IMAGE SUPER RESOLUTION

Faculty of Computers and Artificial Intelligence

Cairo University

Under the supervision of

Prof. Motaz Elsaban

TA. Asmaa Ahmed

Seif Gad	**20180128**
Sherif Magdy Abdellah	**20180133**
Mohab Abdelsallam	**20180294**
Hagar Nabil Anwar	**20180325**
Rana Khaled Abdelazem	**20180103**

Table of Contents

Contents

1. Overview

In this section we are aiming to give an overview of the Image Super Resolution (ISR) domain, the idea behind it, our motivation, research questions that we are seeking to answer, problem definition in general, contributions to a lot of people in this idea and finally our plan to answer these questions.

1.1. Introduction

Image Super Resolution refers to the task of enhancing the resolution of an image from low-resolution (LR) to high (HR) [1]. It is an important class of image processing techniques in computer vision and image processing and enjoys a wide range of real-world applications.

Figure: An Example of low-resolution image (left) and its high-resolution (right) variant [1]

With huge improvement in deep learning techniques in recent years, deep learning-based SR models have been actively explored and often achieve state-of-the-art performance on various benchmarks of SR. A variety of deep learning methods have been applied to solve SR tasks, ranging from the early Convolutional Neural Networks (CNN) [2] based method to recent promising Generative Adversarial Nets based SR approaches [3] [4] [5].

1.2. Datasets

Today there are a variety of datasets available for image super-resolution. They differ in image amounts, quality, resolution, and diversity, etc. Some of them provide LR-HR image

pairs, while others only provide HR images. The DIV2K dataset consists of 800 training images, 100 validation images, and 100 test images.

Dataset	Amount	Avg. Resolution	Avg. Pixels	Format	Category Keywords
BSDS300 [40]	300	(435, 367)	154, 401	JPG	animal, building, food, landscape, people, plant, etc.
BSDS500 [41]	500	(432, 370)	154, 401	JPG	animal, building, food, landscape, people, plant, etc.
DIV2K [42]	1000	(1972, 1437)	2, 793, 250	PNG	environment, flora, fauna, handmade object, people, scenery, etc.
General-100 [43]	100	(435, 381)	181, 108	BMP	animal, daily necessity, food, people, plant, texture, etc.
L20 [44]	20	(3843, 2870)	11, 577, 492	PNG	animal, building, landscape, people, plant, etc.
Manga109 [45]	109	(826, 1169)	966, 011	PNG	manga volume
OutdoorScene [46]	10624	(553, 440)	249, 593	PNG	animal, building, grass, mountain, plant, sky, water
PIRM [47]	200	(617, 482)	292, 021	PNG	environments, flora, natural scenery, objects, people, etc.
Set5 [48]	5	(313, 336)	113, 491	PNG	baby, bird, butterfly, head, woman
Set14 [49]	14	(492, 446)	230, 203	PNG	humans, animals, insects, flowers, vegetables, comic, slides, etc.
T91 [21]	91	(264, 204)	58, 853	PNG	car, flower, fruit, human face, etc.
Urban100 [50]	100	(984, 797)	774, 314	PNG	architecture, city, structure, urban, etc.

Figure: Datasets for super resolution [1]

1.3. Image Quality Assessment

Image quality refers to visual attributes of images and focuses on the perceptual assessments of viewers. In general, image quality assessment (IQA) methods include subjective methods based on humans' perception (i.e., how realistic the image looks) and objective computational methods.

1.3.1. Peak Signal-to-Noise Ratio

Signal to Noise Ratio (PSNR) [6] is the most common technique used to determine the quality of results. It can be calculated directly from the MSE using the formula below, where L is the maximum pixel value possible.

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(I(i) - \hat{I}(i))^2,$$

$$\mathrm{PSNR} = 10 \cdot \log_{10}(\frac{L^2}{\mathrm{MSE}}).$$

1.3.2. Structural Similarity Index Measure

Structural Similarity Index Measure (SSIM) [7] is used to compare the perceptual quality of two images using the formula below, with the mean (μμ), variance (σσ), and correlation (c) of both images.

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

1.4. Upsampling Methods

There are multiple methods for performing image upsampling [8], we list them as follows:

1. Nearest-neighbour Interpolation [9]: The nearest-neighbour interpolation is a simple and intuitive algorithm. It selects the value of the nearest pixel for each position to be interpolated regardless of any other pixels.

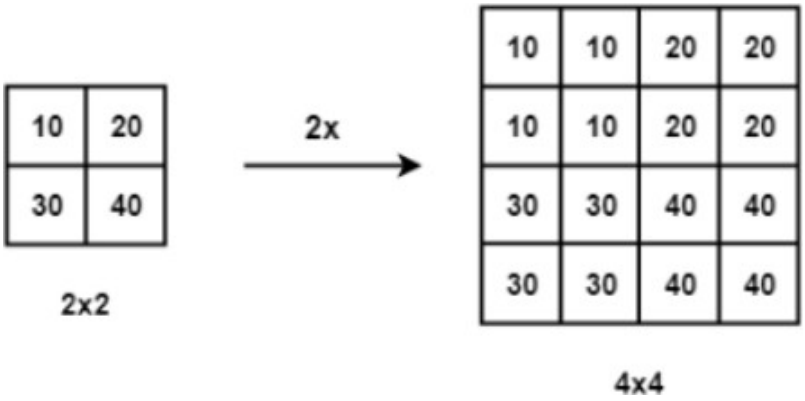


Figure: Nearest-neighbour interpolation with scale 2

2. Bilinear Interpolation [10]: The bilinear interpolation (BLI) first performs linear interpolation on one axis of the image and then performs on the other axis. Since it results in a quadratic interpolation with a receptive field-size 2 × 2, it shows much better performance than nearest-neighbour interpolation while keeping a relatively fast speed.
3. Bicubic Interpolation [11] – Similarly, the bicubic interpolation (BCI) performs cubic interpolation on each of the two axes Compared to BLI, the BCI takes 4 × 4 pixels into account, and results in smoother results with fewer artifacts but much lower speed. Refer to this for a detailed discussion.

1.5. Motivation

Image super resolution is used in many real-world applications that we introduce some of them as follows.

- **Surveillance** [12] [13] [14]**:** to detect, identify, and perform facial recognition on low-resolution images obtained from security cameras.

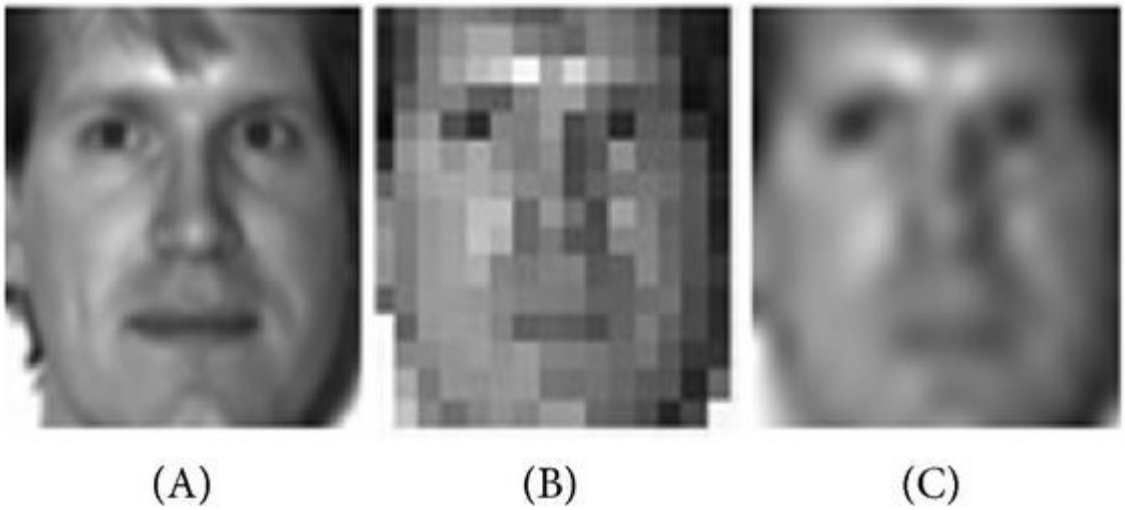


Figure: An Example for facial recognition on low-resolution images (B, C) [14]

- **Medical application** [15] [16] [17]**:** capturing high-resolution MRI images can be tricky when it comes to scan time, spatial coverage, and signal-to-noise ratio (SNR). Super resolution helps resolve this by generating high-resolution MRI from otherwise low-resolution MRI images.

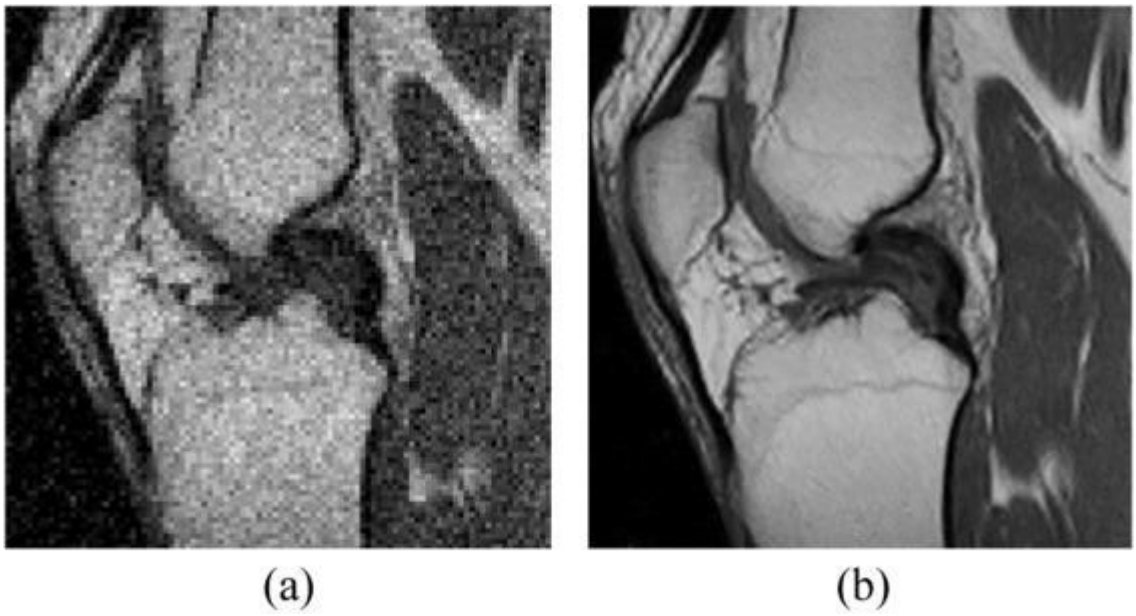


Figure: An Example for generating high-resolution MRI (b) from low-resolution (a)

- **Media streaming** [18] [19]**:** super resolution can be used to reduce server costs, as media can be sent at a lower resolution and upscaled on the fly.

1.6. Problem Definition

Image super-resolution aims at recovering the corresponding HR images from the LR images. Generally, the LR image I_x is modelled as the output of the following degradation:

$$I_x = D(I_y; \delta), \qquad (1)$$

where D denotes a degradation mapping function, I_y is the corresponding HR image and δ is the parameters of the degradation process (e.g., the scaling factor or noise).

Generally, the degradation process (i.e.,D and δ) is unknown and only LR images are provided. In this case, also known as blind SR, researchers are required to recover an HR approximation $\hat{I}_y$of the ground truth HR image I_y from the LR image I_x, following:

$$\hat{I}_y = F(I_x; \theta), \qquad (2)$$

where F is the super-resolution model and θ denotes the parameters of F.

Although the degradation process is unknown and can be affected by various factors (e.g., compression artifacts, anisotropic degradations, sensor noise and speckle noise), researchers are trying to model the degradation mapping. Most works directly model the degradation as a single downsampling operation, as follows:

$$D(I_y; \delta) = (I_y) \downarrow_s, \{s\} \subset \delta, \qquad (3)$$

where $\downarrow_s$ is a downsampling operation with the scaling factor s. As a matter of fact, most datasets for generic SR are built based on this pattern, and the most commonly used downsampling operation is bicubic interpolation with antialiasing. However, there are other works modelling the degradation as a combination of several operations:

$$D(I_y; \delta) = (I_y \otimes \kappa) \downarrow s + n_\varsigma, \{\kappa, s, \varsigma\} \subset \delta, \qquad (4)$$

where $I_y \otimes \kappa$ represents the convolution between a blur kernel κ and the HR image I_y, and $n_\varsigma$is some additive white Gaussian noise with standard deviation ς. Compared to the naive definition of Eq. 3, the combinative degradation pattern of Eq. 4 is closer to real-world cases and has been shown to be more beneficial for SR.

To this end, the objective of SR is as follows:

$$\hat{\theta} = arg_\theta \, min \, L(\hat{I}_y, I_y) + \lambda \Phi(\theta), \qquad (5)$$

where $L(\hat{I}_y, I_y)$ represents the loss function between the generated HR image $\hat{I}_y$and the ground truth image I_y, $\Phi(\theta)$ is the regularization term and λ is the tradeoff parameter.

2. Background

In this section, we overview background concepts related to deep image super resolution models.

2.1. Deep Neural Networks

Neural networks (NNs) form a branch of machine learning models that aims at resembling the working mechanisms of biological neural networks [20]. They can be broadly described as "computing systems made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs." [21]. When NNs get stacked in large number they could be described as Deep Neural Networks (DNNs). We explore the main building blocks and different types of DNNs as follows.

2.1.1. Neural Network Elements

The layers of the neural network are made of something called Nodes which resembles the neuron in the human brain. It fires when there is sufficient stimuli.

The simple neural network consists of an input layer, hidden layer and an output layer. The input layer takes the input vector and multiplies it by a specific weight for each node then takes the sum of the multiplications. This summation is then passed to an activation function to decide to which extend the likelihood this signal is passed to the next layer.

