2 Image Pre-Processing

Image pre-processing is essential before training a machine learning model for several reasons:
Normalization: Scaling pixel values for faster convergence.
Noise Reduction: Removing noise and artifacts for cleaner data.
Data Augmentation: Creating variations to improve generalization.

Standardization: Ensuring consistent data scales.
Resizing: Making images uniform in size.
Feature Extraction: Extracting useful features.
Handling Missing Data: Managing missing or corrupted data.
Color Space: Converting to a common color space.

These steps enhance data quality, model performance, and robustness.

Super Resolution using deep convolution neural network

Super Resolution using deep convolution neural network was the first deep learning model to outperform traditional ones. This neural network consists of 3 convolutional layers shown in Figure 2.16: Patch extraction and Representation, NOn-linear mapping, and reconstruction[32].

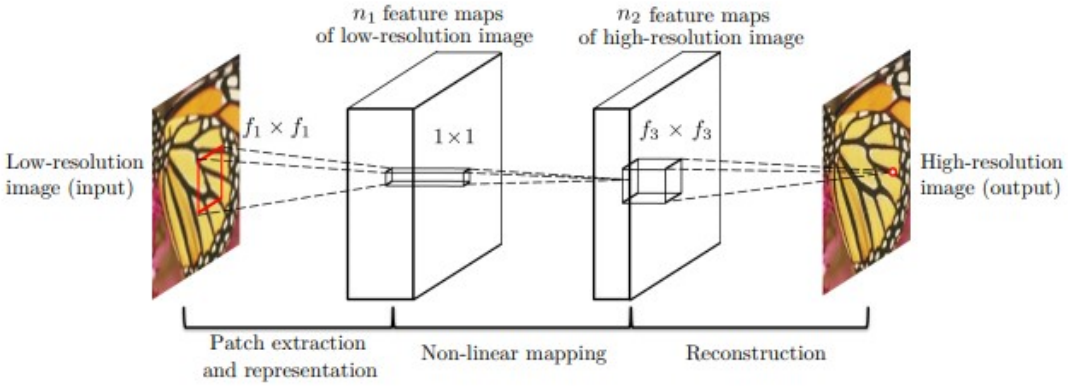


Figure 2.16: SRCNN Architecture[32]

As seen in Figure 2.16 the input image looks the same size as the output image. This input image is pre-processed using a bicubic interpolation. SRCNN aim to upscale LR images to HR dimensions. Having the same size allows the model to learn how to upscale by a specific factor (e.g., 2x or 3x) because it can directly compare the LR input to the HR ground truth.Our goal is to recover from Y an image F(Y) which is as similar as possible to the ground truth high-resolution image X [32]. Before inputting an image into the neural network, it undergoes up-sampling through bicubic interpolation. Subsequently, the image is converted into the YCbCr color space, with only the luminance channel (Y) being utilized by the network. The network's output is then combined with interpolated CbCr channels to generate the final color image. This approach was selected because the objective was not to alter colors (which are stored in the CbCr channels) but to exclusively modify brightness (the Y channel). Moreover, this choice aligns with human vision, as human perception is more sensitive to variations in luminance (i.e., "black and white") than differences in color.

1. Patch Extraction and Representation: In this step, we extract overlapping patches from the low-resolution image Y and represent each patch as a high-dimensional vector. These

vectors form a collection of feature maps, with the number of maps matching the vector's dimensionality.

2. Non-linear Mapping: In this process, each high-dimensional vector is nonlinearly transformed into another high-dimensional vector. Each transformed vector conceptually represents a high-resolution patch. These vectors form a separate set of feature maps.

3. Reconstruction: In this phase, we combine the high-resolution patch-wise representations obtained earlier to create the final high-resolution image. This resulting image is expected to closely resemble the ground truth X.

The final network was trained on DIV2K dataset and implemented the model using the TensorFlow library. The dataset consisted of 1000 images, which were split into an 80-20 ratio for training and validation, with 80% used for training and 20% for validation.

The model was configured with the following parameters:

- Learning Rate: 10^{-4} for all layers.
- Optimizer: Adam optimizer.
- Loss Function: Mean Squared Error (MSE).

MSE was employed to measure the disparity between the generated Super-Resolution (SR) images and the ground truth High-Resolution (HR) images.

Efficient Subpixel Convolution Network

Approaches based on convolutional neural networks, such as SRCNN, come with certain limitations. Firstly, CNN approaches necessitate the use of interpolation methods for up-sampling low-resolution (LR) images. Secondly, CNN methods raise the image resolution either at or before the first layer of the network, leading to increased computational complexity and memory requirements.

A novel approach known as ESPCN (Efficient Subpixel Convolution Network) [33] has been introduced to address the aforementioned issues. ESPCN introduces an efficient sub-pixel convolutional layer to the CNN network, eliminating the need for interpolation methods. The network enhances the image resolution at the very end of the network architecture, which offers advantages. This approach allows the network to learn a superior LR to HR (High-Resolution) mapping compared to pre-processing with interpolation filters before input. Additionally, due to the reduced input image size, smaller filter sizes can be employed for feature extraction. This, in turn, reduces computational complexity and memory usage, greatly enhancing efficiency.

In Figure 2.17 we can see 2 Convolutional layers n1 and n2 both are used to extract feature maps and a sub-pixel convolutional layer that aggregates the feature maps from LR space and builds SR image in a single step[33]. The final network was trained on DIV2K dataset and implemented the model using the TensorFlow library. The dataset consisted of 1000 images,

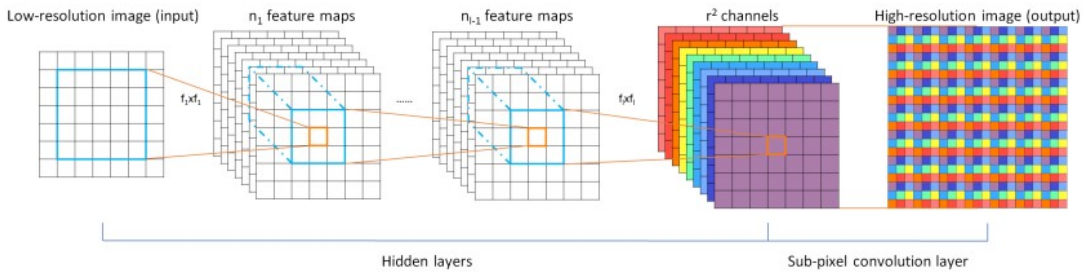


Figure 2.17: Efficient sub-pixel convolutional neural network (ESPCN)[33]

which were split into an 80-20 ratio for training and validation, with 80% used for training and 20% for validation.

The model was configured with the following parameters:

- Learning Rate: 10^{-4} for all layers.
- Optimizer: Adam optimizer.
- Loss Function: Mean Squared Error (MSE).

MSE was employed to measure the disparity between the generated Super-Resolution (SR) images and the ground truth High-Resolution (HR) images.

Autoencoder for Super Resolution

Autoencoders were constructed by incorporating various convolutional neural network layers. Each layer of the neural network was utilized to process the input image at different stages and matrices to remove noise from the image. The model was subsequently trained using a dataset consisting of low-resolution images and their corresponding high-resolution counterparts, allowing it to learn and extract image features within a custom-designed encoder. The trained autoencoder's weights, which represent the encoded images, were saved for later use.

When evaluating the model's performance, a test input image of lower quality was employed. The pre-trained autoencoder model was utilized to enhance the quality of this input image. Below, in Figure 2.18 we highlight the features employed in the model, providing a comprehensive understanding of its functionality and architectural flow [34].

The autoencoder utilized unsupervised learning techniques to denoise the input image. It executed convolution and deconvolution operations on the input image using identical input and output matrices.
The final network was trained on DIV2K dataset and implemented the model using the TensorFlow library. The dataset consisted of 1000 images, which were split into an 80-20 ratio for training and validation, with 80% used for training and 20% for validation.

The model was configured with the following parameters:

- Learning Rate: 10^{-4} for all layers.

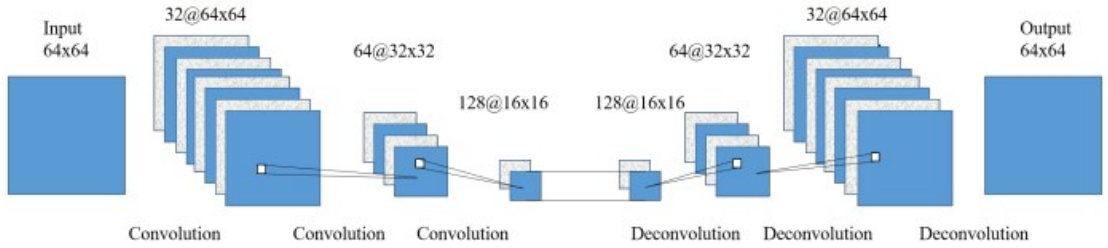


Figure 2.18: Autoencoder Model Architecture[34]

- Optimizer: adadelta optimizer.
- Loss Function: Mean Squared Error (MSE).

MSE was employed to measure the disparity between the generated Super-Resolution (SR) images and the ground truth High-Resolution (HR) images.

Super Resolution GAN

SRGAN (Super-Resolution Generative Adversarial Network) is a deep learning model designed for enhancing the resolution of single images. It employs a perceptual loss function comprising an adversarial loss and a content loss. The adversarial loss guides the model towards generating images that resemble natural photos by leveraging a discriminator network trained to distinguish between super-resolved images and genuine high-quality images. Additionally, SRGAN utilizes a content loss that emphasizes perceptual similarity over pixel-wise similarity, contrasting with conventional super-resolution methods that primarily rely on mean-squared error (MSE) and peak signal-to-noise ratio (PSNR) as optimization targets, which often yield high PSNR values but visually unsatisfying results.

Perceptual Loss

SRGAN replaces the MSE-based content loss with a loss computed on feature maps extracted from the VGG network. These feature maps are more robust to variations in pixel space. Extensive evaluation through mean opinion score (MOS) testing on images from three publicly available benchmark datasets confirms SRGAN's superior performance in generating photo-realistic high-resolution images with significant upscaling factors ($4\times$).

Architecture

The architecture of SRGAN comprises two primary components: the generator and the discriminator. During training, a low-resolution image (LR) is created by applying a Gaussian filter to a high-resolution image (HR) and subsequently down-sampling it by a factor of 'r.' The generator function 'G' is tasked with estimating the corresponding high-resolution image

(SR) when provided with an LR input. The discriminator 'D' is trained to distinguish between super-resolved images and real images.

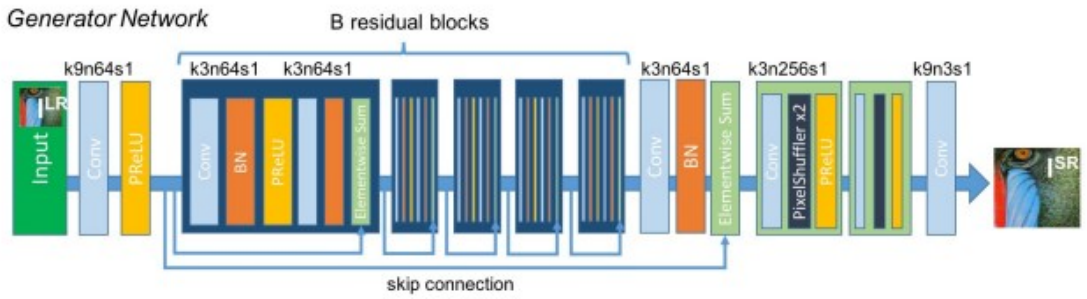


Figure 2.19: Generator Architecture[35]

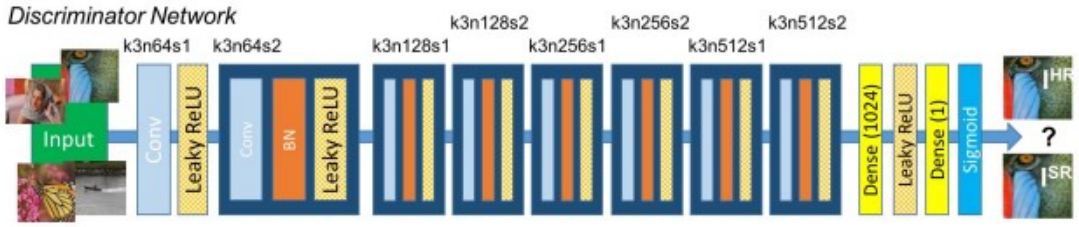


Figure 2.20: Discriminator Architecture[35]

The final network was trained on DIV2K dataset and implemented the model using the TensorFlow library. The dataset consisted of 1000 images, which were split into an 80-20 ratio for training and validation, with 80% used for training and 20% for validation.

The model was configured with the following parameters:

- Learning Rate: 10^{-4} for all layers.
- Optimizer: Adam optimizer.
- Loss Function: Mean Squared Error (MSE).

MSE was employed to measure the disparity between the generated Super-Resolution (SR) images and the ground truth High-Resolution (HR) images.

3. Implementation & Learning Curves

The process of putting theoretical concepts into practice and understanding the trajectory of learning over time. This topic likely delves into the practical aspects of applying knowledge or technology, highlighting the challenges, milestones, and growth experienced along the way. In this section, we briefly outline the training methodologies for the 4 models we've put into action, offering clarity through the presentation of flowcharts, model structures, learning curves, and algorithms.

3.1 Parameters Definitions

Already discussed in Chapter 2. "Key Concepts and Methods"

- **Training Set**: is the dataset from which the algorithm gathers knowledge.
- **Validation Set**: is the dataset employed to gauge the effectiveness of the model's training. Its reliability hinges on various factors such as dataset size, the target prediction, and input characteristics.
- **Epoch**: An epoch represents a full iteration through the training dataset for the learning algorithm.
- **Batch Size**: It signifies the number of training examples used in a single iteration.
- **Optimizer**: This is a tool used to minimize or maximize the loss function.
- **Loss Function**: It quantifies the dissimilarity between the predicted value ($\hat{y}$) and the actual label (y).
- **LR (Learning Rate)**: This value determines the step size used during training. A very low learning rate prolongs training, while a very high rate can lead to suboptimal results or divergence.

3.2 SRCNN

The implementation of Super-Resolution using Deep Convolution Neural Network (SRCNN) has been carried out using two distinct scales, namely 2x and 4x. The model has been trained using the DIV2K dataset and evaluated using an 80-20 cross-validation approach. The model's hyperparameters are presented in Table 3.1, with a total parameter count of 85,889, representing the combined weights and biases within the neural network.

Parameters	SRCNN
Optimizer	Adam
Learning Rate	$3 \cdot 10^{-3}$
Activation Function	ReLU
Loss Function	MSE
Number of Epochs	85
Batch Size	10

Table 3.1: SRCNN Parameters

3.2.1 SRCNN Model Structure

1.The model starts with a Sequential container.
2.The first layer is a Conv2D layer with 128 filters, a kernel size of $(9, 9)$,
3.ReLU activation function, and 'same' padding. It takes input images of shape $(512, 512, 1)$ and learns 128 feature maps.
4.The second layer is another Conv2D layer with 64 filters, a kernel size of $(3, 3)$, ReLU activation, and 'same' padding.
5.The third and final layer is a Conv2D layer with 1 filter, a kernel size of $(5, 5)$, linear activation function, and 'same' padding. This layer produces the super-resolved image.

