Efficient Storage with On-Demand Single-Image Super-Resolution

Rajwinder Mahal
rsm2207@columbia.edu

Abstract

Storing high-resolution images is a challenging problem due to its high storage cost and growing storage demands. To address this issue, we explore the feasibility of using deep convolutional networks for efficient storage and on-demand high resolution image generation. In this paper, we conducted experiments using Super-Resolution Convolutional Neural Network (SRCNN) and Efficient Sub-Pixel Neural Network (ESPCN) to generate high resolution (HR) images given low resolution (LR) images as the inputs. We evaluated the performance of SRCNN and ESPCN with a scaling factor of 4x. Our results show that SRCNN and ESPCN can produce promising results in producing high-resolution images with improved quality compared to traditional upscaling methods such as bicubic interpolation. However, we should carefully considered the trade-off between image quality and computational cost as SRCNN and ESPCN can be computationally intensive for both training and real-time inference. Our findings suggest that single-image super-resolution (SISR) with SRCNN and ESPCN can be a promising approach for efficient storage and real-time high resolution image generation, but further experiments are needed to optimize the performance and scalability of this technique for different applications.

1. Introduction

Single-image super-resolution (SISR) has emerged as a promising technique for increasing the spatial resolution of images. It has various applications in computer vision, including medical imaging, surveillance, and video streaming. One of the significant challenges associated with storing high-resolution images is the high storage cost and large storage space requirements. This challenge becomes even more critical when dealing with social media sites and image-sharing platforms that host millions, if not billions, of user-generated images. Storing a large amount of high resolution images can be very costly for these platforms.

To solve this problem, we propose a solution to store user-generated images as low resolution (LR) and generate high resolution (HR) images on demand using SISR techniques. Our goal is to significantly reduce storage costs while maintaining high image quality when serving to the end users. In this paper, we will use Super-Resolution Convolutional Neural Network (SRCNN) and Efficient Sub-Pixel Neural Network (ESPCN). Our experiments focus on evaluating the performance of SRCNN and ESPCN using much smaller datasets as compared to datasets used in the original publications of these models.

Our proposed solution has the potential to provide a cost-effective and sustainable way to store images at scale because most of the images, especially on social media sites and image-sharing platforms, are typically accessed frequently for a very short period of time (ranging from few days to few weeks). After that period, most of the images are either never accessed or accessed rarely. Therefore, we can store most of those images as LR which will significantly reduce storage cost. To serve HR images efficiently, we can use a smart cache strategy that will store frequently accessed images in the cache as HR. For example, whenever an image in the cache is accessed, we can renew the cache expiry and the image is only removed from the cache if it is not accessed before the cache expiry duration. Setting a cache expiry duration is application specific and if done properly can reduce significant computational cost associated with converting LR image to HR.

2. Related Work

Social media sites and image-sharing platforms such as Facebook and Google Photos uses a combination of techniques to store billions of images uploaded by their users. These techniques include compression and resizing to reduce image size while maintaining quality. Some of the most widely used compression techniques are JPEG and PNG. These compression techniques can reduce significant storage and transmission costs, however, these techniques either lead to a loss of image quality or cause minor reduction in size. For example, lossy compression techniques such as JPEG can degrade image quality, especially at high compression ratios. On the other hand, lossless compression techniques such as PNG may not achieve significant compression ratios especially for high resolution images.

In recent years, deep learning based super-resolution techniques such as SRCNN and ESPCN have gained significant attention due to their ability to generate high quality images. More advanced generative models such as Generative Adversarial Networks (GANs) can generate even higher resolution images with realistic details. However, these generative models are more complex and computationally expensive. Therefore, we are only going to evaluate the performance of SRCNN and ESPCN in this paper.

3. Method

In this paper, we are evaluating two deep learning based models for single image super-resolution. In this section, we describe these models in more details, along with the metrics we have selected for evaluating their performance, and the datasets we have used for training and validation.

