Image Enhancement - Project 1
CS7180 Advanced Perception

20th September 2023

By,
Anirudh Muthuswamy, NUID - 002783250
Gugan Kathiresan, NUID -002756523

Inspired by:

1. C. Dong, C. C. Loy, K. He, and X. Tang, “Learning a Deep Convolutional Network for Image Super-Resolution,” Computer Vision – ECCV 2014, pp. 184–199, 2014, doi: https://doi.org/10.1007/978-3-319-10593-2_13.
2. Christian Ledig, Lucas Theis, Ferenc Huszaŕ, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, Wenzhe Shi, "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network Actions", CVPR, 2017.

Code: https://github.khoury.northeastern.edu/gugank/ImageEnhancement-cs7180

Abstract:

We have combined ideas from two papers in order to obtain better results for single image paper resolution, with additional novelties in the network structure. In order to keep a fairly simple network design, we took the SRCNN network structure as our baseline and made improvements to an already existing model. We use the VGG Perceptual loss instead of the standalone MSE loss used in the SRCNN paper. We further combined multiple input Convolution operations with varying filter sizes and concatenated the output from these input layers to have an understanding of features of different sizes. We trained the model with 32 X 32 patches from 5 images and validated using 50 original 2x bicubic interpolated images from the DIV2K2017 Dataset. The results of the 3 variations of SRCNN with parallel input convolutions and the perceptual loss have been compared and respective PSNR graphs have been plotted. These results show that using just the perceptual loss increases the training PSNR from 40.986 to 41.583 on the logarithmic scale. And employing the parallel input convolution layers increases the training PSNR from 41.583 to 45.027.

Introduction & Prior work:

In 2015, Dong et al. introduced a study that piqued the interest of the computer vision community. Their research focused on harnessing the power of deep learning for the enhancement of single images through super-resolution techniques. Specifically, their objective was to leverage deep convolutional networks to accomplish single image Super Resolution (SR). This research direction was inspired by an observation highlighted in the paper's abstract [1]. The authors posited that traditional example-based Single Image SR (SISR) methods could be interpreted as deep convolutional networks. Moreover, their proposed Convolutional Neural Network (CNN) handled the extraction and fusion of patches, a fundamental component of the conventional example-based methods.

In the realm of super-resolution, traditional example-based approaches consider inherent similarities within the same image and construct mappings between pairs of low and high-resolution images. Meanwhile, traditional sparse coding-based methods involve the utilization of overlapping patches, which are extracted and encoded using a low-level dictionary. These sparse coefficients are subsequently utilized with a high-resolution dictionary to reconstruct high-resolution patches.

The authors of the study introduced a Convolutional Neural Network, referred to as SRCNN, which operates by mapping low-end, high-resolution images through its hidden layers. The study also establishes a connection between Deep Learning (DL)-based and sparse coding-based methodologies, which guides the design of the network structure.

SRCNN, in comparison to Bicubic interpolation and traditional sparse-coding example-based methods, demonstrated superior performance, speed, and potential for further enhancement. The initial step within the network involves patch extraction through the first layer, followed by a non-linear transformation of these extracted patches into higher resolution representations. The resulting overlapping regions from various patches fill the adjacent spaces. An average filter is then applied to flatten this overlapping area, emulating the behavior of convolutional layers. In each instance, the network's motivation can be traced back to its relationship with sparse coding-based methods. Ultimately, the reconstruction phase is achieved by employing a flattening operation, effectively averaging combinations of overlapping patches across the entire image.

Methods:

The goal of the project was to recreate the famed SRCNN network from 2015 [1]. By understanding the fundamental model and its operation, we can explore improvements. It is our opinion that the SRCNN model is simple and easy to understand. Its simplicity allows for quick and efficient performance.

i) Dataset Used:

For this implementation, we wanted to employ datasets used beyond those that were considered in the source paper. The source paper included the Set5, Set14, BDS2000 datasets. We employ the DIV2k dataset [3], which consists of 1000 images of real-world scenes used expressly for single image super resolution. This dataset was unfortunately released in 2017 (2 years after the SRCNN paper was published). Hence we attempt to train our model on this dataset and elucidate its performance.

ii) Pre-processing

The dataset was used in a train test split of the ratio 80:20. The source dataset was split into “High Res” and “Low Res” images. This extends the pre-processing steps from the source paper for the train set which involves creating sub-images (non-overlapping patches) of the original image of size 32x32. After which, the patches undergo Gaussian blur, reduced in resolution by half, and then upscaled back to the same size. For model input x and output y, the training takes patches from the lower resolution images (halved + upscaled ones) for x and have the higher resolution (original sub-images before halving) as the y.

For the validation set, we employ the same halving and upscaling process with respective x and y, but do not do any patching.

The creation of sub-images serves as a method to facilitate the ideology of super-resolution - taking smaller regions of a large image, enlarging but still maintaining the quality.

iii) Implementation of SRCNN:

The SRCNN model consists of three convolutional layers. The model was constructed based on each of the parameters suggested by the source paper. This includes:

- Initial layer with convolutional operation with in-channels of size 3, out channel number of feature maps as 64 of size 32, 32 with padding = 4 and filter size of 9x9
- Second layer, with out-channels of 32 with padding = 0 and a filter size of 1x1
- Last layer, with out-channels of 3 with padding = 2 and a filter size of 5x5

