Improving Image Resolution Using SR-Convolutional Neural Networks

Hetavi Saraiya
SJSU Id: 014508988
hetavi.saraiya@sjsu.edu

Tajbir Sandhu
SJSU Id: 015274974
tajbir.sandhu@sjsu.edu

Sai Ashrith Aduwala
SJSU Id:014427725
saiashrith.aduwala@sjsu.edu

Srikar Narapureddy
SJSU Id:014600742
srikarreddy.narapureddy@sjsu.edu

***Abstract*—Image processing has emerged as a prevalent application of deep learning over the years. Image Processing can be defined as a method to manipulate and change a digital image's pixels, usually to improve its quality and resolution. There are many deep learning architectures that can be used for image processing, and for our project, we put forth a deep learning convolutional neural network model to improve the resolution of an image. This model can convert low resolution images into higher resolution equivalents. For some applications, a higher degree of quality permits specialists to gain more information from an image. An example of this is brain scans. A higher resolution scan can provide a doctor with more details about a patient's condition. Technologies like super-resolution[1] algorithms, multilayer perceptrons, or simple convolutional neural network models have been used to tackle this problem. In our project we use an application specific neural network, Super-Resolution Convolutional Neural Network[2], and OpenCV, to process the images and upgrade them to a higher resolution. The results will be validated using the image-quality metrics: mean squared error, peak signal-to-noise ratio, structure similarity measure, and universal quality index.**

***Index Terms*—Deep Learning, Image Processing, Super Resolution, Convolutional Neural Network.**

I. Introduction

Deep learning, having developed innovative solutions for numerous problems, has become one of the biggest advancements in the field of artificial intelligence. Amongst deep learning's applications, image processing has become popular. According to Wikipedia[3], Image Processing can be defined as "the use of a digital computer to process digital images through an algorithm." It is a subdomain of digital signal processing. There are many uses for image processing, such as feature extraction, censoring, remote sensing, face recognition, etc., but it is primarily used for image Super Resolution (SR), the process of improving the image resolution and clarity. Super Resolution[4] is defined as "the process of reconstructing the high resolution image from its equivalent low resolution image". The low resolution of an image can be caused by low pixel density, small spatial resolution, degradation, masking etc. Thus, to reconstruct the high resolution image, it must be done on the pixel resolution level. This can be difficult for some of the traditional or non-learning methods as it deals with a large number of pixel values for a single image, and the computational complexity is very high. Hence, deep learning methods are used to solve this problem. In our project we develop a deep learning convolutional neural network called Super Resolution Convolutional Neural Network (SRCNN)[5], which is a domain specific network architecture designed specifically for super resolution problems. The original high quality image is first degraded to a specified level by resizing the pixel and increasing its spatial resolution. This degraded image is sent as an input to the model which will then output the high quality image. We also train another network called Efficient Sub Pixel Convolutional Neural Network(ESPCNN)[6] as a baseline to compare SRCNN's results. The performance of the model is tested using the image quality metrics Mean Squared Error(MSE), Peak Signal-to-Noise Ratio(PSNR), Structural Similarity Index(SSIM) and Universal Quality Index(UQI). An analysis was also performed where comparison was conducted using the quality metrics on different weights that were trained using various parameters at different degradation levels. Overall, this project tries to build a model that performs well at the super resolution problem and provides a brief overview of how the performance varies with respect to the model training.

II. Related Works

Improving the image resolution is a broad domain with a diverse set of solutions, each having its own advantages and disadvantages. In this section we discuss a few of these solutions and different types of evaluation metrics that can be used to quantify their performance.

In their paper, Y. K. Badran et al.[7] have proposed a method to produce a high resolution image from a single low resolution image using the learning features to constrain the back projection. The features are learned using a dictionary of low resolution/high resolution patches. These learned features are used as constraints for the reconstruction of the high resolution image. This also satisfies the super resolution assumption, which states that the generated high resolution image, when blurred and down sampled, must be similar to the used low resolution image. The proposed method is divided into two phases and each phase is further divided into sub-processes. The first phase is the learning phase, which is responsible for learning the mapping of low resolution patches to the high resolution features. This phase is divided into two sub-processes: a classification process that clusters the low resolution patches with similar characteristics and a regression process that determines the optimal function to map low resolution patches to high resolution. The second phase

is the super resolution restoration phase, which predicts and constructs the high resolution image. This, once again, is divided into two stages: reconstruction of initial high resolution patch and constraining back projection. The authors were able to achieve an average PSNR of 30.94, SSIM of 0.93 and MSE of 1.64*10-4.