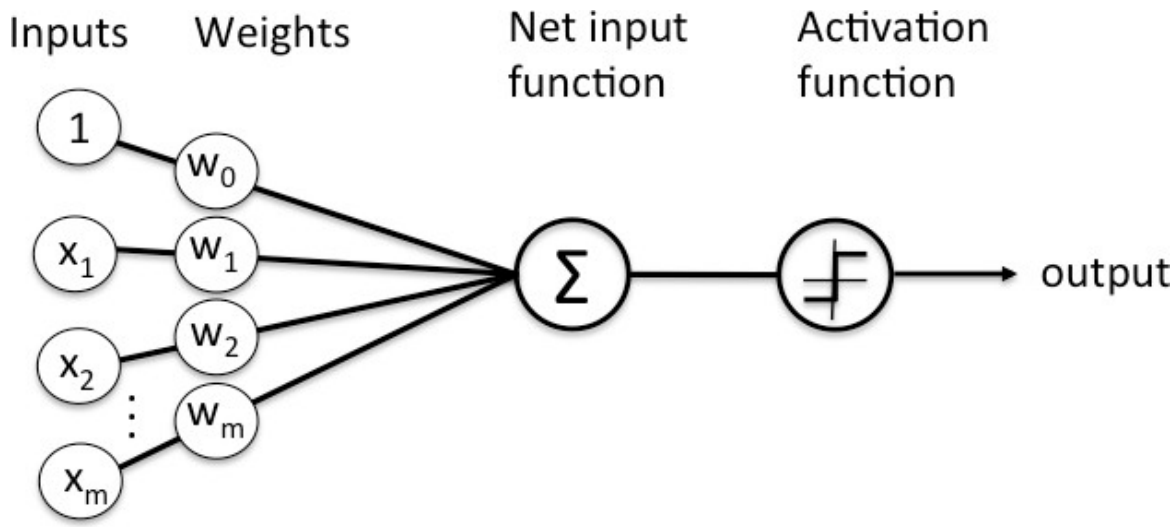


Figure: Neural Network Elements

Deep neural networks are neural networks that have more than 1 hidden layer.

2.1.2. How Neural Networks Work

For a neural network to learn it has to have feedback. This is the job of gradient descent. Gradient is another word for slope, and slope, in its typical form on an x-y graph, represents how two variables relate to each other. In this particular case, **the slope we care about describes the relationship between the network's error and a single weight**, i.e. that is,

how does the error vary as the weight is adjusted. To put a finer point on it, which weight will produce the least error? Which one correctly represents the signals contained in the input data, and translates them to a correct classification? Which one can hear “nose” in an input image, and know that should be labelled as a face and not a frying pan?

As a neural network learns, it slowly adjusts many weights so that they can map signal to meaning correctly. The relationship between network *Error* and each of those *weights* is a derivative, *dE/dw*, that measures the degree to which a slight change in a weight causes a slight change in the error.

Each weight is just one factor in a deep network that involves many transforms; the signal of the weight passes through activations and sums over several layers, so we use the chain rule of calculus to march back through the networks activations and outputs and finally arrive at the weight in question, and its relationship to overall error.

The chain rule in calculus states that

$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx}.$$

Figure: Chain Rule

In a feedforward network, the relationship between the net’s error and a single weight will look something like this:

$$\frac{dError}{dweight} = \frac{dError}{dactivation} * \frac{dactivation}{dweight}$$

Figure: The Relationship Between The Nets’s Error and a Single Weight

That is, given two variables, *Error,* and *weight*, that are mediated by a third variable, *activation*, through which the weight is passed, you can calculate how a change in *weight* affects a change in *Error* by first calculating how a change in *activation* affects a change in *Error*, and how a change in *weight* affects a change in *activation*.

The essence of learning in deep learning is nothing more than that: adjusting a model’s weights in response to the error it produces, until you can’t reduce the error anymore.

2.2. Visual Perception in Humans

Vision begins in the eye which receives the inputs, in the form of light, and finished in the brain which interprets those inputs and gives us the information we need from the data we receive. The components of the eye are pictured below.

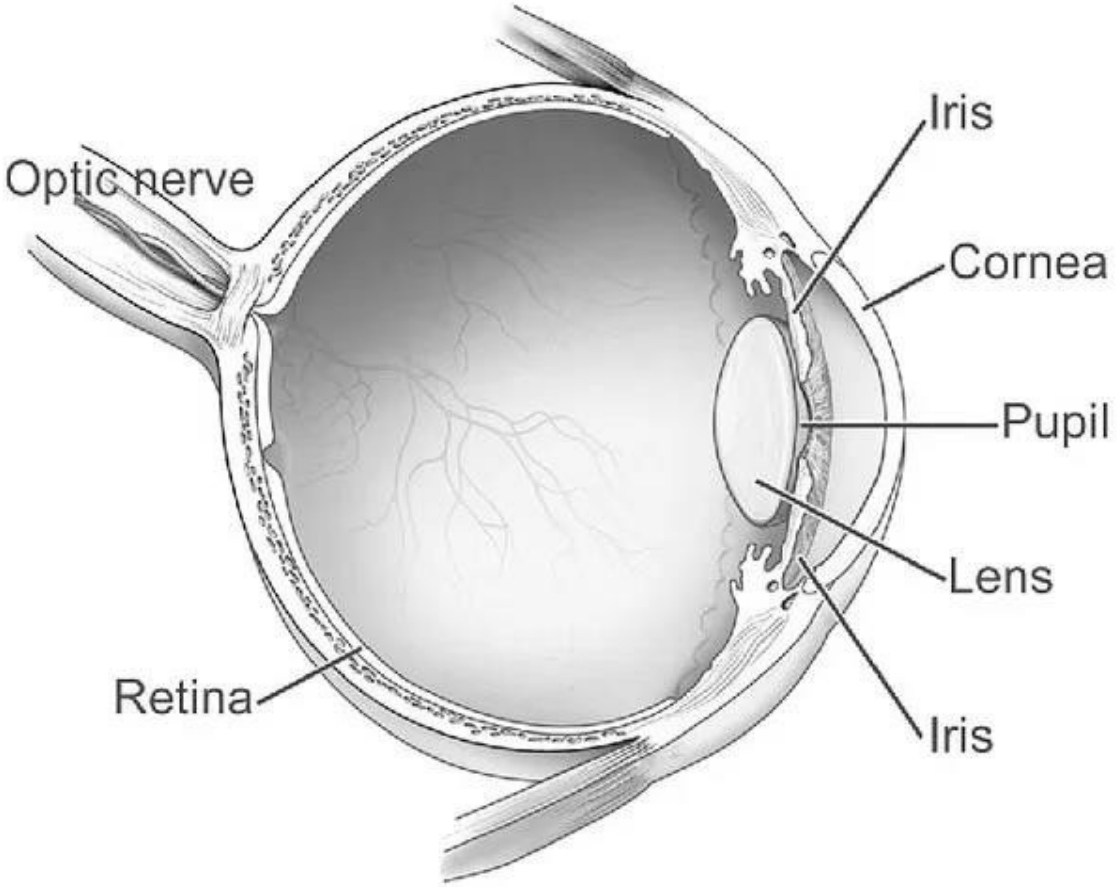


Figure: The Components of The Eye

The eye focuses light on the retina. In the retina there is a layer of photoreceptor (light receiving) cells which are designed to change light into a series of electrochemical signals to be transmitted to the brain [22]. There are two types of photoreceptors – rods and cones.

Rods tend to be found in the peripheral areas of the retina and are designed to respond to low levels of light. They are responsible for our night vision and because of where they are placed on the retina – you can improve your night vision by learning to focus slightly to the side of whatever you are looking at; allowing the light to reach the rod cells most successfully.

Cones cells are found in the fovea (the center of the retina); cone cells handle the high acuity visual tasks such as reading and color vision. Cone cells respond to red, green or blue light and by combining the signals from these three receptors we can perceive a full range of color.

Once the light has been processed by the photoreceptors an electrochemical signal is then passed via a network of neurons to the ganglion cells further back in the retina. The neurons are designed to help detect the contrasts within an image (such as shadow or edges) and the ganglion cells record this (and other information) and pass an amended electrochemical signal, via the optic nerve, to the brain.

2.3. Image Convolution

2.3.1. What Is Convolution?

Convolution is a general-purpose filter effect for images. It is a matrix applied to an image and a mathematical operation composed of integers.

2.3.2. The Process of Image Convolution

A convolution is done by multiplying a pixel's and its neighboring pixels' color value by a matrix. A kernel is a (usually) small matrix of numbers that is used in image convolutions. Differently sized kernels containing different patterns of numbers produce different results under convolution [2]. Convolution is used to smooth, enhance, sharpen the image and many other purposes.

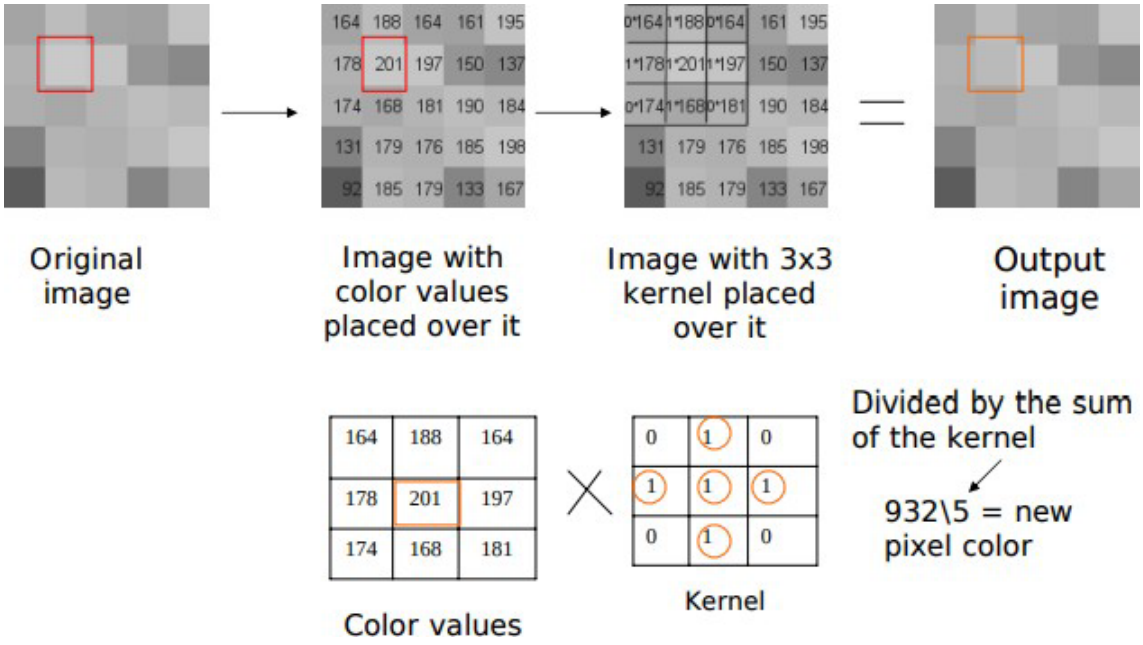


Figure: An Example of Image Convolution

2.4. Convolution Neural Network

A Convolutional Neural Network is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other.

2.4.1. Why ConvNets Over Feed-Forward Neural Nets?

In cases of extremely basic binary images, the method might show an average precision score while performing prediction of classes but would have little to no accuracy when it comes to complex images having pixel dependencies throughout. A ConvNet can **successfully capture the Spatial and Temporal dependencies** in an image through the application of relevant filters.

2.4.2. Convolution Layer - The Kernel

In the figure, we have an RGB image which has been separated by its three-color planes — Red, Green, and Blue. You can imagine how computationally intensive things would get once the images reach dimensions, say 8K (7680×4320). The role of the ConvNet is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction.

