

Preliminary Project Mid Report

Image Resampling and Super-Resolution using Multi Discriminator Generative Adversarial Networks

Rajat Goel,
18103108

Shreya Grover,
18103113

Sarthak Sharma
18103112

Group - G117



Covered in this Report

1. Introduction
2. Literature Survey
3. Motivation
4. Problem Statement
5. Proposed Solution
6. Current Work Done / Experimentation on Dataset
7. Future Work
8. References

Note: This is a preliminary report and methods/approaches/result may change as the project moves further towards completion as seen fit.



Introduction

Deep learning promises to discover rich, hierarchical models that represent probability distributions over the kinds of data encountered in artificial intelligence applications, such as natural images, audio waveforms containing speech, and symbols in natural language corpora.

In this work, we aim at exploiting this area of artificial intelligence to solve the problem of Image Resampling and Super Resolution. The problem is an easy one to understand, but a tricky one to solve. It states, for any low resolution, low-quality image, convert it into a high resolution, high quality one. The tricky part is the fact that doing such a thing requires bringing back lost information, which is physically impossible. So we propose a deep learning algorithm, built upon the ever-popular Generative Adversarial Networks, proposed by Ian Goodfellow et al. 2014^[1]. These make use of modified convolutional neural networks, stacked against each other in a digital kind of fight. One of them is a generative network that generates new data out of noise, and the other identifies it as being fake data. Both of them fight in this manner until the generative model gets smart enough to generate data that the discriminator model identifies as real. Hence, effectively, we estimate what a high-resolution version of an image should be, by teaching computers how to learn by debating and learning from mistakes.

Deep learning-based methods have recently pushed the state-of-the-art on the problem of Single Image Super-Resolution. However, in this project, we also revisit the more traditional interpolation-based methods and compare them against our solution to achieve the best results.

Literature Survey

Interpolation Techniques in Image Resampling^[2]

Manjunatha et. al. demonstrate that resampling establishes a regular interrelationship between pixels by interpolation. To find these interrelationships, a linear model is used under which every pixel is represented in a probability distribution with two classes; it belongs to the non resampled group or the resampled one. The restricted possibility for a specific pixel closeness to the resampled class is suspected to be Gaussian, while the conditional or restricted possibility for a specific pixel associated with the other group is assumed to be identical. An Expectation-Maximization (EM) algorithm can be used to assess a pixel's possibility in linear consortium with its adjacent pixels and the unidentified weights of the grouping. In the Expectation step, the possibility of a pixel belonging to the resampled group is intended. In the step, Maximization the exact method of the correlations between samples is assessed. The stopping condition is prescribed when there is a mismatch in the weights among two succeeding reiterates is quite small. During this stage, the matrix of possible values got in the expectations step for all image pixels is called the "Probability map (p-map)". For that image this p-map is periodic and crest in the 2D Fourier spectrum of the p-map signifies the resampling. In p-map, a probability value is very near to 1 then it shows there is resampling of pixels.

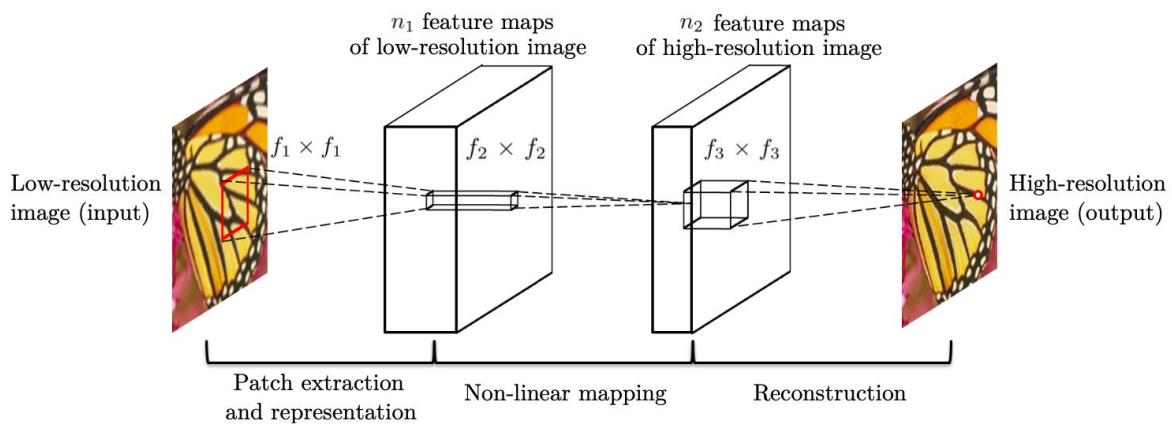
The authors presented the comparative study of Interpolating methods for Image resampling (Nearest neighbour, Bilinear interpolation, Bicubic interpolation, Basic-splines(B-spline)). They compared all interpolation functions and stated that the limited magnitude convolving functions would offer better interpolation. The response was achieved with the high-resolution cubic spline methods. The position of the resampled points with respect to the primary coordinate system has a measurable result on the reply of the sampled interpolation methods ^[2].