In their paper, A. Özçelikkale[8] et al. analyse how limited amplitude resolution can affect the super resolution problem. They consider the case where a high resolution image is reconstructed from a set of images with poor spatial resolution. The authors consider both spatial resolution and amplitude resolution as parameters for their architecture. The main goal of this paper is to construct a high resolution image from multiple quantized low resolution images and to analyse the trade-off between space resolution and amplitude resolution. For the analysis, the authors define their own measurement model, which is a ratio between the representation cost of a number of low resolution images to the representation cost of the target high resolution image. For reconstruction of images, the authors use the norm approximation method and evaluate using SSIM and PSNR. They plot the variations of these evaluation metrics with the change in the representation cost and conclude that it is possible to reach the benchmark performance without a need to know the high resolution image in advance. The authors were able to achieve an average SSIM of 0.89 and an average PSNR of 30.19.

In their paper, T.H. Wei and J.C. Chen propose a traditional method for super resolution using Interpolation. High frequency information is ignored by interpolation. As a result, example-based super-resolution methods can work on any frequency information. The computational complexity of super-resolution is high and takes time to modify the images to high resolution. It varies depending on the size and type of image or video. In this paper, they have proposed a deep learning model for faster super resolution for video. After modifying each frame and passing it through the neural net, they rebuild the video[9].

In their paper, A. Bhowmik et al. propose an approach to solve single-image super-resolution that works on the concept of images in training sets being Non-linear mappings of low-resolution and high resolution. High accuracy is obtained from training-based approaches, but if the images are from different classes or if the HR→LR generative model deviates, it is retrained. They have devised a solution which works with the training dataset and creates a dynamic convolution network to understand the connection between consecutive scales of Gaussian and Laplacian pyramids. These relationships are used to predict the details at a finer scale, resulting in improved Super resolution. This method works on any image of any class and in resource-constrained settings in which this feature is useful. Gains in PSNR are obtained without compromising on the structural fidelity of the image [10].

In their paper, L. Zhao et al. present a fast single SR approach which transfers a LR patch to a HR feature. Many researchers are working on getting HR images from LR images by performing single image SR restoration. These features are used to reconstruct the HR image through a process called constrained back-projection [11] Results using this approach are promising while using the proposed single image SR. It is easy to implement and modify for future enhancement by selecting the optimum features and adding more constraints. Selecting the optimum classifiers that match the features are important turning points.

III. Technical Approach

A. Convolutional Neural Network

A fully connected neural network is not suitable for images since they consider the image as a large, single dimensional vector which leads to increased complexity. A Convolutional Neural Network (CNN) is a special kind of neural network which considers the spatial and temporal dependencies of an image by extracting features using different filters applied repeatedly over the entire image. This process is formally known as convolution and similar to a neural network, These filters are optimized during training to enable the use of features for classification or, in the case of SRCNN, to improve image resolution.

B. SRCNN

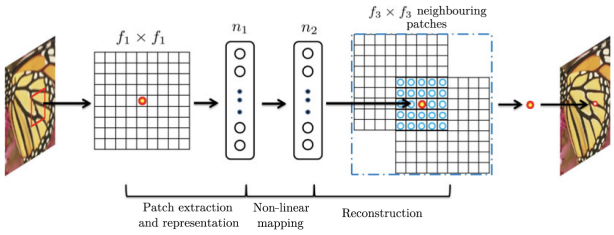

Fig. 1. SRCNN Architecture

The goal is to extract an image F(Y) which represents the high resolution image using the optimized filters learned during the training. The image is initially transformed from RGB to YCR-CB channels and then upscaled to the desired resolution using bicubic interpolation. The Y channel is then passed to CNN. The network can be mainly divided into three stages[1]:

- Patch Extraction
- Non-Linear Mapping
- Reconstruction

Patch extraction and representation: This layer is responsible for the extraction of different features from the input image. This can be achieved using pre-trained bases such as PCA, Haar, etc. In this network, the dense sets of feature maps are extracted from the image using n1 filters optimized during training and the filter outputs are passed through the ReLU activation function. This operation can be expressed as the following equation:

$$F_1(Y) = max(0, W_1 * Y + B_1) \quad (1)$$

where W_1 represents n_1 filters of size $(c * f_1 * f_1)$ and c is the number of channels in the input image.