3.1 SRCNN

Super-Resolution Convolutional Neural Network (SRCNN) is one of the most commonly used deep learning based super-resolution models. SRCNN utilizes a three-layer CNN to learn the mapping between low resolution and high resolution images. The SRCNN model is optimized to minimize the mean squared error (MSE) between the predicted high resolution image and the ground truth high resolution image. There are multiple variants of the SRCNN model as described in the original publication. The 9-5-5 network achieves the best performance but at the cost of the running time. As we are converting low resolution images to high resolution in real-time, we decide to use the 9-1-5 variant which is the fastest among all with marginal difference in performance.

3.2 ESPCN

Efficient Sub-Pixel Neural Network (ESPCN) is a single image super-resolution (SISR) method that increases the spatial resolution of an image by reconstructing high resolution details from a low resolution image. ESPCN uses a sub-pixel convolutional layer to upscale the image by rearranging the low resolution image data into a high resolution grid to increase the number of pixels in the image.

We decide to use ESPCN along with SRCNN because SRCNN model can be computationally expensive and may not be suitable for real-time applications. So, we will also evaluate ESPCN which is considered more efficient than SRCNN while maintaining high image quality.

3.3 Metrics

To evaluate the performance of SRCNN and ESPCN, we are using Peak Signal-to-Noise Ratio (PSNR). This is a widely used metric that measures the quality of the image by calculating the ratio of the maximum possible power of a signal to the power of corrupting noise that affects the representation. A higher PSNR value indicate higher image quality. To have a baseline PSNR value, we use bicubic interpolation to convert low resolution image to high resolution and then use its PSNR value as the baseline to compare the performance of SRCNN and ESPCN. There are other performance metrics such as Structural Similarity Index (SSIM) but given the limited time, we are going to use the PSNR as our evaluation metric.

For training and validation, we used mean squared error (MSE) loss function. It measures the average squared difference between the predicted high resolution image and the ground truth high resolution image.

3.4 Training Data

Deep learning models are usually trained on large datasets. In the SRCNN paper, the author used a dataset of 91 images which was decomposed into 24,800 sub-images and another dataset of 395,909 images which was decomposed into over 5 million sub-images. Similarly, in the ESPCN paper, the author used a randomly selected set of 50,000 images from the ImageNet to train the model.

Given the limited time and resources, we used DIV2K dataset which is a widely used benchmark dataset for image super-resolution. The dataset consists of 800 training, 100 validation, and 100 testing images. We also used a variant of DIV2K dataset by decomposing it into over 21,000 training and 2700 validation sub-images. To create the sub-images, we first resized low resolution images to match high resolution image dimensions and then extracted sub-images with a patch size of 256x256.

Both our datasets consists low resolution images and 4x high resolution images. For ESPCN, we use these images as it is with some normalization and resizing as it takes low resolution input and outputs a high resolution image which is upscaled by a given factor (4x in our case). However, SRCNN requires input and output images to be of same size. So, to upscale low resolution images, we first upsample an image to a bigger size and then downsample it by a factor of 4 and then resize it again to match the output size.

4. Experiments

4.1 Initial Experiment

We train both SRCNN and ESPCN on DIV2K and DIV2K sub-images datasets. For the starting point, we used a learning rate of 1e-3 with a multi-step scheduler that decays learning rate by a factor of 0.1 with milestones at 48, 72, and 88 epochs. We used a batch size of 64 and 16 for the training and validation, respectively for the smaller dataset. For the bigger dataset, we used a batch of 128 and 100 for training and validation, respectively. We have trained our model for 100 epochs.

