Experiments in Single Image Super Resolution using Convolutional Neural Networks and Auto-encoders

Shashwat Gupta u7034693
Sree Venkatesh Yelamolu u7095134
Yashaswi Khandelwal u6846998
Shikhar Mishra u6203537

Abstract

Image enhancement is a fundamental problem in image processing, and deep learning based upsampling methods have evolved to be more effective than conventional interpolation-based techniques[26]. We used the original Super-Resolution Convolution Neural Network (SRCNN)[3] in PyTorch, with the aim of improving final output quality by producing crisp edges. We implemented the perceptual Mean Gradient Error (MGrE) loss function, using which we obtained a testing PSNR of 26.117 and training SSIM of 91.6%. We tested our model by adding noise and blur to inputs, and found that the mixed loss model gives the most aesthetically pleasing results, which we feel is as important as higher scores on conventional metrics. We further implemented an auto-encoder architecture which harnesses the ability to learn compressed representations of high-resolution training data and use the same to reconstruct them. We were able to achieve an average testing PSNR of 26.05 with a max PSNR of 33 on DIV2K dataset using auto-encoders.

1 Introduction

Image quality plays an important role in human and machine perception of the objects present in an image[5]. The quality of the image is important to perform computer vision tasks like object detection[7], facial recognition [24], satellite[19] and medical imaging[6] etc. Image 'quality' is mainly a factor of resolution, however attributes such as noise, blur etc also play a role.

The aim of single-image super-resolution (SISR) techniques is to obtain high-resolution (HR) with better visual quality equivalents of low-resolution (LR) coarse-detailed images[26]. Traditional upsampling techniques have been used for some time, with the most common techniques involving some kind of interpolation between the pixels, such as bilinear or bicubic sampling. These approaches are fast and easy to implement, however their results have scope for improvement. As processing became cheaper in the 2000's, reconstruction based techniques were proposed[28]. Firstly, there were parametric models which analysed keypoints like edges[28]. Later there were data-driven, patch-based algorithms which analysed further details of the scene, like textures, or objects to upsample[28]. The relatively new paradigm of deep learning based super resolution methods have recently evolved to be more effective than conventional techniques[26].

Images can have a smaller resolution due to degradation, such as blurring or noise addition, or because of a small spatial resolution. This degradation can be thought of a function applied on the HR image to obtain the LR image. Theoretically, it is possible to apply the inverse of the degradation function to achieve the HR image, however estimating such a function is not a well posed problem, and therein lies the challenge[26]. Research in this field has increased since the proposal of the Super-Resolution convolution Neural Network (SRCNN) in 2014 by Dong[3] and his research team.

Researchers recently released a survey paper where they classified thirty contemporary convolution Neural Network based super-resolution models into nine broad categories according to their salient features[2]. The earliest models had simple linear designs with a single path for signal flow, which is why they were an excellent choice for our group to experiment, code and learn to innovate upon. The

key difference between these linear architectures is that some perform upsampling early in the net such as SRCNN[3] and VDSR[11], while other others do it later in the network, like FSRCNN[4].

In our work, we refined the training and performance of SRCNN by combining MSE loss with Mean Gradient Error (MGrE) perceptual loss[13] during training. This new mixed loss training regime is able to accurately preserve edges in the images, due to the use of the Sobel operator[22]. Using 100 images of the Div2k dataset[1] for testing, we obtain an excellent score of about 27dB PSNR.

In addition to improving the SRCNN model, we have followed the auto-encoder architecture byMao et al. [14], while fine tuning the same, to generate high quality high definition images from low-resolution images. Auto-encoders have been proven to be effective in tasks like image denoising, color restorations and image compression, while our experimentation show their ability to reconstruct high quality images by achieving an average PSNR of 26.05 when tested with DIV2K dataset.

2 Literature Review

2.1 SRCNN and 'Early' Upsampling Linear Networks

We first focussed on SRCNN which is the original Super Resolution convolution Neural Network[3]. It has three convolution and two Rectified Linear Unit (ReLU) layers for non-linearity. SRCNN is the first successful attempt towards using convolution layers for super resolution. It consists of convolution layers each followed by a non linearity unit. The first convolution layer does patch extraction to create feature maps of the low-resolution input images. The second converts them into high-dimensional feature vectors, and finally the last layer combines it all together to get the high-resolution image. This linear design was the inspiration for later deep learning networks to solve the problem of super resolution. The network uses pixel-wise MSE loss. However, as we will see, using the MSE loss favours a higher PSNR, which is a metric used for 'quantitatively' evaluating restored image quality, but has little relation with the human perception of the image. For further details about the SRCNN architecture, please refer to the original paper by Dong et al. [3].