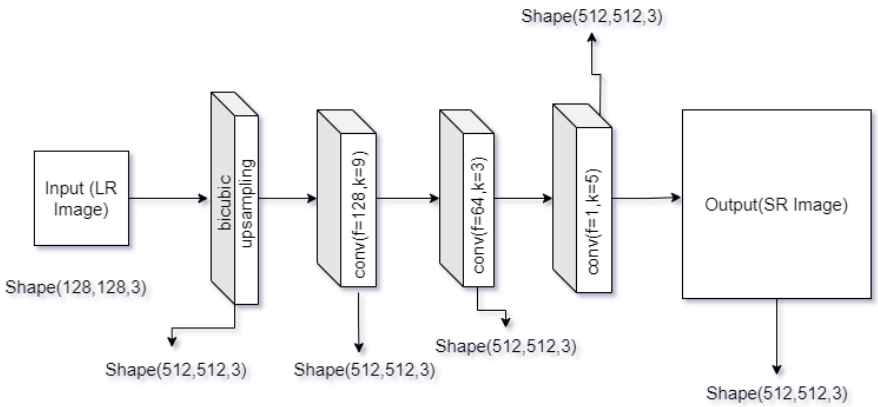


Figure 3.1: SRCNN Model Structure

3.2.2 Algorithm:

1.Input: Begin with a low-resolution input image of any size, such as $(128, 128, 3)$.
2.Upsampling Layer: Apply bicubic upsampling to the low-resolution image, resulting in a size of $(512, 512, 1)$.
3.Convolution Layer 1: Next, employ a convolutional layer with a 9×9 kernel to process the image, resulting in a shape of $(512, 512, 128)$.
4.Convolution Layer 2: Subsequently, apply another convolutional layer with a 3×3 kernel, resulting in a shape of $(512, 512, 64)$.
5.Convolution Layer 3: For the final step, utilize a 5×5 kernel to produce the output image, which is the super-resolution image, with a shape of $(512, 512, 1)$.
6.Output: The output image, representing super-resolution, has dimensions $(512, 512, 3)$.

3.2.3 SRCNN Experiment details

The model takes the low and high-resolution data from the dataset, then resize the images with the desired size and converts bgr to YCrCb, and gets the first channel Y. Then take Y channels float and divides by 255. Last but not least changing images to tensors and starting training the model depending on the model structure that was described in Figure 3.1 and the algorithm before.

3.2.4 Creating High-Low Resolution Images for Model Training Flowchart

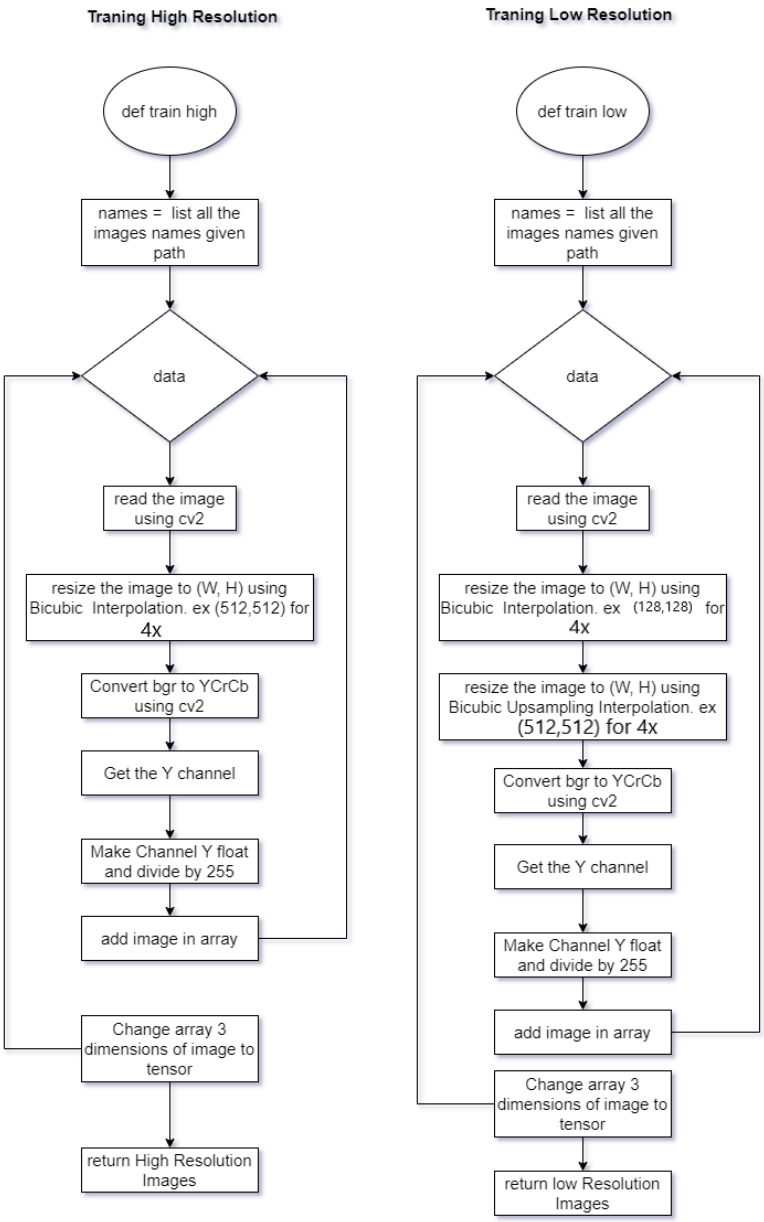


Figure 3.2: High and Low-Resolution Image preprocessing for SRCNN

3.2.5 Model Training & Testing Flowchart

The figure presented in Figure 3.2 depicts a flowchart illustrating the process of Model Training and Testing. This diagram serves to provide a clear understanding of the overall execution of the model. Figure 3.3 depicts the process of preparing High and Low-Resolution Images.

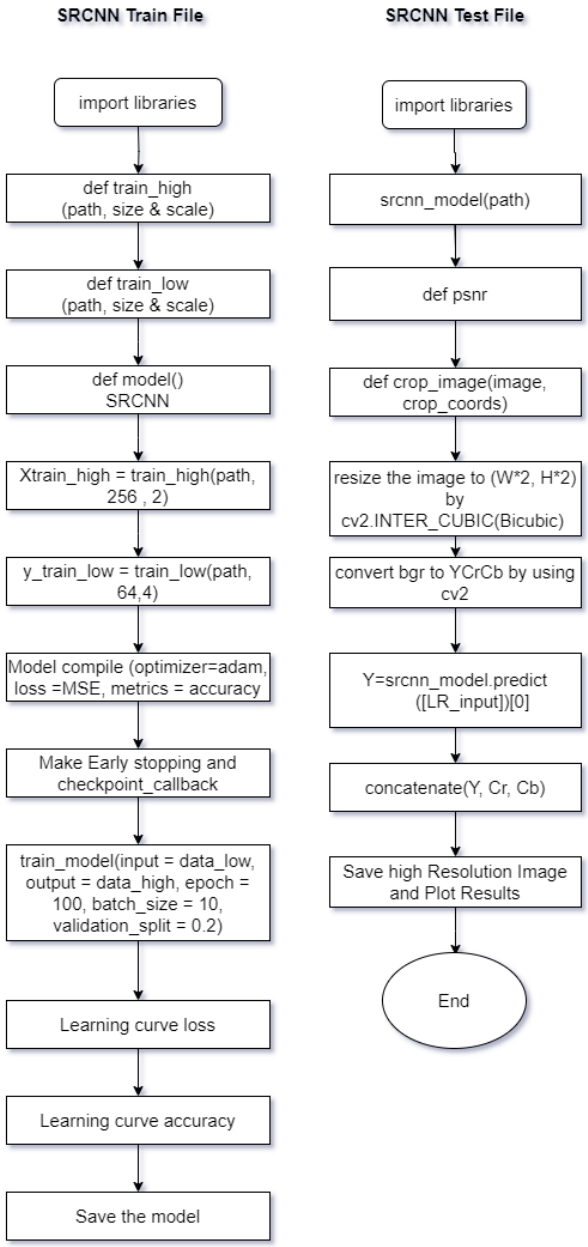
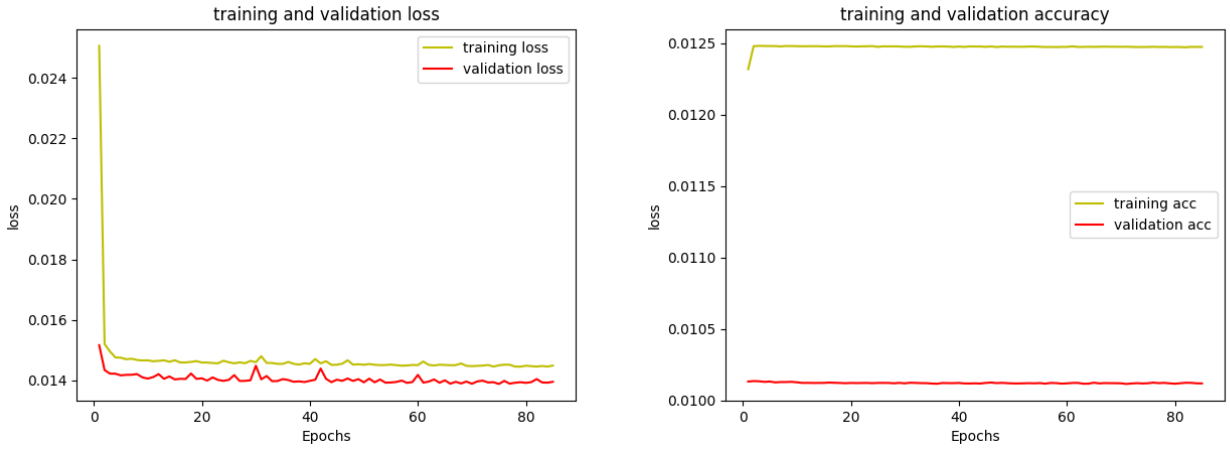


Figure 3.3: Model Training & Testing

3.2.6 Learning Curves

Learning curves are graphs that compare and show the performance of a model on training and validation data over a number of epochs. It shows the training and validation loss in Fig.

3.4(a), and the training and validation accuracy in Fig. 3.4(b) of SRCNN model.
As seen in Figures 3.4(a) and 3.4(b), both training and validation losses are consistently decreasing, while training and validation accuracy are improving. This decreasing loss with increasing epochs trend implies that the model was successfully trained without overfitting. Although the model was initially set to be trained for 100 epochs, it may have been extended further if further loss reduction was possible. However, in order to avoid overfitting, early stopping measures must be considered. As a result, training was stopped at the 85^{th} epoch in order to find a compromise between optimization and avoiding overfitting.



(a) SRCNN training and validation loss learning curve

(b) SRCNN training and validation accuracy learning curve

Figure 3.4: Loss and Accuracy curves

3.3 ESPCN

The Efficient Subpixel Convolution Network (ESPCN) has been successfully built at two distinct sizes, namely 2x and 4x. As previously stated, the model was trained using the DIV2K dataset and a cross-validation approach with an 80-20 split. The total number of parameters for the 2x scale is 96,740, while for the 4x scale it is 61,680. Table 3.2 presents the hyperparameters of the model.

Parameters	ESPCN
Optimizer	Adam
Learning Rate	$1 \cdot 10^{-3}$
Activation Function	ReLU
Loss Function	MSE
Number of Epochs	100
Batch Size	10

Table 3.2: ESPCN Parameters

3.3.1 ESPCN Model Structure

ESPCN model structure Figure 3.5 view the layers of the model, the shape of each layer, and the model flow. After that, is the model algorithm where each layer is described for more clarification. The following model structure is on a specific example of the input image with size (128,128,3) but our model takes any input size.

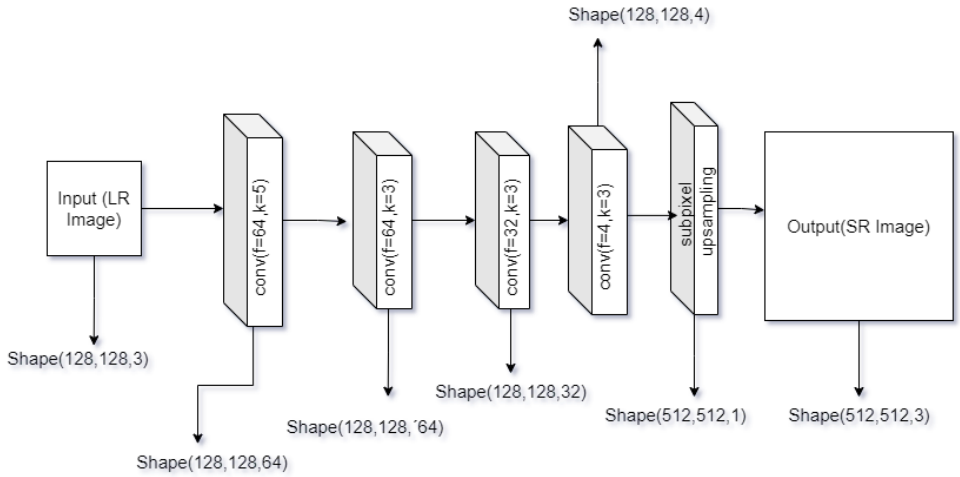


Figure 3.5: ESPCN model structure

3.3.2 Algorith

1.Input Image: Begin with an input image of the desired size, such as (128, 128, 3).
2.First Layer: Start with the first layer, which consists of a convolutional layer featuring 64 filters and a kernel size of 5x5. This layer transforms the input shape to (128, 128, 64).
3.Second Layer: Proceed to the second layer, which is another convolutional layer. This layer also utilizes 64 filters but with a smaller kernel size of 3x3. The output shape remains (128, 128, 64).
4.Third Layer: Continue with the third layer, which is another convolutional layer featuring 32 filters and a 3x3 kernel. The output shape remains (128, 128, 32).
5.Fourth Layer: Move on to the fourth layer, which is a convolutional layer with 4 filters and a 3x3 kernel. This results in an output shape of (128, 128, 4).
6.Sub-Pixel Shuffle: Apply the sub-pixel shuffle function to reconfigure the output. This transformation leads to the final super-resolved image with a shape of (512, 512, 3).

3.3.3 ESPCN Experiment details

The model utilizes images collected from the dataset and performs an 80-20% cross-validation. Subsequently, the images are resized to the specified dimensions, generating the training data. Additionally, the model processes the images to produce both high-resolution and low-resolution versions. The process of achieving low resolution involves dividing the image using a scale factor, whereas high resolution is attained by transforming the RGB color space to YUV and extracting the luminance channel Y. Finally, the process involves converting images into tensors and commencing the training of the model, based on the model structure outlined in Figure 3.5 and the previously mentioned algorithm. The provided figures,3.6, 3.7, and 3.8, depict a High-Low image training flowchart for a Model Training, a Model Training Flowchart and a Model Testing Flowchart, respectively. These diagrams are to provide a clear understanding of the complete implementation process of the model.