Figure 1 shows the training and validation loss. We can see that both SRCNN and ESPCN has lower loss when trained on bigger DIV2K sub-images dataset. Similarly, figure 2

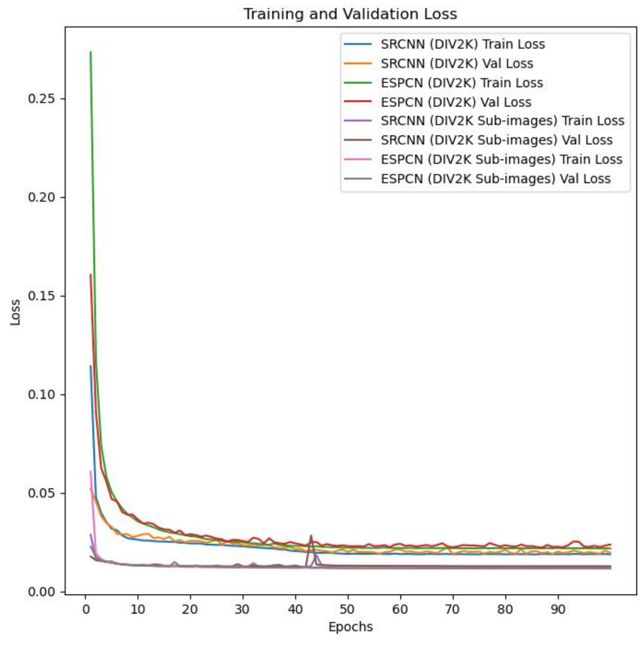


Figure 1: Training and validation loss of models when trained on DIV2K and DIV2K sub-images dataset.

shows that bigger dataset results in better PSNR value. It is clear that training on bigger dataset will result in better

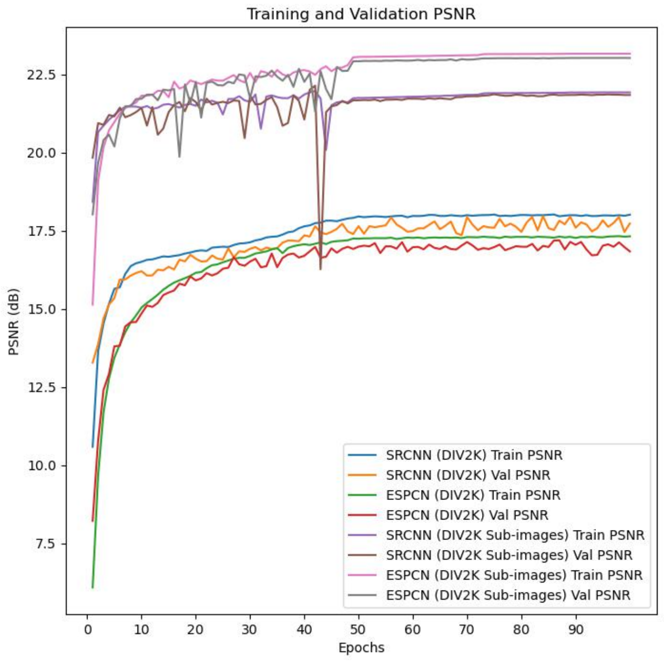


Figure 2: Training and validation PSNR (dB) of models when trained on DIV2K and DIV2K sub-images dataset.

model performance and hence it is critical to train these models on big enough datasets to have better performance.

4.2 Learning Rate

We experimented with different learning rates and found that 1e-3 is the best learning rate resulting in the best PSNR value and minimum loss. We have used Adam as the optimizer and initially experimented with different learning rate weight decay step sizes. In figure 3, after epoch 48, the PSNR value stops improving, especially when using bigger dataset. This was due to learning rate weight decay after epoch 48. So, we decided to use 1e-3 as the learning rate with learning rate decreasing to 1e-4 after epoch 88.