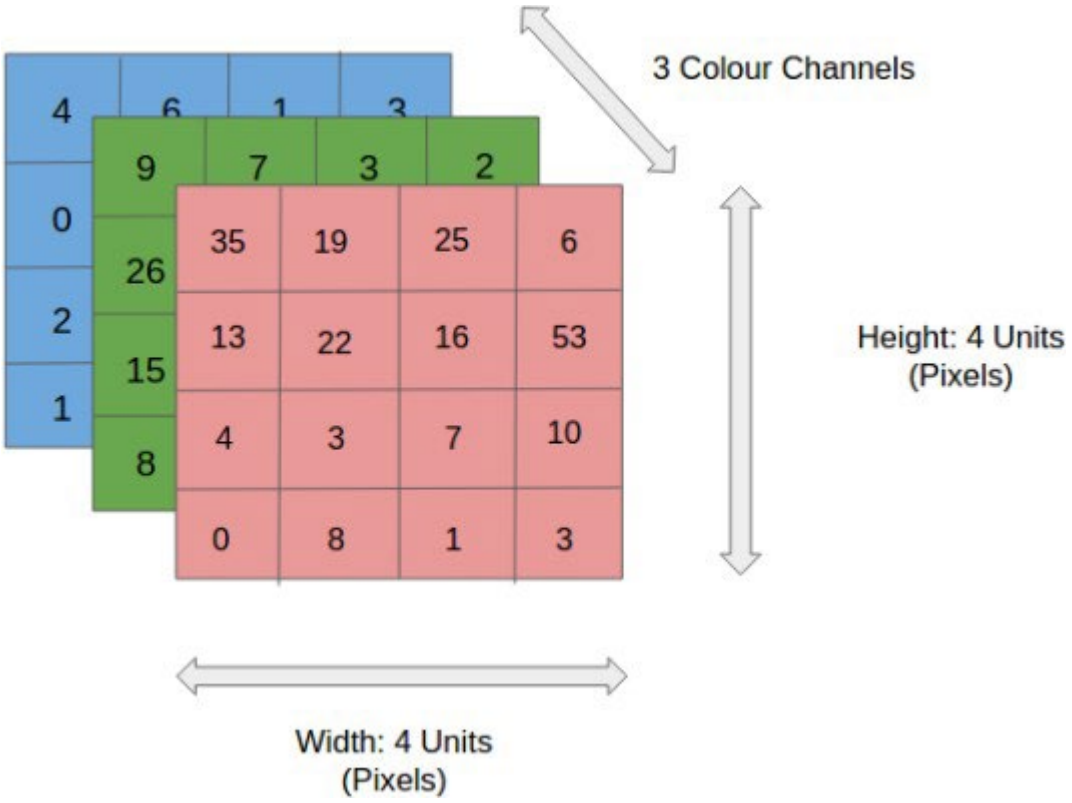


Figure: 4x4x3 RGB Image

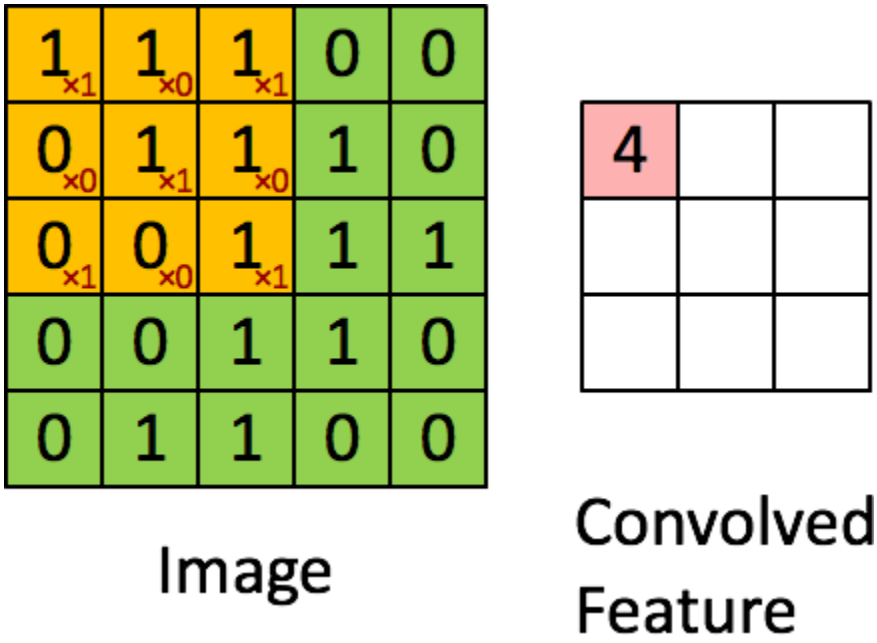


Figure: Convoluting a 5x5x1 Image with a 3x3x1 Kernel to get a 3x3x1 Convolved Feature

In the above demonstration, the green section resembles our 5x5x1 input image, I. The element involved in carrying out the convolution operation in the first part of a Convolutional Layer is called the Kernel/Filter, K, represented in the color yellow. We have selected K as a 3x3x1 matrix. The Kernel shifts 9 times because of Stride Length = 1 (Non-Strided), every time performing a matrix multiplication operation between K and the portion P of the image over which the kernel is hovering. The filter moves to the right with a certain Stride Value till it parses the complete width. Moving on, it hops down to the beginning (left) of the image with the same Stride Value and repeats the process until the entire image is traversed.

There are two types of results to the operation — one in which the convolved feature is reduced in dimensionality as compared to the input, and the other in which the dimensionality is either increased or remains the same. This is done by applying Valid Padding in case of the former, or Same Padding in the case of the latter.

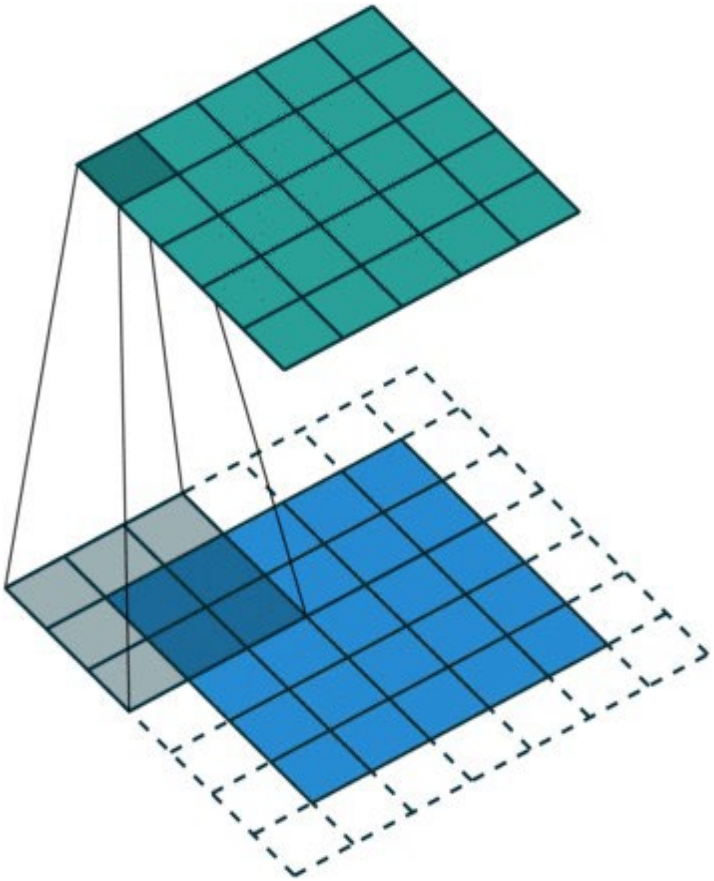


Figure: Same Padding 5x5x1 Image is Padded With 0s to Create a 6x6x1 Image

When we augment the 5x5x1 image into a 6x6x1 image and then apply the 3x3x1 kernel over it, we find that the convolved matrix turns out to be of dimensions 5x5x1. Hence the name — Same Padding. On the other hand, if we perform the same operation without padding, we are presented with a matrix which has dimensions of the Kernel (3x3x1) itself — Valid Padding.

2.4.3. Pooling Layer

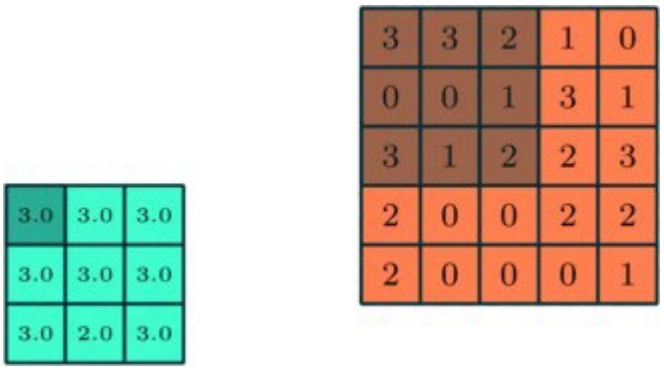


Figure: 3x3 Pooling Over 5x5 Convolved Feature

Like 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 and Average Pooling**. Max 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 performs de-noising along with dimensionality reduction. On the other hand, 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.

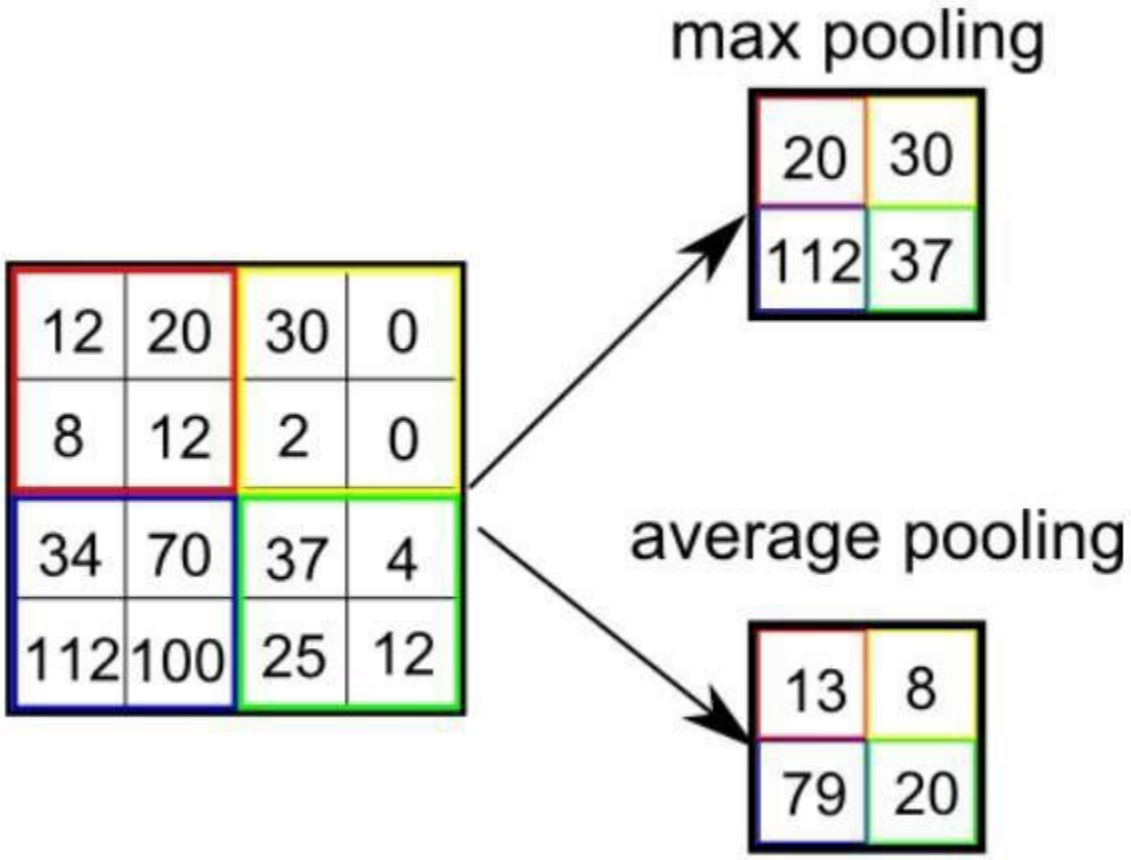


Figure: Types of Pooling

The Convolutional Layer and the Pooling Layer, together form the i-th layer of a Convolutional Neural Network. Depending on the complexities in the images, the number of such layers may be increased for capturing low levels details even further, but at the cost of more computational power.

After going through the above process, we have successfully enabled the model to understand the features. Moving on, we are going to flatten the final output and feed it to a regular Neural Network for classification purposes.

2.4.4. Classification - Fully Connected Layer (FC Layer)

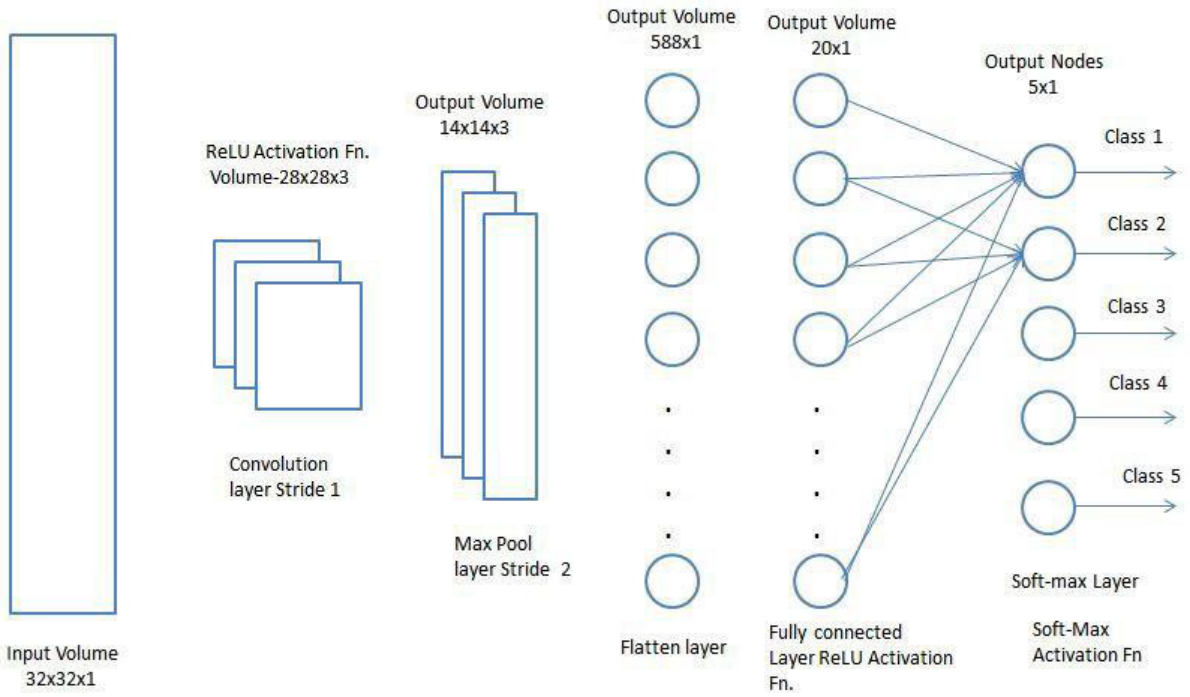


Figure: An example of a Fully Connected Neural Network

Adding a Fully Connected layer is a (usually) cheap way of learning non-linear combinations of the high-level features as represented by the output of the convolutional layer. The Fully Connected layer is learning a possibly non-linear function in that space.