2.2 FSRCNN and 'Late' Upsampling Linear Networks

Upsampling later can improve efficiency, and run-time speed, allowing real-time models for use on say edge devices. Dong et al. [3] followed up with Fast Super-Resolution convolution Neural Network (FSRCNN) at ECCV 2016[4]. Late upsampling designs unlike early upsampling designs aren't computationally expensive as early upsampling designs which upsample the LR image first before learning hierarchial features. FSRCNN is basically SRCNN network designed for late upsampling and ESPCN[20] is based on late upsampling improves computational speed and quality of the output super resolution image than the earlier designs. Later network architectures, like VDSR[11] and DNCNN[31] were based on very deep convolution neural networks introduced to improve the performance, but they rely heavily on the accuracy of the noise estimation.

2.3 Other network architectures

With the success of ResNet[8], people started applying skip-connections to the super-resolution problem, which gave birth to Residual style networks [2] which learn the residue between the input and ground truth images. They utilise skip connections to avoid vanishing gradient problem which makes it feasible to design deep networks. These architectures were found to achieve better performance compared to simpler networks like SRCNN[3] and VDSR[11]. Then, Densely Connected Networks[2] we inspired from DenseNet[9], which achieve high flexibility and richer feature representation by combining hierarchical cues. An example of this type of network is the SRDenseNet[23]. Apart from these, multi-branch networks[2] obtain a diverse set of information at a number of different context scales, with the extra information helping to obtain better HR reconstructions. While all the these networks consider spatial locations and channels to obtain a uniform importance for the super resolution, Attention-based networks[2] only selectively focusses to few features at a given layer and thereby allowing flexibility for the model. Further, since it is not always feasible to assume that the degradation will be bicubic as it is possible for multiple degradations to occur simultaneously, therefore, networks like ZSSR[21] (Zero-Shot Super Resolution) use multiple-degradation handling approach in order to obtain a high-resolution image.

More recently, General Adversarial Networks or GANs have been applied to the super-resolution problem[26]. They have two components, a generator and discriminator. The generator tries to create Super Resolution images to trick the discriminator into believing that they are real high-resolution images instead of being artificially generated, and training progresses in this game theoretic fashion until the discriminative model can't identify the difference between the model generated data and the original data thus giving rise to the best parameters for getting a high-resolution image from a low-resolution image. This approach has produced among the best results so far, and is the main direction of future research in this field. Some examples of important GAN models for super resolution tasks include SRGAN[12], ESRGAN[25] and SRFeat[15].

2.4 Auto-encoders

An auto-encoder is an encoding-decoding framework with symmetric convolution-deconvolution layers which learn end-to-end mappings from low-resolution images to high-resolution Images. This framework can be trained to learn compressed representations from LR image. These representations can be used to retain the most important information from input images and use them to completely reconstruct the image in higher resolution.

A very deep fully convolution auto-encoder network for image restoration was proposed by Mao et al. [14], where they used their architecture to accurately perform tasks like image denoising, super resolution removal of JPEG compression artefacts, non-blind image deblurring and image recolorisation.

The architecture of an auto-encoder consists of three parts, encoder, bottleneck and decoder. In our research we construct an auto-encoder model by only using convolution layers. The encoder consists of stacked convolution layers and thus act as a feature extractor. It aims to learn and preserve primary components of low-resolution images meanwhile eliminating corruptions in the image. Whereas the decoder consists of stacks deconvolution layers which thus are responsible for recovering a clean image from inputs while increasing its resolution.

Auto-encoders provide the ability to represent high- resolution feature maps in compressed representations which can be accessed through the bottleneck layer. These feature maps consist of all the necessary information that is required to reconstruct a sharper and upsampled image. Refer to the appendix for the architecture of the auto-encoder.

Different variants of auto-encoders like variational auto-encoders [14] and coupled deep auto-encoders [30] have been implemented in research for the task of super resolution to get superior effectiveness and efficiency. Though these models have only been trained and tested on standard resolution datasets like Set5, Set14, BSD100 and Cifar-10. In order to understand the effectiveness of auto-encoders on high definition high-resolution images, we have implemented a deep convolution auto-encoder framework which is trained and tested on DIV2K dataset.

2.5 Mean Gradient Error (MGrE) and losses for Super resolution

The MSE loss (L2) function is commonly used for SISR networks. It compares the ground truth image with model output HR image pixel by pixel, hence is a pixel-wise loss. However, the capability of MSE loss to record for the 'perceptually' relevant differences that humans can distinguish is low. To account for these, many novel perceptual losses have been proposed for the SISR problem, such as by Wu et al. [27]. This class of loss functions compare some high level features between the ground truth image with model output HR image.

