Enhancing SRCNN with Perceptual Loss

Sarvesh Prajapati*
College of Engineering
Northeastern University
Boston, MA
prajapati.s@northeastern.edu

Abhinav Kumar*
College of Engineering
Northeastern University
Boston, MA
kumar.abhina@northeastern.edu

Rupesh Pathak*
College of Engineering
Northeastern University
Boston, MA
pathal.r@northeastern.edu

***Abstract*—Single Image SR has been one of the most interesting problem in field of Computer Vision (CV) for a while now. From classical CV to Convolution Neural Networks (CNN), various solutions have been proposed to solve this problem producing some great results. In this paper we try to step back from deep networks and see how changing loss of simple CNNs like Super Resoultion Convolution Neural Network (SRCNN) affects the output. Perceptual loss focuses on improving the visual quality rather than getting the true form of the image. We start by implementing SRCNN with MSE loss, perfrom 2x and 4x SR, and move on further by swapping the loss function and we discuss the results in the conclusion and result section.**

Fig. 1: Using SRCNN for x4 scaling on a cat image, vanishing features can be seen, especially near the moustache of the cat.

I. Introduction

Super Resolution refers to the taking a Low Resolution (LR) image and increasing it's resolution while keeping the features intact, 1 shows example of super-resolution being performed on that of a cat image. After introduction of CNNs [10] and achieving some incredible performance with DeepNets [8], [13] Deep Learning has shown some impressive results for generating Super Resolution images from Low Resolution Images. Classical methods like Nearest Neighbours, Bilinear and Bicubic interpolation performs nicely, but they can be difficult to work with when working with precision critical applications like medical image processing, cybersecurity, etc. Some methods [4] proposed using multiple images and using them for SR, continued research in this domain did produce some pretty good results, but with recent advancement in computation power, introduction of CNNs, the most interesting question was can we do better? Earlier works [5] showed some pretty good results without using CNNs, in this paper, we move study how good were some of the early CNNs models for Image Super Resolution and can we do better without deep networks? A Super Resolution algorithm basically takes in a Low Resolution Image I_{LR}, and produces a Super Resolution image I_{SR}, where we try to minimize the difference between I_{SR} and I_{HR}, where I_{HR} is High Resolution original image. Generally in real world, inorder to test these algorithms, we take a I_{HR}, and we convert it to I_{LR} with bilinear/bicubic interpolation, and test algorithms on them, and try to minimize the loss between I_{SR} and I_{HR}. In this paper we use Div2K dataset [1] [15], and try to enhance them by a factor of 4.

* Authors have equal contribution.

II. Previous Work

Some incredible work has been achieved with Convolution Neural Networks, with introduction of LeNet [10] by LeCun et al, it was fascinating to see how precisely we can solve classification/regression problems in Computer Vision, later on with advancements ImageNet [12], AlexNet [9] motivated us to explore simpler Super Resolution using Deep Learning. Here we explore Super Resolution using Convolution Neural Network (SRCNN) [2] briefly, see how changing the loss function changed the model performance which was motivated by Super-Resolution Using a Generative Adversarial Network (SRGAN) [11] and Perceptual Loss for Style Transfer [6], these papers argued that rather than using something like Mean Squared Error (MSE) loss we can use a perceptual loss function that tries to make image more visually appealing at a cost of higher MSE compared to that of models like VDSR, SRCNN, entire SRCNN was implemented from scratch and pretrained VGG19 [14] layeres were used for implementing perceptual loss. Some papers [7] posed a valid argument whether or not are deeper networks better, and some papers [16] tried to leverage deep CNNs for image enhancement. In this paper we try to see how perceptual loss changes the output of SRCNN, this idea yet trivial sounds great because we might get higher training times, but the deployment of such models can be done pretty much on CPUs.