Non-linear mapping: This layer is responsible for adding non-linearity to the convolutional neural network. The n1 feature maps of the images from the previous layer are transformed into n2 feature maps using n2 filters. These nonlinear mappings represent high resolution patches used in the final layer for reconstruction. Adding more layers in this stage will improve non-linearity, which can improve performance on more complex images at the cost of increased training time. Similar to the previous layer, this operation can be expressed by:

$$F_2(Y) = max(0, W_2 * F_1 * Y + B_2) \tag{2}$$

where W_2 represents n_2 filters of size $(n_1 * f_2 * f_2)$.

Reconstruction: The final layer reconstructs the high resolution image from the n2 high resolution feature maps from the previous layer. This can be done by averaging using a single filter of size n2xf3xf3. This filter projects the set of high resolution patches onto the image domain and performs an averaging function for overlapping patches using the equation:

$$F_3(Y) = max(0, W_3 * F_2 * Y + B_3) \tag{3}$$

where W_3 represents a single filter of size $(n_2 * f_3 * f_3)$.

C. ESPCNN

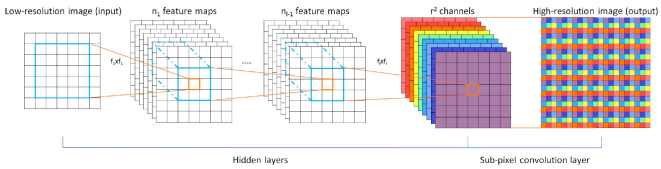

Fig. 2. ESPCNN Architecture

The computational complexity of a CNN is highly dependent on the resolution of an image. SRCNN performs an upscaling of the image before passing it to the network, and this has an effect on performance. ESPCNN proposes a more efficient use of the CNN architecture where the image is upscaled in the final layer.

This network skips the bicubic interpolation and performs feature extraction as well as non-linear mapping directly on the low resolution image. This means that this network must learn the processing necessary for upscaling. Finally, reconstruction is performed at the final layer.

Efficient sub-pixel convolution layer: The efficient sub-pixel convolution layer performs the mapping from Low Resolution (LR) space to the high resolution image. This operation is equal to convolving with filter W with a fractional stride $\frac{1}{r}$ on LR space, which is done by pre-calculated bases in interpolation techniques. It means that different parts of W are periodically activated during the convolution with pixels in the LR image padded in between and can also be viewed as deconvolution. This operation is done by doing general convolutions and finally performing a simple reshuffle from $(h * w * r^2 * C)$ tensor to $((h * r) * (w * r) * C)$ tensor as seen in the last layer from Fig 2. Thus, the interpolation method is automatically learned by the network during training.

IV. Methodology

A. Quality Metrics

In order to evaluate the high resolution images, we selected several error measures which are common in the field of image super resolution. The first and most simple error measure selected was standard Mean Squared Error (MSE) which is determined by finding the total squared error of all images and taking the average. The next measure we chose to use was Peak Signal-to-Noise Ratio (PSNR) where the peak signal, which is 255 for a RGB pixel, is compared to the MSE of the pixel. However, while these error measures are effective at measuring per-pixel error, they fail to quantify human perception of an image's quality [13]. To remedy this, we also selected a number of error measures which account for the overall structure of an image as well. The first of these error measures is PSNR with Block Factor (PSNR-B), which slightly alters the existing PSNR algorithm by factoring in the blockiness of an image relative to the target image [13]. Structural Similarity (SSIM) was also selected as, rather than just comparing pixel values, it also compares the luminance, contrast, and structure of the image. A variant of SSIM known as Multiscale SSIM (MSSSIM) was also chosen to grade images. It functions similar to SSIM, but also scales the target and reference image up and down multiple times and compares the scaled images [14]. Finally, Universal Quality Index (UQI) was used. UQI factors in loss of correlation, luminance, and contrast when grading an image [15].

