Obfuscated Human Face Reconstruction

Massimo Andreetta
massimo.andreetta@studenti.unipd.it

Abstract

This project investigates how distorted images of human faces can be recovered using Convolutional Neural Networks. Prior studies have been done in the field of super-resolution, but this project will concentrate on highly obfuscated faces reconstruction. The input data are obfuscated images of a person's face, obtained from the Labeled Faces in the Wild dataset using two types of obfuscation techniques: pixelation and Gaussian blurs. I investigate the impact of five loss functions on three various models. Since the L2 loss is the default option in image reconstruction, the impact of the loss function has not received much attention overall. In this essay, I highlight additional options for image restoration. In particular, I show the importance of the structural similarity index (SSIM) as a valid loss function. The experiments demonstrate that all three models can generate recognizable faces from inputs that have been altered.

1. Introduction

Human face reconstruction is a powerful technology that enables to reconstruct human faces from distorted or obfuscated images by recovering lost or damaged facial features. It can also be used to identify individuals in photos or videos that have been obscured in some way. Despite being the key factor in the network's learning, the loss function has received little attention from the image processing research community. Pixel loss is a commonly used loss function in image generation and reconstruction tasks. The goal of the pixel loss is to minimize the difference between the generated image and the target image by comparing their pixel-wise values. However, the Mean Squared Error/L2 norm is predominantly used to calculate the difference in pixels between the two images [1][2]. Unfortunately, L2 is not able to accurately predict how good an image appears to a human observer [3]. This is a result of several implicit assumptions made when using L2. First of all, the use of L2 assumes that the effect of noise is independent of the local characteristics of the image. On the contrary, human sensitivity to noise depends on contrast, local luminance, and structure [4]. Additionally, the L2 loss operates under the assumption of white Gaussian noise, which is generally not valid. For this reason, I investigate the use of three alternative error metrics (L1, peak signal to noise ratio (PSNR), and SSIM) and define a new metric that combines PSNR and SSIM. Moreover, I conduct a detailed analysis of the results obtained by three different CNN models, using PSNR and SSIM also as evaluation metrics of the quality of the image reconstruction.

2. Related Work

I based this work manly on information contained in different papers about image reconstruction tasks and Single Image Super-Resolution **(**SR**)**.

2.1. Image Reconstruction

Neural Networks, particularly Convolutional Neural Networks (CNN), have drawn a lot of attention in the context of image reconstruction as a result of their success in a number of computer vision tasks [6] and their effective use for denoising [7], deblurring [8], and demosaicking [2]. However, the main inspiration for this project comes from [9] and [10]. In the first paper the author examines the usage of CNNs to reconstruct obfuscated images of human faces, examining the pixel loss and perceptual loss, which was obtained by minimizing the high-level features captured by a pre-trained VGG network, showing how this loss is preferable to the first. Moreover, he proposed a new loss function obtained by combining these two losses. The model used for this task is a convolutional neural network with an autoencoder structure that uses 3 residual blocks between decoder and encoder. The authors of [10] focused instead on the loss layer, proposing several alternatives to L2. In particular, they show the importance of perceptually motivated losses. Moreover, they also define a novel differentiable loss function based on Multi-Scale Structural Similarity Index (MS-SSIM) combined with the advantages of L1 loss, showing that the quality of the results improves significantly with better loss functions, even when the network architecture doesn't change.

2.2. Image Super-Resolution

Recent research on super-resolution has progressed with the development of deep convolutional neural networks.

SR is the task of increasing the spatial resolution of an image by transforming a low-resolution image into a high-resolution version that maintains the original features while increasing the quality of the image. For this task, the authors of [11] suggested a Super-Resolution Convolutional Neural Network (SRCNN). Their approach directly learns a mapping from low-resolution to high-resolution images. Moreover, they proved the value of deep learning in the traditional super-resolution computer vision problem, demonstrating its ability to produce high-quality results. The authors of [12] instead developed an enhanced deep super-resolution network exploiting the improved performance obtained with residual learning techniques, while removing unnecessary modules in conventional residual networks. They also suggest a brand-new multi-scale deep super-resolution system and training technique that can combine various upscaling factors to reconstruct high-resolution images. In contrast to other earlier approaches, the authors of [13] proposed Residual network for SR tasks named SRResNet that acts as generator network for SRGAN, a generative adversarial network (GAN) for image super-resolution, capable of inferring photo-realistic natural images for 4x upscaling factors. D. Kim et al. in [14] proposed a Progressive Face SR Network, a Facial Attention Loss to restore the attributes of the adjacent area to the facial landmarks and Distilled Face Alignment Network to predict the location of all landmarks including occluded ones.