Table 1: SNR values with different interpolation methods

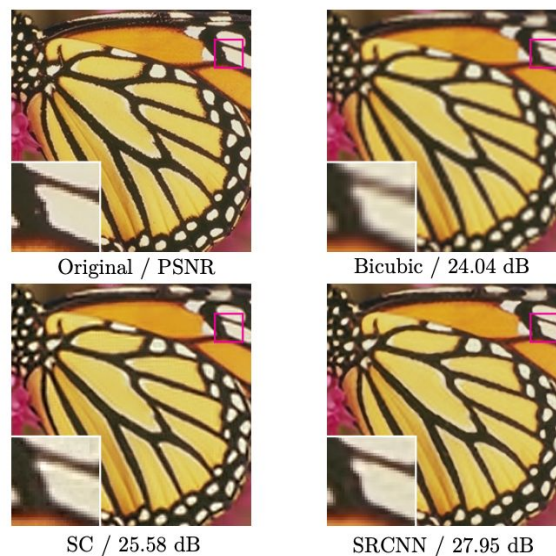
Image Method /	Nearest Neighbor	Bilinear	Bicubic	Spline
Fig1.tif	24.3200	25.2684	25.8365	28.6900 •
Fig2.tif	25.9544	27.3471	27.8160	40.5510
Fig3.tif	25.4760	26.9244	27.3626	32.3114 •
Fig4.tif	22.6283	23.3439	21.9608	30.9051
Fig5.tif	26.7347	27.0129	27.0471	46.8910
Fig6.tif	26.1706	30.2784	32.8654	34.4444 •

Deep Learning for Super-Resolution

Recently, a lot of works have addressed the task of SISR based on Convolutional Neural Networks (CNN). One pioneering work by Chao et al. is the Super-Resolution Convolutional Neural Network (SRCNN). It implicitly learns a mapping between LR and HR images using a fully convolutional network. It takes bicubic interpolation as a pre-processing step and feeds the interpolated result to the network for super-resolution. This slows down the processing speed and increases the memory requirement as all convolution operations are done on HR images. The Efficient Sub-Pixel Convolutional Neural Network (ESPCN) addresses this issue by feeding



the small size LR image to the network and postponing the upscaling to just before the output layer, employing a newly proposed sub-pixel layer. The paper also demonstrates how sparse coding based SR methods can be used as convolutional networks.^[5]



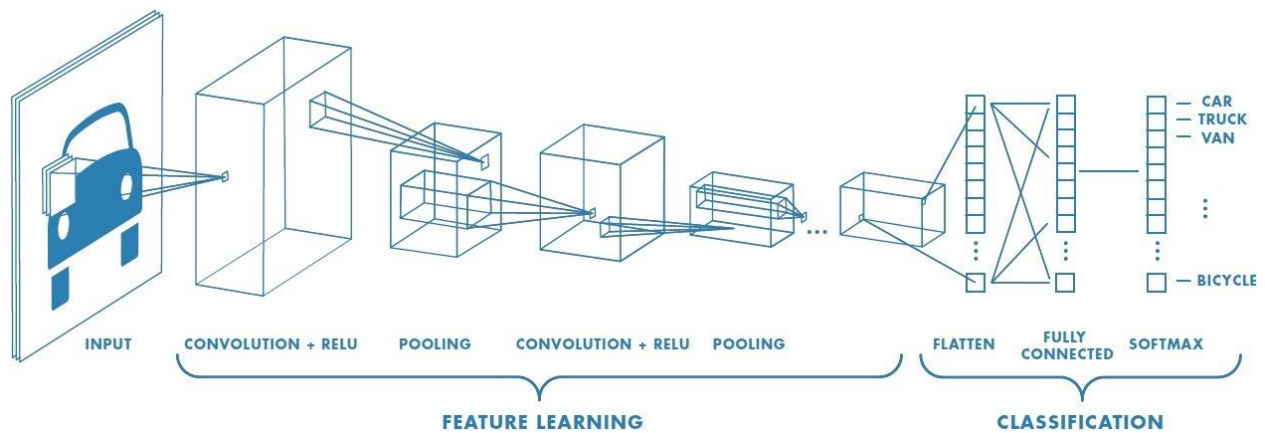
Artificial Neural Networks (ANN)