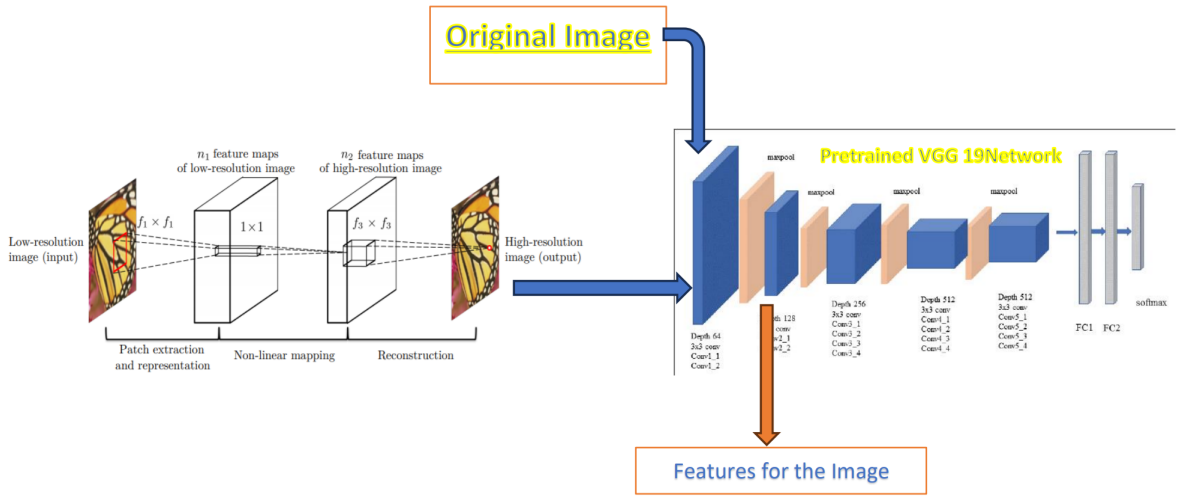

Fig. 2: SRCNN Architecture as described by Dong et al [3]

III. Methodology

A. Implementation of SRCNN

The SRCNN uses a simple 3-layered convolutional network architecture, as illustrated in Figure 2. This architecture represents a deep convolutional neural network designed for super-resolution tasks."

The SRCNN architecture consists of the following layers:

1) **Input Image**: The network takes a low-resolution input image with 3 color channels. On the input image, bilinear interpolation is performed such that resolution of both the images are same.
2) **Convolutional Layer 1**: This layer applies a convolution operation with 64 filters and a kernel size of 9×9. It is followed by a ReLU activation function.
3) **Convolutional Layer 2**: The second convolutional layer has 32 filters and uses a 1×1 kernel. As this layer performs non-linear mapping, it is followed by ReLU activation function.
4) **Convolutional Layer 3**: The final convolutional layer has 3 filters and a 5×5 kernel. This layer produces the high-resolution output image.
5) **Output Image**: The network's output is the high-resolution image with 3 color channels, which is the result of the super-resolution process.

The convolutional layers in the SRCNN architecture learn to extract and enhance image features, leading to improved image resolution. The ReLU activations introduce non-linearity, helping the network model complex relationships within the data. The Loss function to train this network is defined according to Equation 1, which is mentioned in detail further. The SRCNN model was implemented using Algorithm 1 and Algorithm 2.

B. Changing the Loss Function

The loss function for the Super-Resolution Convolutional Neural Network (SRCNN) is typically represented using the mean squared error (MSE), also known as L2 loss. It is defined as follows:

Algorithm 1: SRCNN Training Algorithm

Data: Low-resolution images I_{LR}, High-resolution images I_{HR}, Number of epochs N, Learning rate α, Batch size B

Result: Trained SRCNN model

Perform BiCubic interpolation of I_{LR};

Initialize SRCNN model with random weights;

for *epoch from 1 to* N **do**

  for i *from 1 to* $\frac{|\mathcal{I}_{\mathcal{LR}}|}{B}$ **do**

    Compute the loss: $L_{mse}(I_{LRi}, I_{HRi}) = \|\text{SRCNN}(I_{LRi}) - I_{HRi}\|^2$;

    Compute the gradient: $\nabla L(I_{LRi}, I_{HRi})$;

    Update model weights: SRCNN.update_weights$(\alpha \cdot \nabla L(I_{LRi}, I_{HRi}))$;

$$\mathcal{L}_{\text{mse}}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \|Y_i - \text{SRCNN}(X_i; \theta)\|_2^2 \tag{1}$$

In this equation: - $\mathcal{L}_{\text{mse}}$ represents the Mean-squared loss function. - N is the number of training examples. - Y_i represents the ground truth high-resolution image for the i-th example. - X_i represents the low-resolution input image for the i-th example. - SRCNN$(X_i; \theta)$ represents the output of the SRCNN model with parameters θ for the i-th input image. - $\|\cdot\|_2$ denotes the L2 (Euclidean) norm, which measures the squared pixel-wise difference between the predicted and ground truth images.