For example, in the SRGAN[12] model, a multi task loss is used which consists of three parts namely an MSE loss for pixel-wise similarity, a perceptual similarity metric expressed in terms of distance, and an adversarial loss for the discriminator and generator. This model is then directly checked by the humans and they give a mean opinion score depending on the quality of the output image. EnhanceNet[18] is able to create good textures in the output super resolution image by using perceptual and texture matching losses besides the regular MSE loss. SRFeat[15] is another GAN model which focuses on the realistic perception of the input image, which is done by using an additional discriminator for assisting the generator to generate high-frequency structural features.

We chose to implement the Mean Gradient Error (MGrE) loss function use by Lu and Chen [13] for their modified U-Net SISR model. The MGrE function measures the difference between the edges

present in the two images. It does this by utilising a Sobel edge detection filter to get the gradients of the images, and then calculating the mean square error between the pixel-wise gradients. We use this in conjunction with the MSE loss. In the equation below, (i,j) is the pixel coordinate, G is the gradient for the HR ground truth image and $\hat{G}$ is the gradient for the HR output image, and m and n are the height and width of the images.

$$MGE = \frac{1}{n}\frac{1}{m}\sum_{i=1}^{n}\sum_{j=1}^{m}(G(i,j) - \hat{G}(i,j))^2$$

2.6 Peak Signal-to-Noise Ratio (PSNR) :

It is the ratio between the maximum power of a signal and the power of the corrupting noise that affects the fidelity of its representation. It is usually expressed in therms of the logarithmic decibel scale. The MAX here is the maximum possible pixel value of the image.

$$PSNR = 10LOG_{10}(\frac{MAX^2}{MSE})$$

But a major drawback of PSNR is that it is the estimate of an absolute error and doesn't exactly have a relation to image quality. A higher PSNR means that more noise has been removed from an image, and is thus more biased towards over smoothed or blurry results. An algorithm which removes noise at the cost of textures in the image will still have a high PSNR. A better evaluation metric is thus the Structural Similarity Index Measure or SSIM, which accounts for the structure of the image.

2.7 Structural Similarity Index Measure (SSIM) :

It utilises a perception-based model that incorporates things like contrast ratio and luminosity and other factors, and is more closely related to what humans would perceive in image quality of an image. It is calculated relative to the high-resolution ground truth image, so a value of one means that the output is identical to the ground truth. To achieve a high SSIM measure, an algorithm needs to denoise while preserving the edges and other textual structure like edges of objects in an image.

3 Methodology

Our aim to is conduct a study into the improvement of the SRCNN model trained with the MGrE loss, and demonstrate the new models' effectiveness in 2x SISR tasks. We further wish to learn and demonstrate the effectiveness of auto-encoder training for SISR.

3.1 Datasets and Pipeline:

In order to compare results with Dong et al. [3], we used the same dataset used by them for originally training their model. It consists of 91 images taken from the ILSVRC 2013 Imagenet detection training dataset[17], which is decomposed into 24,800 subimages obtained by striding the originals with a factor of 14. For testing, we used 100 images from the DIV2K[1] dataset. Further, in order to train on 800 high definition high-resolution images of the full DIV2K dataset, we used the auto-encoder architecture due to its ability to generate compressed representations of these images. These images were first resized to a resolution of 512x512 and then further downsampled by a factor of 2 to generate low-resolution images as inputs.

Our experimentation pipeline for both the models proceeded as follows:

1. Make the experimental change in the model, if any
2. Obtain LR input image from HR ground-truth by downsampling
3. Feed LR image through network to get HR output
4. Compare the HR output with HR ground-truth by calculating SSIM and PSNR
5. Do same for all images in the dataset, to find average SSIM and PSNR.

3.2 Implementation Notes:

For the basic SRCNN structure and training, we followed a tutorial[16]. We implemented our own Mean Gradient Error function, as seen in Appendix in fig 9. We used the author's original recommended kernel sizes of 9 -> 1 -> 5. This is the first model we trained. In the original paper, a simple MSE loss is minimised using Stochastic Gradient Descent with standard backpropogation, however used a modern Adam [10] algorithm for better optimisation, with a learning rate of $1e^{-4}$. Adam is used because it provides several improvements over the SGD, and is widely considered by the neural network academia to be a better optimizer than SGD. A batch size of 32 was used throughout experiments.