3.3.4 Creating High-Low Resolution Images for Model Training Flowchart

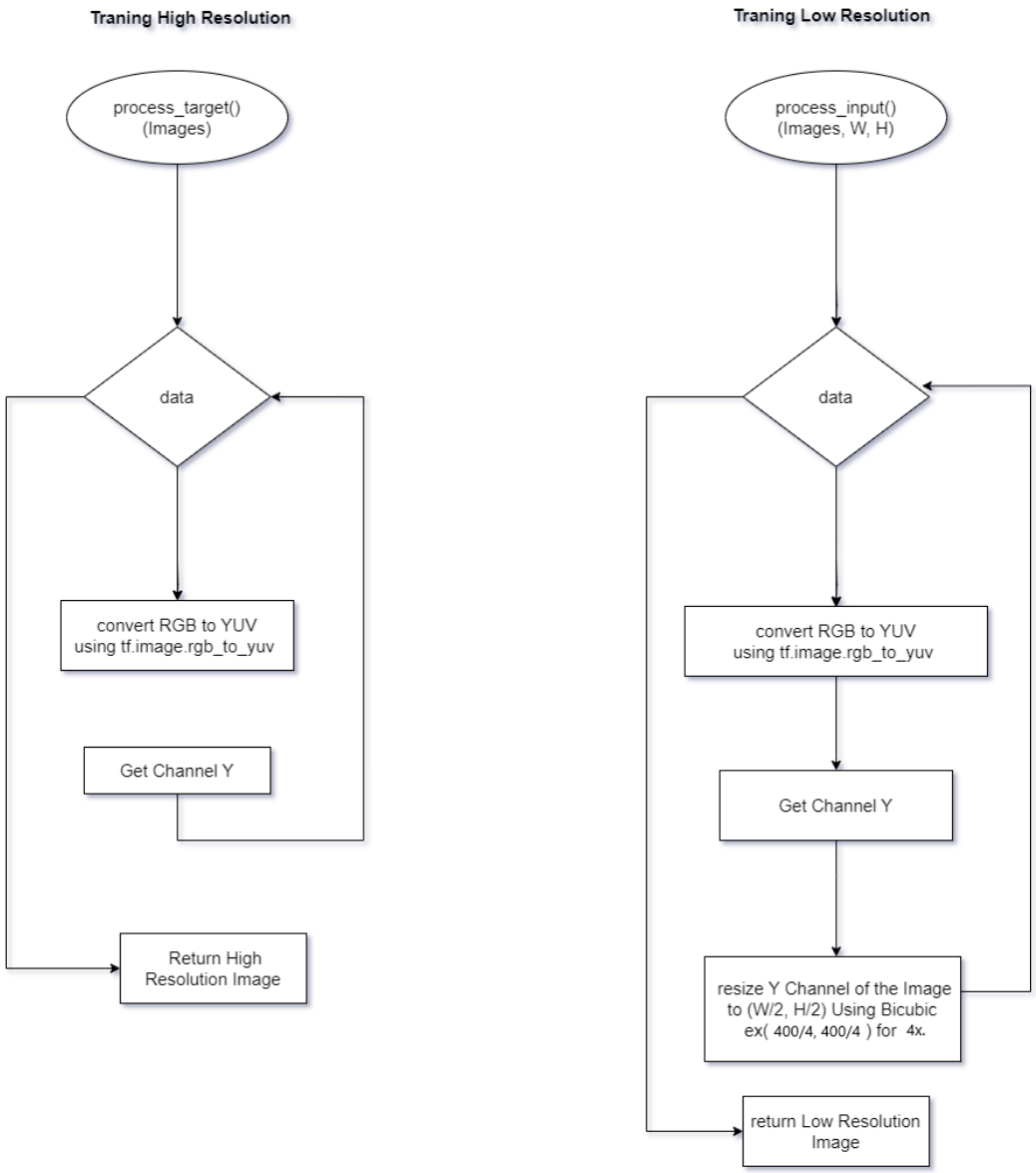


Figure 3.6: High-Low Resolution Images training flowchart

3.3.5 Model Training Flowchart

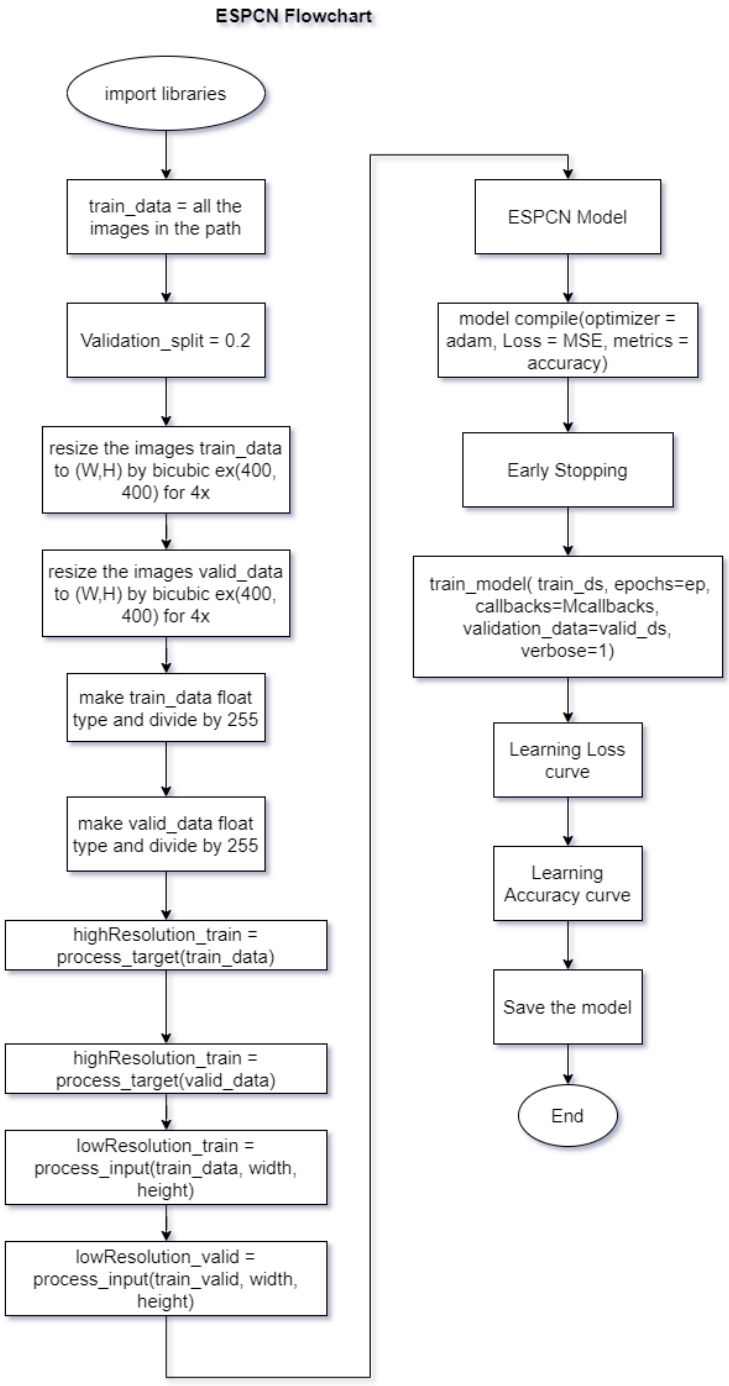


Figure 3.7: ESPCN Model Training Flowchart

3.3.6 Model Testing Flowchart

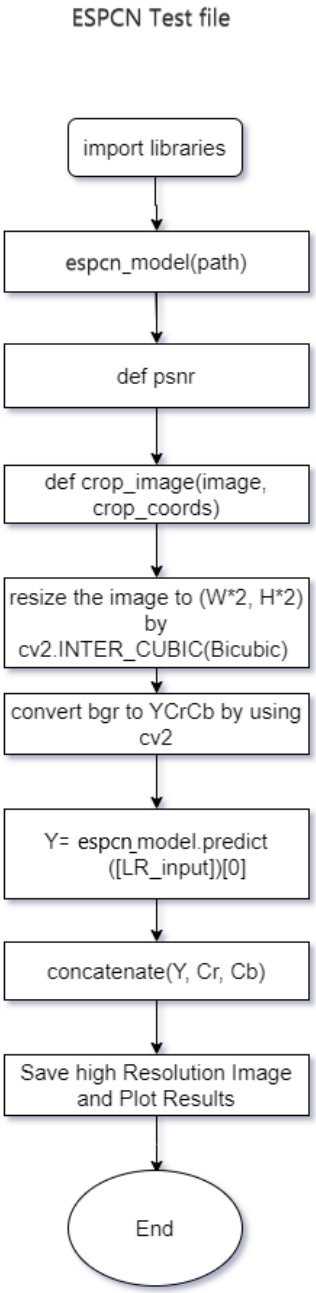
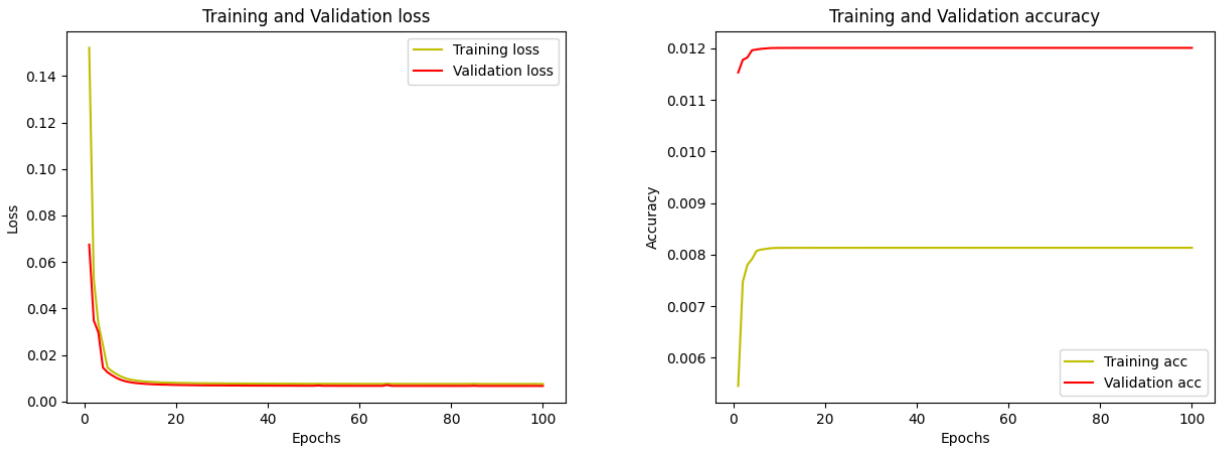


Figure 3.8: ESPCN Model Testing Flowchart

3.3.7 ESPCN Learning Curves

The training and validation loss is shown in Fig. 3.9(a), and the training and validation accuracy in Fig. 3.9(b) of ESPCN model.

As seen in Fig. 3.9(a), and fig. 3.9(b) The training and validation loss is decreasing and the training and validation accuracy is increasing the training and validation loss is decreasing with the increasing number of epochs, which shows that the model is well trained and has no



(a) ESPCN training and validation loss learning curve

(b) ESPCN training and validation accuracy learning curve

Figure 3.9: ESPCN Loss and Accuracy curves

overfit. The model was trained on 100 epochs as mentioned before, we could have increased the number of epochs since there is a possibility that the loss may still decrease, but early stopping (mentioned in ch.2) should be put into consideration to prevent the model from overfitting.

3.4 Autoencoder

Autoencoder has been implemented by two up-sampling techniques (subpixel and deconvolution), The model was trained on two different scales (2x and 4x), They have the same algorithm and structure. As mentioned, before it is trained on the DIV2K dataset and 80-20 cross-validation. With total number of parameters for a 2x scale is 1,586,179, and for a 4x scale is 2,029,315. The following Table 3.3 contains the hyperparameters of the model.

Parameters	Autoencoder
Optimizer	Adam
Learning Rate	$1 \cdot 10^{-3}$
Activation Function	ReLU
Loss Function	MSE
Number of Epochs	120
Batch Size	10

Table 3.3: Autoencoder Deconvolution Parameters

3.4.1 Autoencoder Model Structure

The autoencoder model structure Figure 3.10 shows layers, shapes, and flow. The model algorithm explains each layer next. This model is based on a certain input image size (128,128,3) however could be used with any size.

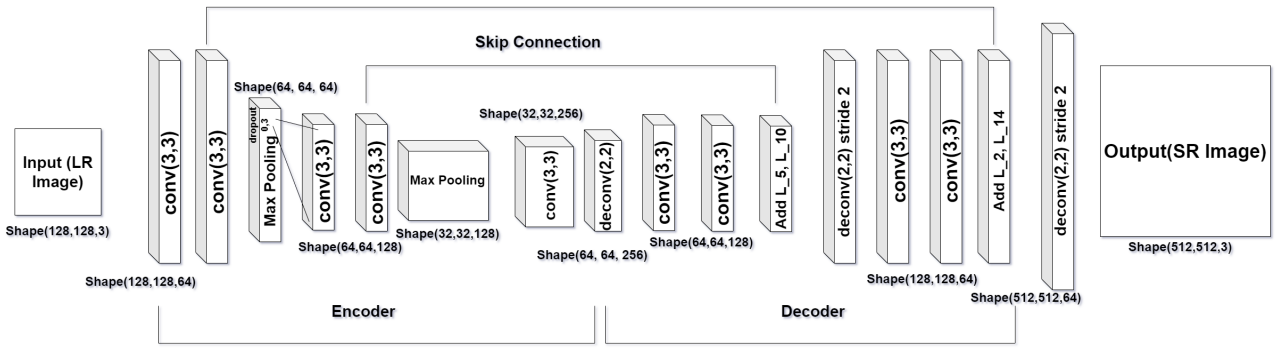


Figure 3.10: Autoencoder model structure

3.4.2 Algorithm

1. Convolutional two-dimensional layers (128, 128, 64) and ReLU activation layer y=max(0,x) were added. Two distinct layers in outer layer 1 extract key features from the input image before descending to a smaller feature space.

2. Connect the max pooling layer (64, 64, 64) to the preceding two-dimensional convolution layer to minimize dimensionality and reduce feature space.

3. The dropout layer (64, 64, 64) regularises concurrent processing of complex neural network layers.
4. Apply two two-dimensional convolutional layers sequentially with the same output shape (64, 64, 128) and the ReLU activation function determined from the formula. In outer layer 2, the layers up to this are saved for further processing.
5. Connect the max pooling layer (output shape: 32, 32, 128) to the preceding two-dimensional convolution layer.
6. Apply a two-dimensional convolution layer with an output shape of (32, 32, 256) after the previous max pooling layer.
7. The up-sampling layer uses transposition technique to enlarge the small picture matrix to a larger one (64, 64, 256 output form).
8. Apply two two-dimensional convolutional layers sequentially with the same output shape (64, 64, 128) and ReLU activation function.
9. Addition operation on previously convolutional layer with outer layer 2, stored for future processing with shape (64, 64, 128).
10. Again apply an upsampling layer with output shape (128, 128, 128).
11. Two two-dimensional convolutional layers are applied sequentially with the same output shape (128, 128, 64) and ReLU activation function.
12. Another procedure is conducted on the convolutional layer with outer layer 1, which is stored for further processing with a shape of (512, 512, 64).
13. Apply the upsampling layer again with output form (512, 512, 64). Applying the final layer of the model to the two-dimensional convolutional layer with an output shape of (512, 512, 3) yields the autoencoder output.
The approach that was previously described is specifically designed for processing data with a shape of (128, 128, 3). However, it is important to note that the model is capable of accepting input data of every shape.

3.4.3 Autoencoder Experiment details

The model utilizes a dataset comprising of photographs, which are then subjected to an 80-20% cross-validation process. Additionally, images are resized to achieve high resolution and subsequently divided into subsets to obtain low-resolution images. The Figure 3.10 model allows for the input of data.