4.3 Batch Size

Large batch sizes can cause the model to converge quickly and can lead to overfitting and poor generalization. In figure 4, we compared SRCNN with batch size of 16 and 64. From the figure, it is clear that a smaller batch size of 16 results in slightly better performance. However, a batch size smaller than 16 doesn't show any differences during our experimentation. We also experimented with a bigger batch size of 128 which resulted in faster converge without significant difference in performance. We also tried to use a

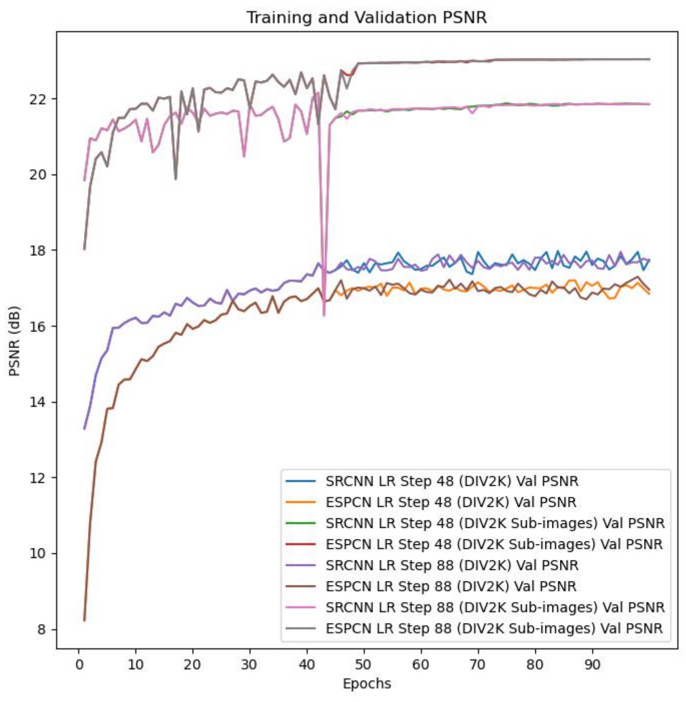


Figure 3: Training and validation PSNR (dB) of models with different learning rate step decay.

batch size of 256 for the bigger dataset but it caused a memory error.

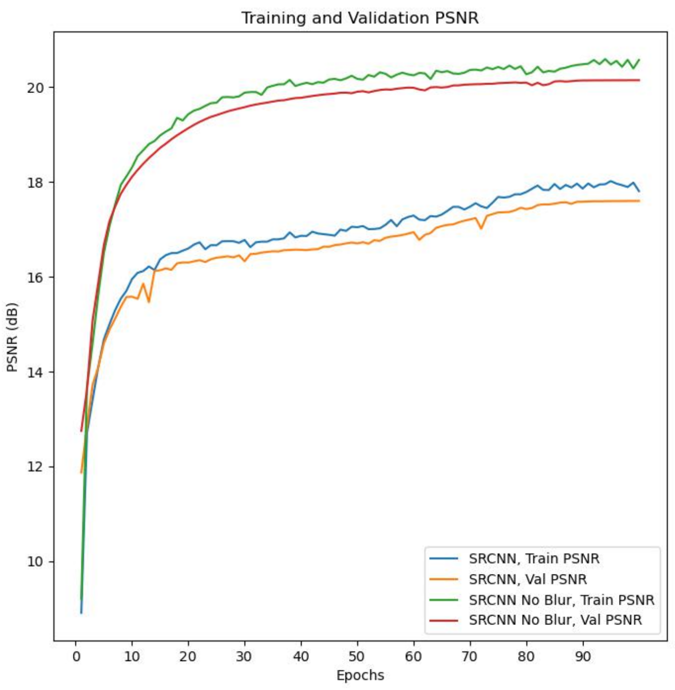


Figure 5: Training and validation PSNR (dB) of models when trained with and without gaussian blur on DIV2K dataset.