We also trained an SRCNN model with a mixed loss, using the MSE as the pixel loss, and the MGrE for the perceptual loss. The MGrE is weighted with a factor lambda, that we kept at 0.1 like the original author's suggestion[13].

$$TotalLoss = MSE + 0.1 * MGrE$$

For training the auto-encoder architecture we implemented the architecture proposed by Mao et al. [14]. The architecture was further fine tuned in order to extract appropriate high-resolution feature maps from input images. Our auto-encoder architecture consist of three symmetric convolution and deconvolution layers of kernel sizes 3, where downsampling operations are implemented after each layer in encoders while upsampling operation is done after each layer in decoders. Unlike the original proposed architecture, our model consists of an additional convolution and deconvolution layer in order to capture high level feature representations from images. Each layer is further followed by a rectified linear unit activation function along with L1 regularization with a regularization factor of 10e-10. This model uses the adam optimizer, same as our SRCNN model, while only using MSE in order to calculate the pixel loss. A batch size of 148 images was used where the input images were further divided into training and validation sets using splitting of 0.15.

4 Experimental Results And Discussion

4.1 Non-neural baseline results

For a baseline comparison, we used bilinear and bicubic upsampling. We used the OpenCV build-in functions, and tested on 100 pictures from the Div2k dataset. Two pictures are shown in appendix figure 14 with these classical types of upsampling.

4.2 Normal SRCNN Training and Testing

The first SRCNN model we trained was using the standard MSE loss. Training for 100 epochs on a single 2080Ti card on the server gave a final training loss of $5.3e^{-5}$, testing PSNR of 26.466, and test SSIM: 0.304. The graph during training of these vs epochs is shown in appendix figure 7. We will now try to improve upon this result by training with both MSE and MGrE loss.

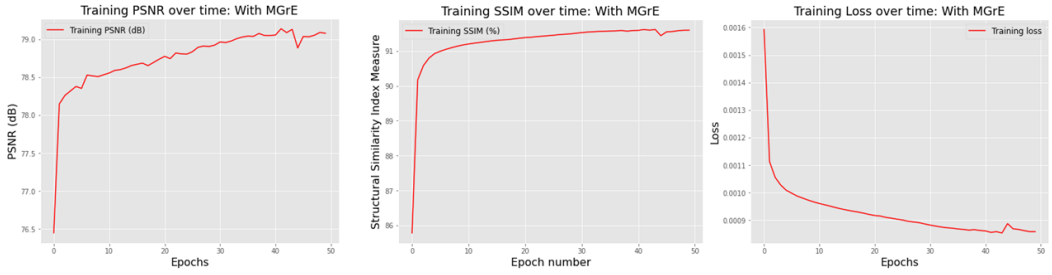
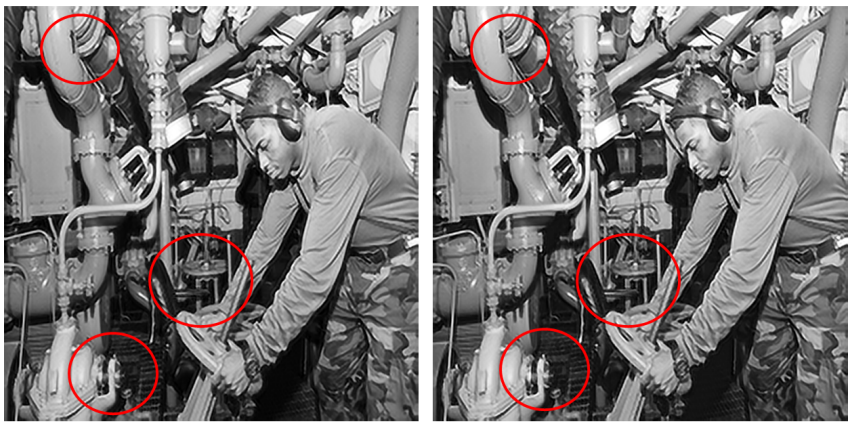


Figure 1: Training Results With MGrE

4.3 SRCNN with MGrE loss

The implemented mean gradient error function was used with MSE loss. The weight hyperparameter was kept at 0.1. The new training was visualised in the following plots1. This resulted in a testing

PSNR: 26.117 and test SSIM: 0.307. There's a clear improvement in the quality of images and SSIM values, as seen in figure. We can see that training progressed smoothly. We can see also the effect of the sobel operation in the encircled sections. The edges are more sharp in the left exhibit that is with the MGrE, and we can see the effect of super resolution on the contours. There is less contrast or dullness in the image without the MGrE loss.