3. Dataset

For this project I have used the Labeled Faces in the Wild dataset [5]. This dataset contains 13,233 images collected from the web divided between pictures of 5,749 different people. A benefit of using this dataset is that the images are already processed to have the faces fully captured and centered around the image. If loaded using 'sklearn.datasets.fetch_lfw_people', every image in this dataset has 62x47 pixels (by default, the resize parameter of this scikit-learn dataset is set to 0.5). The images were then enlarged by setting the resize parameter to 1.2 to obtain 150x112 images, which were then resized using OpenCV to 112x112. Then I obfuscate each image using two methods. The first one is pixelation, also known as mosaicking. Each image is downscaled by a scale factor of 6 using the nearest-neighbour interpolation method, resulting in a pixelated appearance. The downscaled image is then upscaled back to its original size, which returns a pixelated version of the original image. The second method consists in using Gaussian blurs. In this process, each image is convolved with a 2D Gaussian kernel of size 15x15 and with standard deviation of 4. An example of the obfuscated images can be seen in Figure 1.

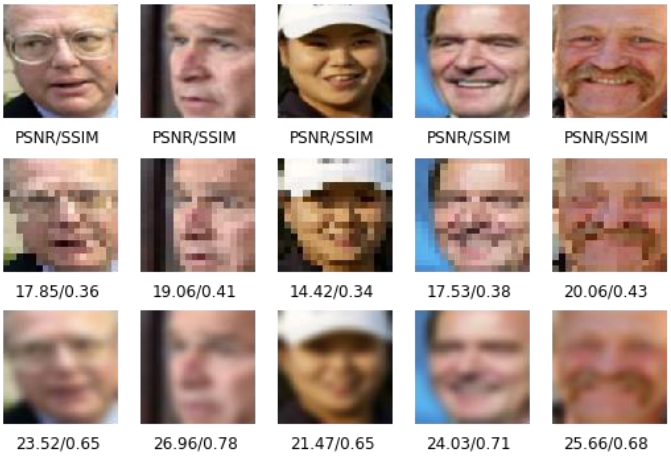

Figure 1. Examples of processed images from the Labeled Faces in the Wild dataset. In the first row we have the 112x112 original images. In the second row we have obfuscated version using pixelation while in the third row we have the obfuscated version using Gaussian blurring. Under each photo are indicated the PSNR and SSIM values compared to its original version.

Due to Google Colab restrictions on RAM and GPU usage, I decided to use only 2500 images, which were randomly split into 2000 training samples and 500 testing samples. These sets will then be preprocessed using Min-Max Scaling to scale the pixel values to the range [0,1]. By scaling the data to this specific range, the model can better learn the relationships between the features and the target variable, leading to improved model performance.

4. Method

For the task of reconstructing obfuscated human faces I decided to use 3 different CNN architectures to test the performance of 5 different loss functions used in image reconstruction to assess the best ones. Two of these models will be highly inspired by two already established architectures, as the objective of this project is to analyze how the different loss functions influence the reconstruction process. As mentioned previously, the input image of shape (112, 112, 3) undergoes a Min-Max scaling step before being fed into the model. The scaling step scales the input image between 0 and 1 to ensure that the image values are within the range of the activation function of the following layers. At the end of the process the output image is then scaled back to the original scale using the min-max scaling values obtained previously from the input images. As evaluation metrics I've used PSNR and SSIM to measure the quality of the image reconstruction. PSNR is a measure of how much the restored image differs from the original image, in terms of mean squared error. However, it is widely accepted that PSNR does not correlate well with human's perception of image quality [15]. On the other hand, SSIM compares the luminance, contrast, and structure of both images to determine how similar the two images are in terms of perceptual structure.