This resulted in 20k parameters.
This model employs the MSE Loss as their loss function. This is due to the fact that MSE and PSNR are closely related and is, in the author's opinion, the better loss function to use.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 64, 32, 32]          15,616
            Conv2d-2          [-1, 32, 32, 32]           2,080
            Conv2d-3           [-1, 3, 32, 32]           2,403
================================================================
Total params: 20,099
Trainable params: 20,099
Non-trainable params: 0
```

Fig 1. Model summary and parameters for SRCNN implementation

iv) Implementation of SRCNN with VGG Loss:

This implementation is the exact same structure as the SRCNN in i). However, we consider employing a more updated approach with Loss functions that are designed for this specific field. One such that came to mind is the loss implemented in the popular SRGAN paper [2].

While the advantages of this loss function are noticeable for GAN type models, some features can be beneficial for our implementation. VGG Loss improves perception quality, preserving textures and structural details, and handling high-frequency information better. It helps reduce common artifacts.

The concept of VGGLoss is similar to MSE Loss, but with a type of pre-processing. During training the ground truth and predicted image are passed through a pretrained VGG Model. The output features maps are then compared with MSE loss and reported back to the training model for optimization.

v) Implementation of SRCNN v2 with VGG Loss:

One of the evident differences between SRCNN and modern single image super resolution methods in deep learning is the size and depth of the model. In fact, the SRCNN is very low in depth compared to popular 2d classification models. The source paper does not explore a larger model to avoid heavy-weight models with slower performance and longer training time. However, while deep models are heavier, wider models are comparatively manageable and can still achieve the impact of a deep model. This impact refers to the use of varying filter sizes to capture varying features of the input image.
In our approach, we create 4 such input layers that vary in filter sizes. Each kind of filter size extracts a different feature, which is then concatenated into a single feature map and then passed to the non-linear mapping layer.

A visualization of the model structure can be seen as a flow chart below with details on its structure and parameters. Fig 4 also depicts an example of the features extracted by the different layers. This figure brings a visual example of the different features these varied filters pick up

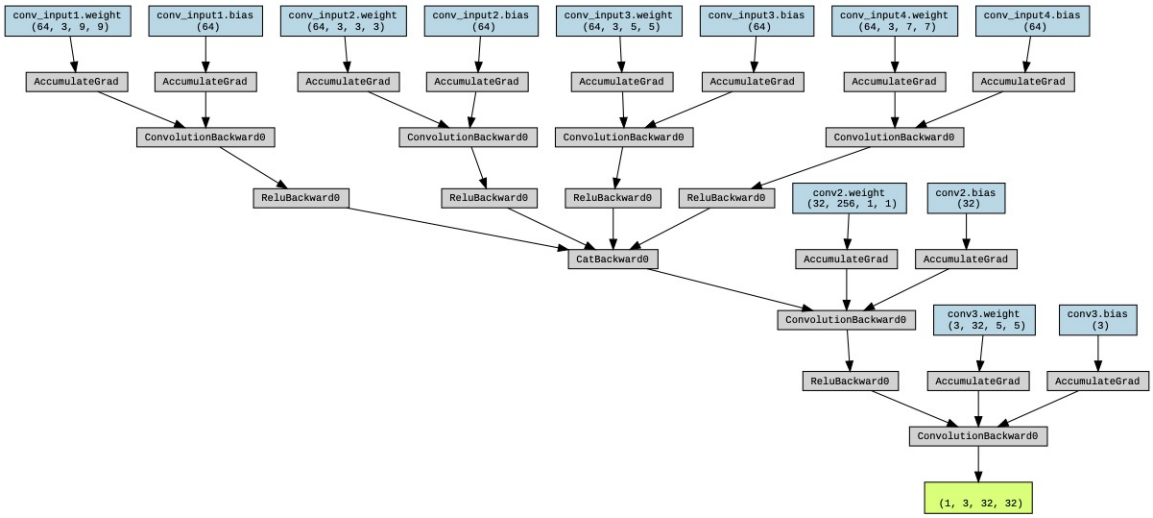

Fig 2. SRCNN v2 model with parallel input conv layer concatenation

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 64, 32, 32]          15,616
            Conv2d-2          [-1, 64, 32, 32]           1,792
            Conv2d-3          [-1, 64, 32, 32]           4,864
            Conv2d-4          [-1, 64, 32, 32]           9,472
            Conv2d-5          [-1, 32, 32, 32]           8,224
            Conv2d-6           [-1, 3, 32, 32]           2,403
================================================================
Total params: 42,371
Trainable params: 42,371
Non-trainable params: 0
----------------------------------------------------------------
```