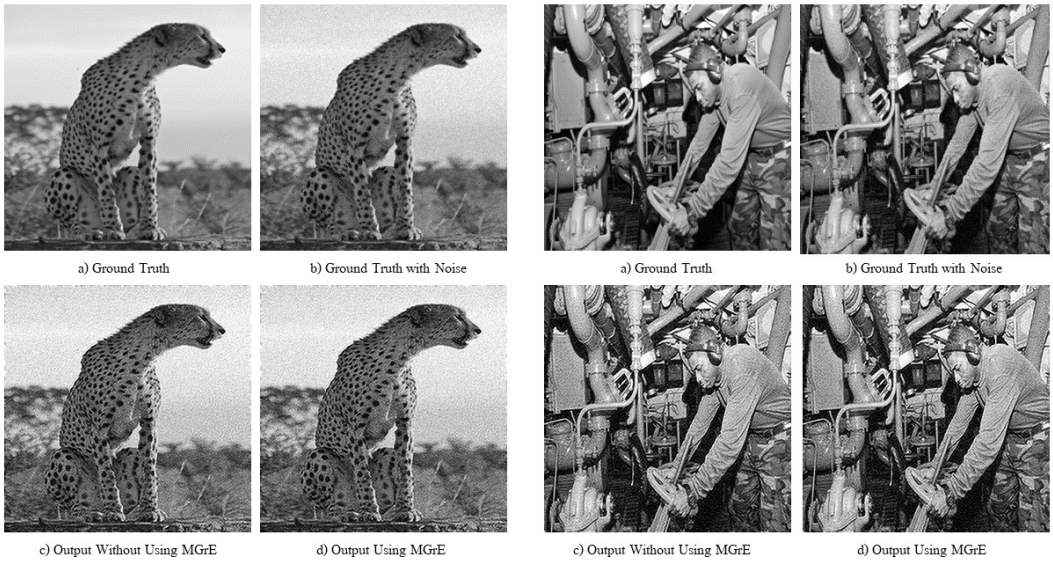


a) With MGrE b) Without MGrE

Figure 2: SRCNN Results

4.4 Testing with Noise

We tested both of our models with Gaussian noise on LR input images. Some results can be seen in figure 3. We find that the model with MGrE is able to provide more crispier images than the model without MGrE, even in the presence of noise. Larger versions of these images can be viewed in the appendix in figures 12 and 13.



a) Ground Truth b) Ground Truth with Noise a) Ground Truth b) Ground Truth with Noise

c) Output Without Using MGrE d) Output Using MGrE c) Output Without Using MGrE d) Output Using MGrE

Figure 3: Testing With Noise

4.5 Testing with Blur

We also tested our models with Gaussian blur on LR input images. Some results can be seen in figure 4. Extra blur can be thought of as an even lower resolution image. With this added, we find that the

model with MGrE gives more accurate images to the ground truth, than the model without MGrE. Further examples can bee seen in appendix in figures 11 and 10.

Figure 4: Testing With Blur

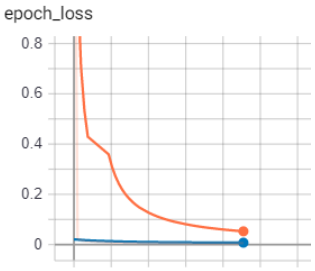


Figure 5: Training Loss - Autoencoders

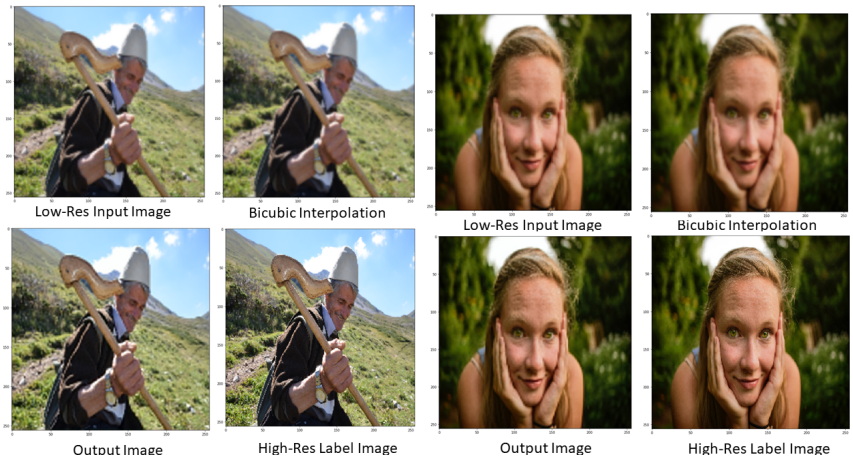


Figure 6: Results of auto-encoder

4.6 Auto-encoder Training and Testing

Several architectural experimentations were performed in order to determine the best performing model for training DIV2K images. The model that gave the best result consists of stacks of 3 convolution and deconvolution layers with kernel sizes of 3. The low-resolution input images were downsampled by 2D max pooling layers placed between encoder layers leading to a compressed encoded image at the end of the encoder. The encoded images were then passed through the decoder layers consisting of 2D upsampling layers between deconvolution layers in order to reconstruct high-resolution outputs. In order to gain the ability of making the model deeper without losing training accuracy due to issues like vanishing gradients, we used skip connections that connect the symmetric convolution-deconvolution layers. Since auto-encoders are prone to overfitting as the dept increases, we used dropout layers having the ability to drop 0.3 input units of convolution layers, thus helping us generalize the model. The model was trained for 100 epochs using 2080Ti GPU, where an average PSNR of 26.046406 was achieved accross the DIV2K testset. The final training loss was recorded at 0.0059 while the final validation loss was recorded as 0.0052. The graph of loss vs epochs is shown in figure 5.