This loss function quantifies the dissimilarity between the predicted high-resolution images and the actual high-

Algorithm 2: SRCNN Testing Algorithm

Data: Low-resolution images I_{LR}, High-resolution images I_{HR}

Result: Output images I_{SR} generated from SRCNN with high resolution, N List of losses of the images

Initialize SRCNN model with learned weights;

for *i from 1 to* I_{LR} **do**

  $I_{SRi} = \text{SRCNN}(I_{LRi})$; `// Use SRCNN to generate image`

  ;

  Compute the loss: $L_{mse(}i)(I_{LRi}, I_{HRi}) = \|\text{SRCNN}(I_{LRi}) - I_{HRi}\|^2$;

  Print the loss:

  Save the image I_{SRi};

  Save the image I_{HRi};

resolution images in the training dataset, with the goal of minimizing this dissimilarity during the training process.

The perceptual loss function for super-resolution is computed based on feature similarity between the high-resolution ground truth image Y and the super-resolved image $\hat{Y}$ using a pre-trained deep neural network, typically VGG. It is defined as:

$$\mathcal{L}_{\text{perceptual}}(\theta) = \sum_{j=1}^{J} \lambda_j \frac{1}{C_j H_j W_j} \left\| \phi_l(Y) - \phi_l(\hat{Y}(\theta)) \right\|_2^2 \quad (2)$$

In this equation: - $\mathcal{L}_{\text{perceptual}}(\theta)$ represents the perceptual loss. - θ represents the parameters of the super-resolution model. - J denotes the number of layers in the VGG network used for feature extraction. - λ_j are scaling factors for each layer's loss term. - If j is the convolutional layer then $\phi_l(x)$ will be the feature maps of shape C_j X H_j X W_j. - $\phi_l(Y)$ and $\phi_l(\hat{Y}(\theta))$ represent the feature maps of the high-resolution ground truth image and the super-resolved image at layer j of the VGG network, respectively.

The perceptual loss aims to encourage the super-resolved image to have similar features to the high-resolution image at multiple layers of the VGG network. The scaling factors λ_l can be used to adjust the importance of different layers in the loss.

This loss function leverages high-level features extracted by VGG to capture perceptual quality, making it a popular choice for super-resolution tasks.

To combine the advantages of both the MSE loss and Perceptual loss we calculated the loss of the network by taking the weighted average of the two losses. It is defined as:

$$\mathcal{L}_{\text{final}} = \lambda_1 * \mathcal{L}_{\text{mse}}(\theta) + \lambda_2 * \mathcal{L}_{\text{perceptual}}(\theta) \quad (3)$$

In this equation: - λ_1 represents the weight of MSE loss - λ_2 represents the weight of perceptual loss

IV. Experiments

A. SRCNN with MSE loss

We trained the model on a Div2K dataset containing 800 images with batch size (B) = 2 and no of epoch equal to 1,500 and then we tested the model on the validation set given by Div2K dataset containing 100 images. This experiment was performed with x2 resolution. Due to complexity of VGGNet, we weren't able to train for more than 40 epochs. Another set of experiments were performed with x4 down-scaling, where we only used 49 image from training set of Div2K and 7 images from validation set, as images in Div2K had varying resolution images, it was diffcult to perform batch training. Training was performed for both the loss function for 150 epochs, and learning rate of all the experiments was set to 0.001.