Now that we have converted our input image into a suitable form for our Multi-Level Perceptron, we shall flatten the image into a column vector. The flattened output is fed to a feed-forward neural network and backpropagation applied to every iteration of training. Over a series of epochs, the model can distinguish between dominating and certain low-level features in images and classify them using Softmax Classification technique.

2.5. Residual Neural Networks

Residual Networks (ResNets) is one of the famous deep learning models that was introduced by He et al. [23]. This type of networks use a technique called skip connections. The skip connection skips training from a few layers and connects directly to the output.

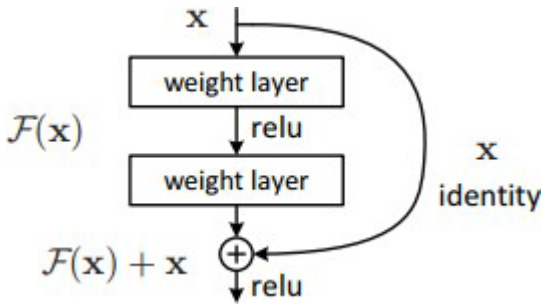


Figure:: Residual block [23]

The very first thing we can notice is that there is a direct connection that skips some layers of the model. This connection is called 'skip connection' and is the heart of residual blocks. The output is not the same due to this skip connection. Without the skip connection, input 'X gets multiplied by the weights of the layer followed by adding a bias term.

Then comes the activation function, f(x) and we get the output as H(x).

H(x)=f(wx + b) or H(x)=f(x)

Now with the introduction of a new skip connection technique, the output is H(x) is changed to H(x)=f(x)+x

But the dimension of the input may be varying from that of the output which might happen with a convolutional layer or pooling layers. Hence, this problem can be handled with these two approaches:

- Zero is padded with the skip connection to increase its dimensions.
- 1×1 convolutional layers are added to the input to match the dimensions.

These skip connections technique in ResNet solves the problem of vanishing gradient in deep CNNs by allowing alternate shortcut path for the gradient to flow through. Also, the skip connection helps if any layer hurts the performance of architecture, then it will be skipped by regularization.

The below graphs compare the accuracies of a plain network with that of a residual network. Note that with increasing layers a 34-layer plain network's accuracy starts to saturate earlier than ResNet's accuracy.

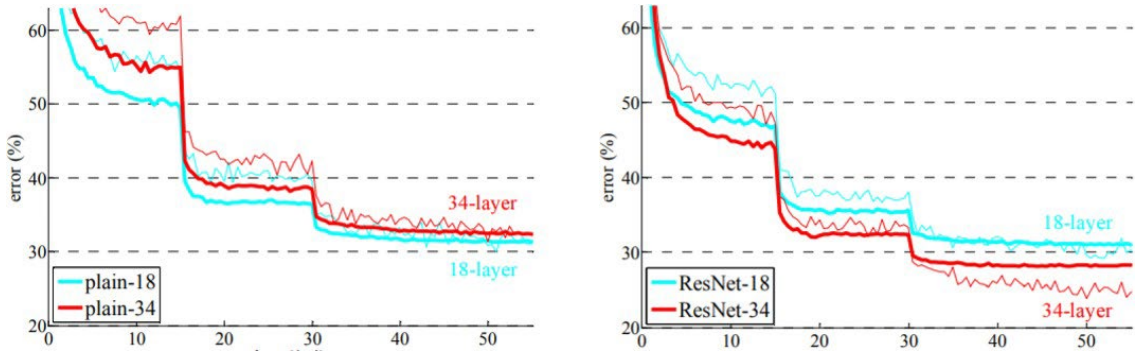


Figure: Difference between normal network and ResNet

3. Literature Review

There are many different methods used to solve the Super Resolution problem. These methods can be categorized according to different criteria. Some methods use pre-upsampling, others use post-upsampling. The network can be residual, recursive, attention-based, or even generative. In this chapter we will talk about each method and understand how it works thoroughly.

3.1. Pre-Upsampling SR

In these methods the feature extraction process occurs in the high-resolution space which requires lots of computational power. They mostly use traditional techniques like bicubic interpolation and deep learning. The most popular method using pre-upsampling is Super-Resolution Using Deep Convolutional Networks (SRCNN) [24].

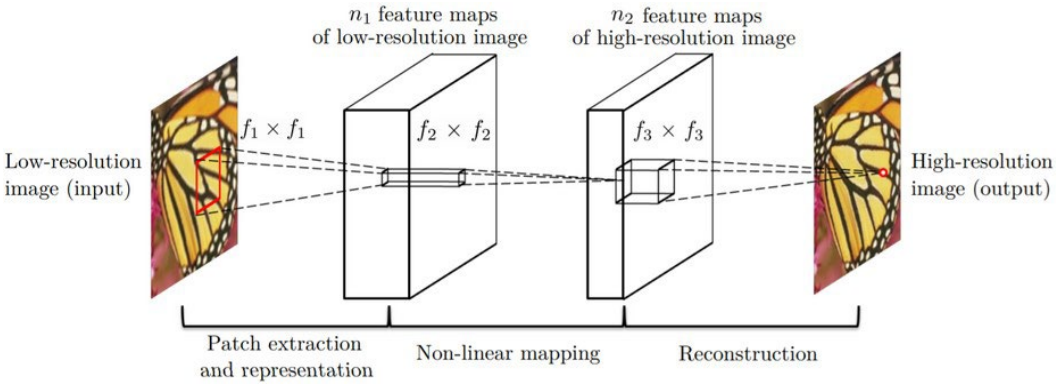


Figure: Visualization for SRCNN architecture [24]

As shown in the figure above, SRCNN consists of three layers. The first layer "Patch Extraction" is expressed by this operation: $F1(Y) = max\ (0, W1 * Y + B1)$.
It extracts dense patches from the low-resolution input and represents them using convolutional filters as high-dimensional vectors. The second layer "Non-linear Mapping" is expressed by this operation: $F2(Y) = max\ (0, W2 * F1(Y) + B2)$.
In this layer, each high-dimensional vector is non-linearly mapped onto another high-dimensional vector. Thus, each mapped vector is the representation of a high-resolution patch. The third layer "Reconstruction" is expressed by this operation: $F(Y) = W3 * F2(Y) + B3$.
This final layer aggregates the above patches to generate the final high-resolution image.
It uses the MSE as a loss function and PSNR as a metric to evaluate the results.

3.2. Post-Upsampling SR

As opposed to Pre-Upsampling, in the Post-Upsampling methods they do the feature extraction process in low-resolution space to solve the problems in Pre-Upsampling methods regarding the high computation required. Upsampling is done using deconvolution or sub-pixel convolution. One of the popular techniques using it is FSRCNN [25].

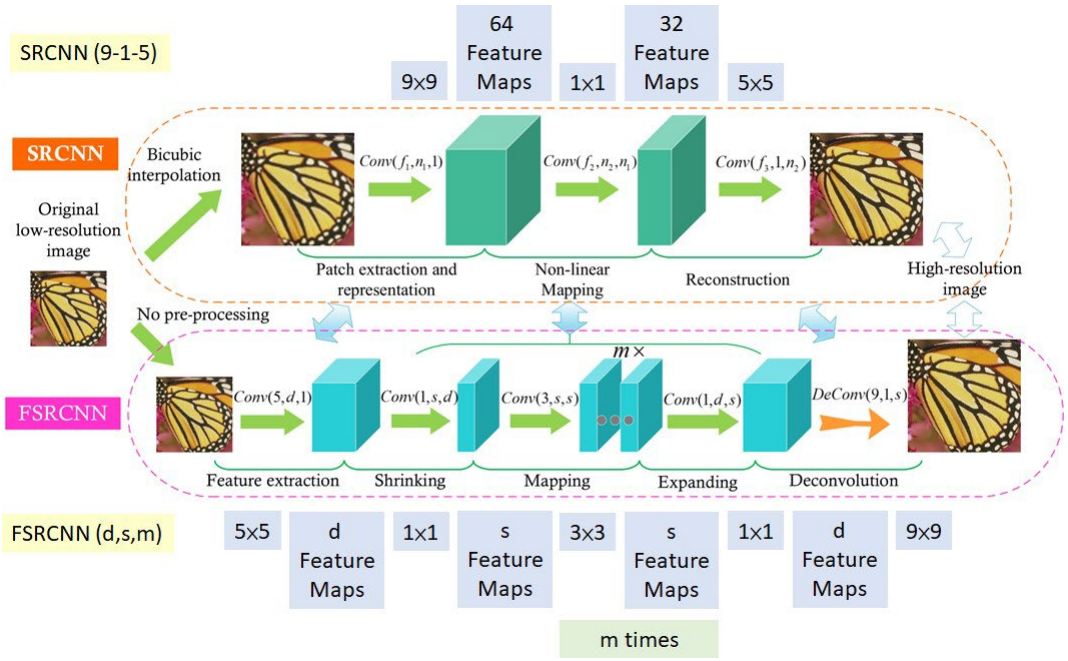


Figure: Different Structures of SRCNN and FSRCNN [25]

From the figure above, we can see that FRSCNN has five layers. The first layer "Feature Extraction" unlike SRCNN, FSRCNN does no pre-processing to the image and performs feature extraction on the original image in low-resolution space. The second layer "Shrinking" helps in reducing the low-resolution dimension after feature extraction, this leads to great reduction in the number of parameters used. The third layer "Mapping", this step affects the SR performance the most and the most influencing factors are the number of layers(depth) and number of filters in a layer(width). To achieve consistency all mapping layers contain the same number of filters. The fourth layer "Expanding" is the inverse process of the Shrinking layer, if we generate the high-resolution image from those shrinked low-dimensional features the restoration will be of poor quality. Thus, we have to expand the high-resolution feature dimension first. The fifth layer "Deconvolution" upsamples and aggregates the previous features to generate the final image. FSRCNN is faster and gives better results than SRCNN.

Original / PSNR

FSRCNN / **33.85** dB

3.3. Multi-Stage Residual Networks

This technique works on splitting the feature extraction process into two stages, one in the low-resolution space and the other in high-resolution space. The first stage predicts the features and the second one improves them. This helps in reducing the computational power. One of the models which use this technique is Balanced Two-Stage Residual Networks (BTSRN) [26].

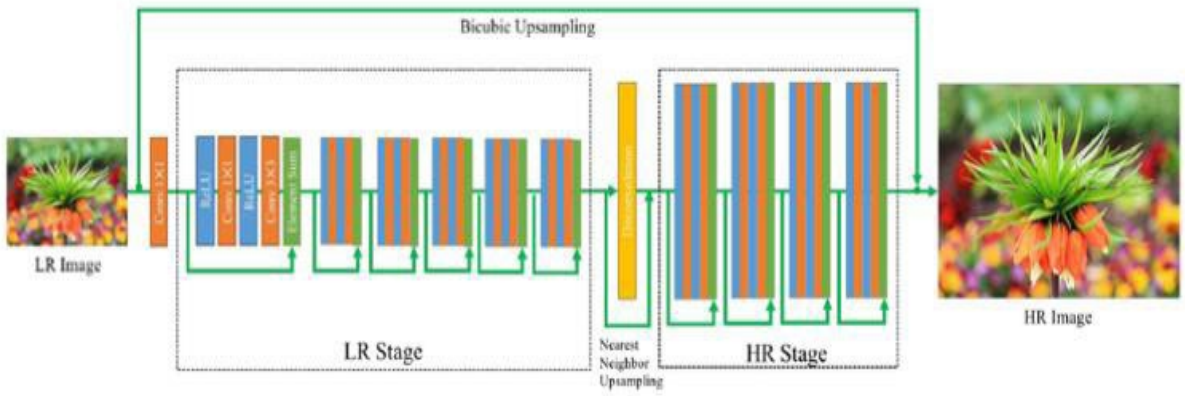


Figure 3: Visualization of BTSRN architecture [26]

BTSRN mainly consists of two stages one in the low-resolution (LR) space and the other in high-resolution(HR). There are 6 residual blocks in the LR space and 4 in the HR. The number of blocks in each stage are decided according to a trade-off between performance and accuracy. The feature maps' sizes are small in the LR stage to prevent high computation. The LR stage's output is upsampled by deconvolution and neighbouring upsampling then sent to the HR stage. Although batch normalization is commonly used in many tasks, in our case it's not needed. Since we found that it makes this network invariant to data re-scaling.

Table 1: Different number of blocks in HR stage

High Stage	PSNR (dB)
3	34.14024
4	34.16181
5	34.12123

Table 2: Different number of blocks in LR stage

Low Stage	PSNR (dB)
5	34.15799
6	34.16181
7	34.15126
8	34.16568
9	34.18892

Original / PSNR

BTSRN / 37.75 dB

3.4. Attention-Based Networks

Throughout our discussion of all previous networks, we saw that they give equal importance to each element in them. Upon further investigation and testing, they found out that giving selective attention to different parts in an image can help enhance the results. One of the networks that uses this technique is Selection Units for Super-Resolution (SelNet) [27].