The initial experimentation was implemented with stacks of 2 convolution and deconvolution layers with kernel sizes of 3. This model was able to achieve an average PSNR of 24.92268. Upon visualization of encoded images we were able to determine that high level features of the image not being extracted. We performed further modifications to the convolution layers adding a convolution and deconvolution layer to the framework while implementing kernel filter sizes of 9 -> 3 -> 5. This framework performed very well for some input images as we were able to record max PSNR of 33. On the other hand, some test images were performing very poorly and thus the min PSNR was observed at 16. As a result, the mean PSNR across the DIV2K testset was computed as 25.15. Several other experimentation were performed that proved ineffective, including data augmentation in order to provide model with inputs of varying brightness and zoom, use of smoothed L1 loss and use of Adadelta optimizer[29].

5 Conclusion and Future Work

The aim of this project was to learn and develop an understanding of advanced computer vision via programming in a deep learning framework like pytorch and keras as well as to further enhance our knowledge in the domain of super resolution. We were able to improve the standard SRCNN model by training with the MGrE loss, and demonstrated its effectiveness in 2x SISR tasks as well as in the presence of noise and blur.

Secondly, use of auto-encoders for the task of super resolution yielded accurate results. We were able to observe outputs with max PSNR of 33 during experimentation, which are on par with other state-of-art models. Due to constraint of time and computation ability, we were unable to make our model robust which leaves us with a huge scope for optimization of the same in the future. The higher PSNR values were observed when convolution and deconvolution layers were implemented using larger kernel sizes.

The main limitation of our work is that there are other contemporary methods which can achieve similar level of results as us. As we described in section:Section 2.6 most GANs and modern SISR architectures already utilise some kind of perceptual loss, as well as other modifications. Future work involves checking the performance of the model using different data augmentation techniques(like random cropping or flipping) on a much larger and more generalisable data set. For example, we could train with some noisy and blurry images to increase robustness of the network. Testing with different hyper parameters such as learning rate and the MGRE loss regularisation parameter. For example, as we know that perceptual loss is important we could try with higher weightage for the MGrE loss, such as 0.15, 0.2 and 0.25. We can implement the same loss function in FSRCNN and other networks and check the performance of the baselines. We would also like to experiment with 3x, 4x and other magnification factors. If time permitted, we would have tried residual learning or perhaps a U-Net style architecture. Another direction would be to use a relevant pre-trained backbone and build our own model from scratch. In future, we could also implement a perceptual loss for auto-encoders and observe the new results.

5.1 Acknowledgements

We would like to thank our lecturer Dylan Campbell for his support during our project. Without his mediation and help, we would not have been able to complete this report!

References

[1] E. Agustsson and R. Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[2] S. Anwar, S. Khan, and N. Barnes. A deep journey into super-resolution: A survey. *CoRR*, abs/1904.07523, 2019. URL `http://arxiv.org/abs/1904.07523`.

[3] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2015.

[4] C. Dong, C. C. Loy, and X. Tang. Accelerating the super-resolution convolutional neural network. In *European conference on computer vision*, pages 391–407. Springer, 2016.

[5] S. Fan, R. Wang, T.-T. Ng, C. Y.-C. Tan, J. S. Herberg, and B. L. Koenig. Human perception of visual realism for photo and computer-generated face images. *ACM Transactions on Applied Perception (TAP)*, 11(2):1–21, 2014.

[6] H. Greenspan. Super-resolution in medical imaging. *The computer journal*, 52(1):43–63, 2009.

[7] M. Haris, G. Shakhnarovich, and N. Ukita. Task-driven super resolution: Object detection in low-resolution images. *arXiv preprint arXiv:1803.11316*, 2018.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[9] G. Huang, Z. Liu, and K. Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016. URL `http://arxiv.org/abs/1608.06993`.

[10] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[11] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang. Fast and accurate image super-resolution with deep laplacian pyramid networks. *IEEE transactions on pattern analysis and machine intelligence*, 41(11):2599–2613, 2018.

[12] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.