3.4.4 Model Training Flowchart

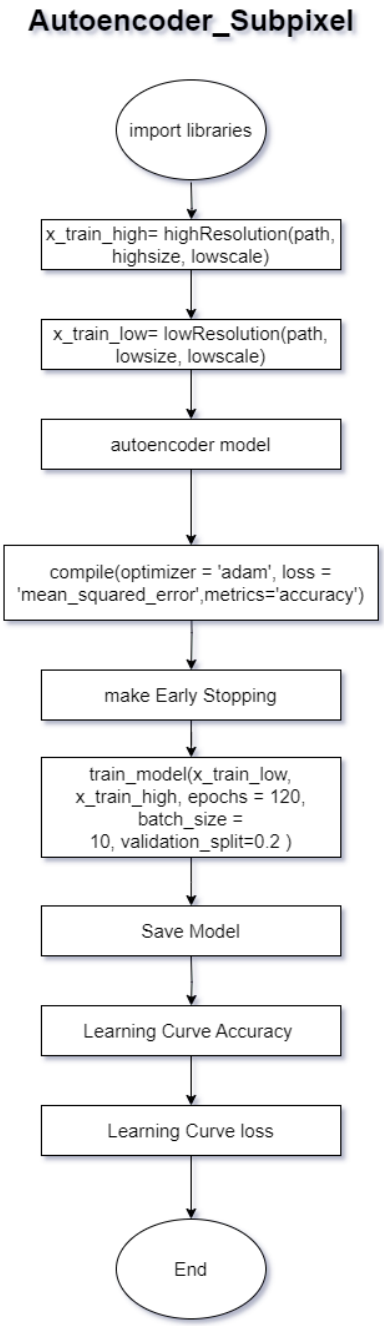


Figure 3.11: Model Training Flowchart

3.4.5 Model Testing Flowchart

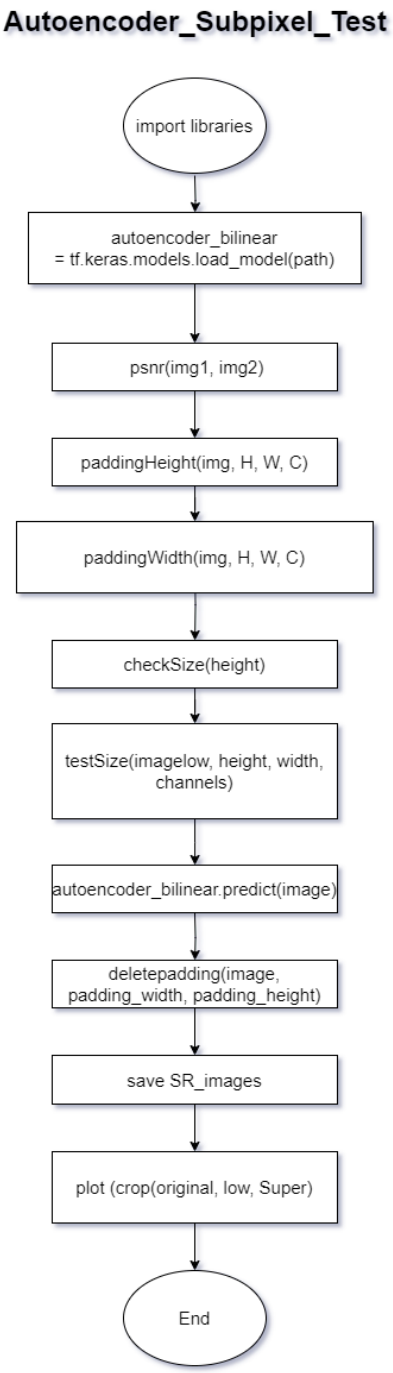
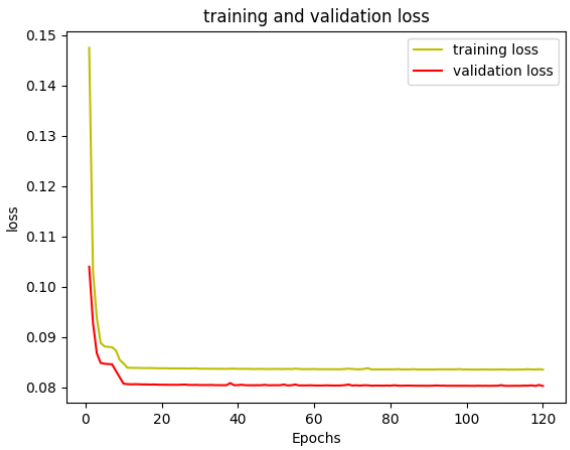


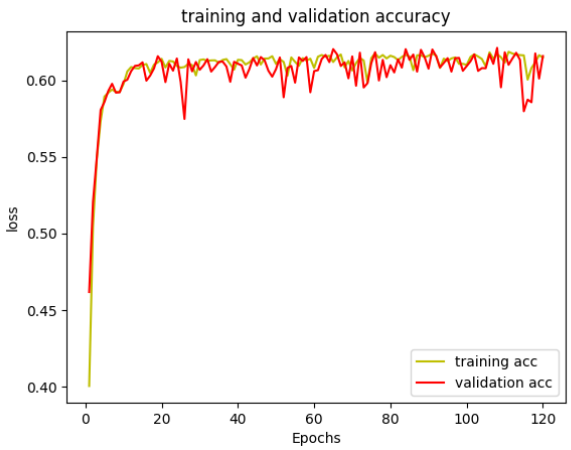
Figure 3.12: Model Testing Flowchart

3.4.6 Autoencoder Learning Curves

The following figures are learning curves of the training and validation loss, and the training and validation accuracy in Figure. 3.13(a) and Fig. 3.13(b) of the Autoencoder model, respectively. The analysis of Figure 3.13 reveals a consistent decrease in both training and validation loss, as well as an increase in training and validation accuracy, as the number of epochs increases.

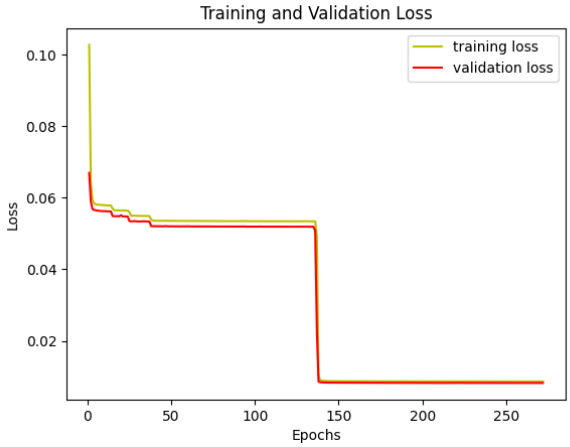


(a) Training and Validation loss learning curve 120 epochs

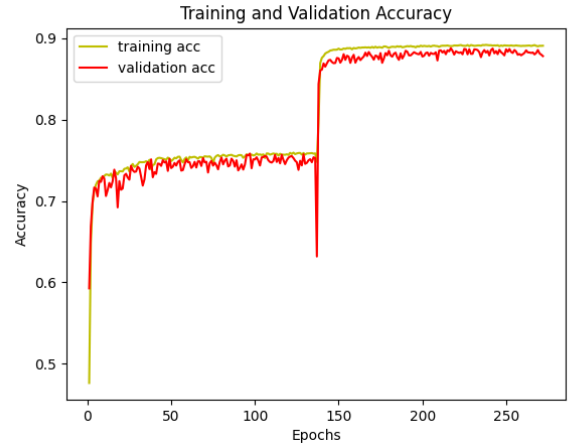


(b) Training and Validation accuracy learning curve 120 epochs

Figure 3.13: Loss and Accuracy curves 120 epochs



(a) training and validation loss learning curve 270 epochs



(b) training and validation accuracy learning curve 270 epochs

Figure 3.14: Loss and Accuracy curves 270 epochs

These findings suggest that the model has been effectively trained and does not exhibit signs of overfitting. The initial configuration of the model involved training it for 120 epochs. However, due to the absence of early stopping, the model was subsequently trained for an additional 1000 epochs. It was observed that the training process terminated at 270 epochs, as indicated by the Learning curve depicted in Figure 3.14. This early termination was likely due to the potential occurrence of overfitting. Fortunately, the implementation of early stopping prevented the model from encountering this issue.

3.5 SRGAN

Super-Resolution Generative Adversarial Networks (SRGAN) has been implemented using scale 4x. The model is trained on the DIV2K dataset. It is divided into generator and discriminator with a total number of parameters for generator is 1,553,667 and for discriminator is 23,569,217. A VGG loss is used to improve the MSE loss that often fails to enforce fine SR image generation The following Table 3.4 contains the hyperparameters of the model. Also, a pretrained SRRESnet model is downloaded and configured for

Parameters	SRGAN
Optimizer	Adam
Boundaries	100,000
Learning Rate G	$1 \cdot 10^{-4}$, $1 \cdot 10^{-5}$
Learning Rate D	$1 \cdot 10^{-4}$, $1 \cdot 10^{-5}$
No. of steps per Epoch	1000
Activation Function	ReLU
Loss Function	MSE
Number of Epochs	1000
Batch Size	10

Table 3.4: SRGAN Generator & Discriminator Parameters

3.5.1 SRGAN Generator Model Structure

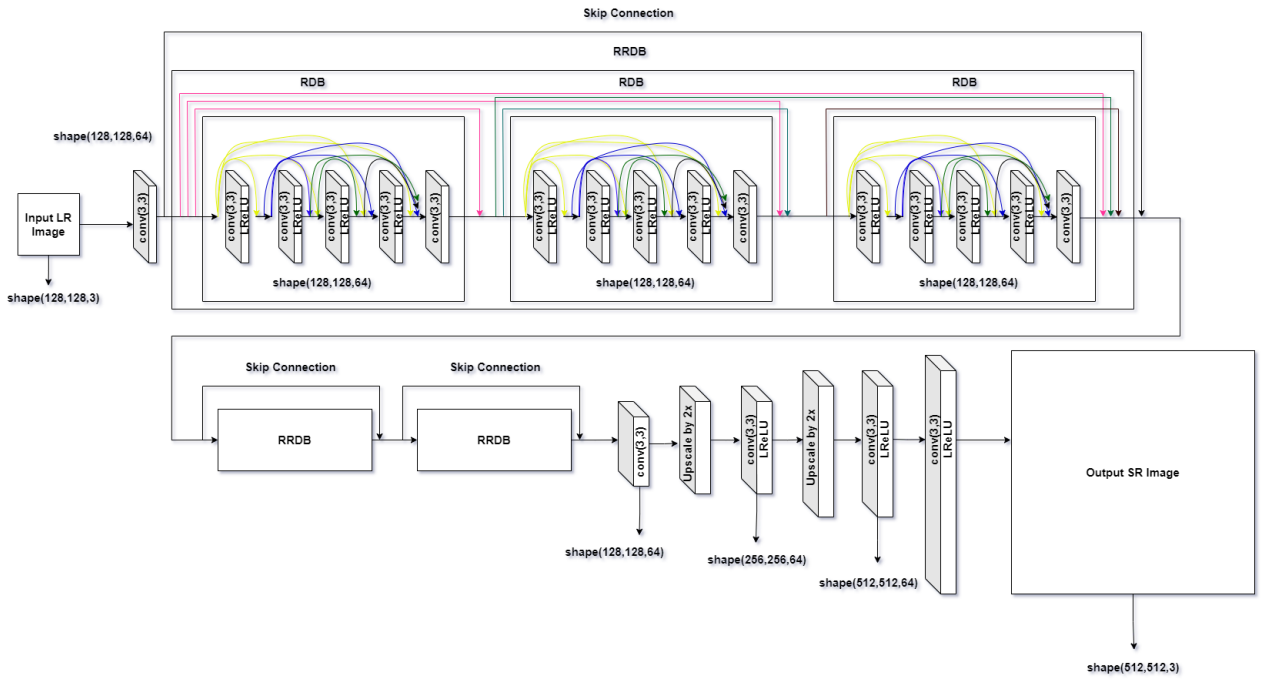


Figure 3.15: Generator Model Structure

3.5.2 SRGAN Generator Algorithm

1. Input: *lr* (Low-resolution image): The input image with shape (*128, 128, 3*).

2. Normalization (Normalize 0-1): Normalize the input image to the range [0, 1].

3. Initial Convolution (Conv2D): Apply a convolutional layer with 64 filters and a 9x9 kernel, with 'same' padding. Apply Parametric Rectified Linear Unit (PReLU) activation.

4. Residual Blocks (Repeated 16 times): Each residual block consists of the following layers:

1. Convolutional layer with 64 filters and a 3x3 kernel, 'same' padding.
2. Batch Normalization with momentum 0.8.
3. PReLU activation.
4. Convolutional layer with 64 filters and a 3x3 kernel, 'same' padding.
5. Batch Normalization with momentum 0.8.
6. Skip connection (Addition) with the block input.

5. Intermediate Convolution (Conv2D):

1. Apply a convolutional layer with 64 filters and a 3x3 kernel, 'same' padding.
2. Batch Normalization.
3. Add the result to the output of the first PReLU activation (skip connection).

6. Upsampling Layers:For each scaling factor (e.g., scale=4), repeat the following:

1. Apply a convolutional layer with 64*4=256 filters and a 3x3 kernel, 'same' padding.
2. Pixel Shuffle Layer (Lambda function): Rearranges pixel values to perform upsampling (depth to space with scale=2).
3. PReLU activation.

7. Final Convolution (Conv2D):

1. Apply a convolutional layer with 3 filters (corresponding to RGB channels) and a 9x9 kernel, 'same' padding.
2. Apply a convolutional layer with 3 filters (corresponding to RGB channels) and a 9x9 kernel, 'same' padding.
3. Apply the hyperbolic tangent (tanh) activation function to ensure output values are in the range [-1, 1].

8. Output (Super-Resolved Image): Denormalize the output image to the range [-1, 1]. The resulting image has a higher resolution and is the super-resolved image.

Model Summary: The generator model takes a low-resolution image as input and produces a high-resolution super-resolved image as output. It consists of 16 residual blocks, each containing two convolutional layers with PReLU activations, Batch Normalization, and a skip connection. The model uses pixel shuffle layers for upsampling to the desired scale.

Block Counts:

1. Initial Convolution: 1 block
2. Residual Blocks: 16 blocks
3. Intermediate Convolution: 1 block
4. Upsampling Layers: Varies based on the desired scaling factor (e.g., 4x upscaling means multiple upsampling layers)

3.5.3 SRGAN Discriminator Model Structure

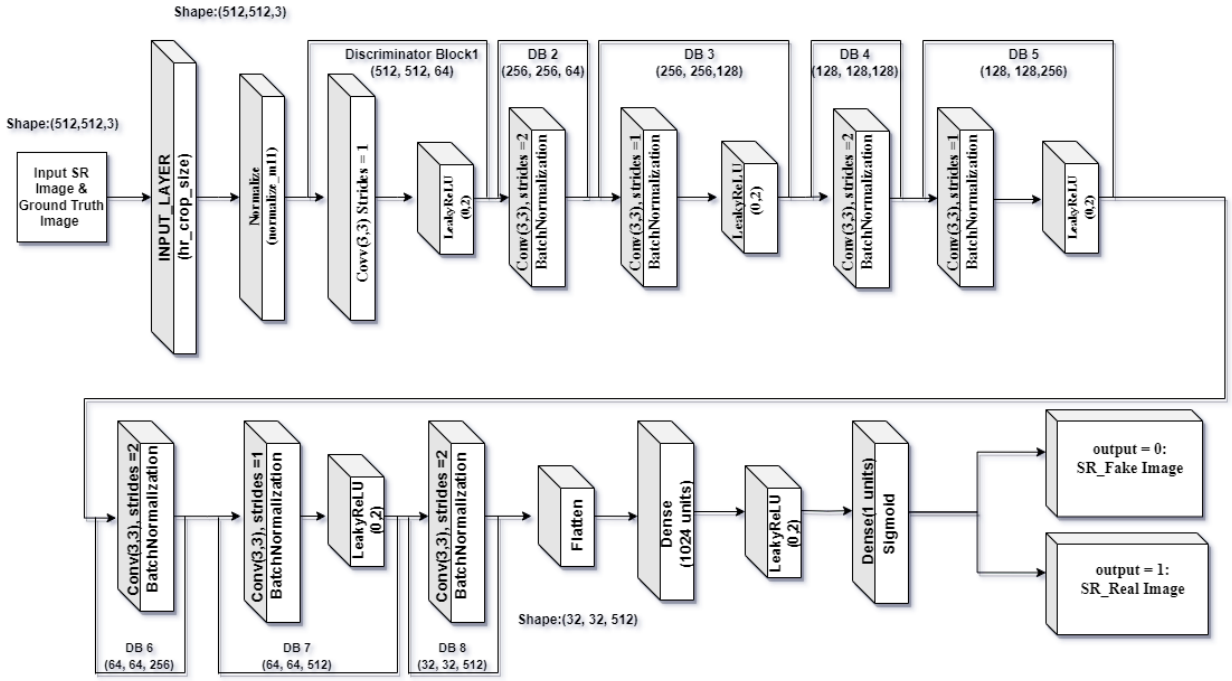


Figure 3.16: Discriminator Model Structure

3.5.4 SRGAN Generator Algorithm

1. Input: The discriminator takes an input image of shape $(\text{hr\_crop\_size}, \text{hr\_crop\_size}, 3)$.

2. Normalization: The input image is normalized using the `normalize_m11` function. This function likely normalizes the pixel values to the range $[-1, 1]$, which is a common practice in GANs (Generative Adversarial Networks).