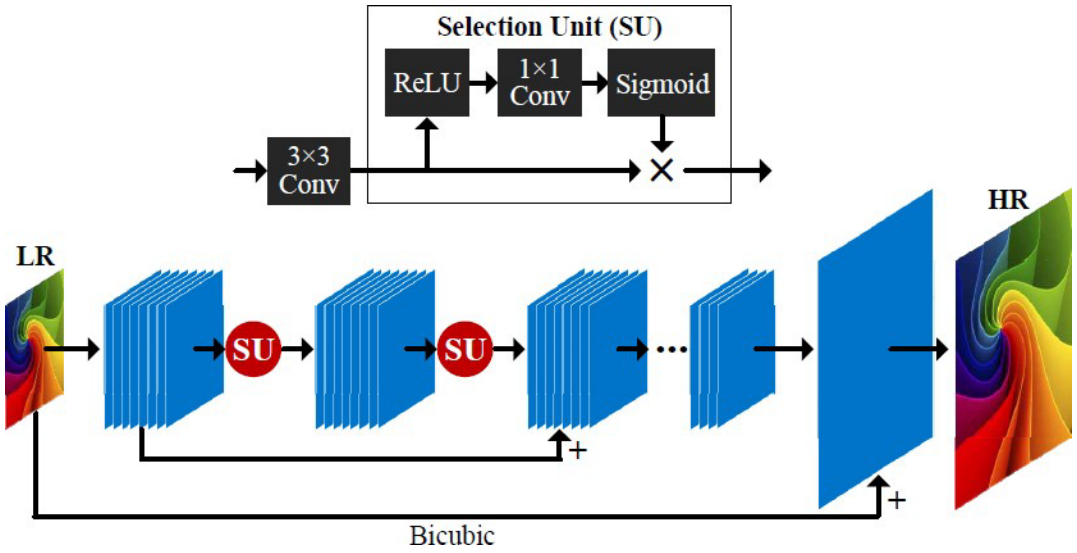


Figure 4: Visualization of SelNet architecture [27]

Normally, ReLU is used to help ensure non-linearity between convolutional layers. But a downside in using ReLU is that its training error can't be back-propagated and thus has very limited control over which data passes. To solve this issue, SelNet proposes a non-linear unit called *Selection Unit (SU)*. This SU is a product of identity mapping multiplied by *Selection Module (SM)* which is inserted between every two convolutional layers. As shown in figure 4, SM is a cascade of a ReLU, 1x1 Conv and a sigmoid. Using this new model, training error can be back-propagated. SelNet consists of a 22-layered deep CNN that ensures a better PSNR results.

Original / PSNR

SelNet / 38.89 dB

4. Methodology

In this section, we will discuss the method we chose to focus on. We have seen in the previous chapter that there are lots of great models and techniques that vary in their complexity, some being too simple and others too complicated. We chose Enhanced Deep Residual Networks for Single Image Super-Resolution (EDSR) [28] due to many reasons; we made sure to choose a network that isn't extremely complex nor too simple, it also has a reasonably good PSNR in comparison to the other networks out there. Let us jump straight into it.

4.1. EDSR

EDSR is a proposed network following the technique of Residual Networks. It is based on the SRResNet architecture but with a few alterations and modifications. The evaluation of this network that will be discussed here is made on the DIV2K dataset.T

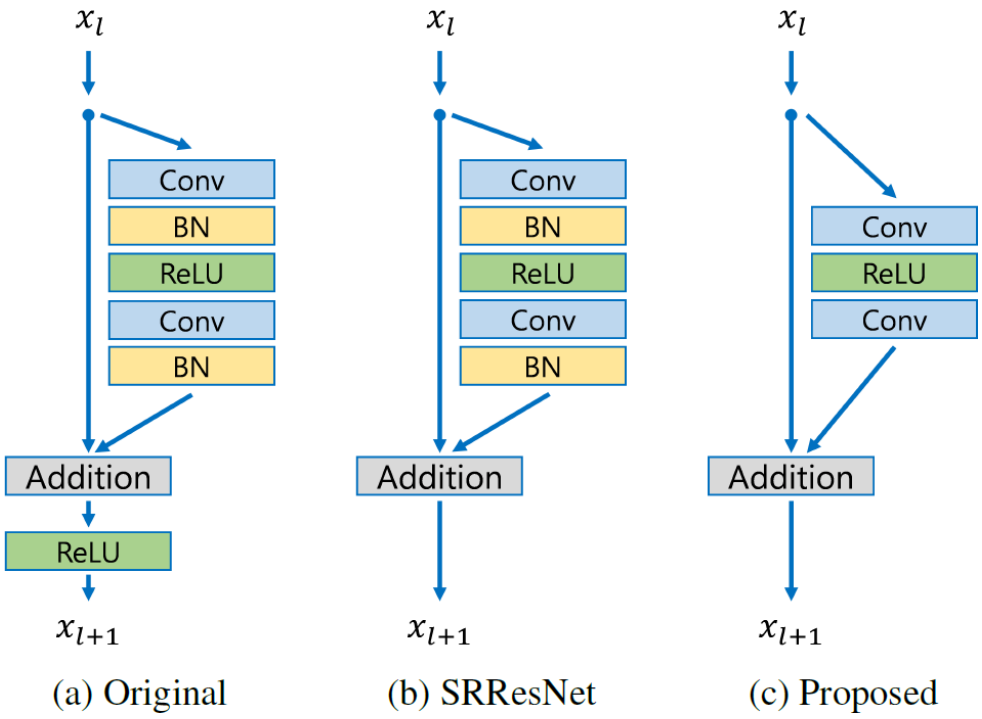


Figure: Architecture of (a)Original ResNet, (b)SRResNet and (c)Proposed Model

It was found that batch normalization gets rid of the flexibility of the network by normalizing the features and it also consumes the same amount of memory as the convolution layers that precedes it. So, it was better to remove them as in the figure above (c). After removing batch normalization, in comparison to SRResNet 40% of the memory usage was saved. This network proposes a single-scale and a multi one. In our project we will focus on the single scale EDSR. In a normal CNN it has B number of layers and F number of feature channels, it occupies $O(BF)$ memory and $O(BF^2)$ parameters. It was found out that increasing the number of features to a certain number makes the training process unstable. This problem was solved in this network by adding residual scaling with factor 0.1 after the last convolution layers in a residual block. This leads to stability in training when using many filters.

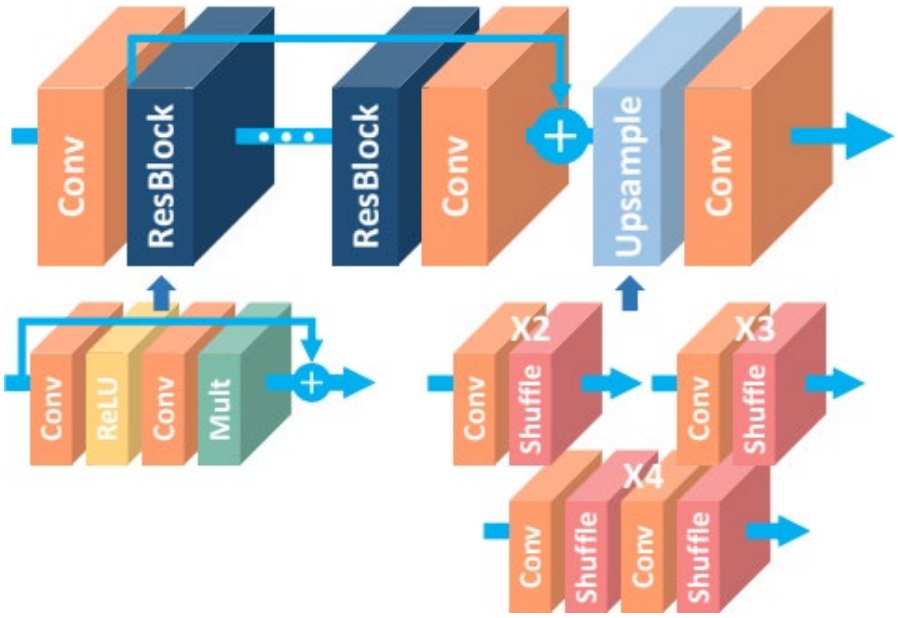


Figure 6: Visualization of Single-Scale SR Architecture [28]

Let's talk about the differences made in this network, ReLU activation function is removed after each residual block as opposed to the Original ResNet and also Batch Normalization is removed inside the residual blocks as discussed above. As seen above in Figure 6, each residual block consists only of Convolution, ReLU then Convolution. Moving to the loss function, this model uses L1 loss (Least Absolute Deviations) instead of L2 loss(Least Square Error). Although it's known that L2 is usually more preferable as it maximizes PSNR. In this model L1 provides better convergence.

Options	SRResNet [14] (reproduced)	Baseline (Single / Multi)	EDSR	MDSR
# Residual blocks	16	16	32	80
# Filters	64	64	256	64
# Parameters	1.5M	1.5M / 3.2M	43M	8.0M
Residual scaling	-	-	0.1	-
Use BN	Yes	No	No	No
Loss function	L2	L1	L1	L1

Table 2: Comparing different models and their specifications [28].

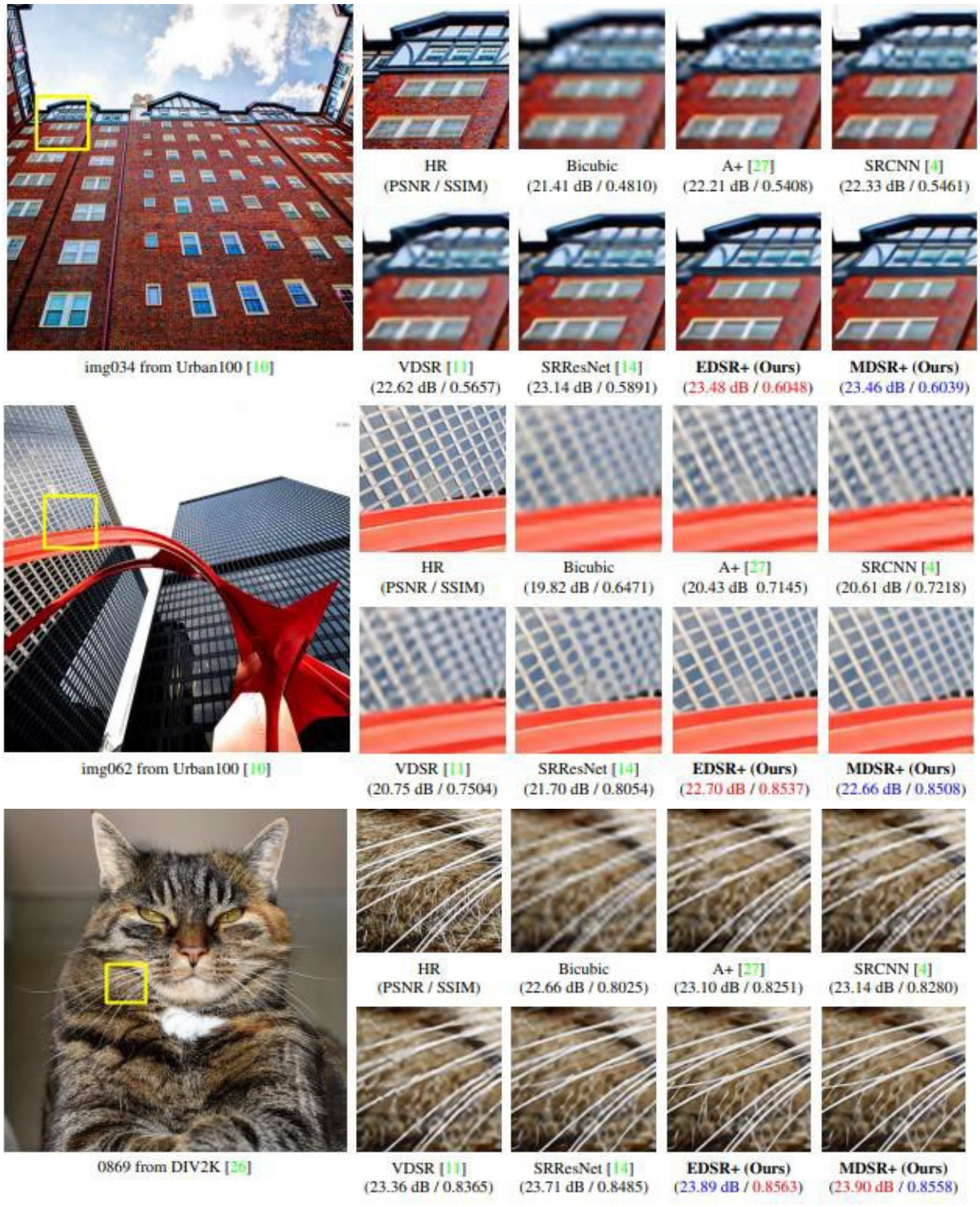
Original / PSNR

EDSR / 29.05 dB

Original / PSNR

EDSR / 30.94 dB

4.2. EDSR vs Other



4.3. Model's output

In this section, let have a look on resultant images using EDSR model versus traditional techniques (e.g., Nearest-neighbor interpolation, Bilinear interpolation, bicubic interpolation,.) on some (generic/evaluation) test cases.

