Reconstruction of Obfuscated Facial Images Using an Autoencoder

Abstract

This study aims to reconstruct obfuscated facial and demonstrates that modern image recognition methods based on neural networks can recover hidden in-formation from images. I have used a convolutional auto-encoder that takes noisy, pixelated, or blurred images as input and attempts to re-construct recognizable facial images. For training, pixel, perceptual and a weighted combination of the two have been used as loss functions. The results demonstrate that our trained model can successfully reconstruct facial images from highly obfuscated images, with some limitations in clarity, relative to the ground truth.

1. Introduction

Facial image obfuscation is technique used primarily to protect the identity of the subject such as faces of subjects in pictures and videos that exist in the public domain and require identity protection, possibly without consent, such as pictures clicked for Google maps' street view (Weyand et al., 2016). Furthermore, obfuscation is often preferred since encryption prevents many common forms of processing. For example, encrypted photos cannot be easily compressed for network transmission or cloud storage.

I attempt to show that deep learning can defeat some common image obfuscation techniques. This has immense policy implications in the privacy and security sphere, in indicating that greater sophistication in privacy protection are required along with also having important applications in the policy space for public domain crime detection and identification. Building on Jacob Conrad's work in the field of reconstructing obfuscated facial images (Trinidad), I attempt to de-obfuscate, three types of common image obfuscation techniques, developing our network model based on three separate loss functions. I use a Convolutional autoencoder to reconstruct **blurred, pixelated, and speckled** facial images.
For training, I use convolutional autoencoder. I used 3 loss functions: **per pixel, perceptual, and weighted combination of pixel and perceptual**. For evaluation I use SSIM and PSNR. Sections 6 and 7, explain in greater detail on our choice of loss functions and evaluation methods. Several experiments have been done, with different levels of obfuscations, loss functions and their combination. I also stress test our model by introducing high levels of obfuscations.

2. Previous work

Jacob Conrad (Trinidad) in his model uses autoencoder networks to reconstruct pixelated and blurred facial images. In training the model using Perceptual loss functions, his results show that perceptual loss functions create more aesthically pleasing faces. Even for high levels of blurring and pixelation, his model was able to construct seemingly recongnizable faces from unrecognizable inputs.

McPherson et.al show that neural networks can be used to classify highly obfuscated images in **MNIST, CIFAR-10, AT&T dataset** McPherson et al. (2016). This proves that obfuscated images contain information which neural networks are able to retrieve information but human eyes can't.

3. Dataset used

I use the Labeled **Faces in the Wild data** set, created at Umass,Amherst (Huang et al., 2008). This dataset contains 13,233 images collected from the web. There are a total of 5,749 people pictured in this dataset, with 1,680 people having two or more pictures in this dataset. Each image is of size 250 × 250 pixels. Pre-processing, has been explained in greater detail in section 5.1.

4. Obfuscation techniques used

I have used 3 obfuscation techniques. These three obfuscation techniques are shown in Figure 1.

4.1 Speckle Noise

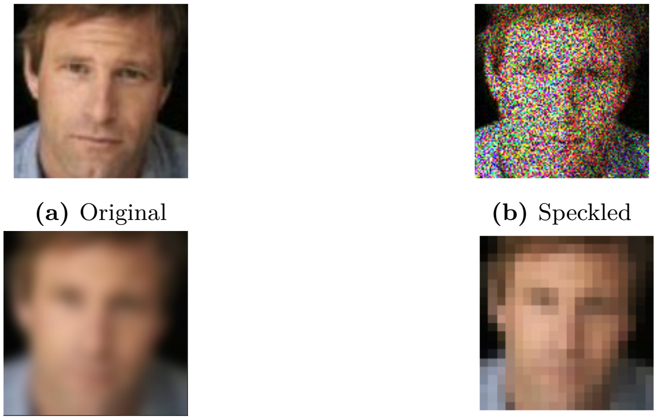
Mathematically there are two basic models of noise: Additive noise and Multiplicative noise. Additive noise is systematic in nature and can be easily modeled and hence removed or reduced easily; whereas multiplicative noise is image dependent, complex to model and hence difficult to reduce. Speckle is not a noise in an image but noise-like variation in contrast. Speckle is mainly a form of multiplicative noise, which occurs when a sound wave pulse arbitrarily interferes with the small particles or objects on a scale comparable to the sound wavelength. Speckle noise is defined as multiplicative noise, having a granular pattern it is the inherent property of SAR (Synthetic Aperture Radar) image (Raju et al., 2013). For this project, I have used the following formula for speckle noise:

$$D(X, y) = I(X, Y) \times S(X, Y)$$

Where, $D(X, Y)$ is the degraded image with speckle, $I(X, Y)$ is the original image and $S(X, Y)$ is the speckle noise; where (X, Y) denotes the pixel location.