3. Discriminator Blocks: The discriminator consists of several blocks, each composed of convolutional layers followed by optional batch normalization and LeakyReLU activation.

Block 1:
1. Conv2D with 64 filters, kernel size 3×3, and no strides.
2. LeakyReLU activation with an alpha of 0.2.

Block 2:
1. Conv2D with 64 filters, kernel size 3×3, and strides of 2.
2. Batch normalization (optional).
3. LeakyReLU activation with an alpha of 0.2.

Block 3:
1. Conv2D with 128 filters, kernel size 3×3, and no strides.
2. Batch normalization.
3. LeakyReLU activation with an alpha of 0.2.

Block 4:
1. Conv2D with 128 filters, kernel size 3×3, and strides of 2.
2. Batch normalization.

3. LeakyReLU activation with an alpha of 0.2.

Block 5:
1. Conv2D with 256 filters, kernel size 3×3, and no strides.
2. Batch normalization.
3. LeakyReLU activation with an alpha of 0.2.

Block 6:
1. Conv2D with 256 filters, kernel size 3×3, and strides of 2.
2. Batch normalization.
3. LeakyReLU activation with an alpha of 0.2.

Block 7:
1. Conv2D with 512 filters, kernel size 3×3, and no strides.
2. Batch normalization.
3. LeakyReLU activation with an alpha of 0.2.

Block 8:
1. Conv2D with 512 filters, kernel size 3×3, and strides of 2.
2. Batch normalization.
3. LeakyReLU activation with an alpha of 0.2.

4. Flatten Layer: After the convolutional blocks, the feature maps are flattened into a 1D vector.

5. Fully Connected Layers: A fully connected layer with 1024 units followed by LeakyReLU activation (alpha = 0.2). A final fully connected layer with a single unit and sigmoid activation produces a binary classification output (real or fake).

6. Block Count:
1. Convolutional Blocks: 8 blocks (labeled as Block 1 to Block 8).
2. Fully Connected Layers: 2 layers.

This discriminator is designed to evaluate the authenticity of super-resolved images, helping to train a GAN for image super-resolution. It assesses whether the input image is more likely to be real (high-resolution) or fake (super-resolved). The LeakyReLU activation functions introduce non-linearity, and batch normalization helps stabilize training.

Keep in mind that this discriminator is a critical component of a GAN-based super-resolution system, where it plays the adversarial role by trying to distinguish between real HR images and generated SR images. The discriminator's loss is used to train the generator network to produce more convincing SR images.

3.5.5 Creating High-Low Resolution Images for Model Training

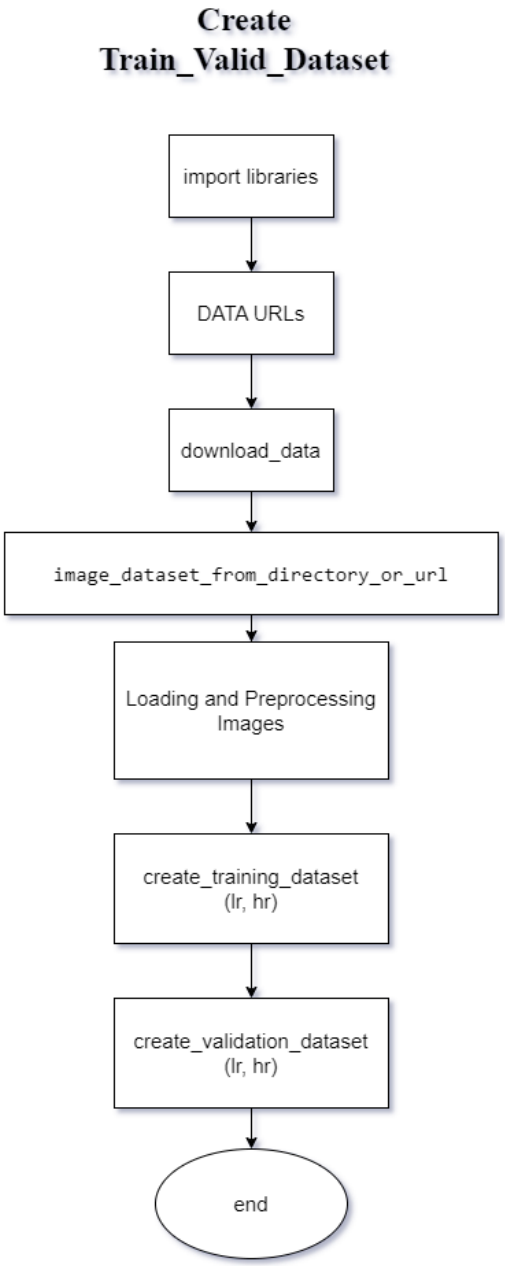


Figure 3.17: High-Low Resolution Images training

3.5.6 Model Training Flowchart

Many separate Python files were created as we can see in this model training flowchart. These files were created to ease the work for the training of the model which is to achieve super-resolution.

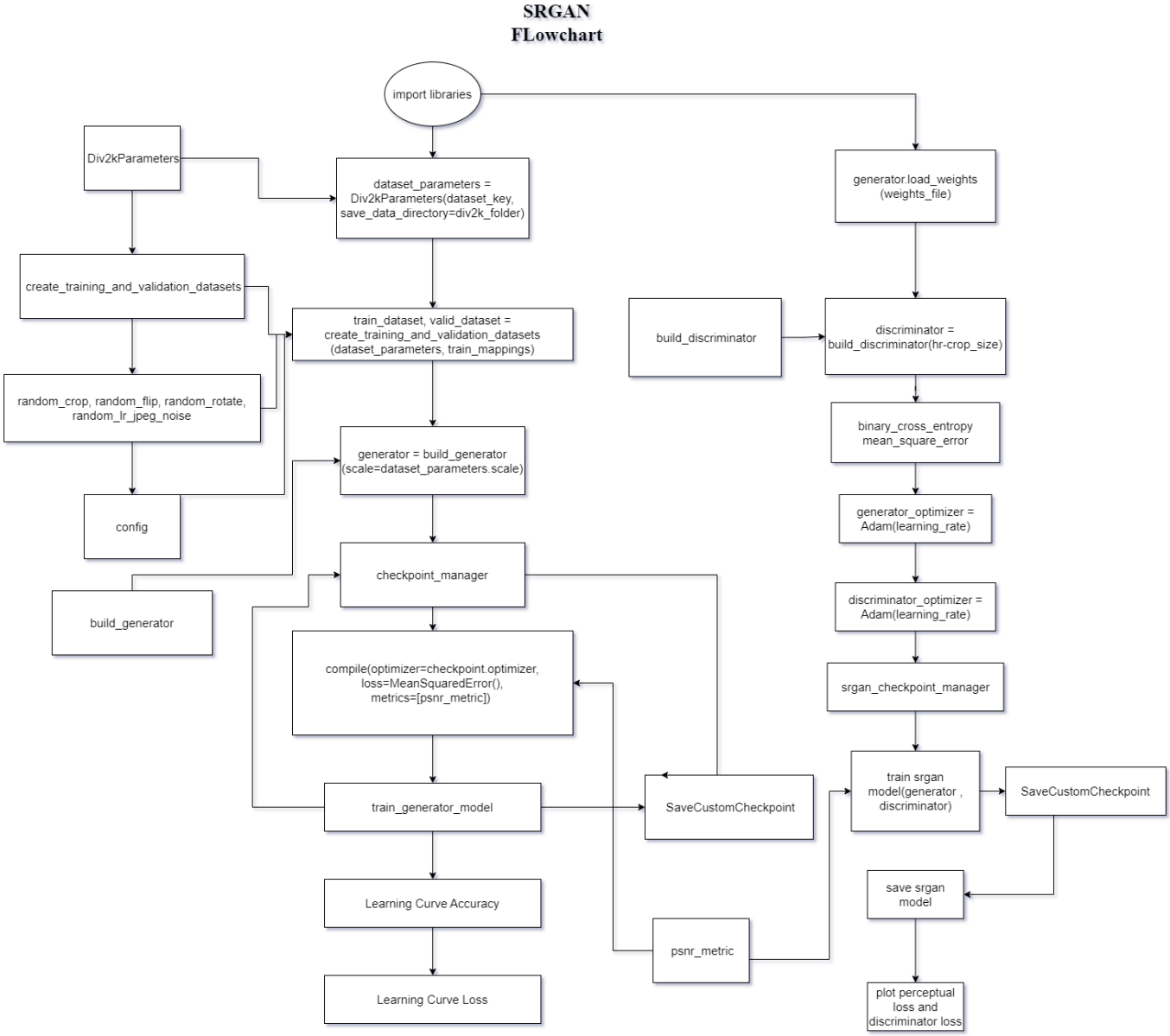


Figure 3.18: Model Training Flowchart

3.5.7 Model Testing Flowchart

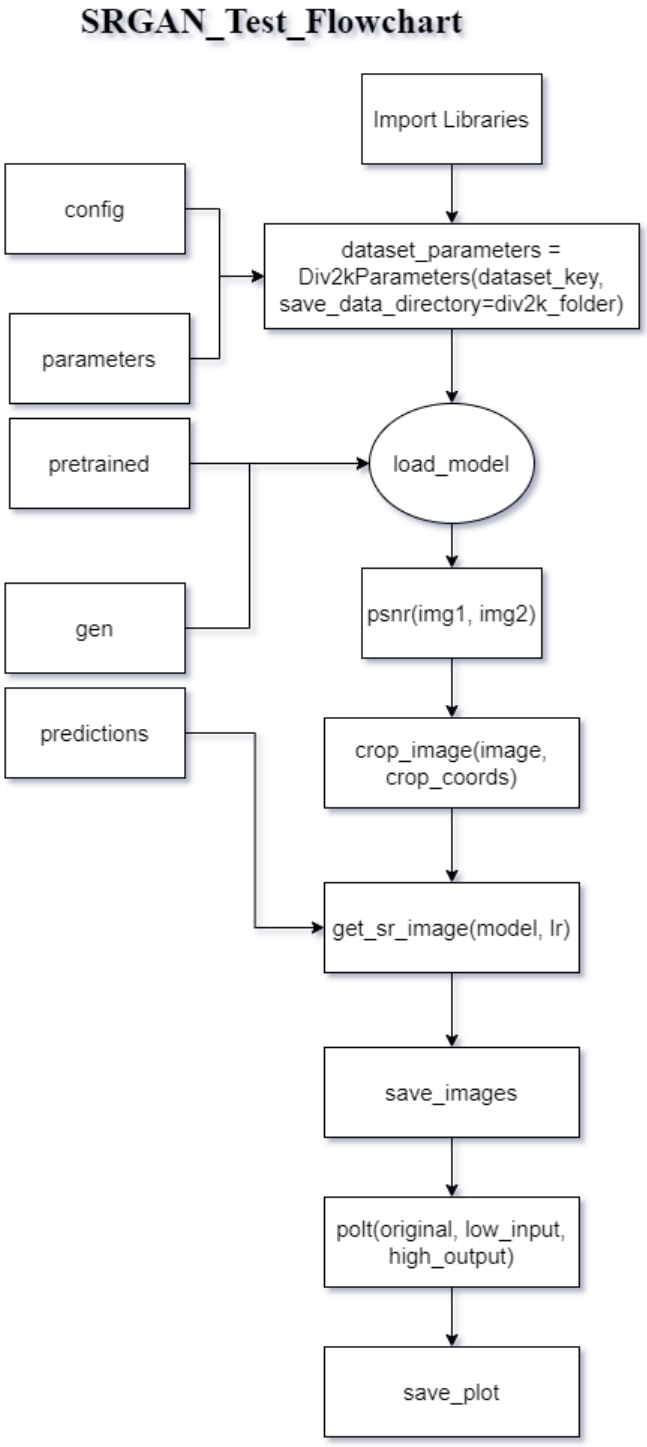
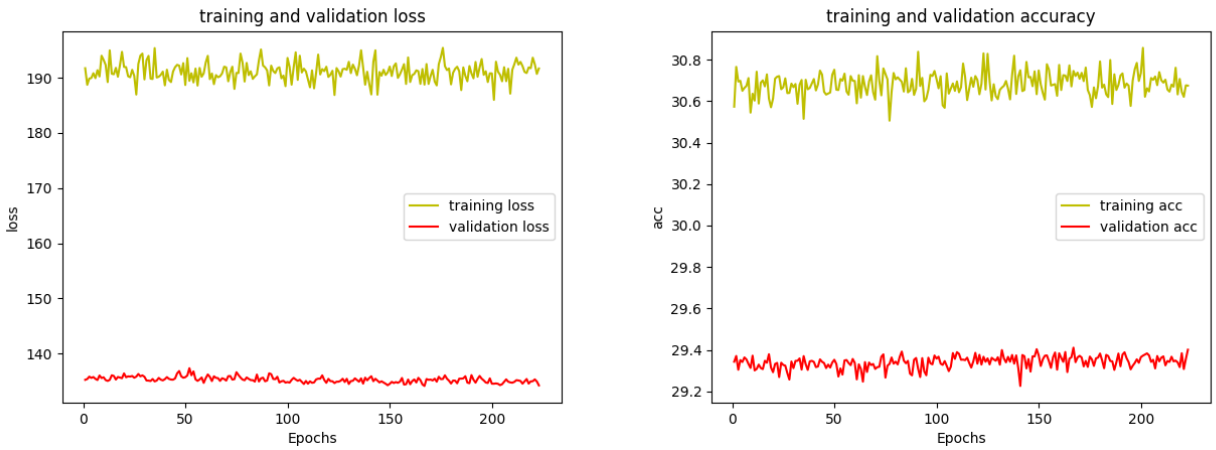


Figure 3.19: Model Testing Flowchart

3.5.8 Learning Curves

Figure 3.20 shows 200-epoch SRGAN generator model training. The model's training loss settled at 0.019 dB and the validation loss was around 0.013 dB. Training accuracy was 30.6–30.8 dB, with a Validation of 29.2–29.4 dB. Given the picture enhancement's complexity, the model's



(a) training and validation loss learning curve (b) Training and validation accuracy learning curve

Figure 3.20: Generator Loss and Accuracy curves

accuracy results show its expansion capacity.

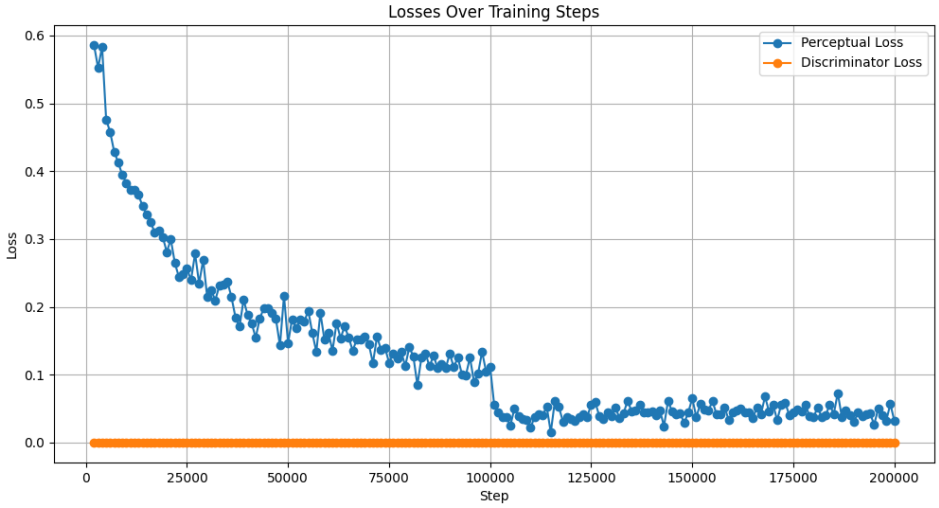


Figure 3.21: Learning Curves Generator-Discriminator training

Figure 3.21 presents a thorough depiction of the integrated training procedure involving both the Generator and Discriminator inside our model. The depicted illustration signifies a significant shift from our prior observations in Figure 3.20, wherein we solely watched the training of the Generator. In contrast to the continuous patterns of loss and accuracy found during the training of the Generator, Figure 3.21 demonstrates a notable development, specifically a decrease in perceptual loss. The observed reduction in perceptual loss is an encouraging indication that our Generator is advancing in its ability to provide output that is both realistic and of high quality.