ANNs are computational processing systems of which are heavily inspired by the way biological nervous systems such as the human brain operate. ANNs are mainly composed of a high number of interconnected computational nodes which are referred to as neurons, of which work entwined in a distributed fashion to collectively learn from the input to optimise its final output. We would load the input, usually in the form of a multidimensional vector to the input layer of 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 detracts 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.^{[4] Yann LeCun et al.}

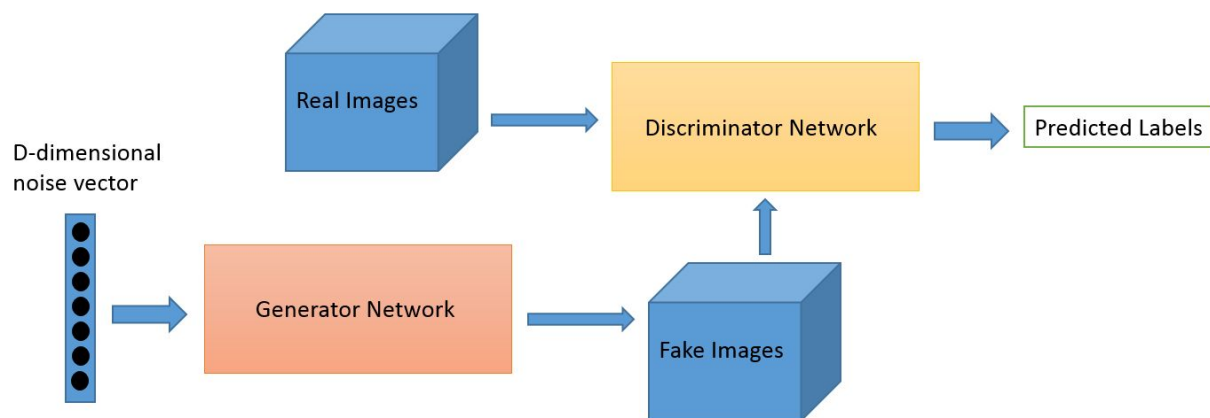
Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNNs) are analogous to traditional ANNs in that they are comprised of neurons that self-optimize through learning. Each neuron will still receive input and operate (such as a scalar product followed by a nonlinear function) - the basis of countless ANNs. From the input raw image vectors to the final output of the class score, the entire of the 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. 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. Common machine learning benchmarking datasets such as the MNIST database of handwritten digits are suitable for most forms of ANN, due to its relatively small image dimensionality of just 28×28 . With this dataset, a single neuron in the first hidden layer will contain 784 weights ($28 \times 28 \times 1$ where 1 bare in mind that MNIST is normalised to just black and white values), which is manageable for most forms of ANN. If you consider a more substantial coloured image input of 64×64 , the number of weights on just a single neuron of the first layer increases substantially to 12, 288. Also, take into account that to deal with this scale of input, the network will also need to be a lot larger than one used to classify colour-normalised MNIST digits, then you will understand the drawbacks of using such models.^{[4] Yann LeCun et al.}

We can visualise a CNN in the following model:



Generative Adversarial Networks - Goodfellow et al. 2014 ^[1]



Above Diagram shows how a Generative Adversarial Network functions. It comprises of two neural networks - Generator and Discriminator. In this network, There are really only 5 components to think about:

- R: The original, genuine data set
- I: The random noise that goes into the generator as a source of entropy
- G: The generator which tries to copy/mimic the original data set
- D: The discriminator which tries to tell apart G's output from R

What happens is, as explained in much more gruesome detail in Goodfellow et al 2014 ^[1] is :

- The generator takes in random input which is basically noise and computes another random output by doing a series of matrix computations in a CNN network.
- The discriminator takes in two inputs the real image, and the image newly created by our generator, and classifies it, telling which one is fake and which one is real. It thus generates probability values which can be used to compute the loss and backpropagate the result to the generator.
- This continues until the generator starts generating images that the discriminator has a very hard time telling apart. At this point we stop training and the result is a generative model that generates a realistic fake image.