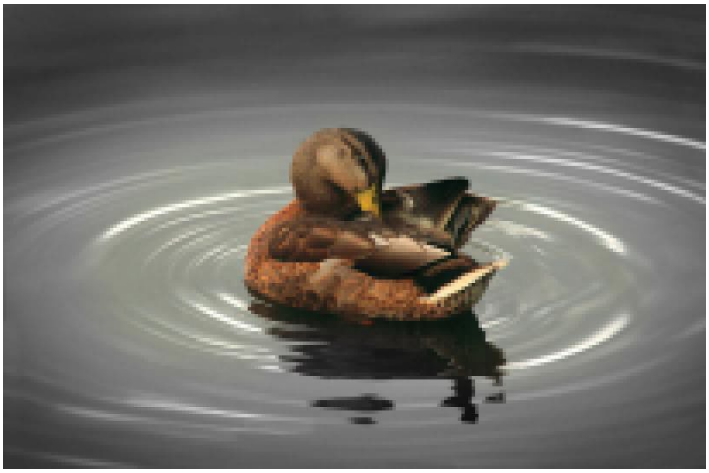
Original

Traditional x2

EDSR x2

Traditional x3

EDSR x3



Traditional x4

EDSR x4

Original

Traditional x2

EDSR x2

Traditional x3

EDSR x3

Traditional x4

EDSR x4

Original

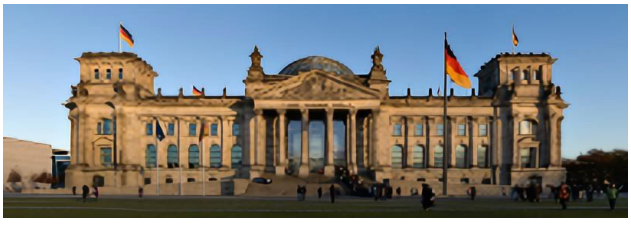
Traditional x2

EDSR x2

Traditional x3

EDSR x3

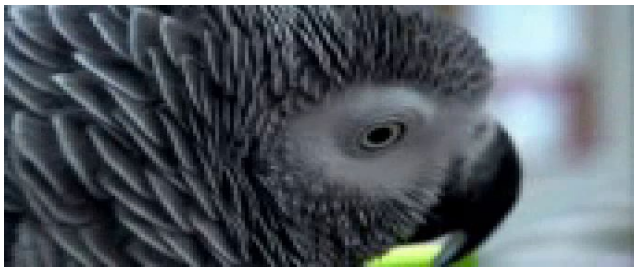
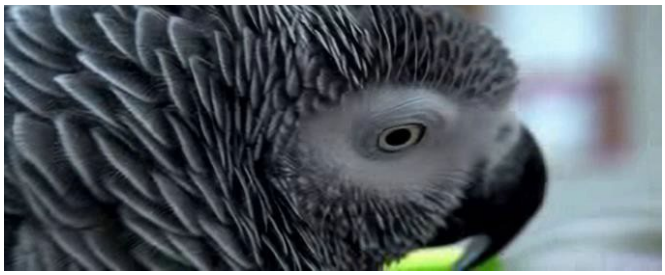
Traditional x4



EDSR x4

4.4. EDSR on Videos

Because of the fact which can consider the video as images changing across time, we can apply our model on videos.

Traditional X4	EDSR X4
	
	
	
	

5. Performance Recovery Score

Our motivation for performance recovery score **PRS** is that image enhancement or improving the visual quality of a digital image can be subjective. Saying that one method provides a better-quality image could vary from person to person. So, we want to reach an objective evaluation metric.

5.1. Introduction

Previously, PNSR was proposed as a quantitative/empirical measure to compare the effects of image enhancement algorithms on image quality. Unfortunately, it's a subjective measure which is not reflective and can be quite tricky, let's have a look at this figure below.

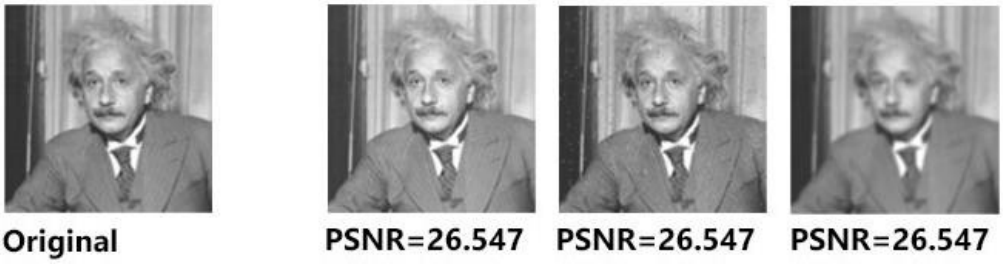


Figure: Evaluating images using PNSR

As we can see, three images have the same PNSR value but there is a markable difference between each of them and the ground truth image.

So, we proposed PRS which is an objective measure with the help of deep learning, it's a classifier as a metric. The higher the classification accuracy ''PRS'' the better degraded image has been reconstructed to match the original image and the better the reconstructive algorithm.

5.2. Dataset

The CIFAR-10 are labeled subsets of the 80 million tiny images dataset. They were collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. Figure below has samples from each class.

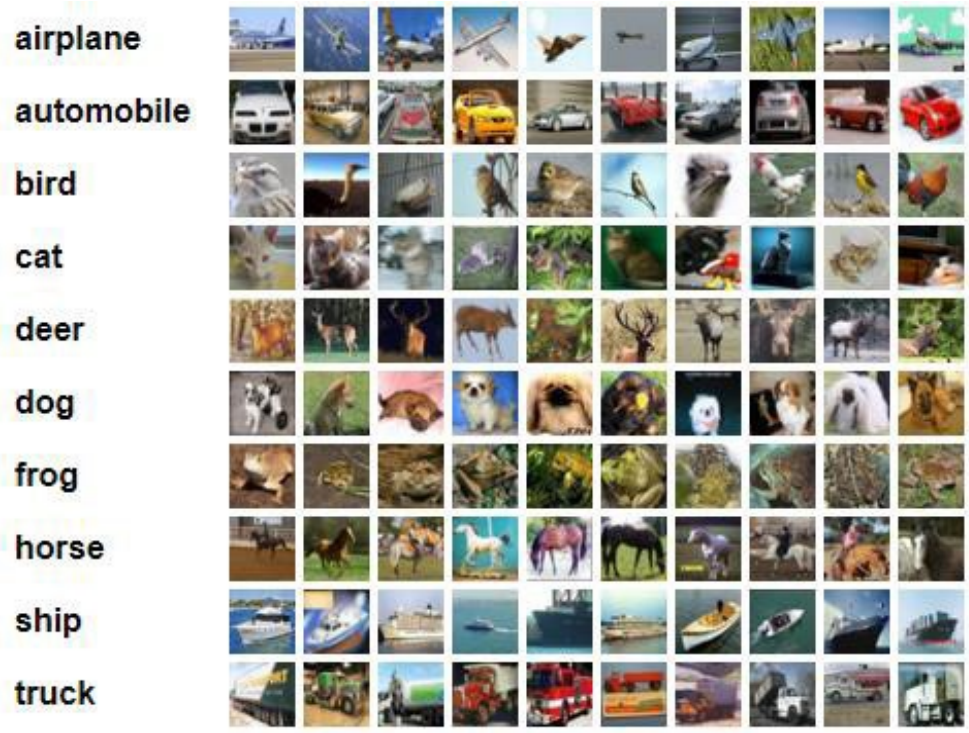
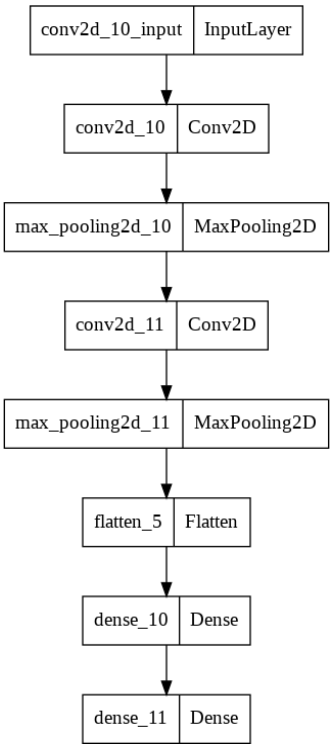


Figure: Cifar-10 dataset

5.3. Model

We used a simple model with a simple architecture consisting of 2 convolution layers and 2 dense layers, as shown in figures below.




```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_10 (Conv2D)          (None, 30, 30, 32)        896

 max_pooling2d_10 (MaxPoolin  (None, 15, 15, 32)       0
 g2D)

 conv2d_11 (Conv2D)          (None, 13, 13, 64)        18496

 max_pooling2d_11 (MaxPoolin  (None, 6, 6, 64)         0
 g2D)

 flatten_5 (Flatten)         (None, 2304)              0

 dense_10 (Dense)            (None, 64)                147520

 dense_11 (Dense)            (None, 10)                650

=================================================================
Total params: 167,562
Trainable params: 167,562
Non-trainable params: 0
_________________________________________________________________
```

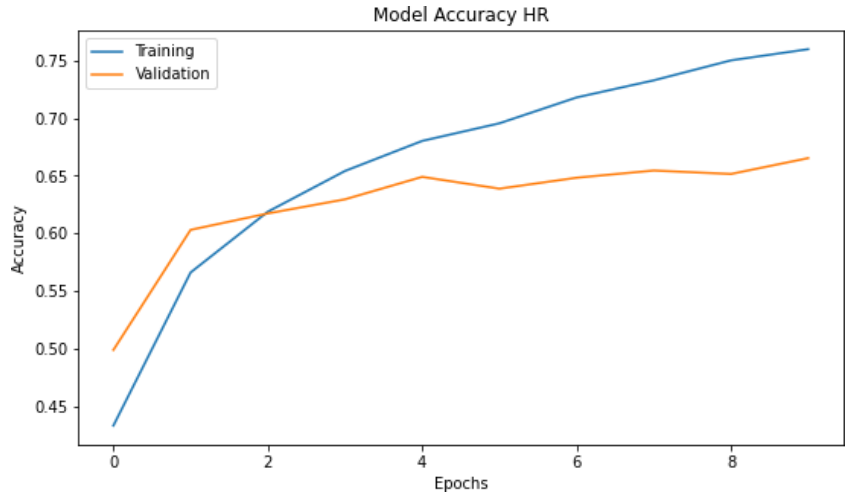
Figure: Model Summary

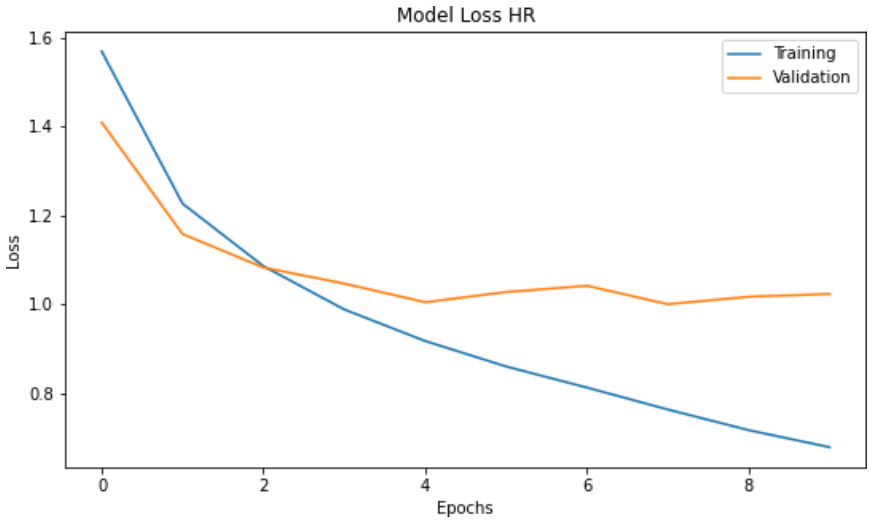
5.4. Experiment

In our experiment we want to show to what extent our model is capable of reconstructing LR images to HR images, the effect or enhancement of using our model on classification tasks and what ratio of evaluation accuracy between high resolution images and original images we succeed to preserve using PRS.

First, we applied our model on ”50000 images” from CIFAR-10 dataset, to get degraded and high-resolution images. Degraded images are 16x16x3 noisy images, high resolution “x2” images are 32x32x3 reconstructed images by our EDSR model supposed to be like original images which are 32x32x3. In the following three stages we divided our dataset 20% “10000 images” for testing and 80% “40000 images” for training with a validation ratio 0.1 “4000 images” from training dataset for validation.

In the first stage, we used the HR images “50000 32x32x3 images” to train/evaluate our model, figures below showing training/evaluation accuracy and loss.



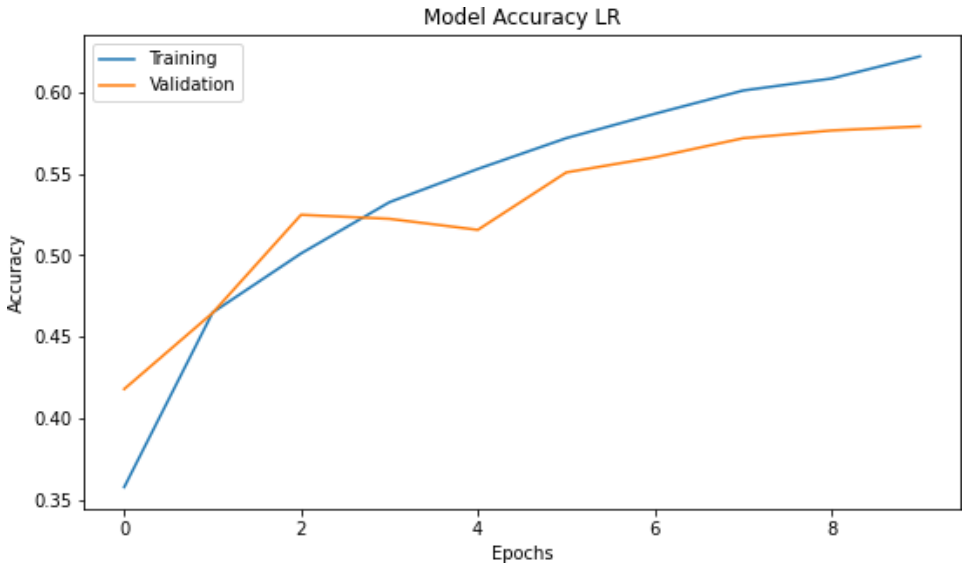


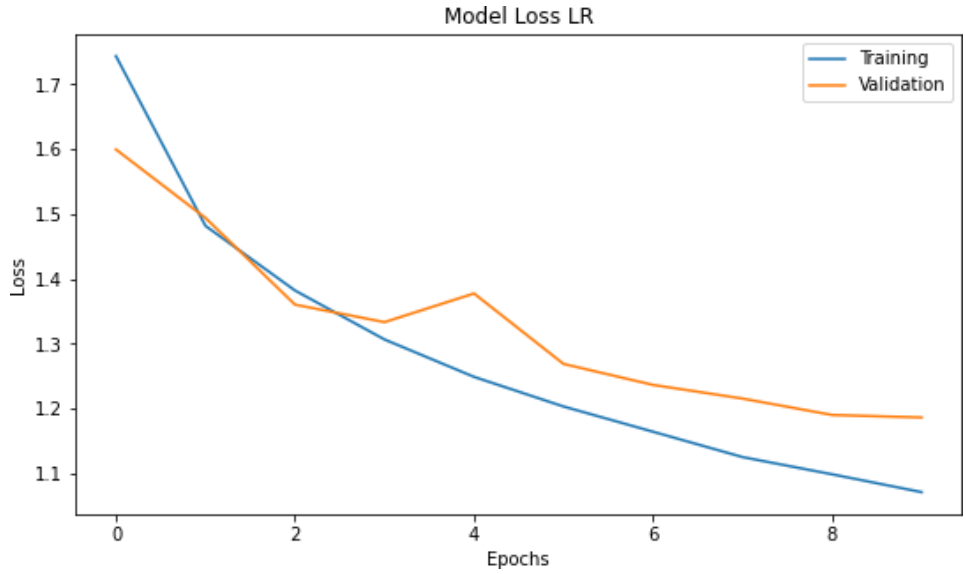
As we can see, we got accuracy nearly 76% on training dataset and 66% on validation dataset. And nearly 67% on testing dataset.

```
[ ]  cnn.evaluate(X_test,y_test)

     313/313 [==============================] - 1s 3ms/step - loss: 0.9738 - accuracy: 0.6707
     [0.9738052487373352, 0.6707000136375427]
```

Second stage, we used LR images “50000 16x16x3 images” to train/evaluate our model, figures below showing training/evaluation accuracy and loss.