4.2 Gaussian Blur

I used a Gaussian Blur function to blur the images. Gaussian blur is equivalent to convolving the original image with a kernel containing 2-dimensional Gaussian function. The Gaussian



(a) Original

(b) Speckled

(c) Blurred

(d) Pixelated

Figure 1: 3 Obfuscation techniques

function is defined as follows:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{\frac{x^2+y^2}{2\sigma^2}}$$

In the above function, σ controls the extent of obfuscation.Hence, working as a parameter for blurring operation.

4.3 Pixelation

Pixelation of an image involves downsampling an image and then upsampling it by using nearest-neighbor interpolation. To down sample by a factor of n the image is first filtered by an appropriate low-pass filter, a grid of size $n \times n$, and n is the parameter of the pixilation. If the output size of the image is not being changed, then it means I divide the image into blocks (the big resulting pixels), subsequently replacing all the pixels in each block with the average value of the pixels in that block.

5. Methodology

5.1 Dataset and Pre-processing

Before training our model, I pre-processed the images. I first cropped each image from 250×250 to 128×128, maintaining the center of the image. This is the facial region I will focus on for obfuscation. The purpose of this is so that our machine is able to better focus on the facial features and their reconstruction, since that is the problem at hand.

I obfuscate the image and create 4 separate arrays: original, speckle, blurred, and pixelated. I convert each array into pickle files which significantly improves processing efficiency during the training process.This is repeated for both levels of obfuscation. Therefore, each kind of obfuscation is associated with 2 distinct pickle files.

Speckle noise: I pass each image in the original numpy array to the 'speckle' function. This function adds speckle noise to each image and returns a noisy image. This noisy image is appended to a new array which I further convert into a pickle file for later use.

Pixelation: For a given 128 × 128 image, I pixelate it by first downsampling the image through bi linear interpolation, to a size $n^{'} \times n^{'}$ and scaling this image back up through nearest neighbor interpolation to 128×128. This method is an alternative to the previously mentioned low pass filter. The downsample factor can be seen as $n^{'}$ in this case and the smaller is $n^{'}$ more information will be lost in the final up-sampled image, cause greater image obfuscation, through pixelation. In this project two levels of obfuscation $n^{'} = 10$ and $n^{'} = 30$ have been considered.

Blurring: I used PIL package in python to apply a gaussian blur to the images. The library has a built in parameter r, which allows us to specify the radius of blur, instead of σ explained before. I have used r = 4.0 and 2.5 as the two levels of blur intensity, an example is shown in the Table 1.

5.2 Architecture

Our model input and output are of the shape (128 × 128 × 3) A convolutional neural network has been used to un-obfuscate the input facial images which are obfuscated. For the autoencoder, our input image is encoded using 2 convolutions layers, each of which are followed by a max pool layer. Each convolutional layer has 32 3 × 3 feature maps with a RELU activation function. The maxpooling windows are of size 3 × 3
The encoded image, then, is passed through 2 convolutions layers, each of which are followed by an upsampling layer. Each convolutional layer has 32 3 × 3 feature maps with a RELU activation function and the up sampling window is of size 2×2. The final layer convolution layer consists of 3 filter maps of 3×3, with the sigmoid activation function. I use the Adam optimizer to update parameters in our network. I I use two different loss functions i.e. the Mean squared error/pixel by pixel loss and perceptual loss.

6. Loss functions

Our autoeconder model is trained on two different loss functions:

6.1 Per Pixel Loss

The per pixel loss function calculates the mean squared error between between each pixel value from the real image and each pixel value from the generated image, i.e the squared Euclidean distance. Since our model input and output is in the shape (128 × 128 × 3), mathematically, this is our loss function

$$\mathcal{L} = \frac{||y - \hat{y}||^2}{128 \times 128 \times 3} \tag{1}$$

6.2 Perceptual Loss

Our second loss function is perceptual loss. It uses a pre-trained image classifier to find the difference between the predicted and the original image. It uses the layers in the pretrained trained model to get a feature representation of both predicted image $\hat{y}$ and the original image y. Then it calculates the mean squared error between the two feature representations. The formula of the loss function is given below:

$$J = \frac{1}{P \times Q \times R} ||\phi(y) - \phi(\hat{y})||$$

(P, Q, R) is the shape of the output of a chosen layer and $\phi(y)$, $\phi(\hat{y})$ are the feature representations of the original image and the predicted image respectively.