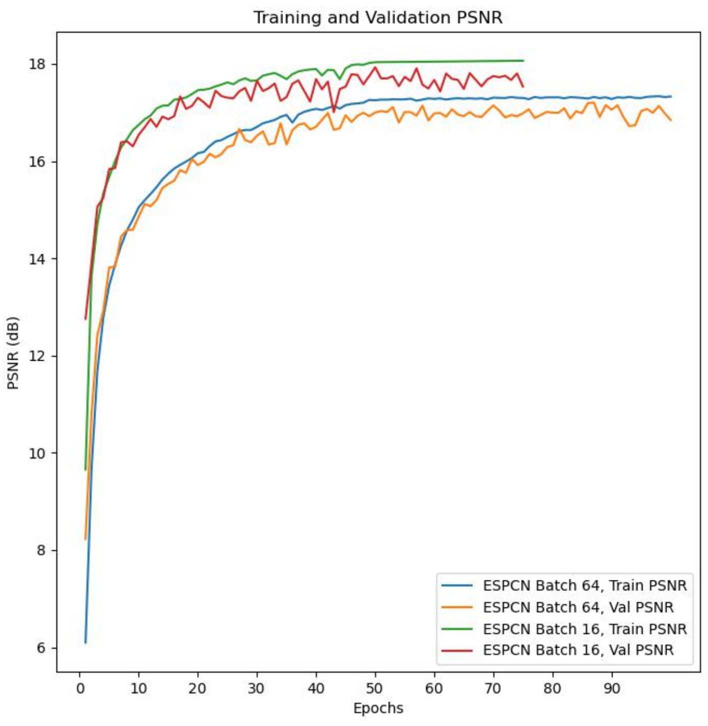


Figure 4: Training and validation PSNR (dB) of models with different batch size when trained on DIV2K dataset.

4.4 Gaussian Blur

Gaussian blur is a common image processing technique for generating low resolution images for super-resolution tasks. It blurs a given image by averaging the pixel values. However, when we used it as a preprocessing step, we saw a decrease in performance of both SRCNN and ESPCN. So, we conducted an experiment to compare the performance with and without gaussian blur. Figure 5 shows that model without gaussian blur gives significantly better performance as compared to model using gaussian blur as the preprocessing step. So, we decided to not use it in our final model.

4.5 Input Size

When using larger image input size, the SRCNN and ESPCN can capture more detailed features and textures which can help in producing more realistic high resolution outputs. We experimented with two different sets of input sizes, 112x112 and 224x224. Figure 6 shows that larger input size results in better performance in both SRCNN and ESPCN.

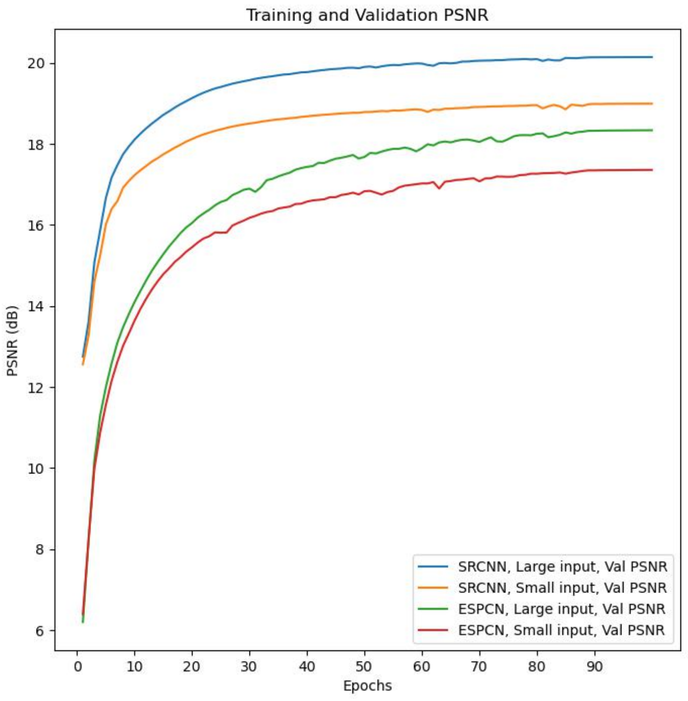


Figure 6: PSN (dB) with different input size.