B. Datasets

In order to produce the weights and conduct testing, several different datasets were used: train91 [1], BSD [16], and Set14 [1]. train91 and BSD were both used for training while Set14 was used for testing. train91 was also used for two separate models, one model which was trained with train91 having a degradation factor of 2 applied and another with train91 having a degradation factor of 10 applied. The degradation factor simulates the blockiness caused by image compression. For testing, 8 different versions of Set14 were also used with increasing degradation factors from 2 to 9.

C. Neural Network Parameters

The neural network used Adam optimizer with a learning rate of 0.001. MSE was picked as the loss function with ReLU as the activation function.

D. Test Process

SRCNN was tested using the various versions of Set14 with ESPCNN acting as a baseline for SRCNN's upscaling performance. To test the neural networks, the images from the set are fed into the models resulting in upscaled images. The upscaled images are then compared to their reference images using the previously mentioned error measures. This process was repeated with multiple sets of weights.

V. Analysis

A comparative analysis was performed where the different evaluation metrics were compared for the various degradation levels of the image. This comparison was done across multiple network weights that were trained on different parameters and datasets.

The **online weights** were provided by the author of one of the reference papers. This was mainly used to perform mode testing rather than to finalize the model.

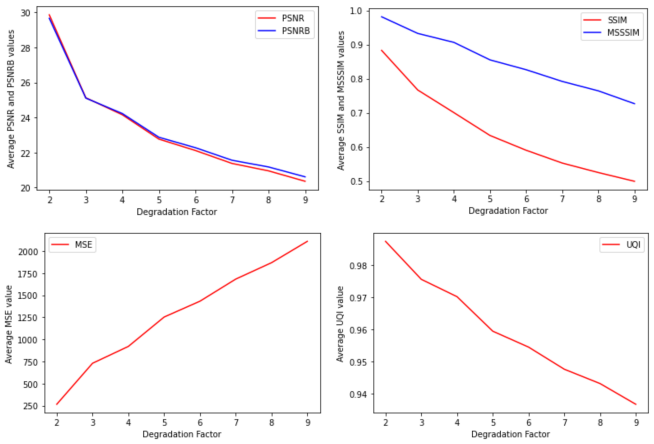

Fig. 3. Average Quality Metrics for Online Weights

As the graphs in Fig.3. show, the average PSNR and PSNRB values tend to decrease with the degradation level. This trend can be seen for both SSIM and UQI. This is due to the model being trained on the images that have a degradation level of 2. Thus, the average metric values for the images with degradation factor 2 is higher than the rest. The error measure of MSE is inverse to the other similarity metrics, meaning it has a lower value for the image with degradation factor of 2 and increases hence forth.

For the project, we have trained the model four times with different parameters and two datasets, and saved the weights for the analysis.

The weights **trained_91.h5** were the result of training the model on the train91 dataset explained above. The images for this training were degraded to level 2 and the model was trained for 300 epochs with 128 batch size.

The graphs in Fig.4. show the trends of the various evaluation metric values for our first trained weights. We can see a trend that is similar to the previous weights, with high metric values for lower degradation levels and lower metric values for higher degradation levels.

The weights **trained_bsd.h5** were the result of training the model on the BSD200 dataset explained above. The images for this training were degraded to level 2 and the model was trained for 200 epochs with 128 batch size.

The graphs in Fig.5. show the trend of the evaluation metric for the weights trained as explained above. These graphs also follow a similar trend as the previous weights, except for a sharp spike in the graph at a degradation level of 4. This