4.1. Architecture

4.1.1 Simple CNN

The first model is a simple CNN which is going to be used as baseline for my tests. The model consists of 5 Convolutional layers. The first Convolutional layer has 128 filters of size 9x9 while the next three Convolutional layers have 128 filters of size 4x4. Each of these layers is followed by a Batch Normalization layer and uses ReLU as activation function. Moreover, they all use zero padding to keep the spatial dimensions of the output the same as the input. The Convolutional layers perform the convolution operation between the input image and the learned filters to extract features from the image, while the Batch Normalization layers normalizes the output of each convolutional layer to improve the stability and training speed of the model. ReLU introduces non-linearity to the model, allowing it to learn more complex representations of the input image. The last layer is the output layer, which has 3 filters of size 4x4, and uses a sigmoid activation function, which produces the output image in the range [0,1]. This is a very simple CNN model, but it can still achieve good results when trained on image reconstruction tasks.

4.1.2 SRResNet

As mentioned before, the authors of [13] proposed SRResNet, a Residual Network for SR tasks. I decided to repurpose it for image reconstruction tasks by removing the 2 upsampling blocks at the end of the network, which are used to increase the resolution of the output image. In image reconstruction instead, the output size is the same of the input. The model is essentially the same, except for the output layer, where I used the sigmoid activation function rather than tanh to account for the use of Min-Max Scaling. The input image is passed through a Convolutional layer with 64 filters and a 9x9 kernel size, followed by a parametric ReLU (PReLU) activation function. Next, we pass the output of the Convolutional layer through a series of 5 residual blocks, where the output of each residual block is added to its input. This helps the model to learn the difference between the input and output image. Each residual block consists in two Convolutional layers with 64 filters 3x3, each followed by a Batch Normalization layer and a PReLU activation function. After passing through the residual blocks, the output is passed through another Convolutional layer with 64 filters and a 9x9 kernel size and then normalized using a Batch Normalization layer. The output of this layer is then added to the output of the initial Convolutional layer to form the final output. The final output is then passed through one last Convolutional layer with 3 filters, a 9x9 kernel size and a sigmoid activation function. All layers use stride = 1 and zero padding.

4.1.3 Residual Autoencoder

For the last model we use the architecture proposed in [9]. It could be defined as Residual Autoencoder as it uses an Encoder-Decoder structure with residual blocks in between. The architecture of the residual autoencoder consists of multiple Convolutional and Transpose Convolutional layers with zero padding, each of them followed by a Batch Normalization Layer and ReLu as activation function. The Convolutional layers are used to reduce the spatial dimensions of the input image, while the Transpose Convolutional layers are used to increase the spatial dimensions of the representations learned by the network. The network starts by processing the input image through a series of Convolutional layers. The first convolutional layer uses a filter of size 9x9 with 64 filters. After the first convolutional layer, there are other three Convolutional layers, each with a filter size of 4x4 and a stride of 2. The number of filters is, respectively, 64,128 and 256. Each of them reduces the spatial dimensions of the input by a factor of 2. These layers also use Batch Normalization and ReLU activation functions. The network then includes three residual blocks followed by an element-wise addition with the input tensor. Each residual block consists in two Convolutional layers with 64 filters 3x3, each followed by a Batch Normalization layer and a ReLU activation function. The purpose of these residual blocks is to allow the network to learn more complex representations of the input data. After the residual blocks, the network uses 4 transpose convolution layers to increase the spatial dimensions of the representations learned by the network. The first three layers have 256, 128, and 64 4x4 filters respectively with stride=2. The last one has 3 filters of size 9x9 and stride=1, and uses a sigmoid activation function. As said before, I' used Min-Max Scaling as input preprocessing instead of subtracting the mean training image and dividing each channel by 255 as the author of the paper did. For this reason, the activation of the output layer was changed from tanh to sigmoid. This choice allowed an increase in the reconstruction performance as we'll see in the section 5.