4.6 Final Model

We have trained our final SRCNN and ESPCN models for 100 epochs using a training batch size of 128. For validation, we used a batch size of 100 for bigger dataset and a batch size of 50 for smaller dataset. We used a

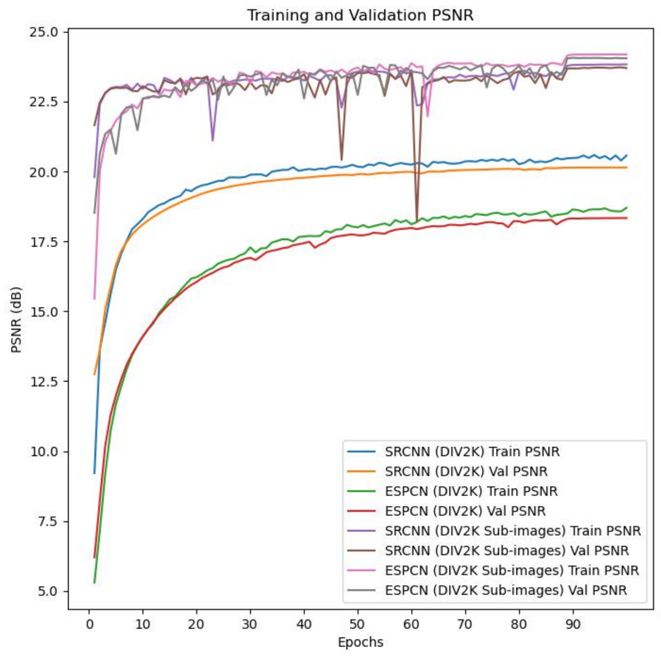


Figure 8: Training and validation PSNR (dB) of fine-tuned models trained on DIV2K and DIV2K sub-images dataset.

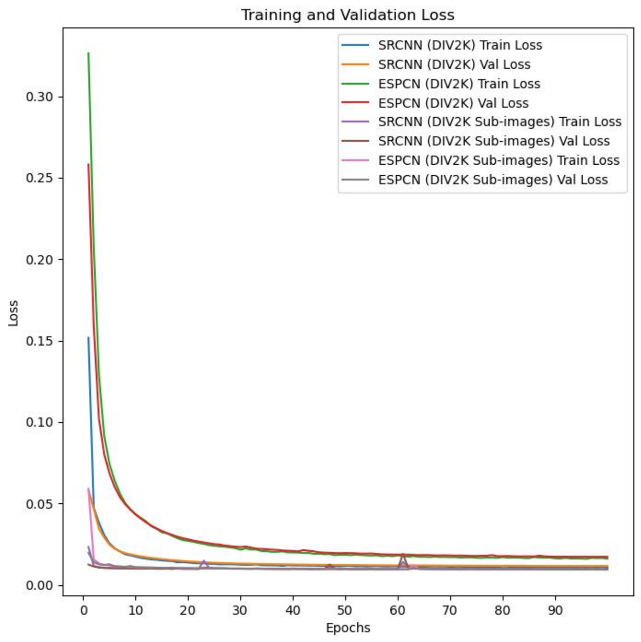


Figure 7: Training and validation loss of fine-tuned models when trained on DIV2K and DIV2K sub-images dataset.

starting learning rate of 1e-3 with Adam as the optimizer and a learning rate decay by a factor of 0.1 after 88 epochs.