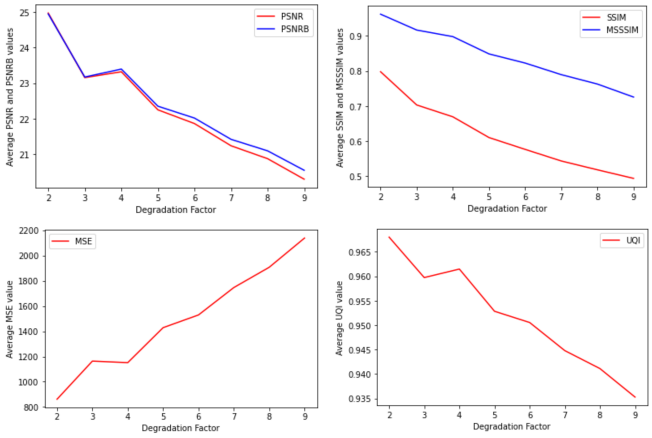

Fig. 4. Average Quality Metrics for trained_91.h5

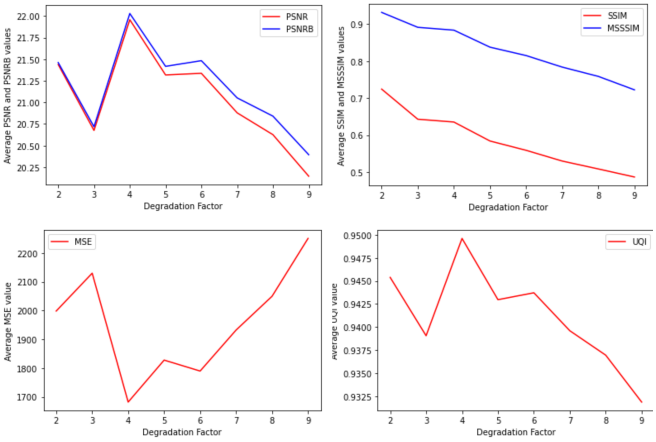

Fig. 5. Average Quality Metrics for trained_bsd.h5

behavior is likely due to the quality of the dataset as this behavior cannot be observed with the other datasets.

The weights **trained_deg2.h5** were the result of training the model on the train91 dataset explained above. The images for this training were degraded to level 2 and the model was trained for 200 epochs with 128 batch size.

The graphs in Fig.6. show the trend of the evaluation metrics with the degradation levels for the new set of trained weights as explained above. These weights also show us a similar trend with the degradation levels.

The weights **trained_deg10.h5** were the result of training the model on the train91 dataset explained above. The images for this training were degraded to level 10 and the model was trained for 50 epochs with 1 batch size.

These graphs in Fig.7. show the trend of the evaluation metrics with the degradation factor for the last set of trained weights. These graphs also show the similar decreasing trend, but we can observe that the decrease is more gradual and smooth. The reason for this smooth decrease can be the lower pixel density of the training images. These weights were trained on higher degradation factor training images, which

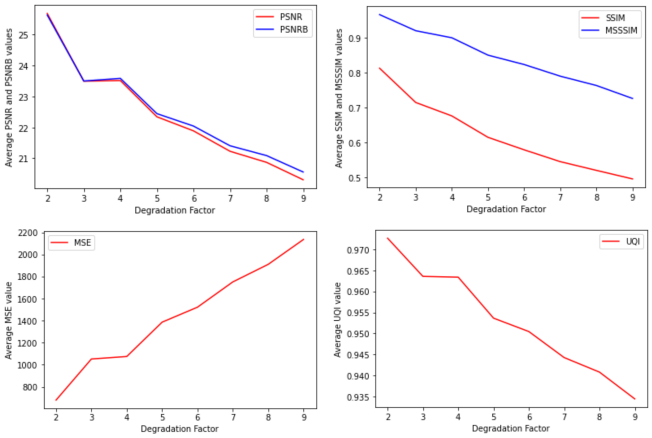

Fig. 6. Average Quality Metrics for trained_deg2.h5

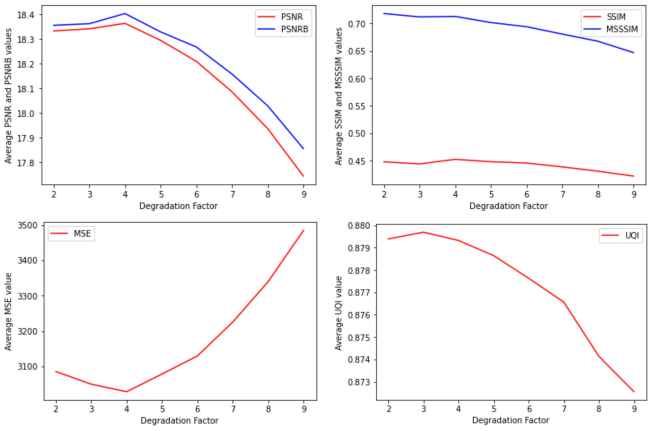

Fig. 7. Average Quality Metrics for trained_deg10.h5

might have caused the model to overfit the lower density of pixels. It can also be noted that the decrease in the metric values is low. The maximum difference we can observe is around 0.7 for PSNR and similar low differences for the other metrics. This again can be the result of the training images having higher degradation levels.

