

# Super-Resolution Project Individual Report

Mentors :

Sahasra Ranjan, Adarsh Raj  
Sibasis Nayak, Aniket Gudipaty

Mentee :

Arpon Basu (arpon.basu@gmail.com)

## Contents

<b>1</b>	<b>Some Interesting Points to Note</b>	<b>4</b>
<b>2</b>	<b>General Procedure of the Project</b>	<b>4</b>
<b>3</b>	<b>My Contribution to the Project</b>	<b>5</b>
<b>4</b>	<b>Problems faced</b>	<b>6</b>
<b>5</b>	<b>Results</b>	<b>6</b>
<b>6</b>	<b>Acknowledgements</b>	<b>8</b>
<b>7</b>	<b>Bibliography</b>	<b>11</b>

### **Abstract**

This report is to present a personalized view of what I learnt during this project, which includes image processing in python, training of neural networks as well as many meta-skills such as how to debug neural networks, or how to document code in a professional manner.

**Note1:-** I have provided the links to all the code I have referenced in this report through hyperlinks. Also, the links are to our common project repository, but all the same files have also been committed to my personal repository which I have submitted.

**Note2:-** Many of the talking points in this report naturally coincide with those in our common team report. Thus, the content of some sections has been reproduced verbatim from there.

## 1 Some Interesting Points to Note

I, as a team member of my team had already written in quite some detail about Convolutional Neural Networks, and how their need arises (to reduce the number of training parameters as well as through convolution, link related pixels in the input to related ones in the output (a simple neural network would link **every** pixel of the input to **every** pixel of the output, which is obviously unnecessary and excessive, and also the reason why simple networks have such large number of trainable parameters), thus **establishing correlation**) naturally in projects such as ours. We, though a course taken by Andrew Ng also learnt of the various possibilities that CNNs offered to us, such as LeNet-5, resNets and VGGs.

I would also like to note some interesting points regarding image processing, such as the point of dense versus non-dense patches : In non-dense (non-overlapping) patches, one takes non-overlapping chunks from an image, processes them separately, and then places them side by side as they were. This has the disadvantage that sometimes the reconstructed image looks a bit "blocky", ie:- one can differentiate between two blocks in the final image too. On the other hand, with dense patches, one takes overlapping chunks from the image, processes them, and then "averages" them together to get the final image. In practice it's often observed that this averaging leads to blurriness in the final image. Moreover, it's also observed that in terms of metrics such as PSNR, non-dense patches work almost as well as dense ones [12].

Finally, one comes to the issue of channels, which are basically how images are expressed as a some of three linearly independent components. The most famous modes of expression are RGB, BGR, YCbCr, etc. The interesting part about these channels is that often it suffices to train the network for only one of the channels among the three (as we did : we only trained our network for the luminance channel in YCbCr) and the final reconstructed image (with one newly produced channel component from the NN and the two older ones) usually look much better than the original one despite the fact that two of the channels have the same components.

## 2 General Procedure of the Project

So according to [1] we implement a **Super-Resolution Convolutional Neural Network (SRCNN)** consisting of **3 layers**[6][7], whose jobs can be broadly described to be Patch extraction and representation[4], non-linear mapping and reconstruction of the patches[8].

At this juncture it's also important to mention a baseline with which we will compare our performance, which is that of **bicubic interpolation**. What does this entail? Bicubic interpolation is basically a mathematical process by which a low resolution image can be **up-scaled** to a higher resolution image by interpolating (low-information) patches in the image to a high-information (resolution) one through a cubic polynomial. If one thinks about it, it's not too different from how successively better solutions of Laplace's equation are obtained in numerical analysis from an initial seed : small patches are taken, the value at the centre of the patch is replaced by the **average** of the values at the boundary, and the process is repeated till satisfaction. Now, instead of taking the average (which is a **linear** function of it's inputs), we simply replace it by a **cubic polynomial** (obtained through the Lagrange interpolation formula) and repeat the same process to get an up-scaled image.

The net point of the above digression is that since modern languages like MATLAB and Python can implement bicubic interpolation on images in  $< 1$  second, whatever neural network we choose to implement must give a performance at least as good as the bicubic one.

Thus, in our neural network, we immediately scale the input image via bicubic interpolation, and the neural network then operates on the scaled up image (to hopefully) produce a better result.

The first step of the neural network mapping is **patch extraction and representation**, which basically generates patches from our image and applies a simple ReLU filter on them. Then comes the **non-linear mapping**, which basically takes the patch and maps it to a higher dimension (this is basically where the extra information is added to the image), and finally in the last layer (**reconstruction**), all the patches from the images are collated back together to form our output image.

### 3 My Contribution to the Project

During the initial phase of the project where we had to go through the Coursera lecture series, all of us covered the material on our owns though continuously animated by discussions among ourselves about the course content which deepened our understanding.

After that, we had to train our networks (using images from the tensorflow dataset) using a neural network architecture that I designed. In this, me and Harsh split up our ways to train a network, one with, and one without applying what is known as a DCT basis (which is basically a fancy name for the Fourier transform applied on images, whcih is widely used in image compression softwareS like JPEG).

After the network was trained, Harsh then used both mine and his networks to produce results [12] which one can see in the sections below.

Finally, in python, since the language is dynamically typed, there is no need of declaring data types for functions, and while this makes the life of developers easier, it makes things for people trying to comprehend the code tougher, and thus a robust documentation is needed, which I duly wrote. Finally, many sections in the team report were authored by me, which summed up our entire project and presented results in a coherent form.