The significance of the Discriminator's function in the combined training process becomes apparent when examining the graphical representation of the discriminator loss. It is worth mentioning that the Discriminator model, in its current architecture, does not have its own loss function. However, its primary role is to discern and differentiate between genuine and created data.

4. Results

This chapter will present an analysis of the results and provide a concise overview of the test dataset employed for evaluating the aforementioned models. Each model was implemented using Python scripts and developed using the Google Colab editor. Colab is a cloud-based platform designed for Jupyter Notebooks, which eliminates the need for any initial setup and provides free access to computing resources, including Graphics Processing Units (GPUs) and Tensor Processing Units (TPUs). Colab exhibits a notable level of compatibility between systems with machine learning, data science, and educational applications. The aforementioned methodology was utilized to facilitate the training and evaluation of the model.

4.1 Test Datasets

4.1.1 Set5

Introduced by Marco Bevilacqua et al. in Low-Complexity Single-Image Super-Resolution based on Non-negative Neighbor Embedding. The Set5 dataset is a dataset consisting of 5 images (“baby”, “bird”, “butterfly”, “head”, “woman”) commonly used for testing performance of Image Super-Resolution models [36].

4.1.2 set14

Introduced by Roman Zeyde et al. in On Single Image Scale-Up Using Sparse-Representations The Set14 dataset is a dataset consisting of 14 images commonly used for testing the performance of Image Super-Resolution models [37].

4.1.3 Medical Images

Helping Deep Learning and AI Enthusiasts like me to contribute to improving COVID-19 detection using just Chest X-rays. It is a simple directory structure branched into test and train and further branched into the respective 3 classes which contains the images. From this, we have used 5 HR images and downscaled them by a scale factor of 4 using Bicubic Interpolation and tested these 5 LR images against our models [38].

4.1.4 Satellite Images

Satellite imagery used to build this dataset is made available through Planet's Open California dataset, which is openly licensed[39].

4.2 Objective & Subjective Evaluation

As given in Chapter 2, Subsection 2.5.2 provides a detailed view of metrics evaluation using the peak signal-to-noise ratio(PSNR), structure similarity index method(SSIM), and universal image quality(Q/UQI). The test datasets are evaluated with these metrics.

4.2.1 Set5 Dataset Results

Metrics Based Evalution

Set 5 Results	butterfly.png			bird.png			baby.png		
	PSNR	SSIM	UQI	PSNR	SSIM	UQI	PSNR	SSIM	UQI
Nearest Neighbour	30.12	0.67	0.94	30.49	0.65	0.88	30.96	0.64	0.95
Bilinear	29.92	0.69	0.94	30.78	0.76	0.91	30.87	0.64	0.95
Bicubic	29.92	0.71	0.95	30.81	0.77	0.91	30.88	0.64	0.94
SRCNN	29.85	0.65	0.92	32.08	0.87	0.94	30.40	0.57	0.93
ESPCN	29.49	0.66	0.92	32.76	0.90	0.94	30.29	0.54	0.92
Autoencoder	30.14	0.77	0.96	32.30	0.88	0.95	31.36	0.71	0.96
SRGAN	29.65	0.67	0.92	32.24	0.90	0.94	29.97	0.52	0.93

Table 4.1: Set5 PSNR SSIM UQI

As shown in Table 4.1, The Autoencoder model has superior performance compared to the other models, with highly favorable outcomes. The image "baby.png" was seen at a scale of 4x, resulting in a peak signal-to-noise ratio (PSNR) of 31.36dB, a structural similarity index (SSIM) of 0.71, and a universal quality index (UQI) of 0.96. These metrics indicate that this particular model beats all other models. Additionally, it is evident that the SRGAN model has the poorest performance compared to all other models. The image quality metrics for the given data are as follows: Peak Signal-to-Noise Ratio (PSNR) is measured at 29.97 decibels, Structural Similarity Index (SSIM) is calculated to be 0.52, and Universal Quality Index (UQI) is determined to be 0.93.

Time based Evaluation

The processing time for each model is programmed utilizing the Python programming language and the technique of transfer learning.
As shown in Table 4.2 the SRCNN model exhibits faster speed compared to other models evaluated using the Set5 dataset. The image file named "baby.png" required a processing time

Model Processing Time	butterfly.png	bird.png	baby.png
Nearest Neighbour	1.2649433612823486	1.4369845390319824	2.81901216506958
Bilinear	4.0726094245910645	5.131830453872681	23.461042165756226
Bicubic	2.8188066482543945	4.843384265899658	12.380435943603516
Scrnn	9.29832458496093 8e-06	1.0251998901367188e-05	4.76837158203125e-06
ESPCN	0.07583236694335938	0.39519762992858887	0.12412190437316895
Autoencoder	0.2870447635650635	0.3173179626464844	1.316498041152954
SRGAN	0.6860742568969727	1.447791576385498	7.225991725921631

Table 4.2: Set5 Image Processing Time

of 4.76837158203125 microseconds when subjected to a scaling operation of 4x. The bilinear interpolation method required the longest processing time of 23.461042165756226 seconds when applied to the image "baby.png" at a scaling factor of 4x.

Subjective Evaluation

When subjectively comparing the performance of different models in terms of metrics evaluation, it appears that SRGAN has a more favourable subjective view. On the other hand, the output from Autoencoder appears to be smoother but lacks the distinctive features of the image. ESPCN, which has the highest PSNR of 31.01dB, exhibits good details but its model output is not superior to that of SRGAN.
The following Table 4.1gives us the metrics as well as time based evaluation for "head.png".

head.png	PSNR(dB)	SSIM	UQI	Time(s)
Nearest Neighbour	**31.11**	**0.70**	**0.95**	**1.3669040203094482**
Bilinear	**31.10**	**0.71**	**0.95**	**6.513258934020996**
Bicubic	**31.15**	**0.72**	**0.95**	**3.5147593021392822**
SRCNN	**30.46**	**0.63**	**0.93**	**8.58306884765625e-06**
ESPCN	**30.46**	**0.63**	**0.92**	**0.07977461814880371**
Autoencoder	**31.17**	**0.71**	**0.95**	**0.29107117652893066**
SRGAN	**30.13**	**0.57**	**0.89**	**0.6872272491455078**

Table 4.3: head.png evaluation

The optimal approach to enhance resolution is accomplished by utilizing the Super-Resolution Convolutional Neural Network (SRCNN) algorithm on the specified image file, "head.png", yielding a processing duration of around 8.58306884765625 microseconds. The dimensions of the image titled "Original head.png" are (280, 280, 3). The low-resolution image has dimensions of (69, 69, 3). The Autoencoder model indicates superior performance in terms of Peak Signal-

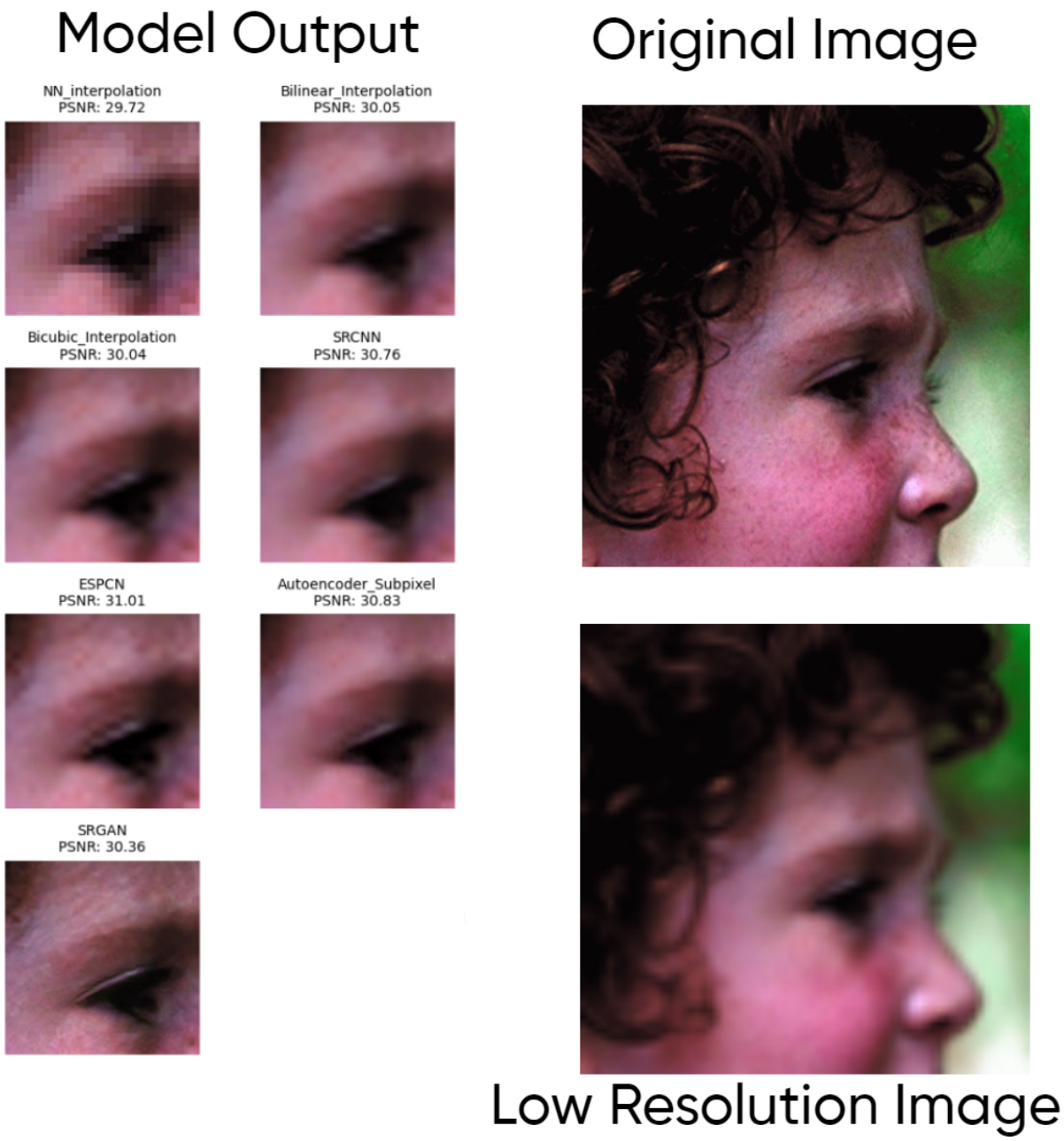


Figure 4.1: Set5/head.png

to-noise ratio (PSNR), achieving a value of 31.17dB, as well as UQI metrics with a score of 0.95. On the other hand, when the bicubic approach is used for the head.png image, it demonstrates the highest Structural Similarity Index (SSIM) with a value of 0.72. Upon subjective evaluation, it appears that the results obtained with the SRGAN model for the image "head.png" exhibit a higher level of detail and superior quality compared to alternative models.

4.2.2 Set14 Dataset Results

Metrics Based Evalution

As shown in Table 4.4, the Autoencoder model exhibits greater performance in comparison to the other models, yielding highly favorable results. The image denoted as "baboon.png"

Set14 Results	baboon.png			man.png			flowers.png		
	PSNR	SSIM	UQI	PSNR	SSIM	UQI	PSNR	SSIM	UQI
Nearest Neighbour	28.75	0.32	0.92	29.66	0.46	0.88	30.34	0.58	0.94
Bilinear	28.74	0.33	0.93	29.57	0.45	0.87	30.24	0.58	0.94
Bicubic	28.74	0.34	0.93	29.57	0.45	0.87	30.28	0.59	0.94
SRCNN	28.64	0.28	0.92	29.34	0.36	0.83	29.97	0.48	0.91
ESPCN	28.63	0.28	0.91	29.26	0.33	0.82	29.91	0.47	0.90
Autoencoder	28.92	0.45	0.95	29.88	0.53	0.89	30.03	0.48	0.90
SRGAN	28.43	0.25	0.90	29.25	0.31	0.82	29.76	0.43	0.90

Table 4.4: Set14 PSNR SSIM UQI

underwent a 4x upscaling process, which produced a peak signal-to-noise ratio (PSNR) of 28.92 decibels (dB), a structural similarity index (SSIM) of 0.45, and a universal quality index (UQI) of 0.95. The stats demonstrate that this specific model outperforms all other models. Furthermore, it is apparent that the SRGAN model exhibits the least satisfactory performance in comparison to all other models. The picture quality measures utilised for the SRGAN are as follows: The Peak Signal-to-Noise Ratio (PSNR) is seen to have a value of 28.43 dB, while the Structural Similarity Index (SSIM) is computed to be 0.25. Additionally, the Universal Quality Index (UQI) is discovered to be 0.90.

Time based Evaluation

Model Processing Time	**baboon.png**	**man.png**	**flowers.png**
Nearest Neighbour	41.1389844417572	3.4345169067382812	1.8568639755249023
Bilinear	15.66460633277893	16.587475061416626	12.454272508621216
Bicubic	18.41284966468811	16.482320547103882	12.186731338500977
Scrnn	7.3909759521484375e-06	7.152557373046875e-06	6.4373016357421875e-06
ESPCN	0.19222450256347656	0.10503244400024414	0.16424989700317383
Autoencoder	1.3267014026641846	1.3168034553527832	0.2998049259185791
SRGAN	3.663444757461548	5.172736644744873	1.5003535747528076

Table 4.5: Set14 Image Processing Time

As shown in Table 4.5 The SRCNN model demonstrates superior computational efficiency in comparison to other models that were assessed using the Set14 dataset. The image file titled "baboon.png" underwent a scaling procedure of 4x, resulting in a processing time of 7.3909759521484375 microseconds. The execution of the nearest neighbor interpolation method to the image "baboon.png" with a scaling factor of 4x resulted in a processing time of 41.1389844417572 seconds, which was the longest among the tested methods.

Subjective Evaluation

When subjectively comparing the performance of different models in terms of metrics evaluation, it appears that SRGAN has a more favorable subjective view. On the other hand, the output from Autoencoder appears to be smoother but lacks the distinctive features of the image. ESPCN, which has the highest PSNR of 29.25dB, exhibits good details but its model output is not superior to that of SRGAN.