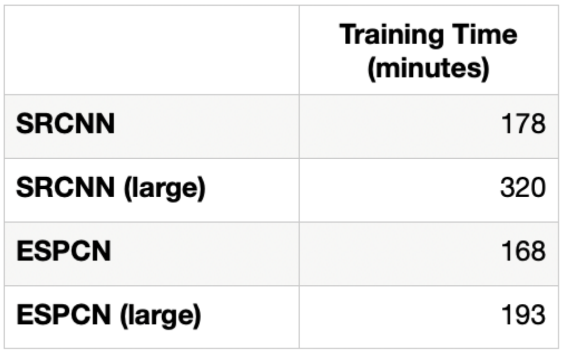
Figure 7 shows that bigger dataset results in lower loss. We also noticed that ESPCN results in bigger loss when trained on smaller dataset as compared to SRCNN. In figure 8, we can see that both models result in significantly lower performance when trained on smaller dataset. It also shows that ESPCN performs significantly worse than SRCNN when trained on a smaller dataset. When trained on bigger dataset, both models have identical performance with ESPCN performing slightly better than SRCNN.

	PSNR (dB)				
	DIV2K Validation	**DIV2K Sub-images Validation**	**Set5**	**Set14**	**Average**
SRCNN	20.14		16.28	15.57	**17.33**
SRCNN (large)		23.73	14.42	12.66	16.94
ESPCN	18.33		15.74	15.23	16.43
ESPCN (large)		24.06	14.37	12.86	**17.1**
Bicubic	18.83		16.10	15.73	16.89

Figure 9: PSNR (dB) comparison on different datasets with bicubic interpolation as the baseline metrics.

4.7 Evaluation

We used Set5 and Set14 to test to performance of SRCNN and ESPCN along with the DIV2K validation set. Both Set5 and Set14 are standard benchmark datasets used in



	Training Time (minutes)
SRCNN	178
SRCNN (large)	320
ESPCN	168
ESPCN (large)	193

Figure 10: Training time comparison

super-resolution tasks, and consists of 5 and 14 images, respectively. Figure 9 shows the performance of 4 models on different datasets. Generally, CNN models perform worse on smaller datasets and figure 9 shows that both SRCNN and ESPCN perform worse when trained on smaller datasets. We also noticed that SRCNN performs better than ESPCN when both models are trained on smaller dataset but ESPCN slightly performs better when trained on large dataset. When compared to the baseline bicubic PSNR value, we found that bicubic interpolation results in better PSNR value as compared to models trained on smaller dataset.