Sub Pixel Convolutional Neural Networks-Wenghe Shi et al. [6]

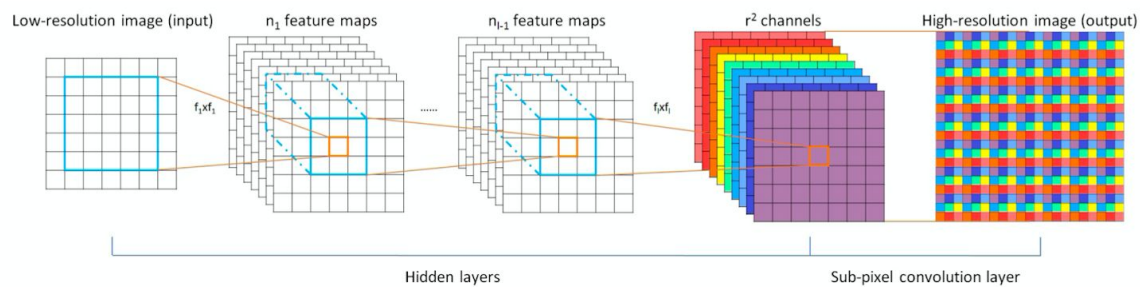


Figure 1. The proposed efficient sub-pixel convolutional neural network (ESPCN), with two convolution layers for feature maps extraction, and a sub-pixel convolution layer that aggregates the feature maps from LR space and builds the SR image in a single step.

This process is also known as Pixel shuffling. Pixel shuffling rearranges the tensor of shape (N, C, H, W) into $(N, C/r^2, H*r, W*r)$ where r is the shuffling factor. It basically converts the depth(channel) into space(height and width). In Generator, pixel shuffling is used to upsample the size of the image. It is a method to basically combine multiple computed channels from a small image into a lower dimension channel, with the same size, by squishing and recombining elements of each matrix into a single matrix.



Motivation

Every pattern and algorithm we try to recreate can be found in nature. Basing the whole project with the help of neural networks that are based on mammalian brain networks, we also have derived the inspiration for the topic from nature itself. You look upon the sky and you spot an eagle line following a straight path, to a far off observer the image is a lot less clean than an observer which might be another eagle moving with the same velocity behind our object. The image seems clearer up close because the noise is reduced, there's not as much light scattering, not as much distance than the observer at the ground.

A clearer image helps us experience objects, memories, explorations, in an unquestionably better manner. A high-resolution image is as close to what our eyes can see, as close to what we consider natural, so boundaries between what is man-made and what is natural starts to shake hands and vanish.

Digital Image Forensics is another great beneficiary of high-resolution images. Digital Image forensics is a relatively challenging and hot research area aiming at gathering information on the history of an image. In the current digital era, images have played an important role in today's regular life. As a consequence, doubt is bound to arise on the integrity of an image. With the propagation of digital images and due to availability of easy-to-use media manipulating tools such as Photoshop, anyone can manipulate the information of their digital record to attain their intention. The demonstrative potential of visual media and availability of their, storage, acquisition, and distribution is such that they are extremely advantageous to deliver the information. As a result of that, nowadays images and videos play a vital role as proof for both trials and everyday life argument. Some video clips in television program or news are generally recognized as a certification of the genuineness of that program or news. Similarly, a CCTV footages can be a part of fundamental probationary material in a trail.

This brings our attention to the fact that a high-resolution image can be more than just a technical achievement. It can be the difference between an innocent being caught and let free, the difference between the discovery of a new star or identifying it as a fluke, and this is our primary motivator for this project where we aim to exploit the vast potential of deep neural networks, to create high-resolution images, from previously low-resolution ones.