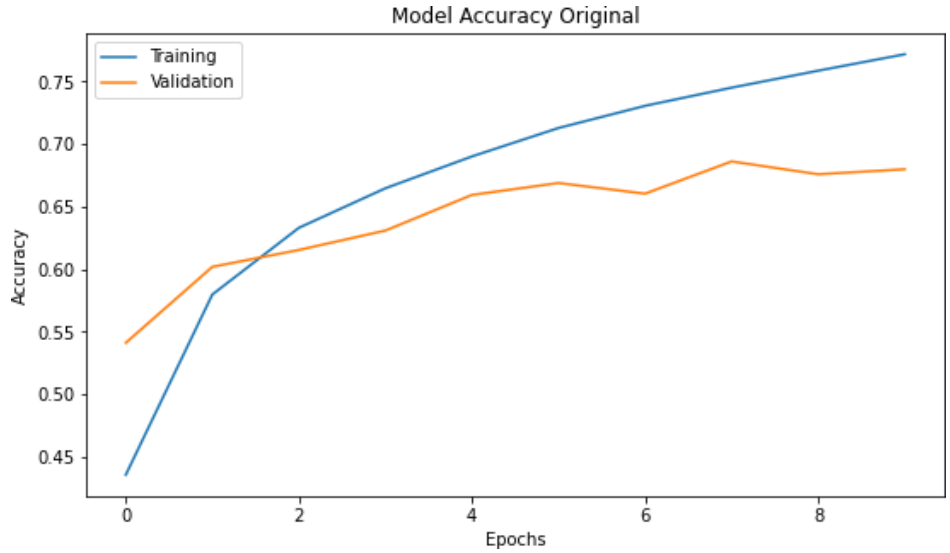
We got accuracy nearly 62% on training dataset and 58% on validation dataset. And nearly 60% on testing dataset.

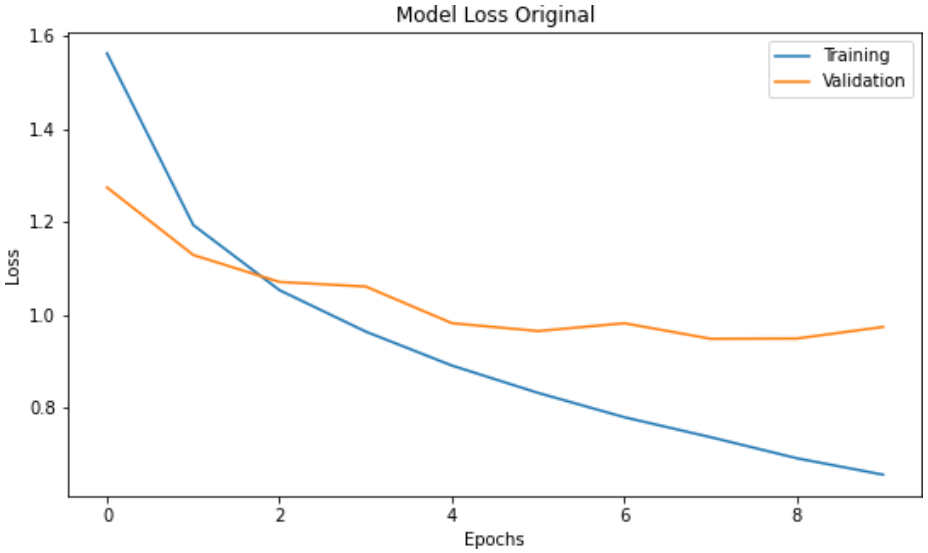
```
[ ]  cnn.evaluate(X_test,y_test)

     313/313 [==============================] - 1s 3ms/step - loss: 1.1576 - accuracy: 0.6000
     [1.1576499938964844, 0.6000000238418579]
```

As we can notice here, our EDSR model did a great job reconstructing these LR images, also this difference in accuracy between HR and LR on testing/evaluation datasets can be quite large and remarkable when using a deep classification model.

Third stage, we used ORIGINAL images “50000 32x32x3 images” to train/evaluate our model, figures below showing training/evaluation accuracy and loss.





We got accuracy nearly 77% on training dataset and 68% on validation dataset. And nearly 68% on testing dataset.

```
[ ]  cnn.evaluate(X_test,y_test)
```

```
313/313 [==============================] - 2s 5ms/step - loss: 0.9595 - accuracy: 0.6799
[0.9594609141349792, 0.6798999905586243]
```

So, the ratio between testing accuracy between HR images stage and ORIGINAL images stage is nearly 98% which means that our HR images is so close to ORIGINAL images.

```
[ ]  0.6707000136375427 / 0.6798999905586243
```

```
0.9864686320799584
```

6. Colorization

In this section, we will talk about a colorization model we applied and whether there is an effect in using our EDSR model before it or not. This is a colorization model that uses CNN, takes a grayscale image, and converts it into a colored version of the original one. Everything mentioned in this part is referenced from 'Colorful Image Colorization' paper by Richard Zhang, Phillip Isola, Alexei A. Efros.

6.1. Introduction

This paper proposes a fully automatic approach that produces vibrant and realistic colorizations. It exploits the underlying uncertainty of the problem by posing it as a classification task and uses class-rebalancing at training time to enhance the results' accuracy.

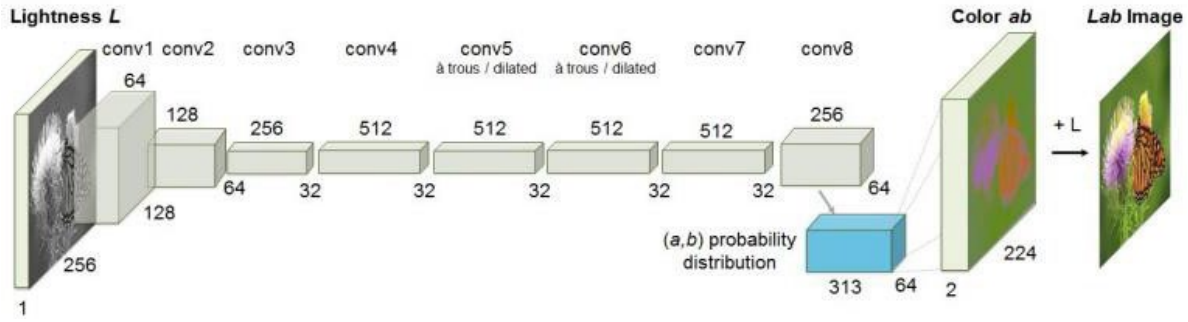
The system is trained on over a million of colored images. It is then evaluated using a "Colorization Turing Test"; where we ask human participants to choose from images, which is the generated image, and which is the ground truth. This method is successful in fooling 32% of the humans which is significantly higher than previous methods. This demonstrates that the model produces nearly photorealistic results.

6.2. How it works

The goal in the paper is not to recover the actual ground truth but to produce a plausible colorization that could potentially fool a human observer. Given the lightness channel *L.* the system predicts the corresponding *a* and *b* colour channels of the image in the Lab colorspace. In this model, any coloured photo can be used as a training image simply by taking its *L* channel as an input and outputting it's *a* and *b* channels.

There were a few problems with the previous colorization models, one of them was the loss function used. It caused the results to look desaturated. That was due to the loss function being inherited from standard regression problems. In this paper, they utilize a loss tailored to the colorization problem. A loss function that is inherently multimodal. We first predict a distribution of possible colours for each pixel then re-weight the loss to emphasize rare colours. At the end we produce a final colorization by taking the annealed-mean of the distribution that is more vibrant and realistic than previous approaches.

The below figure is the model's architecture. Each conv layer refers to a block of 2 or 3 repeated conv and ReLU layers, followed by a BatchNorm [30] layer. This network has no pool layers. All changes in resolution are achieved through spatial downsampling or upsampling between conv blocks.



6.3. Comparison

In this section we will see some outputs before the image was passed into our EDSR model and after being passed into it.

Original black and white images:

Normal colorization:

Colorization after being passed into EDSR model:

EDSR x2 EDSR x3 EDSR x4

EDSR x2 EDSR x3 EDSR x4

According to the above figures, we can conclude that the EDSR model didn't affect the colorization results. It stayed the same as it was before, no matter the super resolution scale used.

The only difference noticed is the quality of the image itself and not the colorization and this is due to the image needing to be resized before being passed onto the EDSR model.

We expected this experiment to enhance the colorization models' results but unfortunately that was not the case here. The colorization model works well on its own as well as the EDSR model. Combining them didn't turn out to be the best option.

7. Application

To enable users to use the model we created a simple application that takes an image/video and the model to be used then returns the enhanced image/video.

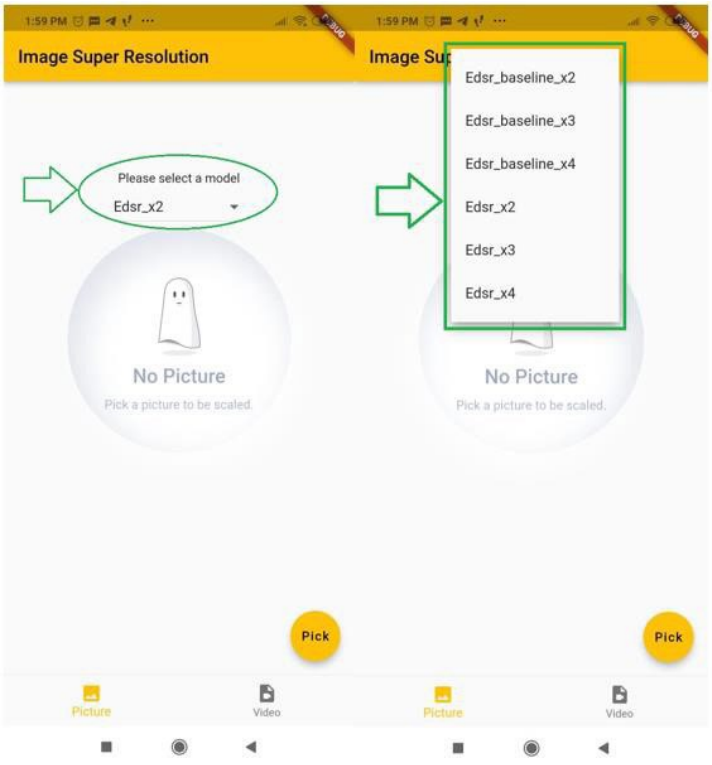
7.1. Tool Used

Flutter: Used to build the User interface.
Fast API: Used to build the backend of the application.
Pytorch: Used to build the super-Resolution model.
Colab: Used to write and test the Super-Resolution model.
VS code: Used the build the whole project.

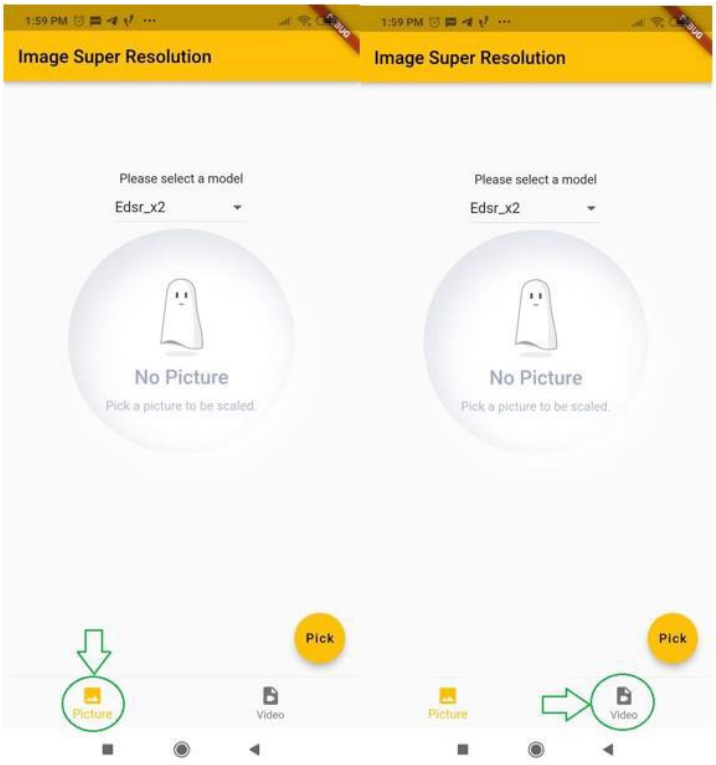
7.1. Overview

The home page consists of 4 actions.