4.2. Loss functions

4.2.1 L2 loss function

In a wide range of disciplines, including signal and image processing, pattern recognition, and regression problems, the Mean Squared Error, or L2, is arguably the most common error measure. Its convexity and differentiability are two of the primary factors contributing to its popularity. However, it has been demonstrated that L2 does not fully

reflect how people perceive the quality of an image [15]. This loss is obtained using the squared Euclidean distance between the pixels of the predicted output (*x*) and the ground truth (*y*).

$$L_{L2} = \frac{1}{n}\sum_{i=1}^{n}(x^{(i)} - y^{(i)})^2 \quad (1)$$

Where n = 112x112x3 is the number of pixels in our images.

4.2.2 L1 loss function

This loss is obtained instead using the absolute Euclidean distance between the pixels of the network's predictions (*x*) and the ground truth (*y*).

$$L_{L1} = \frac{1}{n}\sum_{i=1}^{n}\left|x^{(i)} - y^{(i)}\right| \quad (2)$$

Networks trained with the Mean Absolute Error, or L1, perform better than L2 because it doesn't overly penalize larger errors like L2. However, the results are still sub-optimal as we will see in section 5.

4.2.3 PSNR loss function

PSNR is a measure of image quality that compares the predicted image to the ground truth image. It measures the difference between the two images in terms of the logarithmic ratio of the MSE to the maximum possible pixel value defined by *MAX*. In our case, this value is set to 255.

$$PSNR = 10\log_{10}\frac{MAX^2}{MSE} \quad (3)$$

A high PSNR generally indicates that the restored image is very similar to the original, so the loss is simply defined as

$$L_{PSNR} = 1/PSNR \quad (4)$$

4.2.4 SSIM loss function

SSIM is a measure of image quality that compares the predicted image to the ground truth image. It is based on the idea that the perceived quality of an image is related to the structural information present in the image. It can be expressed as:

$$SSIM = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{x,y} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \quad (5)$$

Where the constants c_1 and c_2 are used to stabilize the division. The output of SSIM ranges between -1 and 1, with 1 indicating identical images, so we can define the SSIM loss function as:

$$L_{SSIM} = 1 - SSIM \quad (6)$$

4.2.5 PSNR + SSIM loss function

Even though PSNR and SSIM have poor relationships with how people perceive image quality, we will see in the next section that, if used as loss functions for reconstruction tasks, they can achieve even better results than L2 and L1. This is true in particular for SSIM. For this reason, I decided to combine the image reconstruction capabilities of SSIM and PSNR in the following loss function:

$$L_{PSNR+SSIM} = L_{SSIM} + 0.5 * L_{PSNR} \quad (7)$$

The SSIM loss is weighted twice as much as the PSNR loss in order to prioritize the quality of the restored images.

5. Experiments

Each model uses Adam with learning rate = 0.0001 as optimizer and was trained with a batch size of 16 for 20 epochs because the PSNR and SSIM metric flattened at around this number of iterations. Moreover, I used early stopping with patience = 5. One of the models' limitations is their inability to include very small face characteristics like wrinkles, as these tiny details are lost in the obfuscation process. Overall, the models tend to have better performance when reconstructing blurred images instead of pixelated ones. This is because blurring can reduce detail while preserving the relative positions of various facial characteristics, whereas pixelated obfuscation loses a lot of location-specific detail in each pixelated block, especially around the eyes and mouth.

5.1. L2 loss

From Table 1 we can see that all the models have very decent results, even the very simple CNN. Moreover, the results on the training set are very similar to the ones of the test set, showing that the models can perform well also on faces they have never seen. The choice of using Min-Max Scaling and the sigmoid activation function in the output layer proved to be beneficial, as all the three models perform better than the residual autoencoder proposed in [9]. Moreover, the residual autoencoder proved to be the best model among the three for both deblurring and demosaicking, except on the test set where SSResNet is better. In Figure 2 we can see a quick view of the results obtained by my residual autoencoder on the reconstruction

of blurred and pixelated images.