6.3 Weighted combination of Pixel loss and Perceptual loss

For the last experiment, I combine the perceptual and pixel loss function in an attempt to gain benefit from both the loss functions. I define our combined loss function as:

$$L_{combined} = \alpha \times L_{pixel} + L_{perceptual}$$

α is the weight given to the pixel loss.

7. Evaluation Process

I have used two different methods to evaluate our training process.

- **PSNR** or peak signal to noise ratio measures the quality of a reconstruction in terms of absolute errors. the higher the PSNR the better the reconstruction, generally. The formula of the PSNR is the following:

  $$PSNR = 10 \times \log_{10} \frac{R^2}{MSE}$$

  R is the maximum possible pixel value of the image

- **SSIM** or Structural Similarity Index measures perceived changes in structural information as well as in luminescence and contrast. The output of SSIM ranges between -1 and 1, with 1 indicating identical images. Unlike PSNR, SSIM is based on visible structures in the image. PSNR only estimates absolute errors.

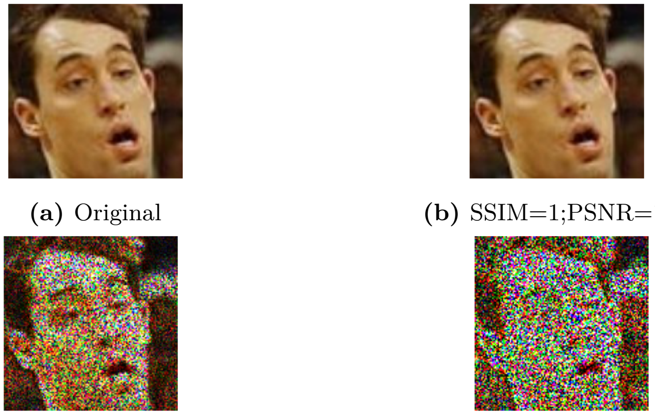
8. Experiment Results

The results discussed in the following sub-sections are a result of experimentation with **2 intensities (high and low) for each of the 3 methods of obfuscations alongside results from use of 3 different loss functions, namely per pixel/MSE loss, perceptual loss and a weighted combined loss function of both**.

8.1 Training using Pixel Loss function

I first train our auto encoder using per pixel loss function. The results presented in Table 1 give us some very interesting insights:

- SSIM and PSNR values suggest that the reconstructed images are more similar to the original image than obfuscated images



(a) Original

(b) SSIM=1;PSNR=100

(c) SSIM=.3845; PSNR=55

(d) SSIM=.3380; PSNR=43.5

Figure 2: PSNR and SSIM trend

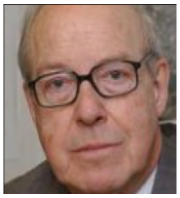
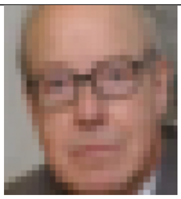
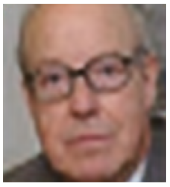
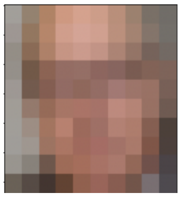
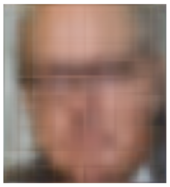
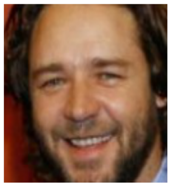
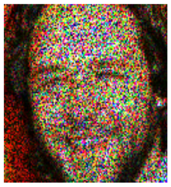
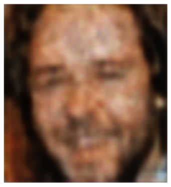
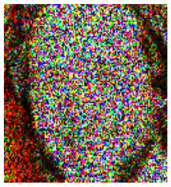
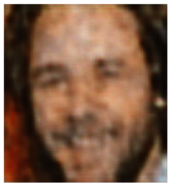
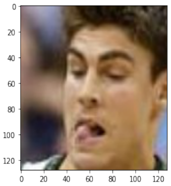
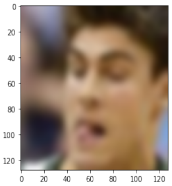
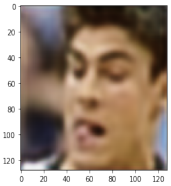
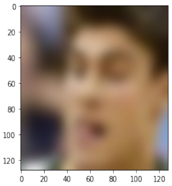
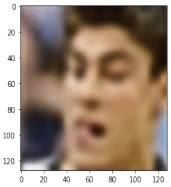
- Our auto encoder is able to reconstruct highly speckle image. This is surprising, as it is almost impossible for the naked to figure out the facial features in the highly speckled image. The reason could be that highly speckled images don't lose outlines and edges.