[13] Z. Lu and Y. Chen. Single image super resolution based on a modified u-net with mixed gradient loss. *arXiv preprint arXiv:1911.09428*, 2019.

[14] X.-J. Mao, C. Shen, and Y.-B. Yang. Image restoration using convolutional auto-encoders with symmetric skip connections. *arXiv preprint arXiv:1606.08921*, 2016.

[15] S.-J. Park, H. Son, S. Cho, K.-S. Hong, and S. Lee. Srfeat: Single image super-resolution with feature discrimination. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 439–455, 2018.

[16] S. R. R. R. Rath. Image super-resolution using deep learning and pytorch, Aug 2020. URL `https://debuggercafe.com/image-super-resolution-using-deep-learning-and-pytorch/`.

[17] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[18] M. S. Sajjadi, B. Scholkopf, and M. Hirsch. Enhancenet: Single image super-resolution through automated texture synthesis. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4491–4500, 2017.

[19] J. Shermeyer and A. Van Etten. The effects of super-resolution on object detection performance in satellite imagery. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 0–0, 2019.

[20] W. Shi, J. Caballero, F. Huszár, 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. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1874–1883, 2016.

[21] A. Shocher, N. Cohen, and M. Irani. "zero-shot" super-resolution using deep internal learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3118–3126, 2018.

[22] I. Sobel and G. Feldman. A 3x3 isotropic gradient operator for image processing. *a talk at the Stanford Artificial Project in*, pages 271–272, 1968.

[23] T. Tong, G. Li, X. Liu, and Q. Gao. Image super-resolution using dense skip connections. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4799–4807, 2017.

[24] 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)*, pages 437–440. IEEE, 2016.

[25] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018.

[26] Z. Wang, J. Chen, and S. C. Hoi. Deep learning for image super-resolution: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[27] Q. Wu, C. Fan, Y. Li, Y. Li, and J. Hu. A novel perceptual loss function for single image super-resolution. *MULTIMEDIA TOOLS AND APPLICATIONS*, 2020.

[28] C.-Y. Yang, C. Ma, and M.-H. Yang. Single-image super-resolution: A benchmark. pages 372–386, 09 2014. ISBN 978-3-319-10592-5. doi: 10.1007/978-3-319-10593-2_25.

[29] M. D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012. URL `http://arxiv.org/abs/1212.5701`.

[30] K. Zeng, J. Yu, R. Wang, C. Li, and D. Tao. Coupled deep autoencoder for single image super-resolution. *IEEE transactions on cybernetics*, 47(1):27–37, 2015.

[31] W. Zuo, K. Zhang, and L. Zhang. *Convolutional Neural Networks for Image Denoising and Restoration*, pages 93–123. Springer International Publishing, Cham, 2018. ISBN 978-3-319-96029-6. doi: 10.1007/978-3-319-96029-6_4. URL `https://doi.org/10.1007/978-3-319-96029-6_4`.