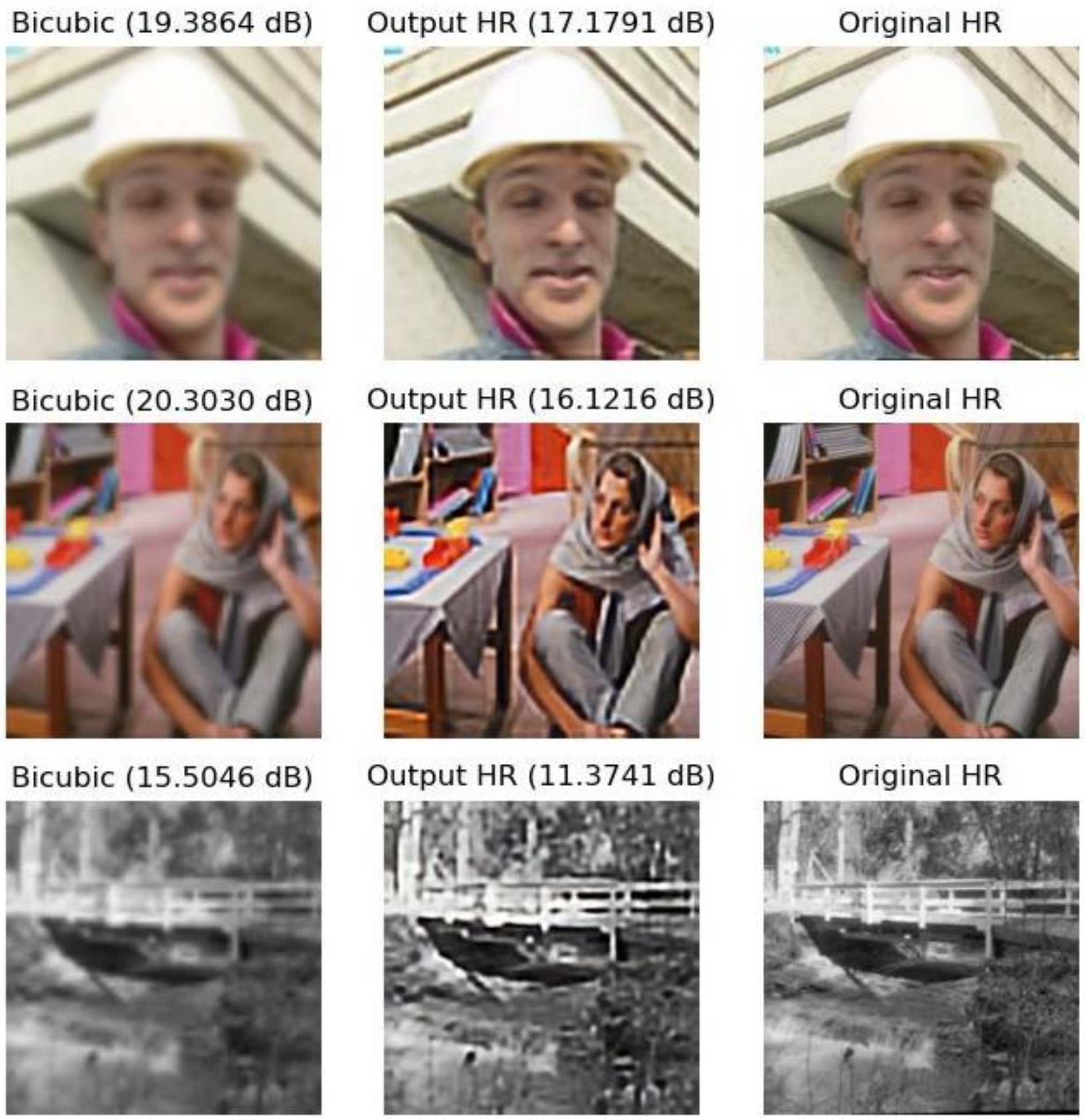


Figure 11: Comparing predicted images on Set5 on SRCNN model trained on smaller dataset.

One interesting thing we noticed after testing the models on Set5 and Set14 datasets is that they perform much worse as compared to performance on div2k validation dataset. The baseline bicubic interpolation PSNR value is better in these cases. This shows that both SRCNN and ESPCNN are not well tuned to handle images that are from completely different dataset with different features.

When comparing the training time, we notice that ESPCN model trains much faster as compared to SRCNN. Figure 10 shows that ESPCN trains significantly faster when trained on large dataset as compared to SRCNN.

The figure 11 shows a comparison of predicted high resolution output by SRCNN model as compared to simple bicubic interpolation and the ground truth high resolution image. From the figure, we can see that images with bicubic interpolation have high PSNR values as compared predicted model even though we can see that predicted images look way better than the images upsampled using bicubic interpolation. One reason may be that images upsampled using bicubic interpolation are much smoother than predicted which resulted in better PSNR value.

5. Conclusion

In this paper, we have experimented with different deep learning based super-resolution models such as SRCNN and ESPCN. Although our proposed models were not able to perform significantly better than the baseline bicubic interpolation, we still think that these models can produce promising results if trained on large enough datasets. For example, authors of SRCNN and ESPCN have trained models on much bigger datasets as compared to datasets we used. Apart from using bigger datasets, we can experiment with different preprocessing techniques such as laplacian pyramid super-resolution, progressive upsampling, etc. We can even use transfer learning by using pre-trained models to fine-tune them for image super-resolution.

GitHub Link: https://github.com/mahalrs/super-scaler

References

[1] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks, CoRR. abs/1501.00092 (2015).

[2] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network, CoRR. abs/1609.05158 (2016).

[3] C. Ledig et al. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network, CoRR. abs/1609.04802 (2016).