Problem Statement

Images are immensely powerful data visualisations. One needs to look no further than the success of social media such as Instagram - designed entirely around the power of images - to exemplify this. Be it dynamic data visualisation, criminology and police work or medical science, a picture truly says a thousand words.

However, a problem arises when images of varying resolution and dimensionality are produced. A low-resolution image can be created by cheaper cameras, bad quality CCTV cameras, or other low quality generally low-cost electronic devices.

Making an image high-quality is very beneficial as it can help reduce strain on eyes, help see crime scenes more clearly for the police, create more immersive closer to real-life experiences for entertainment shows, etc. The drive to create better quality images can also be a pure exploratory and interest-driven pursuit as can be demonstrated by the many photography enthusiasts.

However, taking a low quality, low-resolution image and turning into a high-resolution one is not an easy task. If we see the reverse process, converting an HD image into a low-quality one leads to loss of details, which is information lost, and if a method came by that could recover lost information from this universe, it would surely send all the physicists of our generation in a frenzy. (Conservation of Information is a fundamental law of physics).

So the best we can do is make an estimate. Take an educated guess, fueled by the thousands of years of human evolution to recognise how things should more or less look like, in general, and create a high-quality version. To automate this process, and make our computers do the same, we need to be a bit smarter and aid them in learning how to guess.

Thus, in this project, we aim to use an industry-leading technique in image generation and processing, Generative Adversarial Networks, and make the smartest guesses and recover as much detail as feasible.

Proposed Solution

Through great deliberation, a few important key points were extracted from various papers. It was then thought about in terms of both feasibilities and expected increase in output quality whether to employ these ideas or not.

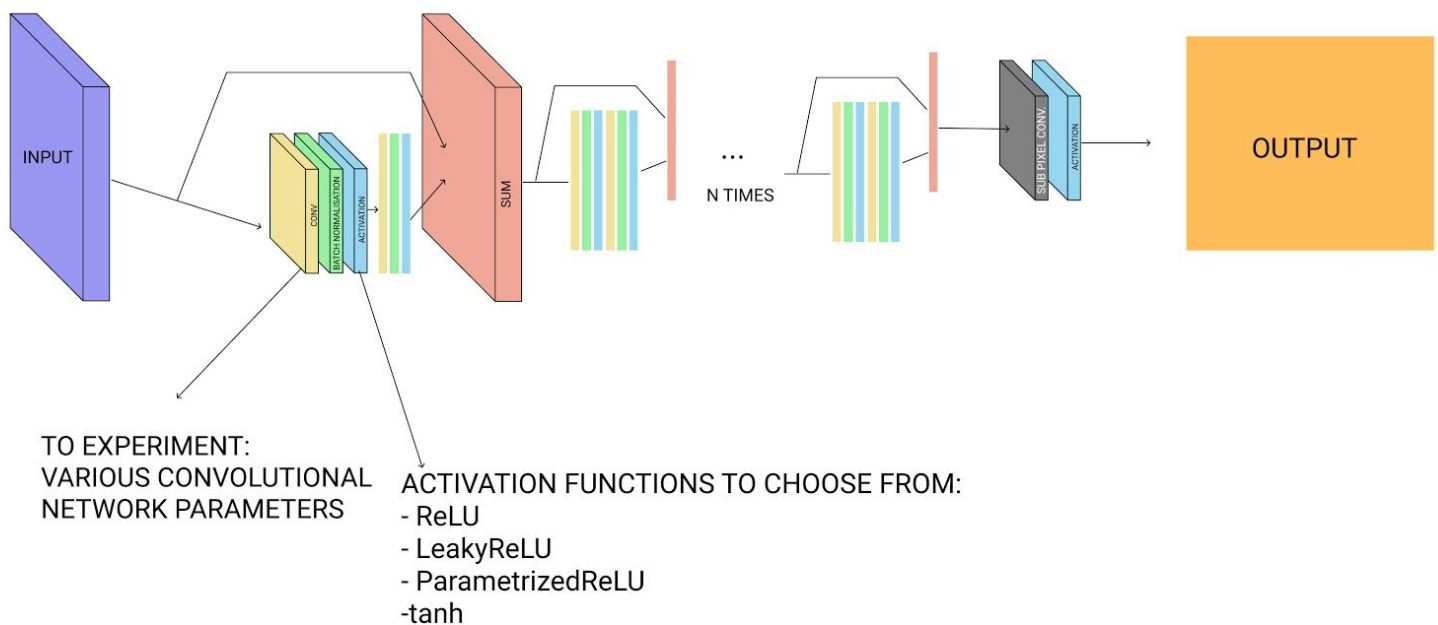
These ideas were isolated as to divert from general Generative Adversarial Networks and CNN's:

1. Use of Deep Residual learning - Xiangyu et al. 2015 [7]
 - a. It utilises summing of residual network outputs with original or semi altered output
 - b. It is our speculation that this would increase bits of images in terms of quality without modifying the image too much
2. Use of Sub-Pixel Convolutional 2D networks - Wenzhe Shi et al. 2016 [6]
 - a. It is a method that squishes a tensor of (N, C, H, W) to $(N, C/r, H*r, W*r)$ where r is the shuffling factor.
 - b. It has very promising results in terms of upscaling an image.
 - c. We could now isolate features, improve upon them using convolutional networks, and then squish them into an upscaled image using this method.
3. Expanding the channels while reducing the kernel matrix in successive CNN's.
 - a. It has been observed in many papers that this seems to have an effect of isolating a large number of features and then slowly recombining them while applying linear as well as non-linear transformations.
4. Using Multi Discriminatory networks.
 - a. We speculate that using two discriminators, one as generally intended in a GAN, and other, which computes how much the generated image differs mathematically from the original image (a content loss function), can give us a better loss estimation, and lead to better results, as well as faster training.

Thus, a generative model of following rough idea has been constructed by us and implementation would reveal our success and/or problems and discrepancies in this network.

This generator network comprises of layers of residual convolutional neural networks, whose

GENERATOR NETWORK

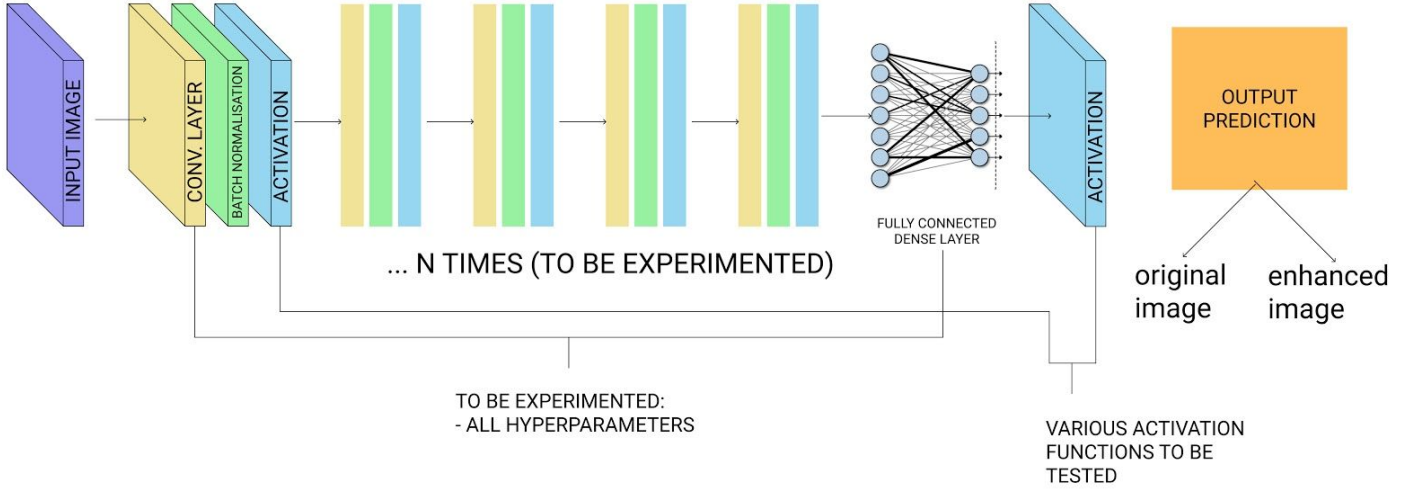


results are being added into the original and then processed further. We intend to use the simple operation of elementwise sum for this. In the end, the whole result is passed into a sub-pixel convolutional network and then into an activation layer. This increases the final output resolution by upscaling and then rounding off values into [0:255] for RGB channels

The choice of the first discriminator is simple. We choose a relatively simple discriminator, i.e, a standard CNN network with the following structure:

This model is a basic one. We need to test upon the hyperparameters, the activation function to

DISCRIMINATOR NETWORK:




use, and the no of layers of the convolutional network.