Fig 3. Model summary and parameters for SRCNN v2 implementation

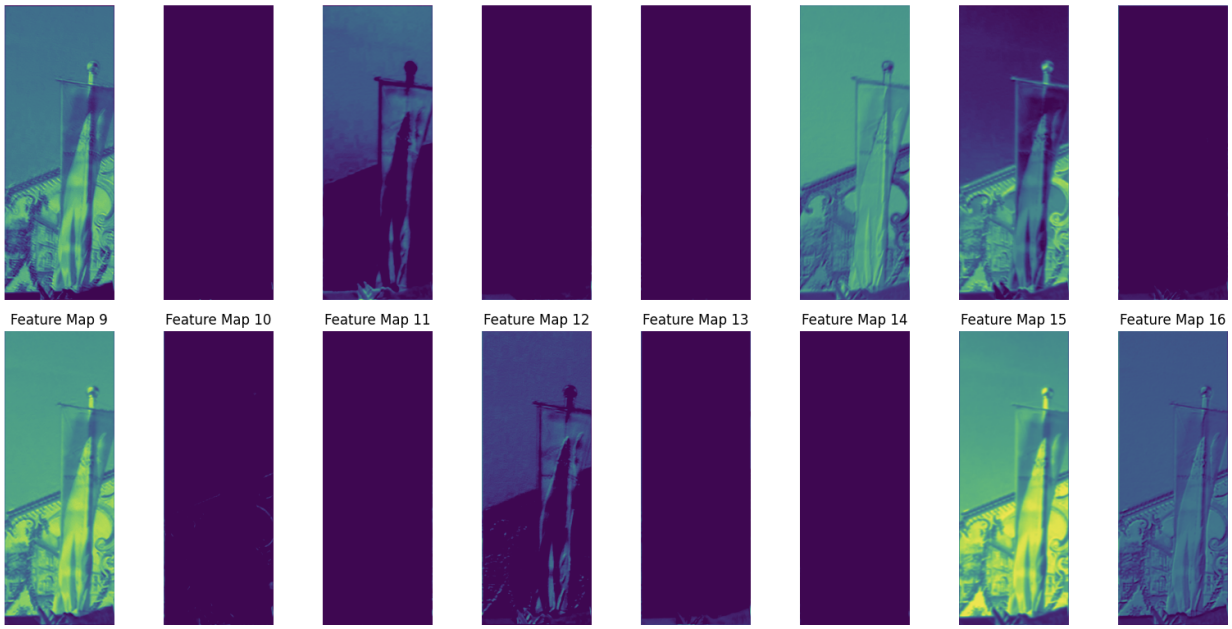

Fig 4. a) Output of initial Conv layer #1 with filter of 9 (First 16 of 32)

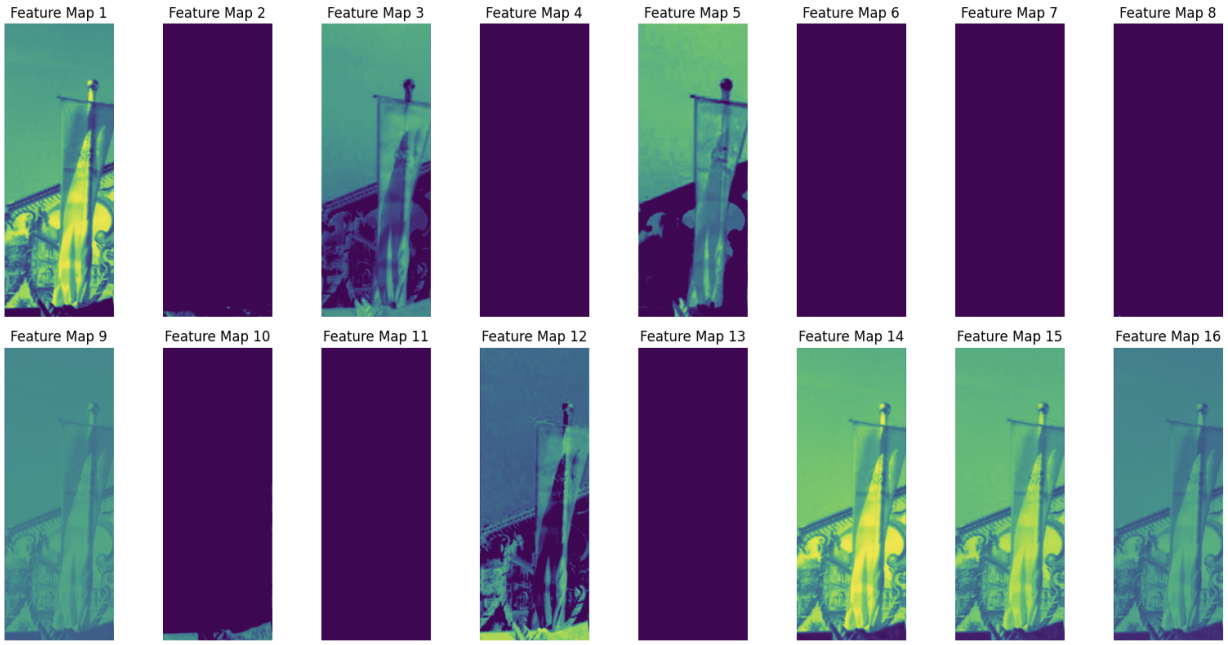

Fig 4. b) Output of initial Conv layer #2 with filter of 3 (First 16 of 32)

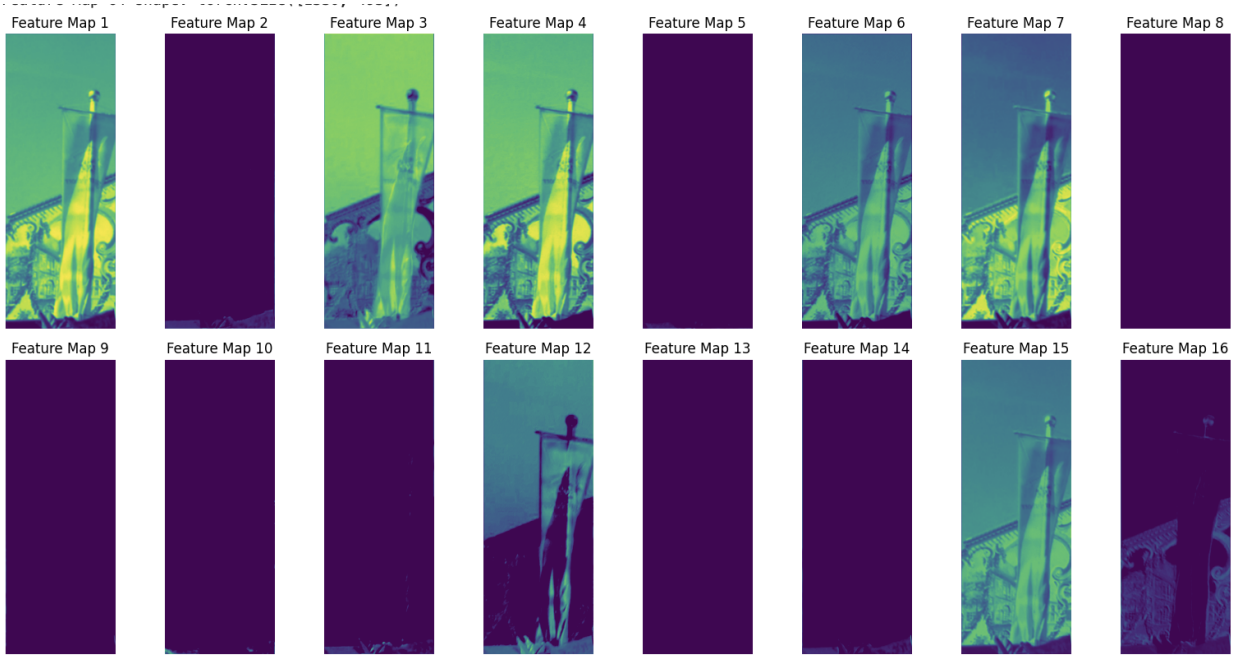

Fig 4. c) Output of initial Conv layer #3 with filter of 5 (First 16 of 32)