- I can notice that the model is not able to reconstruct sharp features such as wrinkles. This might be because it is difficult to learn for the system a definite place for wrinkles. Naturally, due to definite placement of features such as lips, eyes, nose, the model is able reconstruct these features correctly.

- I observe that feature reconstruction blurred images is better than that of pixelated images. Though blurring reduces details, it still maintains the relative position of various facial features, allowing for a more accurate reconstruction in comparison to pixelated obfuscation because a lot of specific location detail is lost within each pixelated block.

Table 1: Inputs and outputs for two levels of obfuscation

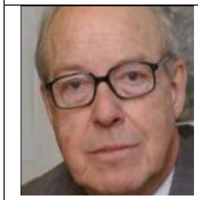
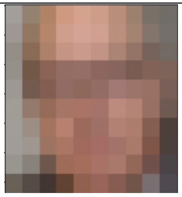
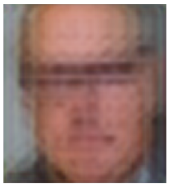
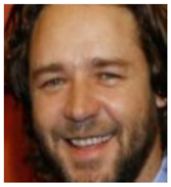
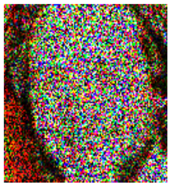
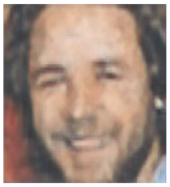
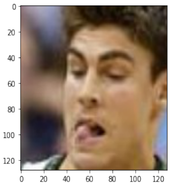
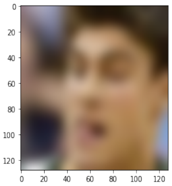
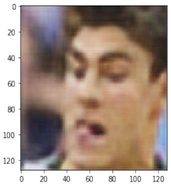
Ground Truth	Obfuscation type	Input	Output

 Original Image	**Pixelation** $n^{'} = 30$	 SSIM:0.66;PSNR:69.37	 SSIM:0.80;PSNR:70.21
	$n^{'} = 10$	 SSIM:0.32;PSNR:69.41	 SSIM:0.347;PSNR:69.16
 Original Image	**Speckle Noise** Low noise	 SSIM=0.33;PSNR=55	 SSIM:0.71;PSNR:73.9
	High noise	 SSIM=0.30;PSNR=43	 SSIM:.58;PSNR:62
 Original Image	**Gaussian Blur** $r = 2.5$	 SSIM=0.71;PSNR=73.8	 SSIM=0.78;PSNR=75.2
	$r = 4$	 SSIM=0.52;PSNR=71.5	 SSIM=0.67;PSNR:73.1

8.2 Training using Perceptual loss

In this experiment, I train our autoencoder using the Perceptual loss function. Unlike Pixel loss function which operates on each pixel and tries to capture local features, perceptual loss function captures high level features by minimizing the high level features captured by the VGG network (a deep CNN network which I import). Results in table 2 call for some discussion:

Table 2: Model predictions of high obfuscation inputs when trained with Perceptual loss function

Ground Truth	Obfuscation type	Input	Output
 Original Image	**Pixelation** $n^{'} = 10$	 SSIM:0.32;PSNR:69.41	 SSIM:0.394;PSNR:69.77
 Original Image	**Speckle Noise** High noise	 SSIM=0.30;PSNR=43	 SSIM:0.70;PSNR:62
 Original Image	**Gaussian Blur** $r = 4$	 SSIM=0.52;PSNR:71.5	 SSIM=0.69;PSNR:73

- I can see that the output images in Table 2 have more well defined outlines and edges than output in Table 1. This proves that our perceptual loss function is indeed capturing high level features more efficiently.
- Similar to the trend in Table 1, I can see that SSIM and PSNR of reconstructed images is greater than that of obfucated images. Moreover, reconstructed images in Table 2 are more similar to the original images that reconstructed images in Table 1 even though SSIM and PSNR in some of the reconstructed images in Table 2 is lower than reconstructed images in Table 1. Therefore, both PSNR and SSIM are not the best metrics to evaluate structural similarity
- I also observe that perceptual loss function is better suited to especially reconstruct blurred image. This is because blur is not a local phenomena which makes it difficult to be captured by the pixel loss function.