L2 Loss	Blurred Images			
Architecture	Train set PSNR / SSIM		Test set PSNR / SSIM	
Simple CNN	26.5861	0.7900	26.5198	0.7868
SRResNet	27.9575	0.8101	27.7065	0.8000
Original Residual Autoencoder	26.56	0.700	25.96	0.696
Residual Autoencoder	27.5846	**0.8301**	27.3430	**0.8224**
	Pixelated images			
Architecture	Train set PSNR / SSIM		Test set PSNR / SSIM	
Simple CNN	23.9009	0.7562	23.8404	0.7529
SRResNet	24.3446	0.7650	24.1000	**0.7555**
Original Residual Autoencoder	23.94	0.600	23.52	0.585
Residual Autoencoder	26.1573	**0.7866**	24.3998	0.7492

Table 1. Performance of the various models on the reconstruction of blurred and pixelated images using L2 loss.

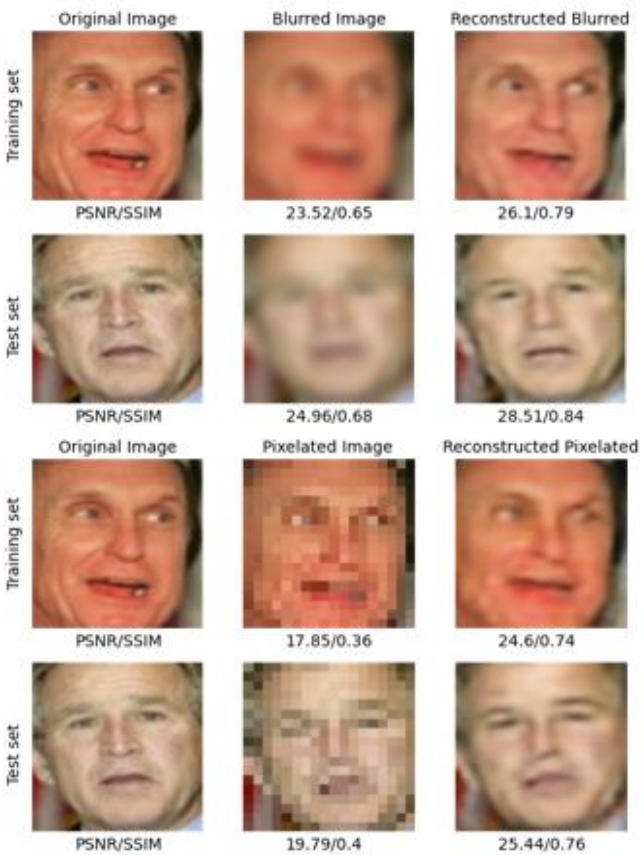

Figure 2. Reconstruction results from blurred and pixelated images of the residual autoencoder using L2 loss. The relative PSNR / SSIM results compared to the ground truth are provided below the pictures.

5.2. L1 loss

From Table 2 we can see that using L1 improves the overall performance of all the models, except for the residual autoencoder on pixelated images, which has worse performance compared to L2. For this reason, the SRResNet performs better than the residual autoencoder on pixelated images.

L1 Loss	Blurred Images			
Architecture	Train set PSNR / SSIM		Test set PSNR / SSIM	
Simple CNN	26.7774	0.7948	26.7322	0.7918
SRResNet	27.3167	0.8171	27.1332	0.8076
Residual Autoencoder	28.0884	**0.8368**	27.9211	**0.8301**
	Pixelated images			
Architecture	Train set PSNR / SSIM		Test set PSNR / SSIM	
Simple CNN	23.7566	0.7513	23.7075	0.7481
SRResNet	24.1818	**0.7813**	24.0060	**0.7729**
Residual Autoencoder	24.2364	0.7689	23.9235	0.7541

Table 2. Performance of the various models on the reconstruction of blurred and pixelated images using L1 loss.

5.3. PSNR loss

Here we can see in Table 3 that compared to L2 and L1, there is a small overall improvement of the performance of all the three models, both for blurred and pixelated image reconstruction, with residual autoencoder being the best. As expected, the PSNR values tend to be higher.