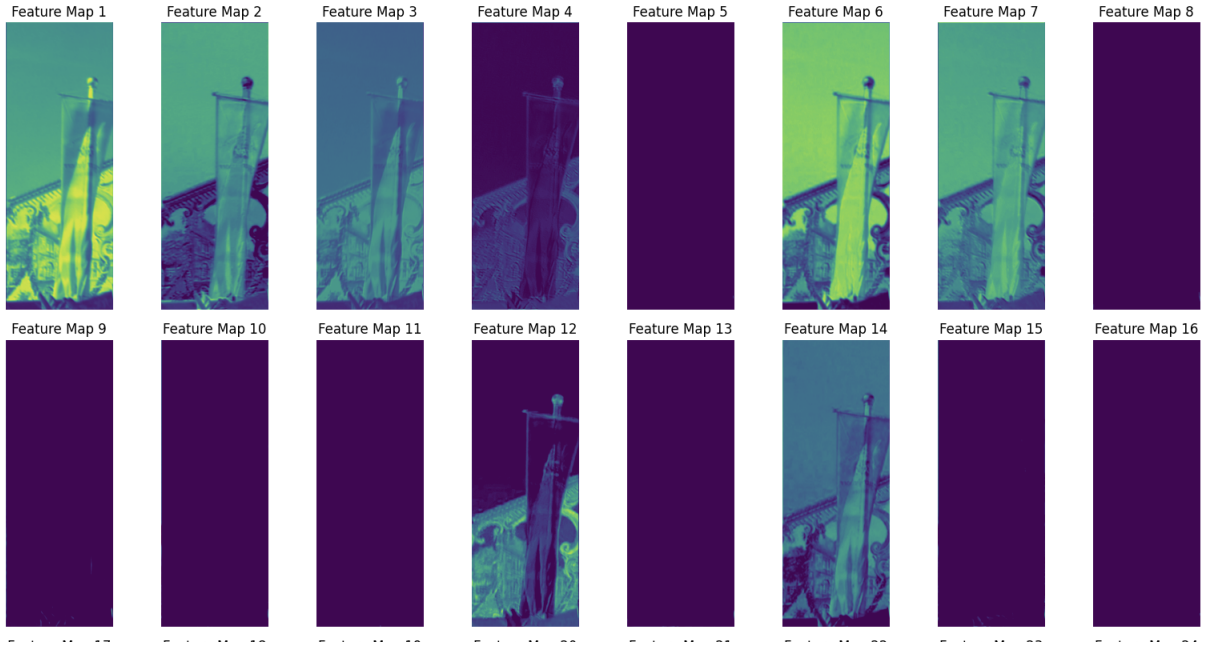

Fig 4. d) Output of initial Conv layer #4 with filter of 7 (First 16 of 32)

Results:

i) Metrics Used:

For this implementation we also consider the PSNR metric, which deals with comparing pixel values between images. It is often used to understand restoration quality of images in such tasks.

Eq 1) $PSNR = 20 * log(Max\ pixel\ value) - 10 * log(MSE)$

ii) Performance of SRCNN with visualization:

The baseline SRCNN with MSE Loss achieved a training PSNR of 40.986 and a validation PSNR of 31.781. This is not as good in comparison to the PSNR reported by the SRCNN paper [1] on the Set5 dataset, a 32.60 PSNR.