For the second discriminator function, we looked around and found the best results to be provided by the VGG loss function (Johnson et al. 2016). It is the most used content loss function in style transfer algorithms and gives a result nearest to the actual perceptual loss. It is given as:

$$l_{VGG/i,j}^{SR} = \frac{1}{W_{i,j}H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} (\phi_{i,j}(I^{HR})_{x,y} - \phi_{i,j}(G_{\theta_G}(I^{LR}))_{x,y})^2 \quad (5)$$

Here $W_{i,j}$ and $H_{i,j}$ describe the dimensions of the respective feature maps within the VGG network.



Another contender for a loss function is the pixel-wise Mean Square Error Loss function. This is given as:

$$l_{MSE}^{SR} = \frac{1}{r^2WH} \sum_{x=1}^{rW} \sum_{y=1}^{rH} (I_{x,y}^{HR} - G_{\theta_G}(I^{LR})_{x,y})^2$$

Current Work (Experimentation on Dataset)

Dataset was collected from various sources for high-quality HD images as they seem to fit the criteria. Dataset was elaborated by downsampling the photos by a factor of 4. AntiAliasing was used to downsample and create the most realistic low-quality images. Datasets used include:

- A. [CoCo Image dataset](#)
- B. [Flickr-Faces-HQ Dataset](#)
- C. [Landscape Pictures Dataset](#)
- D. [Natural Scene Statistics Dataset - UArizona](#)

EXPERIMENTATION:

A sample image was taken ^[B] and simple mathematical upscaling techniques were applied to them, to judge what general methods amount to, what they are capable of, and what our approach stands against. Here we show a sample where we use the nearest neighbour interpolation technique ^[2] to generate a 4x upscaled image using a previously 4x downsampled image (using state of the art Anti-Aliasing technique).

Note: Python 3 is used here

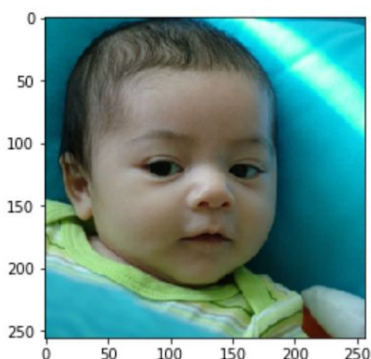
```
import matplotlib.image as img #basic image loading and performing math
import matplotlib.pyplot as plt #plotting image

import numpy as np

kid = img.imread("kid.jpg") #sample image
plt.imshow(kid) #original 256x256 image

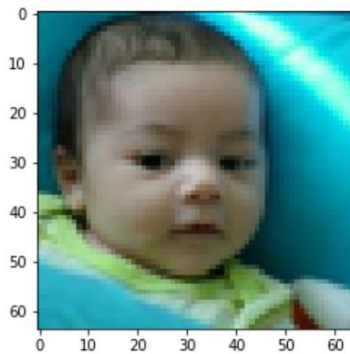
print("Dimensions : {} x {}".format(kid.shape[0],kid.shape[1]))
```

Dimensions : 256 x 256



```
kid = img.imread("kiddownscaled.jpg") #sample downscaled image image
plt.imshow(kid) #4 x downscaled 64x64 image
print("Dimensions : {} x {}".format(kid.shape[0],kid.shape[1]))
```

Dimensions : 64 x 64



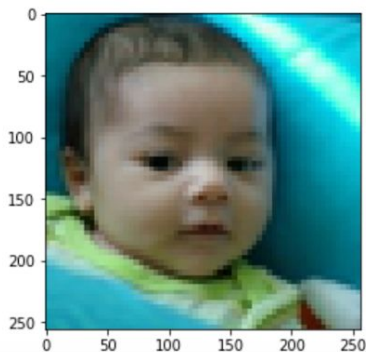
```
width,height = kid.shape[:2]
factor = 4

#nearest neighbor upscaling
upscaledKid = np.zeros([width*factor,height*factor,3])

for x in range(0,height*factor):
    for y in range(0,width*factor):
        upscaledKid[x][y] = kid[int(x/factor)][int(y/factor)]
upscaledKid = upscaledKid.astype(np.uint8)
print("New Dimensions : {} x {}".format(upscaledKid.shape[0],upscaledKid.shape[1]))
plt.imshow(upscaledKid)

#saving the image
img.imsave("kid@NearestNeighbor.jpg", upscaledKid)
```