6 Appendix:

Figure 7: Training Results Without MGrE

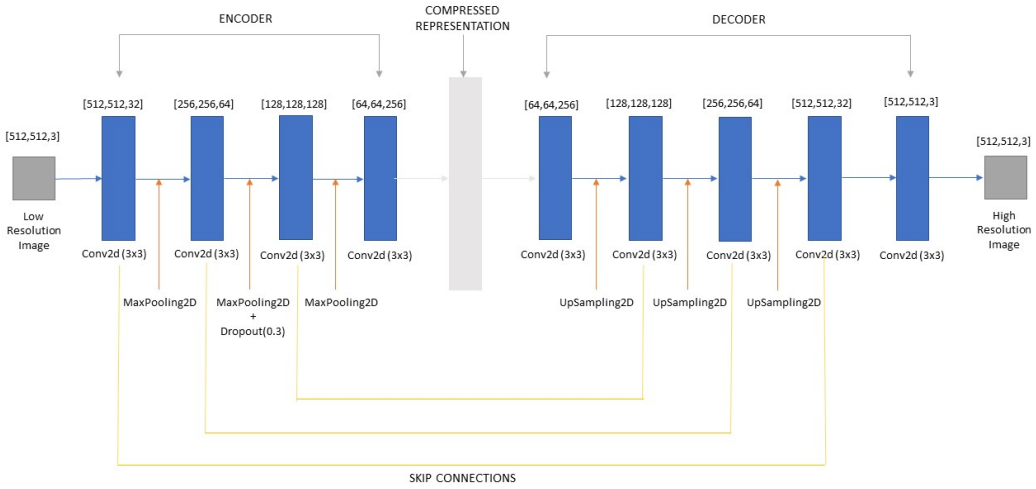


Figure 8: Auto-encoder architecture

```python
def MGE(outputs, targets, lambda_weightage = 0.1):

    outputs = outputs.cpu().detach().numpy()
    targets= targets.cpu().detach().numpy()
     # Creating a 1-d Gaussian Kernel of
    gauss_kernel_1d = cv2.getGaussianKernel(3, 1.5)
    #Taking the transpose of the 1-d gaussian kernel
    gauss_kernel_1d_t = np.transpose(gauss_kernel_1d)
    #Converting the Gaussian kernel to 2-d
    gauss_kernel_2d = np.dot(gauss_kernel_1d,gauss_kernel_1d_t)
    #Sobel filter for x and y
    sobel_y = np.array([[1, 0, -1], [2, 0, -2], [1, 0, -1]])
    sobel_x = np.transpose(sobel_y)
    # Performing convolution of sobel kernels by gaussian kernel
    d_x = signal.convolve2d(gauss_kernel_2d, sobel_x, mode='same', boundary='symm')
    d_y = signal.convolve2d(gauss_kernel_2d, sobel_y, mode='same', boundary='symm')

    # applying vertical and horizontal convolution to the image I
    mge=0
    for i in range(outputs.shape[0]):
        dx = signal.convolve2d(outputs[i].reshape(outputs.shape[2],outputs.shape[3]), d_x, mode='same', boundary='symm')
        dy = signal.convolve2d(outputs[i].reshape(outputs.shape[2],outputs.shape[3]), d_y, mode='same', boundary='symm')
    # finding the gradient magnitude
        mag = np.sqrt(np.square(dx) + np.square(dy))

        dx2 = signal.convolve2d(targets[i].reshape(targets.shape[2],targets.shape[3]), d_x, mode='same', boundary='symm')
        dy2 = signal.convolve2d(targets[i].reshape(targets.shape[2],targets.shape[3]), d_y, mode='same', boundary='symm')
        mag2 = np.sqrt(np.square(dx2) + np.square(dy2))

    # compute mean gradient error
        mge+= (np.sum(np.square(mag-mag2)))/outputs.shape[2]*outputs.shape[3]
    mge=mge/outputs.shape[0]
    return mge * lambda_weightage
```

Figure 9: MGrE Code

Figure 10: Blur image output with MGrE

Figure 11: Blur image output without MGrE

Figure 12: Noisy image output with MGrE

Figure 13: Noisy image output without MGrE

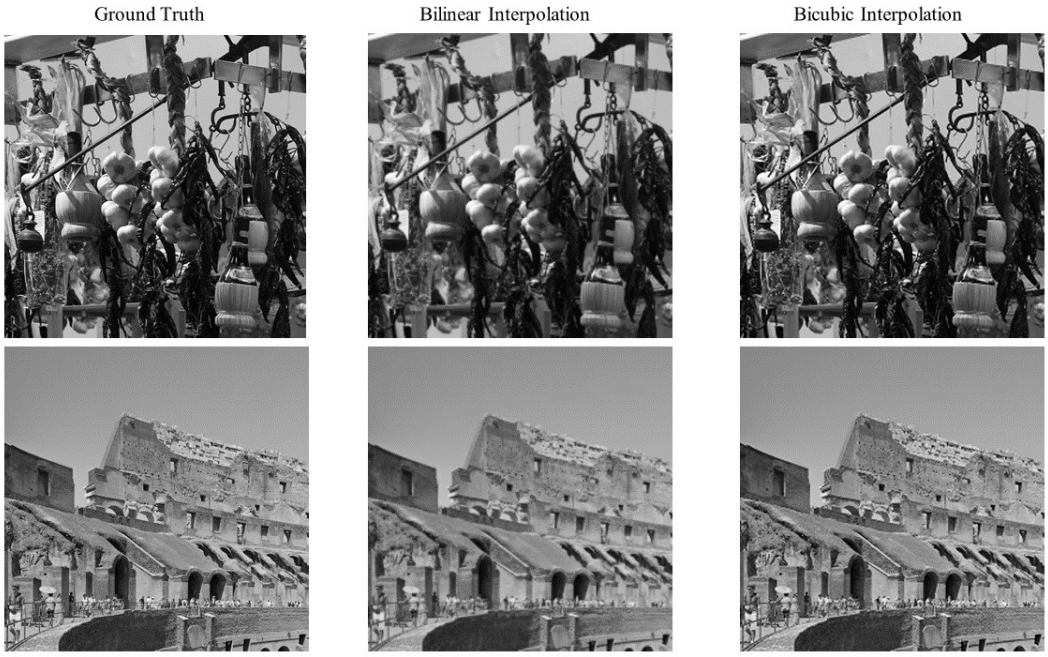


Figure 14: Baseline- Upsampling with Linear and Bicubic Interpolation