B. SRCNN with Perceptual loss

We used the same values and trained and evaluated on the same dataset as in the case of MSE loss. For Perceptual loss the value for λ_i is [1], λ_1 is 0.1 and λ_2 is 0.9.The layers of the VGG19 pre-trained model used by us is :

1) `block2_conv2`: This layer captures some texture and structural information.

We tried using multiple layers but our gpu didn't have enough memory. After getting the loss we trained the SRCNN model (Algorithm 1) with L_{final} instead of L_{mse} and generated the results using Algorithm 2 for both the cases of x2 and x4.

V. Results

TABLE I: Mean Squared Errors of the images

MSE Loss	Image1	Image2	Image 3	Image 4	Image 5
Modified SRCNN	**0.0006**	**0.0032**	**0.0038**	**0.0024**	**0.0053**
MSE Loss (SRCNN)	**0.0003**	**0.0031**	**0.0036**	**0.0024**	**0.0053**

As shown in Figure 5, the SRCNN consistently outperformed other methods when using the perceptual loss in most cases, it is worth noting that for low epochs, there were square tiles visible in most of the cases. However, it's worth noting that the gap between the two methods is considerably narrower compared to the results we obtained during our experimentation with x2 factor scaling. In Figure 6, we observe a case where the perceptual loss produced superior results because the input images contained numerous features that the VGG network could leverage, resulting in more effective utilization of the perceptual loss.

On the other hand, in Figure 7, we encounter a scenario where, although the image generated by the perceptual loss appears visually superior, there is an unusual texture on the wall that was absent in the original image. This phenomenon is known as hallucination and is one of the challenges associated with employing perceptual loss. To address this issue, we have mitigated it by combining the two losses with a weighted sum in our final loss function.

In Table I, we can observe that the mean squared error (MSE) loss is either equal to or lower in the case of SRCNN

Fig. 3: These images are the Ground Truth images

Fig. 4: These images are results from SRCNN with MSE Loss(for x4)

Fig. 5: These images are results from SRCNN with Perceptual Loss(for x4)

Fig. 6: Case when Perceptual loss (right) performed better

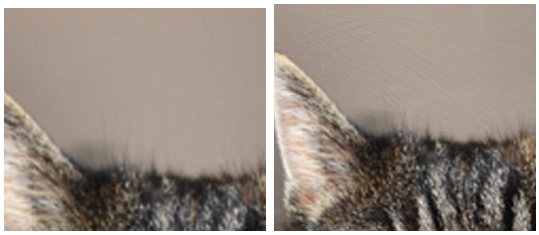

Fig. 7: Case when Perceptual loss (right) performed worse

without modification, even though the final output image is perceptually superior. This suggests that MSE loss may not be a suitable metric for obtaining higher-quality super-resolution images.

VI. Conclusion

We can conclude that using Perceptual loss we can get a better appealing images compared to the images generated

by the MSE lose. And adjusting the weights in the weighted average of two losses we can find the balance between better appealing image and a true image.

VII. Reflections and Acknowledgements

During the course of this project we learned what is the significance of defining a loss function on a deep convolutional neural network and what are the types of perceptual loss and how to find them and use them. Also, We learned the architecture and working of VGG network as well how to use its intermediate layers to calculate the perceptual loss.

References

[1] Eirikur Agustsson and Radu 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] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *CoRR*, abs/1501.00092, 2015.

[3] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):295–307, 2016.

[4] S. Farsiu, M.D. Robinson, M. Elad, and P. Milanfar. Fast and robust multiframe super resolution. *IEEE Transactions on Image Processing*, 13(10):1327–1344, 2004.

[5] Daniel Glasner, Shai Bagon, and Michal Irani. Super-resolution from a single image. In *2009 IEEE 12th International Conference on Computer Vision*, pages 349–356, 2009.

[6] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution, 2016.

[7] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1646–1654, 2016.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc.

[10] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[11] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network, 2017.

[12] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge, 2015.

[13] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[15] Radu Timofte, Eirikur Agustsson, Luc Van Gool, Ming-Hsuan Yang, Lei Zhang, Bee Lim, et al. Ntire 2017 challenge on single image super-resolution: Methods and results. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[16] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. *CoRR*, abs/1802.08797, 2018.