PSNR	Blurred Images			
Architecture	Train set PSNR / SSIM		Test set PSNR / SSIM	
Simple CNN	27.5200	0.7966	27.4431	0.7934
SRResNet	28.1864	0.8380	27.9722	0.8301
Residual Autoencoder	29.3583	**0.8544**	29.0591	**0.8459**
	Pixelated images			
Architecture	Train set PSNR / SSIM		Test set PSNR / SSIM	
Simple CNN	24.2428	0.7623	24.1503	0.7588
SRResNet	24.4730	0.7748	24.2071	0.7652
Residual Autoencoder	25.5830	**0.7902**	24.5705	**0.7664**

Table 3. Performance of the various models on the reconstruction of blurred and pixelated images using PSNR loss.

5.4. SSIM loss

Using SSIM loss provides a significant boost in the performance of all the models on both datasets, as shown in Table 4. All models have now a SSIM value well over 0.8 with the residual autoencoder passing 0.9, at least on the blurred dataset. However, as we can see in Figure 3, SSIM reconstructions struggle a bit to maintain the original color of the faces.

SSIM	Blurred Images			
Architecture	Train set PSNR / SSIM		Test set PSNR / SSIM	
Simple CNN	27.4119	0.8369	27.3281	0.8326
SRResNet	28.5495	0.8878	28.4434	0.8806
Residual Autoencoder	27.0003	**0.9108**	26.8274	**0.8946**
	Pixelated images			
Architecture	Train set PSNR / SSIM		Test set PSNR / SSIM	
Simple CNN	24.1874	0.7766	24.0753	0.7715
SRResNet	24.5532	0.8052	24.2078	0.7841
Residual Autoencoder	25.3908	**0.8612**	23.9914	**0.7876**

Table 4. Performance of the various models on the reconstruction of blurred and pixelated images using SSIM loss.

5.5. PSNR+SSIM loss

The slight lack of color of the SSIM loss is unfortunately still present also here, as we can see in Figure 3.

PSNR+SSIM	Blurred Images			
Architecture	Train set PSNR / SSIM		Test set PSNR / SSIM	
Simple CNN	27.8359	0.8336	27.7357	0.8290
SRResNet	29.2775	0.8892	29.1476	0.8817
Residual Autoencoder	28.6523	**0.9155**	28.3727	**0.8990**
	Pixelated images			
Architecture	Train set PSNR / SSIM		Test set PSNR / SSIM	
Simple CNN	23.9799	0.7763	23.9083	0.7715
SRResNet	24.9042	0.8109	24.5279	**0.7905**
Residual Autoencoder	25.2175	**0.8464**	24.0194	0.7857

Table 5. Performance of the various models on the reconstruction of blurred and pixelated images using PSNR+SSIM loss.

From Table 5 we can see a general increase of the PSNR and SSIM metrics. The performance of the CNN remained similar to the previous ones. Instead, the performance of the residual autoencoder improved on the deblurring task while the performance of SRResNet improved on both tasks, proving that this loss can actually be utilized with good results.