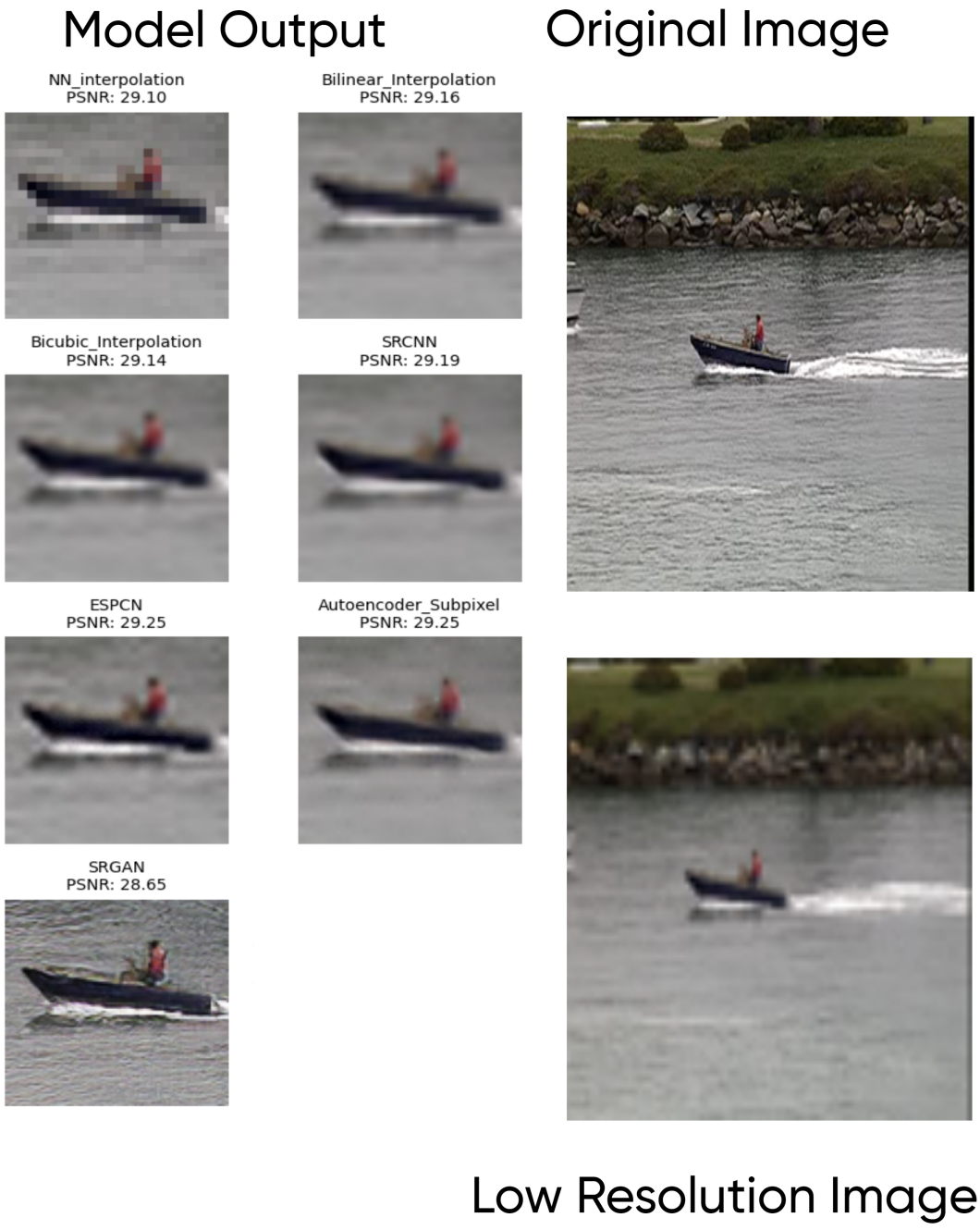


Figure 4.2: Set14/coastgaurd.png

The following Table 4.6 gives us the metrics as well as time-based evaluation for "coastguard.png".

coastgaurd.png	PSNR	SSIM	UQI	Time(s)
Nearest Neighbour	**29.53**	**0.42**	**0.97**	**1.897402048110962**
Bilinear	**29.63**	**0.46**	**0.97**	**5.911610841751099**
Bicubic	**29.64**	**0.47**	**0.97**	**8.034470558166504**
SRCNN	**29.72**	**0.48**	**0.98**	**5.9604644775390625e-06**
ESPCN	**29.80**	**0.53**	**0.98**	**0.1802520751953125**
Autoencoder	**29.85**	**0.52**	**0.98**	**0.35178327560424805**
SRGAN	**29.00**	**0.38**	**0.97**	**1.3255438804626465**

Table 4.6: coastguard.png evaluation

The resolution enhancement method can be effectively achieved by employing the Super-Resolution Convolutional Neural Network (SRCNN) algorithm on the designated image file, "coastguard.png". This procedure results in a processing time of around 5.9604644775390625 microseconds. The Figure 4.2 "coastguard.png" has dimensions of (288, 352, 3). (72, 87, 3) are the measurements of the low-resolution image. The Autoencoder model has excellent performance in terms of Peak Signal-to-noise ratio (PSNR), obtaining a value of 29.85dB, along with UQI metrics, which obtained a score of 0.98. In contrast, the utilisation of the bicubic method for the "coastguard.png" image yields the most notable Structural Similarity Index (SSIM) of 0.52.
Based on a subjective assessment, it is seen that the outcomes achieved using the SRGAN model for the image "coastguard.png" demonstrate a greater level of detail and superior quality in comparison to alternative models.

4.2.3 Medical Images

Metrics Based Evalution

As shown in Table 4.7, the SRCNN model exhibits greater performance in comparison to the other models, yielding highly favorable results. The image denoted as "01.png" underwent a 4x upscaling process, which produced a peak signal-to-noise ratio (PSNR) of 34.44 decibels (dB), a structural similarity index (SSIM) of 0.84, and a universal quality index (UQI) of 0.98. The stats demonstrate that this specific model outperforms all other models. Furthermore, it is apparent that the SRGAN model exhibits the least satisfactory performance in comparison to all other models. The picture quality measures utilised for the SRGAN are as follows: The Peak Signal-to-Noise Ratio (PSNR) is seen to have a value of 29.98 dB, while the Structural Similarity Index (SSIM) is computed to be 0.46. Additionally, the Universal Quality Index

	05.png			01.png			02.png		
	PSNR	**SSIM**	**UQI**	**PSNR**	**SSIM**	**UQI**	**PSNR**	**SSIM**	**UQI**
Nearest Neighbour	**31.29**	**0.61**	**0.99**	**32.15**	**0.70**	**0.97**	**30.99**	**0.59**	**0.99**
Bilinear	**32.21**	**0.71**	**0.99**	**33.20**	**0.79**	**0.98**	**31.73**	**0.68**	**0.99**
Bicubic	**32.04**	**0.70**	**0.99**	**33.04**	**0.78**	**0.98**	**31.57**	**0.67**	**0.99**
SRCNN	**33.12**	**0.77**	**0.99**	**34.44**	**0.84**	**0.98**	**32.52**	**0.74**	**0.99**
ESPCN	**32.15**	**0.72**	**0.99**	**32.20**	**0.72**	**0.98**	**31.67**	**0.70**	**0.99**
Autoencoder	**30.56**	**0.55**	**0.98**	**31.95**	**0.69**	**0.97**	**30.93**	**0.58**	**0.98**
SRGAN	**29.49**	**0.40**	**0.97**	**29.98**	**0.46**	**0.90**	**29.43**	**0.36**	**0.97**

Table 4.7: Medical Images PSNR SSIM UQI

(UQI) is discovered to be 0.90.

Time based Evaluation

	03.png	04.png	05.png
Nearest Neighbour	**23.040222883224487**	**61.74701738357544**	**55.48414468765259**
Bilinear	**159.3715033531189**	**443.43003511428833**	**402.25062465667725**
Bicubic	**173.8292374610901**	**486.56131625175476**	**438.5942533016205**
Scrnn	**1.5020370483398e-05**	**8.58306884765e-06**	**6.67572021484e-06**
ESPCN	**2.598209857940674**	**1.4106853008270264**	**1.5716898441314697**
Autoencoder	**25.653207302093506**	**9.293129444122314**	**41.09856700897217**
SRGAN	**114.73309469223022**	**41.9976749420166**	**97.98771381378174**

Table 4.8: Medical Images Image Processing Time

As shown in Table 4.8 The SRCNN model demonstrates superior computational efficiency in comparison to other models that were assessed using the Medical Image dataset. The image file titled "03.png" underwent a scaling procedure of 4x, resulting in a processing time of 1.5020370483398438 microseconds. The execution of the bicubic interpolation method to the image "03.png" with a scaling factor of 4x resulted in a processing time of 173.8292374610901 seconds, which was the longest among the tested methods.

Subjective Evaluation

When subjectively comparing the performance of different models in terms of metrics evaluation, it appears that SRGAN has a more favorable subjective view. On the other hand, the output from Autoencoder appears to be smoother but lacks the distinctive features of the image. ESPCN, which has the highest PSNR of 29.25dB, exhibits good details but its model output is not superior to that of SRGAN.
The Figure 4.3 gives us the metrics as well as time-based evaluation for "04.png".
As shown in Table 4.9 the resolution enhancement method can be effectively achieved by employing the Super-Resolution Convolutional Neural Network (SRCNN) algorithm on the desig-

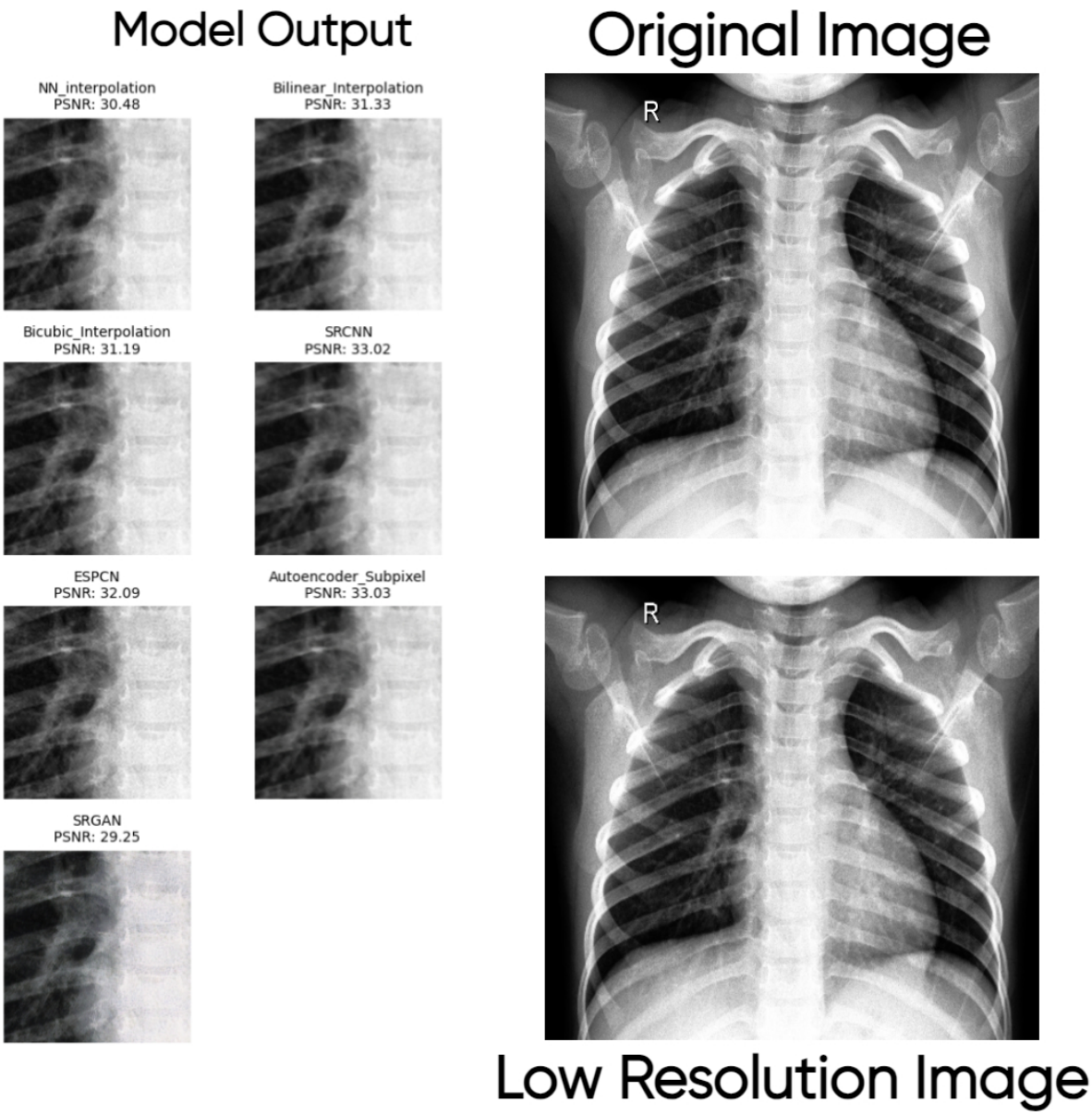


Figure 4.3: Medical Images/04.png

nated Figure 4.3, "04.png". This procedure results in a processing time of around 8.58306884765625 microseconds. The image has dimensions of (1372, 1742, 3). (343, 435, 3) are the measurements of the low-resolution image. The Autoencoder model and SRCNN has excellent performance in terms of Peak Signal-to-noise ratio (PSNR), obtaining a value of 33.03dB and 33.02dB, along with UQI metrics, which obtained a score of 0.97 and 0.99 respectively. In contrast, the utilization of the SRCNN method for the "04.png" image yields the most notable Structural Similarity Index (SSIM) of 0.81.

Perceptually, it is quite difficult to reach a conclusion regarding medical images. This is the primary reason subjective comparison is invalid for the majority of purposes.

04.png	PSNR	SSIM	UQI	Time(s)
Nearest Neighbour	30.48	0.67	0.98	61.74701738357544
Bilinear	31.33	0.76	0.99	443.43003511428833
Bicubic	31.19	0.75	0.99	486.56131625175476
SRCNN	33.02	0.81	0.99	8.58306884765625e-06
ESPCN	32.09	0.80	0.98	1.4106853008270264
Autoencoder	33.03	0.66	0.97	9.293129444122314
SRGAN	29.25	0.46	0.95	41.9976749420166

Table 4.9: 04.png evaluation

4.2.4 Satellite Images

Metrics Based Evalution

Satellite Images	scene_3.png			scene_1.png			scene_2.png		
	PSNR	SSIM	UQI	PSNR	SSIM	UQI	PSNR	SSIM	UQI
Nearest Neighbour	29.45	0.47	0.99	29.54	0.50	0.97	29.62	0.52	0.96
Bilinear	29.70	0.62	0.99	29.83	0.63	0.98	29.79	0.65	0.98
Bicubic	29.65	0.63	0.99	29.81	0.64	0.98	29.79	0.66	0.98
SRCNN	30.50	0.77	1.00	30.36	0.75	0.99	30.35	0.76	0.99
ESPCN	30.40	0.79	1.00	30.19	0.77	0.99	30.26	0.79	0.99
Autoencoder	28.67	0.19	0.97	29.06	0.33	0.95	28.84	0.23	0.91
SRGAN	28.52	0.48	0.98	28.70	0.52	0.97	28.83	0.59	0.96

Table 4.10: Satellite Images PSNR SSIM UQI

As shown in Table 4.10, the image denoted as "scene_01.png" underwent a 4x upscaling process, which produced a peak signal-to-noise ratio (PSNR) of 30.36 decibels (dB) for SRCNN model which is highest among all, a structural similarity index (SSIM) of 0.77 was registered for ESPCN model, and both ESPCN and SRCNN model has universal quality index (UQI) of 0.99. Furthermore, it is apparent that the SRGAN model exhibits the least satisfactory performance in comparison to all other models. The picture quality measures utilized for the SRGAN are as follows: The Peak Signal-to-Noise Ratio (PSNR) is seen to have a value of 28.70 dB, while the Structural Similarity Index (SSIM) is computed to be 0.52. Additionally, the Universal Quality Index (UQI) is discovered to be 0.97.

Time based Evaluation

As shown in Table 4.11 The SRCNN model demonstrates superior computational efficiency in comparison to other models that were assessed using the Medical Image dataset. The image file titled "Scene_1.png" underwent a scaling procedure of 4x, resulting in a processing time of 6.198883056640625 microseconds. The execution of the bicubic interpolation method to the im-

	Scene_1	Scene_2	Scene_3
Nearest Neighbour	**9.499075174331665**	**6.589518785476685**	**13.16812777519226**
Bilinear	**70.20917987823486**	**44.553818464279175**	**94.7231674194336**
Bicubic	**77.62449622154236**	**47.019455432891846**	**97.07889294624329**
SRCNN	**6.198883056640e-06**	**5.9604644775390e-06**	**7.152557373046e-06**
ESPCN	**0.3917245864868164**	**0.2386934757232666**	**0.7563259601593018**
Autoencoder	**10.358985185623169**	**2.230616331100464**	**3.477984666824341**
SRGAN	**14.466397047042847**	**10.31320309638977**	**10.323824167251587**

Table 4.11: Satellite Images Image Processing Time

age "Scene_1.png" with a scaling factor of 4x resulted in a processing time of 77.62449622154236 seconds, which was the longest among the tested methods.