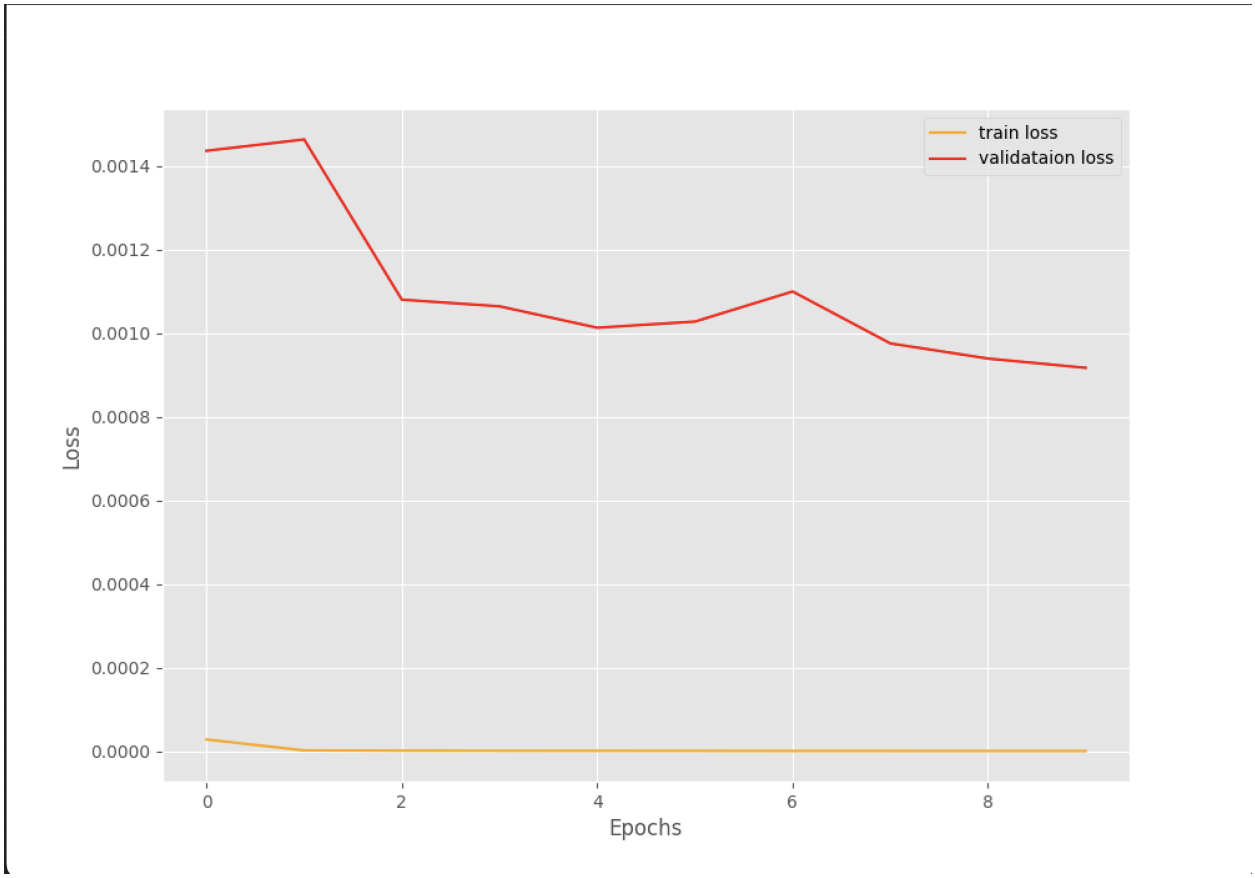

Fig. Loss Graph vs epochs for SRCNN + MSE Loss

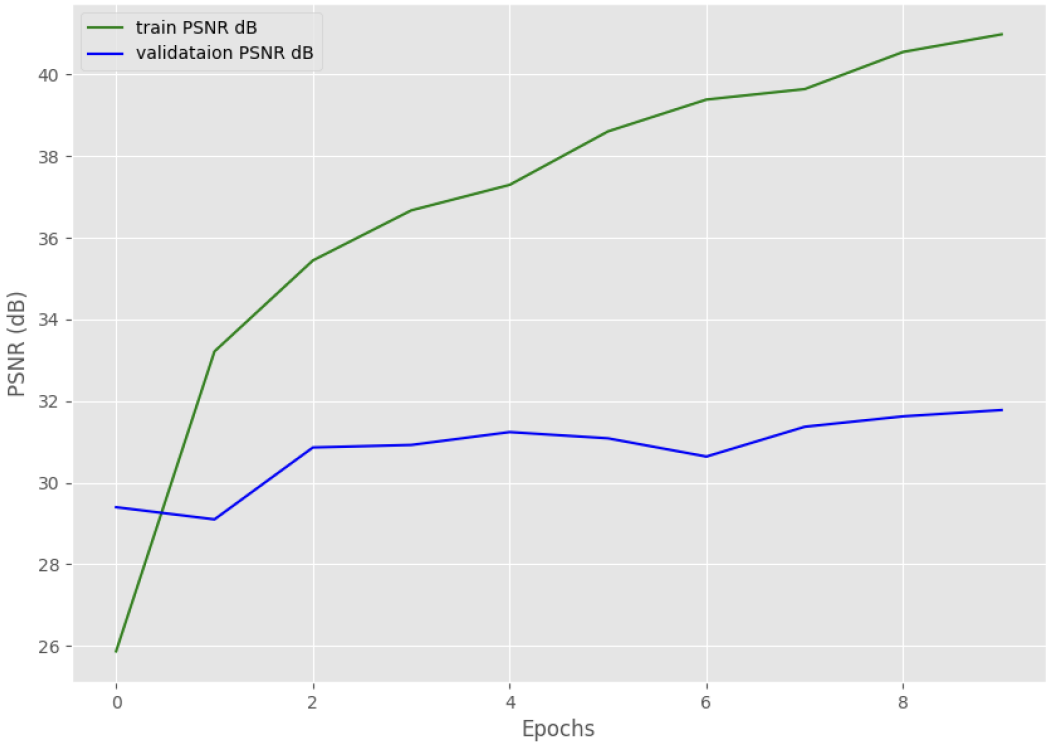

Fig. Train vs Validation PSNR vs epochs for SRCNN + MSE Loss

iii) Performance of SRCNN + VGG Loss with visualization:

The baseline SRCNN with VGG Loss achieved an improved training PSNR of 41.583 and a validation PSNR of 31.238. This is better in comparison to the PSNR reported by the MSE Loss.