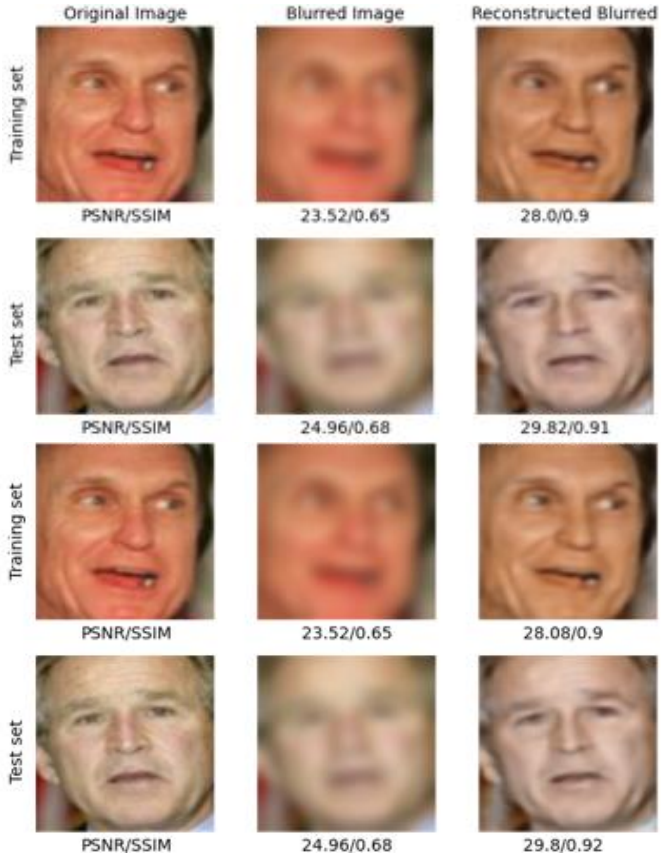

Figure 3. Reconstruction results from blurred images of the residual autoencoder. The results using SSIM loss are in the first and second row while in the third and fourth row we have the results of using PSNR+SSIM loss The relative PSNR / SSIM results compared to the ground truth are provided below the pictures.

6. Conclusion

In this project I concentrate on the loss functions, an aspect of neural networks that is typically neglected when exploring image restoration. I proposed several alternatives to L2, like L1, PSNR and SSIM and analyzed their effects on 3 different CNN architectures to show their improved performance on the networks, highlighting the improvement obtained with SSIM loss compared to L2. Moreover, I also define a novel loss that combines the SSIM and PSNR loss functions. For future work, it would be interesting to see the impact of a SRGAN and new state-of-the-art architectures for SR and image reconstruction tasks and the addition of perceptual loss and the Facial Attention Loss [14], as they have been shown to be effective not only for training but also be able to improve the quality of generated images.

References

[1] Jain, V., Seung, S.: Natural image denoising with convolutional networks. In Koller, D., Schuurmans, D., Bengio, Y., Bottou, L., eds.: Advances in Neural Information Processing Systems 21. Curran Associates, Inc. 2009, 769 776.

[2] Wang, Y.Q.: A multilayer neural network for image demosaicking. In: IEEE International Conference on Image Processing. 2014, 1852–1856.

[3] Zhang, L., Zhang, L., Mou, X., Zhang, D.: A comprehensive evaluation of full reference image quality assessment algorithms. In: IEEE International Conference on Image Processing. 2012, 1477–1480

[4] Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing 13(4), 2004, 600–612

[5] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[6] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS. 2012, 1097–1105

[7] Burger, H., Schuler, C., Harmeling, S.: Image denoising: Can plain neural networks compete with BM3D? In: IEEE Conference on Computer Vision and Pattern Recognition. 2012, 2392–2399

[8] Xu, L., Ren, J.S., Liu, C., Jia, J.: Deep convolutional neural network for image deconvolution. In: NIPS. 2014, 1790–1798

[9] Jacob Conrad Trinidad, "Reconstructing Obfuscated Human Faces.", 2017

[10] Hang Zhao, Orazio Gallo, Iuri Frosio, Jan Kautz. Loss Functions for Image Restoration with Neural Networks. arXiv:1511.08861v3, 2018

[11] Chao Dong, Chen Change Loy, Kaiming He, Xiaoou Tang. Image Super-Resolution Using Deep Convolutional Networks, arXiv:1501.00092v3, 2015

[12] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, Kyoung Mu Lee. Enhanced Deep Residual Networks for Single Image Super-Resolution, arXiv:1707.02921v1, 2017

[13] Ledig, Christian, et al. "Photo-realistic single image super-resolution using a generative adversarial network." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017

[14] D. Kim, M. Kim, G. Kwon et al.: Progressive Face Super-Resolution via Attention to Facial Landmark, arXiv:1908.08239v1, 2019

[15] Zhang, L., Zhang, L., Mou, X., Zhang, D.: A comprehensive evaluation of full reference image quality assessment algorithms. In: IEEE International Conference on Image Processing. 2012, 1477–1480