Subjective Evaluation

When subjectively comparing the performance of different models in terms of metrics evaluation, it appears that SRGAN has a more favorable subjective view. On the other hand, the output from Autoencoder appears to be smoother but lacks the distinctive features of the image. Although it gives the highest PSNR value of 36.82dB and exhibits good details its model output is not superior to that of SRGAN. In terms of processing time, SRCNN model produces a super resolution image in 1.049041748046875 microseconds of "Scene_4.png".
The following Table 4.12 gives us the metrics as well as time-based evaluation for "Scene_4.png".

scene_4.png	PSNR	SSIM	UQI	Time(s)
Nearest Neighbour	**32.53**	**0.65**	**1.00**	**7.6174116134643555**
Bilinear	**33.90**	**0.75**	**1.00**	**45.244446754455566**
Bicubic	**33.63**	**0.75**	**1.00**	**47.795045614242554**
SRCNN	**36.16**	**0.84**	**1.00**	**1.049041748046875e-05**
ESPCN	**35.32**	**0.84**	**1.00**	**0.22671008110046387**
Autoencoder	**36.82**	**0.67**	**1.00**	**2.6133012771606445**
SRGAN	**29.79**	**0.58**	**1.00**	**11.198408842086792**

Table 4.12: Scene_4.png evaluation

As shown in Table 4.12 the resolution enhancement method can be effectively achieved by employing the Super-Resolution Convolutional Neural Network (SRCNN) algorithm on the designated image file, "04.png". This procedure results in a processing time of around 8.58306884765625 microseconds. The image has dimensions of (1372, 1742, 3). (343, 435, 3) are the measurements of the low-resolution image. The Autoencoder model and SRCNN has excellent performance in terms of Peak Signal-to-noise ratio (PSNR), obtaining a value of 33.03dB and 33.02dB, along with UQI metrics, which obtained a score of 0.97 and 0.99 respectively. In contrast, the utilization of the SRCNN method for the "04.png" image yields the most notable Structural Similarity

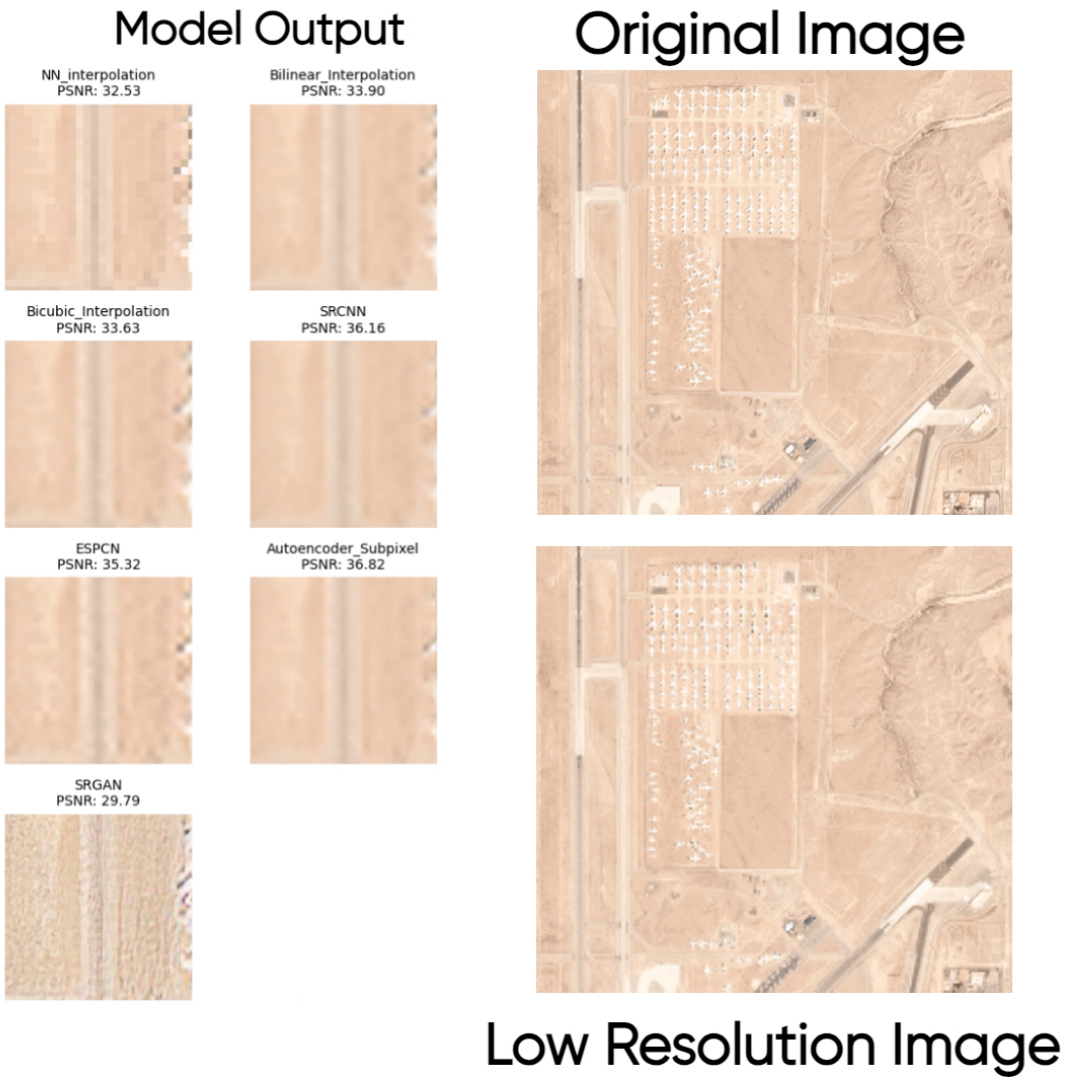


Figure 4.4: Satellite Images/Scene_4.png

Index (SSIM) of 0.81.

From Figure 4.4 we can subjectively tell that the image results produced by SRGAN showcase rich details.

5. Conclusion

This thesis provides a complete description of traditional upscaling methods, along with an examination of deep learning approaches. The traditional methods currently employed for image super-resolution include Nearest Neighbour, Bilinear, and Bicubic Interpolation. On the other hand, deep learning-based models such as Image Super-Resolution Convolutional Neural Network (SRCNN), Efficient Sub-Pixel Convolutional Neural Network (ESPCN), Autoencoder for Single Image Super Resolution, and Super Resolution Generative Adversarial Network (SRGAN) have been developed with distinct architectures and have shown promising results. These results are evaluated based on criteria such as reconstruction efficiency, reconstruction accuracy, and perceptual quality.

This dissertation provides a detailed introduction to the topic of Image Interpolation and its techniques implementation and compares the results of the implemented models. Further, a detailed comparison of reconstruction results was displayed.

Based on Chapter 4, it can be inferred that the Autoencoder model has superior performance in terms of metrics evaluation. Autoencoders have been observed to have strong performance in terms of metric evaluation when it comes to minimizing reconstruction error. This implies that they have the capability to generate images that closely resemble high-resolution ground truth photos on a pixel-level basis. Conversely, the SRCNN model exhibits the most computational efficiency among all the models considered. Having the least numbers or convolutional layers the model performs the fastest computation for the upscale of 4 also outperforming the traditional methods. Additionally, The SRGAN model introduces adversarial training by incorporating a discriminator network that serves the purpose of discerning between authentic high-resolution images and the high-resolution images generated by the model. The utilization of an adversarial loss serves to encourage the generated images to possess a high degree of perceptual realism.

Although SRGAN does not demonstrate exceptional performance in conventional metrics such as PSNR, it has remarkable proficiency in generating visually fascinating outcomes characterized by detailed features, textures, and sharpness.

6. Future-Scope

- Enhancing the performance of the models can be achieved by increasing the number of training epochs, varying the batch sizes, and utilizing larger datasets.
- Customizing models with other models to enhance computational efficiency and observe the results.
- Develop a web application or a mobile application to address the challenge of upscaling.
- The field of Image Super Resolution has been the subject of extensive research for a considerable period of time, resulting in the development of several cutting-edge methodologies. The focus of future research should be on the development and implementation of video super-resolution techniques.
- Despite the fascinating outcomes produced by SRGAN, it has not demonstrated notable success in metrics-based assessments. Future research should focus on the investigation of evaluation systems based on metrics for the purpose of enhancing performance.

Bibliography

[1] I. Pekkucuksen and Y. Altunbasak, "Multiscale gradients-based color filter array interpolation," *IEEE Transactions on Image Processing*, vol. 22, no. 1, pp. 157–165, 2012.

[2] T. M. Lehmann, C. Gonner, and K. Spitzer, "Survey: Interpolation methods in medical image processing," *IEEE transactions on medical imaging*, vol. 18, no. 11, pp. 1049–1075, 1999.

[3] S.-L. Chen, H.-Y. Huang, and C.-H. Luo, "A low-cost high-quality adaptive scalar for real-time multimedia applications," *IEEE Transactions on circuits and systems for video technology*, vol. 21, no. 11, pp. 1600–1611, 2011.

[4] A. Giachetti and N. Asuni, "Real-time artifact-free image upscaling," *IEEE Transactions on Image Processing*, vol. 20, no. 10, pp. 2760–2768, 2011.

[5] S. P. Jaiswal, V. Jakhetiya, A. Kumar, and A. K. Tiwari, "A low complex context adaptive image interpolation algorithm for real-time applications," in *2012 IEEE International Instrumentation and Measurement Technology Conference Proceedings*, pp. 969–972, IEEE, 2012.

[6] C.-C. Huang, P.-Y. Chen, and C.-H. Ma, "A novel interpolation chip for real-time multimedia applications," *IEEE Transactions on circuits and systems for video technology*, vol. 22, no. 10, pp. 1512–1525, 2012.

[7] L. Yue, H. Shen, J. Li, Q. Yuan, H. Zhang, and L. Zhang, "Image super-resolution: The techniques, applications, and future," *Signal processing*, vol. 128, pp. 389–408, 2016.

[8] H. Shen, M. K. Ng, P. Li, and L. Zhang, "Super-resolution reconstruction algorithm to modis remote sensing images," *The Computer Journal*, vol. 52, no. 1, pp. 90–100, 2009.

[9] S. Anwar, S. Khan, and N. Barnes, "A deep journey into super-resolution: A survey," *ACM Computing Surveys (CSUR)*, vol. 53, no. 3, pp. 1–34, 2020.

[10] S. L. Tanimoto, *An interdisciplinary introduction to image processing: pixels, numbers, and programs*. MIT Press, 2012.

[11] T. Acharya and A. K. Ray, *Image processing: principles and applications*. John Wiley & Sons, 2005.

[12] S. Lee, "Digital image warping," 1994.

[13] S. H. Mahajan and V. K. Harpale, "Adaptive and non-adaptive image interpolation techniques," in *2015 International Conference on Computing Communication Control and Automation*, pp. 772–775, IEEE, 2015.

[14] V. Patel and K. Mistree, "A review on different image interpolation techniques for image enhancement," *International Journal of Emerging Technology and Advanced Engineering*, vol. 3, no. 12, pp. 129–133, 2013.

[15] S.-L. Chen, "Vlsi implementation of an adaptive edge-enhanced image scalar for real-time multimedia applications," *IEEE Transactions on circuits and systems for video technology*, vol. 23, no. 9, pp. 1510–1522, 2013.

[16] Y. Zhang, Y. Li, J. Zhen, J. Li, and R. Xie, "The hardware realization of the bicubic interpolation enlargement algorithm based on fpga," in *2010 Third International Symposium on Information Processing*, pp. 277–281, IEEE, 2010.

[17] Y. Bengio, I. Goodfellow, and A. Courville, *Deep learning*, vol. 1. MIT press Cambridge, MA, USA, 2017.

[18] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *arXiv preprint arXiv:1511.08458*, 2015.

[19] "Neural network: Architecture, components & top algorithms." https://www.upgrad.com/blog/neural-network-architecture-components-algorithms/.

[20] S. Sharma, "Neural network: Architecture, components & top algorithms | upgrad blog." https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9/.

[21] A. W. Services, "What is amazon machine learning?," Accessed 2023. https://docs.aws.amazon.com/machine-learning/latest/dg/what-is-amazon-machine-learning.html.

[22] M. Mandal, "Introduction to convolutional neural networks (cnn)," May 2021.

[23] H. Yingge, I. Ali, and K.-Y. Lee, "Deep neural networks on chip - a survey," pp. 589–592, 02 2020.

[24] J. Jordan, "Autoencoders." https://www.jeremyjordan.me/autoencoders/.

[25] A. Aggarwal, M. Mittal, and G. Battineni, "Generative adversarial network: An overview of theory and applications," *International Journal of Information Management Data Insights*, vol. 1, no. 1, p. 100004, 2021.

[26] S. Yamaguchi, S. Kanai, A. Kumagai, D. Chijiwa, and H. Kashima, "Transfer learning with pre-trained conditional generative models," *arXiv preprint arXiv:2204.12833*, 2022.

[27] P. Mohammadi, A. Ebrahimi-Moghadam, and S. Shirani, "Subjective and objective quality assessment of image: A survey," *arXiv preprint arXiv:1406.7799*, 2014.

[28] Z. Wang and A. C. Bovik, "A universal image quality index," *IEEE signal processing letters*, vol. 9, no. 3, pp. 81–84, 2002.

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

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

[31] R. Zeyde, M. Elad, and M. Protter, "On single image scale-up using sparse-representations," in *Curves and Surfaces: 7th International Conference, Avignon, France, June 24-30, 2010, Revised Selected Papers 7*, pp. 711–730, Springer, 2012.

[32] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part IV 13*, pp. 184–199, Springer, 2014.

[33] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1874–1883, 2016.

[34] J. Kim, S. Song, and S.-C. Yu, "Denoising auto-encoder based image enhancement for high resolution sonar image," in *2017 IEEE underwater technology (UT)*, pp. 1–5, IEEE, 2017.

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

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

[37] R. Zeyde, M. Elad, and M. Protter, "On single image scale-up using sparse-representations," in *Curves and Surfaces: 7th International Conference, Avignon, France, June 24-30, 2010, Revised Selected Papers 7*, pp. 711–730, Springer, 2012.

[38] P. Raikokte, "Covid-19 image dataset." https://www.kaggle.com/datasets/pranavraikokte/covid19-image-dataset, April 2020. Kaggle Dataset.

[39] R. J. Hammell, "Planes in satellite imagery." https://www.kaggle.com/datasets/rhammell/planesnet, January 2023. Kaggle Dataset.

List of Figures

List of Tables

List Of Acronyms

SR Super Resolution

CNN Convolutional Neural Network

DCT Discrete Cosine Transform

DFT Discrete Fourier Transform

SRGAN Super Resolution Generative Adversarial Network

SRCNN Super Resolution Convolutional Neural Network

ESPCN Efficient Sub-Pixel Convolutional Network

LR Low Resolution

PSNR Peak Signal to Noise Ratio

SSIM Structural Similarity Index Method

UQI Universal Image Quality Index

HR High Resolution

UAV Unmanned Aerial Vehicle

CT Computed Tomography

MRI Magnetic Resonance Imaging

PET Positron Emission Tomography

DVR Digital video recorder

MSE Mean Square Error

CCD Charge-Coupled Device

CMOS Complementary Metal Oxide Semiconductor

Declaration

I hereby declare that I completed this work independently and that I have used no aids other than those referenced.

The parts of the work, which include phrases or points taken from other sources, are clearly marked with the origin of the information. This also applies to diagrams, sketches, visual representations as well as for sources from the internet.

I also declare that I have not submitted this work in any other testing procedure as an examination paper, nor will in the future.

The submitted written work corresponds to the electronic version. I agree that an electronic copy may be made and stored to enable verification by anti-plagiarism software.

Place, Date

Signature