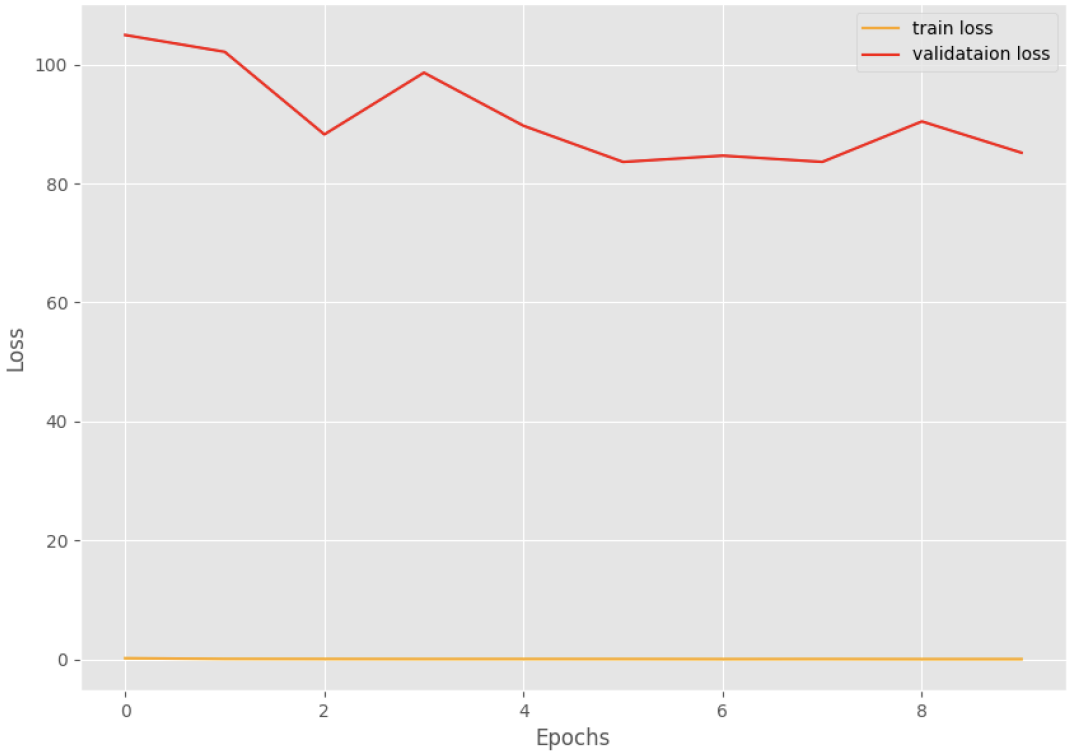

Fig. Loss Graph vs epochs for SRCNN + VGG Loss

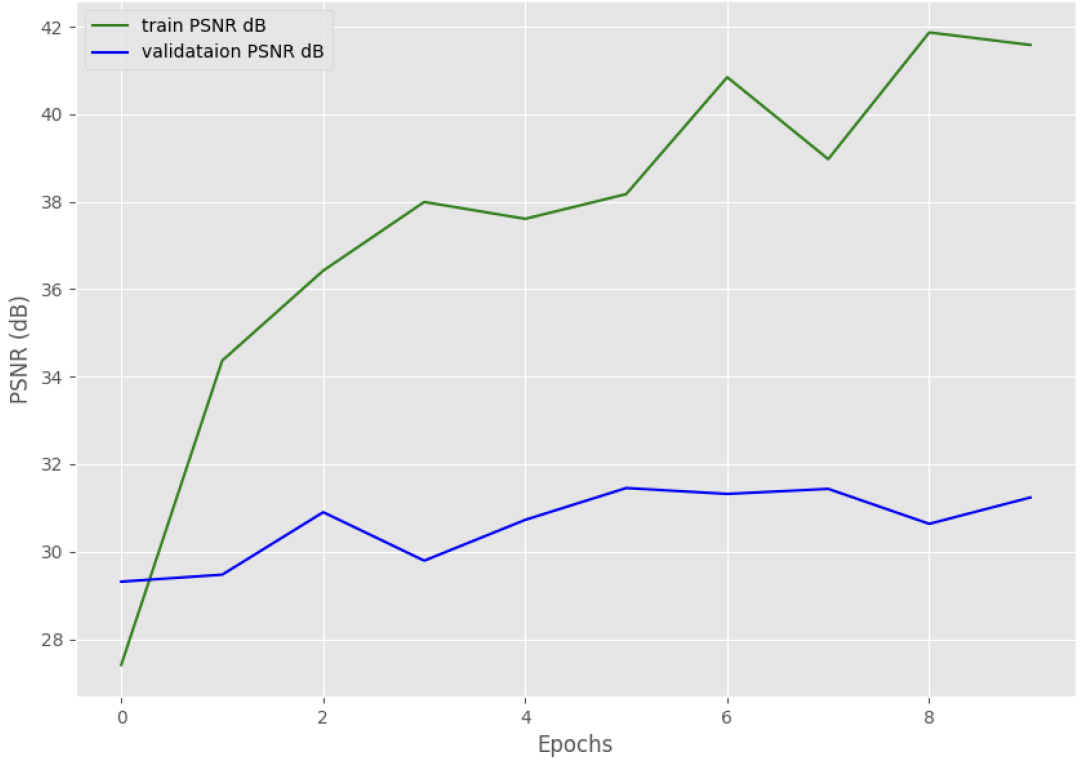

Fig. Train vs Validation PSNR vs epochs for SRCNN + VGG Loss

iv) Performance of SRCNN v2 + VGG Loss with visualization:

The SRCNN2 with VGG Loss achieved an improved training PSNR of 45.027 and validation PSNR of 31.673. This is slightly better in comparison to the PSNR reported by the SRCNN with VGG Loss.