PSNR and PSNR-B also gave surprising results. Since PSNR-B should give lower scores to blockier images than PSNR, it seemed likely that the PSNR-B would be lower due to the degradation added to the input images. However, our experimentation shows that opposite. This might be due to the interpolation that SRCNN uses. Larger blocks are graded worse by PSNR-B, and the interpolation methods might be breaking up these blocks and smoothing them. Also, MSSSIM and SSIM appear to have less steep slopes than the other error measures. Since these error measures attempt to quantify human perception of an image, this may suggest that a human might not grade SRCNN's performance on highly degraded images as harshly as most of our error metrics.

The common trend we observe for all the weights is the decrease of similarity metric values and increase in the mean square error over the different degradation levels. The images with a degradation factor of 2 have a relatively lower pixel density than the original high quality images but still have a better pixel density than images with a degradation factor of 10. The model that is trained on these images can easily learn to reconstruct the image from the relatively low pixel density image. However, it cannot perform on the same level when dealing with images that have 10 times lower pixel density compared to the original image. Thus, the model performs better for images at lower degradation levels than at higher levels, resulting in the decreasing trend in the graphs. Since MSE is an error measure rather than a similarity metric, it has an increasing trend across all the graphs.

VI. Tools Used

A. *Keras[17]*

Keras is an open source library for python that provides an interface for neural networks. It has the implementations of many commonly used neural network components like the layers, optimizers, activation functions etc. In our project we use keras to build the neural network framework. We make use of three of its sub components: keras.Sequential, which is used to stack the layers of the model linearly, keras.Conv2D, which is used to create a convolution, and keras.optimizers, which allows us to select an optimization function for the model.

B. *cv2[18]*

The cv2, or opencv, is a widely used computer vision library for python. It is used to integrate images into the python code. For our project this library was used to read the degraded image for the network and save the reconstructed high quality image.

C. *Numpy[19]*

The numpy is a mathematical library for python that helps perform mathematical operations on multidimensional arrays and matrices and provides other high-level mathematical functions. For our project, numpy was used in the evaluation metric calculations. It was used for the implementation of mse() and psnr() functions.

D. *Sewar[20]*

Sewar is a python library used for image quality assessment by calculating the image evaluation metrics. It has implementations for many such metrics such as Mean Square Error, Peak Signal-to-Noise Ratio, Structural Similarity measure, Spatial Correlation Coefficient, Spectral Angle Mapper, Spatial Distortion Index, etc. For our project, we used this library to calculate some of the complex evaluation metrics of PSNR-B, MSSSIM and UQI.

E. *Matplotlib[21]*

The matplotlib is a popular graphing library for python. It has the capabilities to plot static functions, animations, and even interactive visualizations. It also has 3D plot capabilities. For our project, we used one of its sub-components called

pyplot. Pyplot is used for plotting statistical graphs that resembles the plots from MATLAB. We used this library to plot the graphs between the different evaluation metrics and the degradation levels for the analysis.

VII. Conclusion

Throughout this paper, we have discussed several interesting methods of improving image quality using neural networks, particularly SRCNN and the newer ESPCNN as a baseline for our analysis. We have explained the architecture and working of the SRCNN used in the project followed by a brief explanation on the various evaluation metrics, datasets and the testing process. An analysis was also performed that compared the trends of the average values of the evaluation metrics, with the different degradation levels. SRCNN and similar Super Resolution algorithms are effective at recovering images with relatively low degrees of degradation. However, as the quality of the input image continues to decrease, the effectiveness of Super Resolution also starts to deteriorate. Regardless, throughout the project we were able to achieve satisfactory results with input which has lower levels of degradation, and since improvements can be observed from the move to the newer ESPCNN from SRCNN, we can conclude that better results can be found by continuing research and development of new algorithms. The code and the output images for the project can be found in the GitHub link[22]

VIII. Contributions

All group members contributed equally to this project. Srikar and Hetavi focused on the implementation of the neural network whereas Tajbir and Sai focused on the testing and analysis. Research was equally divided amongst all participants.