New Dimensions : 256 x 256





Original Image

256 px x 256 px



Downscaled @ 4x

64px x 64 px



Upscaled by a factor of 4

256px x 256px

Method : Nearest Neighbor

These results show how one of the traditional methods work. The nearest neighbour method uses a simple technique where we just calculate the nearest pixel value and replicate it to generate a new upscaled image. The interpolation kernel for this algorithm is defined as :


$$f(x) = \begin{cases} 0, & x < 0 \\ 1, & x > 0 \end{cases}$$

where x is the distance between grid point and interpolated point.

This interpolation function appears more akin to a sharpening filter in commercial photo editing software suites like Adobe Photoshop. Its effect can be varying, e.g. here, it appears to de-blur the downsampled image somewhat and sharpens the edges. Generally, in photos with straight lines, it appears to make the lines jagged if they are curved or tilted a lot and sometimes make a line perfectly straight if there is a very slight non-trivially noticeable tilt to it. ^[2]

Hence, scope reveals for our work as it is a completely different approach from traditional methods and GAN's have been in general able to prove superiority over traditional simple mathematical methods in complex image processing and generation problems due to their non-determinism.

Apart from this, major time was spent on figuring out the complex workings of neural networks, convolutional neural networks, Generative Adversarial Networks, and figuring out the optimal approach, deciding the various blocks, layers, and optimising substructures within them.



Thus, a lot of time was used in expanding on our idea and thus landing on this approach of using multi discriminator neural networks.

Future Work

Having done analysis using traditional techniques and an extensive literature review, we will now move this project into its most important stage, i.e., implementation. We have isolated a solution that we believe will work to produce our desired outputs. We will now be constructing the various neural networks required for this and generating the outputs.

The languages and libraries we will be using are:

- Python as Programming Language
- Numpy and Pandas for matrix computations
- Matplotlib for graphing
- PIL for image processing, loading, saving, etc
- Pickle for saving and loading models
- And more as seen necessary

We will be implementing our models directly in python.


The first step would be to generate and isolate appropriate data for training and testing. We will be using various datasets as mentioned [here in references](#).

These datasets contain high-quality images and the following preprocessing steps will be taken for these:

1. Isolate and filter only HD images out of the complete dataset and gather a very large amount of such images.
2. Downscale all images by 4x using AntiAliasing
3. Separate into training and testing data
4. Make sure the images are multi-sectional, i.e., they are from various domains, containing a vast variety of data, as to avoid overfitting to a particular type. E.g. It should include trees, nature, faces, man-made and natural objects, indoors, outdoors, abstract images, art, handwriting, among other things.

We intend to generate models and tune their parameters including:

- CNN hyperparameters
 - Kernel size
 - Channel Size
 - Overlapping (Fully, Semi, etc.)
- No. of CNN layers

- 
- No. of total residual blocks
 - Activation Functions to be used
 - Sigmoid
 - ReLU
 - LeakyReLU
 - PReLU
 - Tanh
 - Etc
 - Content Loss function

In the end, we will use a set of images to estimate our losses in details using various standard loss functions.



References

Papers:

1. Generative Adversarial Networks - Ian J. Goodfellow et al. 2014
2. Interpolation Techniques in Image Resampling - Manjunatha. S, Malini M Patil
3. ImageNet Classification with Deep Convolutional Neural Networks - Alex Krizhevsky et al.
4. Convolutional network for images, speech, and time series - Yann LeCun et al.
5. Image Super-Resolution Using Deep Convolutional Networks - Chao Dong, Chen Change Loy
6. Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network - Wenzhe Shi et al.
7. Deep Residual Learning for Image Recognition - Xiangyu Zhang et al.

Data:

- A. CoCo Image dataset
- B. Flickr-Faces-HQ Dataset
- C. Landscape Pictures Dataset
- D. Natural Scene Statistics Dataset - UArizona