- I have also found that the loss values for pixelation and blurring are close to each other, for the same level of obfuscation. Since, the obfuscation in pixelation causes much higher information loss and therefore the levels of obfuscation for $r = 1$ and $n^{'} = 30$ is not exactly the same, the loss values are not **very** close.

8.3 Training using combination of Pixel and Perceptual Loss

In this last experiment, I combine the perceptual and pixel loss function in an attempt to gain benefit from both the loss functions. I define our combined loss function as:

$$L_{combined} = \alpha \times L_{pixel} + L_{perceptual}$$

In our model, I take $\alpha = 0.5$. This means perceptual loss is weighted twice as much as pixel loss. I observe that the output images in Table 3 are more similar to the original image than the output images in Table 1. This is because the new loss function takes the best of Perceptual and Pixel loss function. Pixel loss function focuses on the local features and perceptual loss function works on the face outline and edges.However, the difference between the output of table 2 and table 3, most probably because perceptual loss dominated over pixel loss, which is preferred.

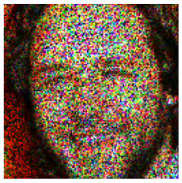
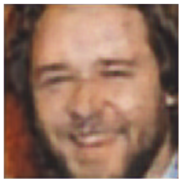
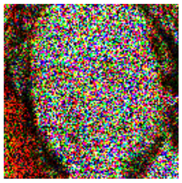
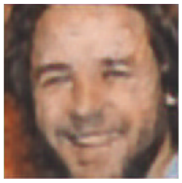
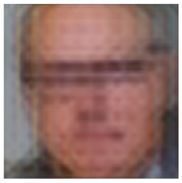
Original Image	**Obfuscation Type**	**Input**	**Output**
	Low Speckle Noise	 SSIM = 0.33	 SSIM = 0.70
	High Speckle Noise	 SSIM = 0.30	 SSIM = 0.65
	Pixelation$n^{'} = 10$	 SSIM = 0.323	 SSIM = 0.383

Table 3: Our results on using the combined loss function

9. Conclusion

In our study, I work on 3 types of obfuscations: Speckle Noise, Pixelation, and Gaussian Blur over 3 loss functions: Per Pixel, Perceptual, and combination of Pixel and Perceptual. I demonstrate that high intensity blurred and noisy images tend towards greater similarity after reconstruction, to the ground truth whereas for highly pixelated images the de-obfuscation is weaker and only provides a general contour of the the subject's face. While I do observe better performance when utilising combined loss functions, I posit, that this greater difficulty in reconstruction occurs, because essentially in pixelation there is a 'loss ' of information from the ground truth , while in speckle noise and bluring the ground truth information is simply 'distorted' and therefore, contains more information for training. With future progress, better parametrization of hyperparameters such as the learning rates, as well as greater architectural depth, will hopefully lead to cleaner de-obfuscation and reconstruction of lost information, aiding the immense privacy and policy scope this technique presents

References

Gary B Huang, Marwan Mattar, Tamara Berg, and Eric Learned-Miller. Labeled faces in the wild: A database forstudying face recognition in unconstrained environments. 2008.

Richard McPherson, Reza Shokri, and Vitaly Shmatikov. Defeating image obfuscation with deep learning. *arXiv preprint arXiv:1609.00408*, 2016.

KM Sharavana Raju, Mohammad Shahnawaz Nasir, and T Meera Devi. Filtering techniques to reduce speckle noise and image quality enhancement methods on satellite images. *IOSR Journal of Computer Engineering (IOSR-JCE)*, 15(4):10–15, 2013.

Jacob Conrad Trinidad. Reconstructing obfuscated human faces. *Standford University*.

Tobias Weyand, Ilya Kostrikov, and James Philbin. Planet-photo geolocation with convolutional neural networks. In *European Conference on Computer Vision*, pages 37–55. Springer, 2016.