References

[1] L. Yue, H. Shen,J. Li, Q. Yuan, H. Zhang, L. Zhang, (2016). "Image super-resolution: The techniques, applications, and future" *Signal Processing*, 128, 389-408.

[2] N. M. Elsaid,Y. C. Wu (2019, July). "Super-Resolution Diffusion Tensor Imaging using SRCNN: A Feasibility Study." In 2019 *41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)* (pp. 2830-2834). IEEE.

[3] "Digital image processing," *Wikipedia*, 29-Nov-2020. [Online]. Available: https://en.wikipedia.org/wiki/Digital_image_processing. [Accessed: 15-Dec-2020].

[4] B. Raj, "An Introduction to Super Resolution using Deep Learning", 2019. [Online]. Available: https://medium.com/beyondminds/an-introduction-to-super-resolution-using-deep-learning. [Accessed: 12-Dec- 2020].

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

[6] W. Shi et al., "Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network," 2016 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, 2016, pp. 1874-1883, doi: 10.1109/CVPR.2016.207.

[7] Y. K. Badran, G. I. Salama, T. A. Mahmoud, A. Mousa and A. E. Moussa, "Single image super resolution based on learning features to constrain back projection," 2019 *International Conference on Innovative Trends in Computer Engineering (ITCE)*, Aswan, Egypt, 2019, pp. 23-28, doi: 10.1109/ITCE.2019.8646324.

[8] A. Özçelikkale, G. B. Akar and H. M. Ozaktas, "Super-resolution using multiple quantized images," 2010 *IEEE International Conference on Image Processing*, Hong Kong, 2010, pp. 2029-2032, doi: 10.1109/ICIP.2010.5651039.

[9] T.H. Wei and J.C. Chen, "Video Super-Resolution via Convolution Neural Network," 2016 *3rd International Conference on Green Technology and Sustainable Development (GTSD)*, Nov. 2016.

[10] A. Bhowmik, S. Shit, and C. S. Seelamantula, "Training-Free, Single-Image Super-Resolution Using a Dynamic Convolutional Network," *IEEE Signal Processing Letters*, vol. 25, no. 1, pp. 85–89, Jan. 2018.

[11] L. Zhao, Q. Sun, and Z. Zhang, "Single Image Super-Resolution Based on Deep Learning Features and Dictionary Model," *IEEE Access*, vol. 5, pp. 17126–17135, 2017.

[12] W. Shi et al., "Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network," 2016 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, 2016, pp. 1874-1883, doi: 10.1109/CVPR.2016.207.

[13] C. Yim and A. C. Bovik, "Quality Assessment of Deblocked Images," in *IEEE Transactions on Image Processing*, vol. 20, no. 1, pp. 88-98, Jan. 2011, doi: 10.1109/TIP.2010.2061859.

[14] C. Charrier, K. Knoblauch, L. T. Maloney, A. C. Bovik and A. K. Moorthy, "Optimizing Multiscale SSIM for Compression via MLDS," in *IEEE Transactions on Image Processing*, vol. 21, no. 12, pp. 4682-4694, Dec. 2012, doi: 10.1109/TIP.2012.2210723.

[15] Zhou Wang and A. C. Bovik, "A universal image quality index," in *IEEE Signal Processing Letters*, vol. 9, no. 3, pp. 81-84, March 2002, doi: 10.1109/97.995823.

[16] P. Arbelaez, C. Fowlkes, and D. Martin, The Berkeley Segmentation Dataset and Benchmark, Available: https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/

[17] Keras. [Online]. Available: https://keras.io/. [Accessed: 15-Dec-2020].

[18] OpenCV, 13-Oct-2020. [Online]. Available: https://opencv.org/. [Accessed: 15-Dec-2020].

[19] NumPy. [Online]. Available: https://numpy.org/. [Accessed: 15-Dec-2020].

[20] "sewar," PyPI. [Online]. Available: https://pypi.org/project/sewar/. [Accessed: 15-Dec-2020].

[21] "Visualization with Python," Matplotlib. [Online]. Available: https://matplotlib.org/3.3.3/index.html. [Accessed: 15-Dec-2020].

[22] https://github.com/SAI-ASHRITH/MachineLearning-Project