## 4 Problems faced

There were many instances during the project when we faced a standstill and had to brainstorm our ideas and explore to break-through. Some of them are listed below:

- **Reading the images:** There many subtleties involved in how different libraries treat images, and one of the major issues was the resizing of images. Numpy resize requires (height,width) as a parameter, while OpenCV resize requires (width,height) as a parameter which made debugging a strenuous task.
- **Dimension Errors:** Dimensions of data types are very important as mentioned above too, and we would frequently run into dimension mismatch type errors, and consequently we have had to typecast our data at many places (like from say  $(w, h)$  to  $(w, h, 1)$ ), resize it (using bicubic interpolation and other methods), etc.
- **Google Colaboratory:** We also learnt that large scale ML projects are best done in this fascinating service offered by google instead of hosting it on our local device Jupyter notebooks.

## 5 Results

We present a tabulation of our results here in terms of both numerical metrics (**decibels**, which is the unit in which PSNR gives answers[12]) as well as one visual confirmation for each type of network trained for the reader to peruse through (1, 2, 3, 5). Also all networks here have been trained for a dataset of 100,000 images and a scale factor of 2, as mentioned above. This has also been included here [2].

The tables may intersperse with the flow of other sections.

Project Results		
Ground Truth	Grayscale (dB)	RGB image (dB)
Bicubic	49.63	45.42
With DCT bases (without dense patches)	17.95	— (Grayscale is already low)
Without DCT and without dense patches	28.79	28.14
Without DCT + Dense patches	29.85	28.98

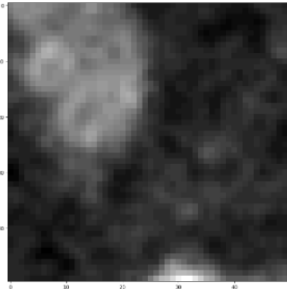
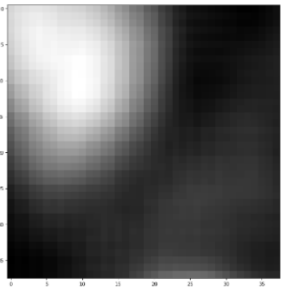
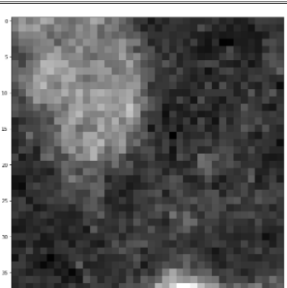
	Processed Input
	Predicted
	Ground Truth

Table 1: Without DCT + dense Patch Extraction (Before Reconstruction)


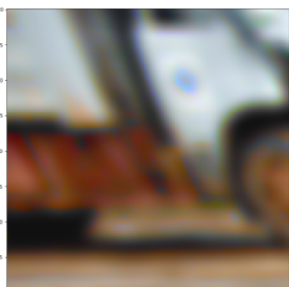
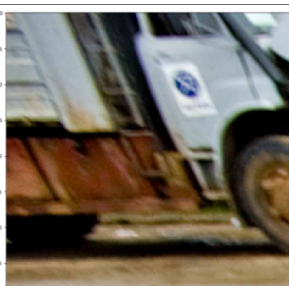
	Bicubic Interpolation
	Predicted (Blurring happened mainly due to reconstruction)
	Ground Truth

Table 2: Without DCT + dense Patch Extraction (After Reconstruction)

## 6 Acknowledgements

We would like here to briefly acknowledge all the help we have received in our journey in this project : We would first like to thank our mentor for this project, **Sahasra Ranjan**, who was very committed and guided us ably throughout the project, even looking up for relevant pieces of code (such as the tensorflow function `tfds.list_builders()`, and many more along the way) when we couldn't find it.

We would also like to thank the other mentors for the CNN Project, **Adarsh Raj**, **Aniket Gudipaty** and **Sibasis Nayak**, who diligently prepared quite thorough tutorials (in the learning phase) for us, before this project began, which helped us clear many of our concepts and develop a clear understanding about how things proceed in this field of neural networks.




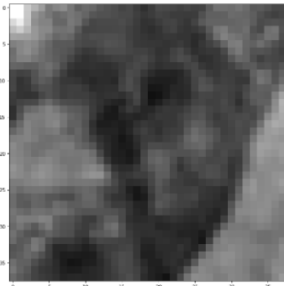
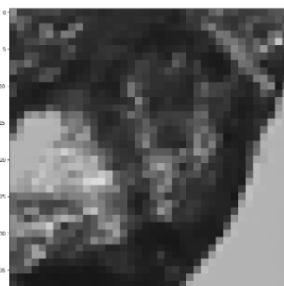
	Bicubic Interpolation
	Predicted (Notice the bright top-left corner)
	Ground truth (But down sampled from $50 \times 50$ to $38 \times 38$ to match network output dimensions)

Table 3: Using DCT bases (Only Grayscale)

I would also like to thank all members of WnCC who are organizing or helping in organizing this entire event, for providing such a wonderful opportunity to freshers for getting an early exposure in the various avenues that coding can lead one down to, as well as continuously supervising us along the way through READMEs and videos, thus keeping progress steady.

Finally, I would also like to apologize to anybody I may have missed on my way inadvertently.




	Bicubic Interpolation
	Predicted (Notice that there are distinct blocks formed due to non-dense path extraction)
	Ground Truth

Table 4: Without DCT and not using dense patch extraction

## 7 Bibliography

- [1] [Referred CNN Paper](#)
- [2] [README.md](#)
- [3] [First draft of helper functions](#)
- [4] [Block patch extraction](#)
- [5] [Tensorflow dataset](#)
- [6] [Model Trained without DCT](#)
- [7] [Model Network with DCT](#)
- [8] [First Reconstruction](#)
- [9] [CNN](#)
- [10] [ImageSuperResolution Class](#)
- [11] [Callbacks in Tensorflow](#)
- [12] [Results](#)