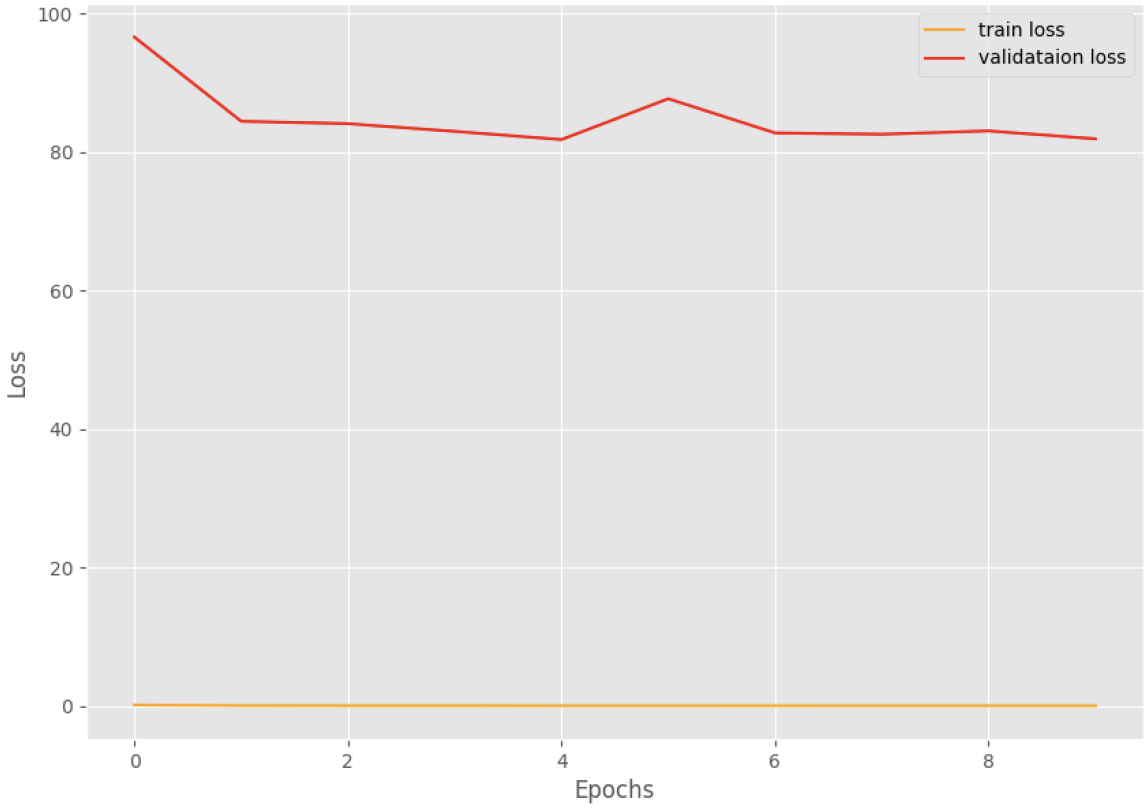

Fig. Loss Graph vs epochs for SRCNN2 + VGG Loss

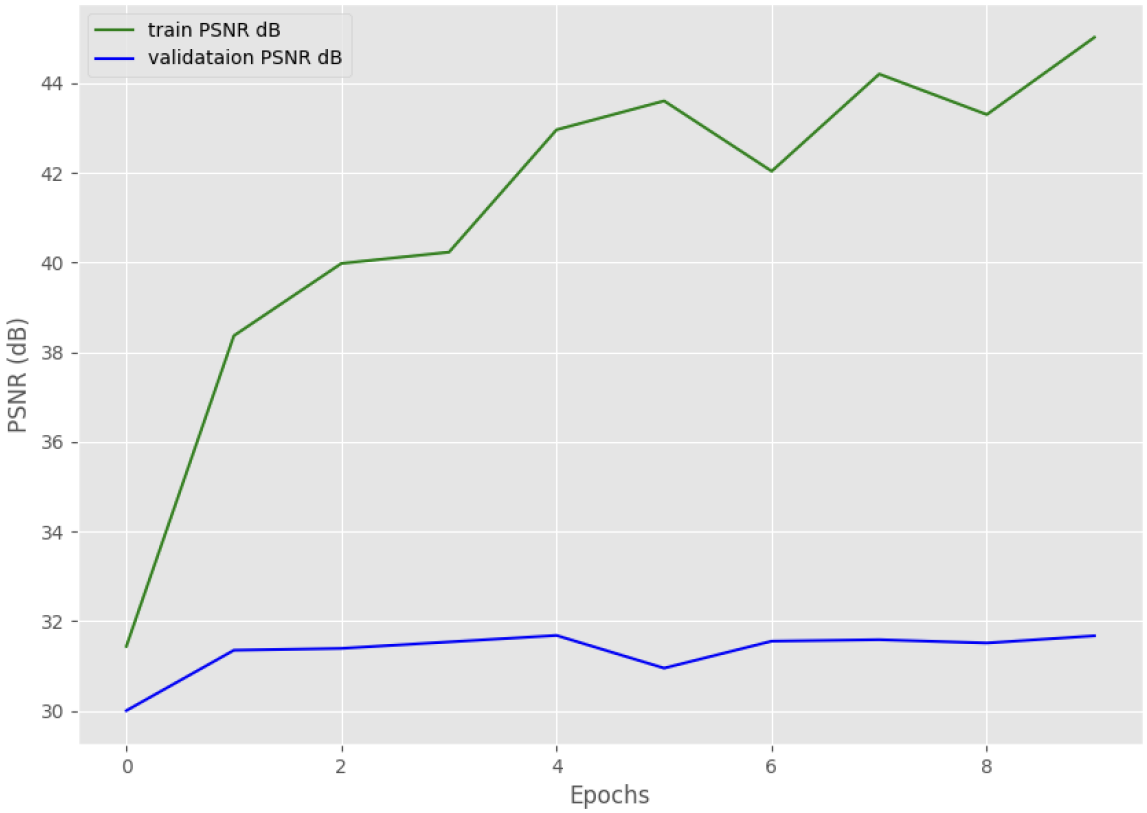

Fig. Train vs Validation PSNR vs epochs for SRCNN2 + VGG Loss

v) Visualization of results

Below is a side by side comparison of the results of the 3 models tested vs the ground truth. The top left is the ground truth, top right is SRCNN with VGG Loss, bottom left is SRCNN with MSE Loss, bottom right is SRCNN2 with VGGLoss.
It is notable to mention that the predicted images are lesser in file size but still report better resolution than the ground truth zoomed in image.