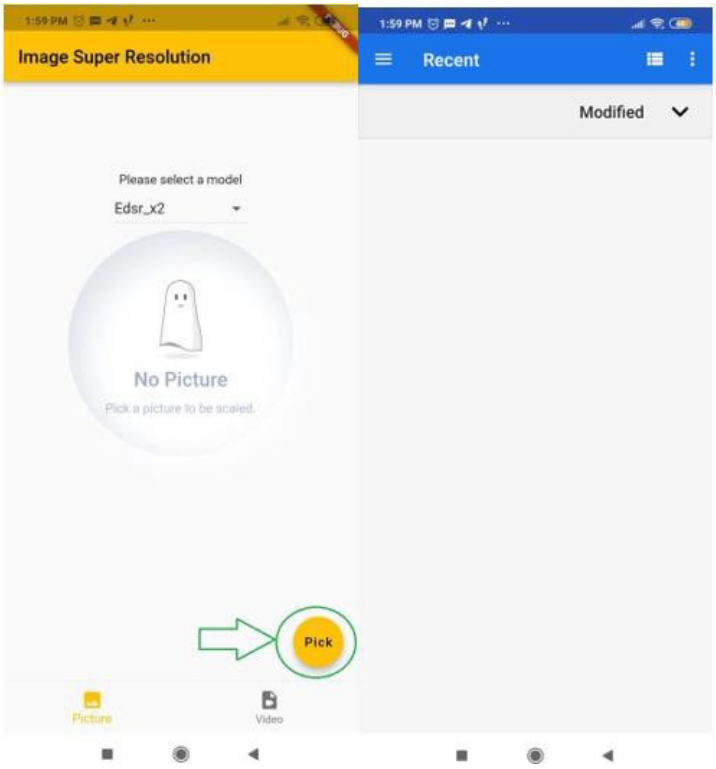
1. Selecting the model type from a dropdown menu.



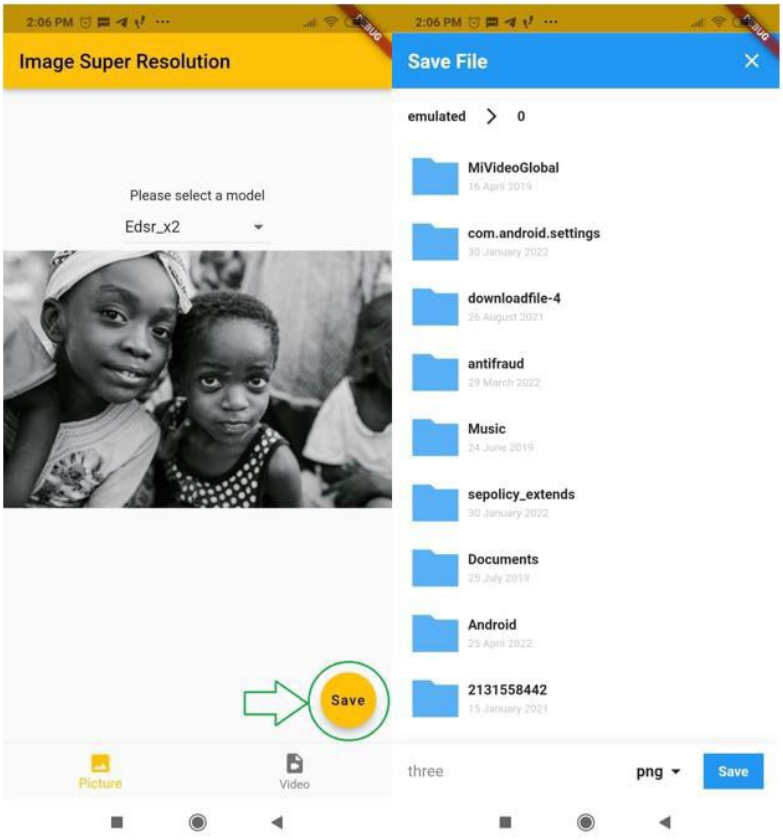
2. The user either selects the image or the video option from the bottom bar.



3. The user presses the pick button and selects the desired image/video.

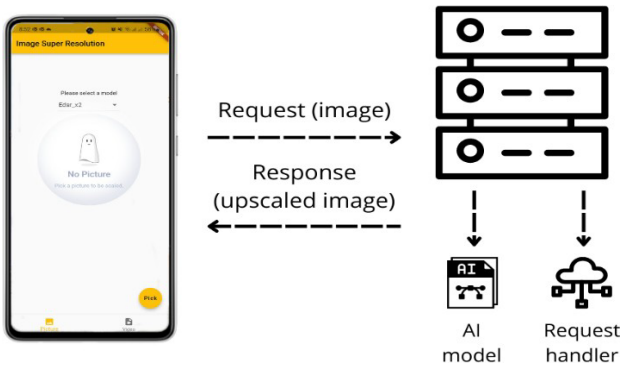


4. After the user receives the image/video, the user can then save it to the phone.



Background process:

1. The application converts the image/video to a base64 as a json object.
2. The application sends the http a POST request to the server.
3. The server then receives the image/video in base64.
4. The server then sends the image/video to the model and returns the result.



7.3. Server

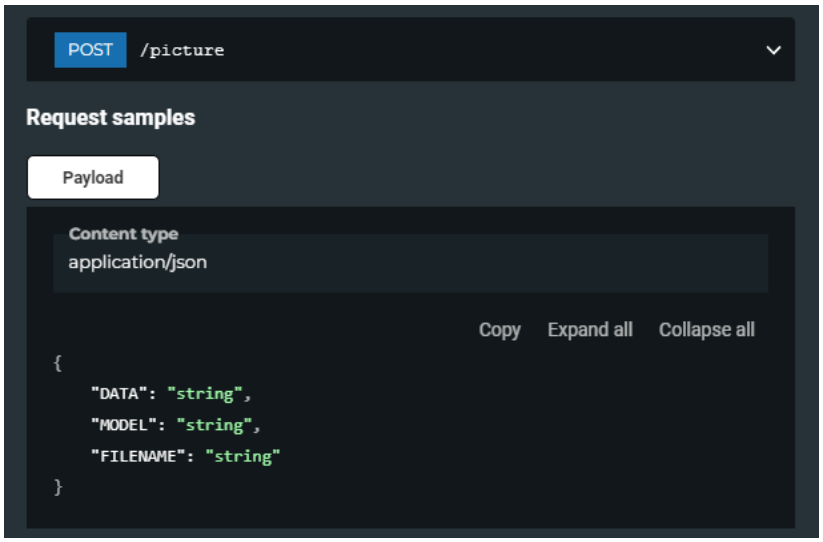
The server is hosted by the website https://www.linode.com/

7.3.1 Specs

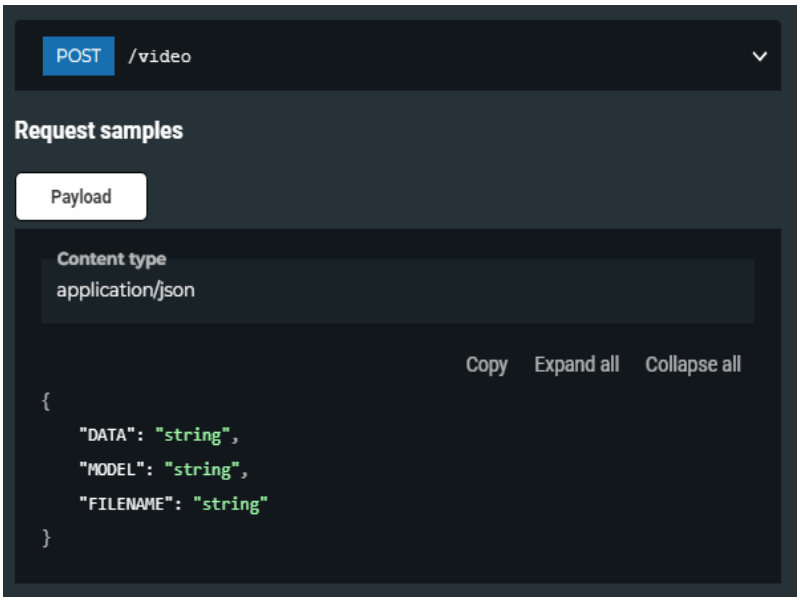
8GB Ram
4 CPUs
160GB Disk

7.3.2 Functions

The server consists of two main functions.
The first receives a json object consists of the image, the model type and the filename.



The second also receives a json object consists of the video, the model type, and the filename.



References

[1] Z. Wang, J. Chen and S. C. H. Hoi, "Deep Learning for Image Super-Resolution: A Survey," *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 43, p. 3365–3387, 2021.

[2] Z. Li, F. Liu, W. Yang, S. Peng and J. Zhou, "A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects," *IEEE Trans. Neural Networks Learn. Syst.,* vol. 33, p. 6999–7019, 2022.

[3] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. P. Aitken, A. Tejani, J. Totz, Z. Wang and W. Shi, "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, 2017.

[4] Y. Yuan, S. Liu, J. Zhang, Y. Zhang, C. Dong and L. Lin, "Unsupervised Image Super-Resolution Using Cycle-in-Cycle Generative Adversarial Networks," in *2018 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2018, Salt Lake City, UT, USA, June 18-22, 2018*, 2018.

[5] W. Zhang, Y. Liu, C. Dong and Y. Qiao, "RankSRGAN: Generative Adversarial Networks With Ranker for Image Super-Resolution," in *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, 2019.

[6] J. Korhonen and J. You, "Peak signal-to-noise ratio revisited: Is simple beautiful?," in *Fourth International Workshop on Quality of Multimedia Experience, QoMEX 2012, Melbourne, Australia, July 5-7, 2012*, 2012.

[7] D. Brunet, E. R. Vrscay and Z. Wang, "On the Mathematical Properties of the Structural Similarity Index," *IEEE Trans. Image Process.,* vol. 21, p. 1488–1499, 2012.

[8] D. Han, "Comparison of commonly used image interpolation methods," in *Conference of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE 2013)*, 2013.

[9] O. Rukundo and H. Cao, "Nearest Neighbor Value Interpolation," *CoRR,* vol. abs/1211.1768, 2012.

[10] P. R. Smith, "Bilinear interpolation of digital images," *Ultramicroscopy,* vol. 6, p. 201–204, 1981.

[11] Z. Dengwen, "An edge-directed bicubic interpolation algorithm," in *2010 3rd international congress on image and signal processing*, 2010.

[12] T. Uiboupin, P. Rasti, G. Anbarjafari and H. Demirel, "Facial image super resolution using sparse representation for improving face recognition in surveillance monitoring," in *2016 24th Signal Processing and Communication Application Conference (SIU)*, 2016.

[13] F. Liu, Q. Yu, L. Chen, G. Jeon, M. K. Albertini and X. Yang, "Aerial image super-resolution based on deep recursive dense network for disaster area surveillance," *Personal and Ubiquitous Computing,* p. 1–10, 2021.

[14] L. Zhang, H. Zhang, H. Shen and P. Li, "A super-resolution reconstruction algorithm for surveillance images," *Signal Processing,* vol. 90, p. 848–859, 2010.

[15] D. Mahapatra, B. Bozorgtabar and R. Garnavi, "Image super-resolution using progressive generative adversarial networks for medical image analysis," *Computerized Medical Imaging and Graphics,* vol. 71, p. 30–39, 2019.

[16] J. S. Isaac and R. Kulkarni, "Super resolution techniques for medical image processing," in *2015 International Conference on Technologies for Sustainable Development (ICTSD)*, 2015.

[17] N. Zhao, Q. Wei, A. Basarab, D. Kouamé and J.-Y. Tourneret, "Single image super-resolution of medical ultrasound images using a fast algorithm," in *2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI)*, 2016.

[18] Y. Chen, Q. Li, A. Zhang, L. Zou, Y. Jiang, Z. Xu, J. Li and Z. Yuan, "Higher quality live streaming under lower uplink bandwidth: an approach of super-resolution based video coding," in *Proceedings of the 31st ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2021.

[19] A. Zhang, C. Wang, B. Han and F. Qian, "Efficient volumetric video streaming through super resolution," in *Proceedings of the 22nd International Workshop on Mobile Computing Systems and Applications*, 2021.

[20] Y. LeCun, Y. Bengio and G. Hinton, "Deep learning," *nature,* vol. 521, p. 436–444, 2015.

[21] R. Hecht-Nielsen, "The mechanism of thought," in *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, 2006.

[22] N. J. Wade and M. Swanston, Visual perception: An introduction, Psychology Press, 2013.

[23] K. He, X. Zhang, S. Ren and J. Sun, "Identity mappings in deep residual networks," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, 2016.

[24] C. Dong, C. C. Loy, K. He and X. Tang, "Image Super-Resolution Using Deep Convolutional Networks," *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 38, p. 295–307, 2016.

[25] C. Dong, C. C. Loy and X. Tang, "Accelerating the Super-Resolution Convolutional Neural Network," in *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II*, 2016.

[26] Y. Fan, H. Shi, J. Yu, D. Liu, W. Han, H. Yu, Z. Wang, X. Wang and T. S. Huang, "Balanced Two-Stage Residual Networks for Image Super-Resolution," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2017, Honolulu, HI, USA, July 21-26, 2017*, 2017.

[27] J.-S. Choi and M. Kim, "A Deep Convolutional Neural Network with Selection Units for Super-Resolution," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2017, Honolulu, HI, USA, July 21-26, 2017*, 2017.

[28] B. Lim, S. Son, H. Kim, S. Nah and K. M. Lee, "Enhanced Deep Residual Networks for Single Image Super-Resolution," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2017, Honolulu, HI, USA, July 21-26, 2017*, 2017.