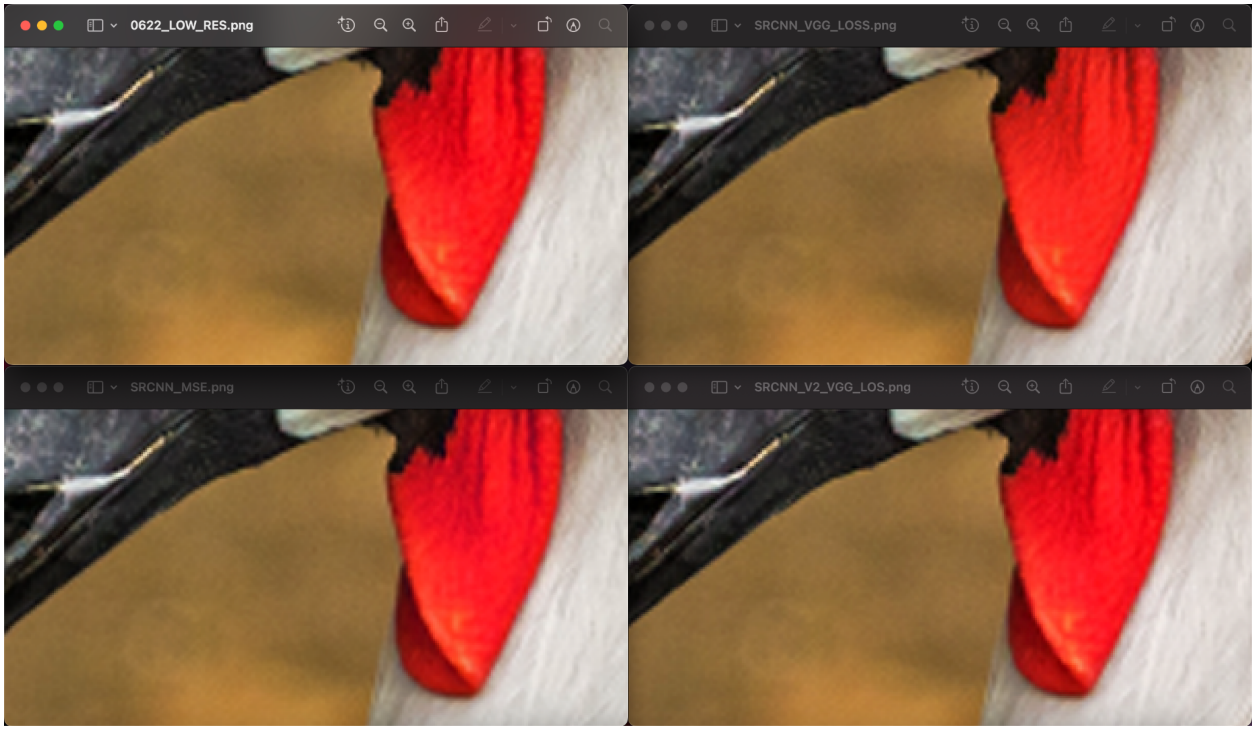

Fig. Comparison of results

vi) Summary of strengths and weakness of implementation:

The paper addresses one of the limitations of the base paper - improving the width of the model with more convolutional layers.
However, there are certain limitations that are still not addressed. This includes the variability of the inputs due to the use of bilinear interpolation as a preprocessing step. The bilinear interpolation is an additional step, that could be a key component that achieves the good performance, rather than the network itself.
In addition, there is another minor limitation, which is, the training of the model on a specific upsample scale only (2x). However, multi-filter training is included.

While the proposed model did not beat the validation performance for 10 epochs, it can be seen from the training PSNR improvements that further training with larger samples can result in better validation results.

Reflection & Acknowledgments:

Through the implementation of this project we were able to understand certain aspects of SISR using the SRCNN that we were previously unsure about.
For example, one source of misunderstanding for us, was on why images needed to be patched into sub images for the dataset when we are trying to obtain super-resolution

on a full image. We learnt that the sub-imaging facilitated the model to learn specific instances or patches of larger images, referring to the example-based method assumption (there are inherent repetitions of patterns in a natural image).

Through trial and error we were able to understand how different filter sizes and padding impacted the dimensions of the image passing through the model. That is, if we were to obtain a reconstructed image at the end of the network specific parameters had to be used.

It was interesting to work on a project that was aimed to achieve something proven, rather than work on something who's result is unpredictable. Through this we achieved a thorough understanding of why each parameter and layer was used. This allowed us to explore improvements like additional parallel layers and loss functions.

Some resources that we used to implement the project include,

1. C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a Deep Convolutional Network for Image Super-Resolution," Computer Vision – ECCV 2014, pp. 184–199, 2014, doi: https://doi.org/10.1007/978-3-319-10593-2_13. **- For the SRCNN implementation**
2. Christian Ledig, Lucas Theis, Ferenc Husza ́r, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, Wenzhe Shi, "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network Actions", CVPR, 2017. **- For the VGGLoss implementation**
3. Eirikur Agustsson and Radu Timofte. NTIRE 2017 chal- lenge on single image super-resolution: Dataset and study. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 126–135, 2017. **- DIV2k dataset**
4. https://medium.com/coinmonks/review-srcnn-super-resolution-3cb3a4f67a7c **- To understand the basis of claims in the SRCNN paper and its strengths and weaknesses.**
5. https://debuggercafe.com/image-super-resolution-using-deep-convolutional-networks-paper-explanation/ **- insight on paper structure**
6. https://debuggercafe.com/srcnn-implementation-in-pytorch-for-image-super-resolution/ **- For ideas on the implementation of the model and training scripts only**
7. https://github.com/sgrvinod/a-PyTorch-Tutorial-to-Super-Resolution/blob/master/utils.py **- To understand inference through Pytorch**