A Machine Learning Approach to Beating Image Obfuscation

CS-408 PROJECT REPORT

Dhruv Agarwal
Department of Computer Science
Ashoka University

Harsh Karamchandani
Department of Computer Science
Ashoka University

Prakhar Jain
Department of Computer Science
Ashoka University

May 28, 2019

Abstract

This paper aims to beat Image Obfuscation techniques used to hide private information in facial images. We have implemented an Autoencoder that takes an obfuscated image as input and reconstructs the facial image as the output. Our model has been trained on Labeled Faces in the Wild dataset using three types of obfuscation techniques on faces: pixelation, blurring and blocking (a technique devised to beat our current model). Previous work has been done on reconstruction of images and we build on that work to provide better results by experimenting with two loss functions: Pixel-to-pixel loss and Perceptual loss. Experiments conducted on our model demonstrate that our model can produce recognizable images from obfuscated facial images and beat commonly-used obfuscation techniques.

***K*eywords** Image Obfuscation · Auto-encoder · CNN · Blurring · Pixelation · Blocking

1 Introduction

In today's world, when everyone is generating millions of bits of data each day, it has become more important than ever to preserve the privacy of such data. A primary component of this data comprises of facial images that are shared willingly or unwillingly on social media platforms. To preserve the privacy of the people photographed in such images, many **Image Obfuscation** methods have been devised. The aim of these methods is to hide private information and make the identity of the image's subject indecipherable. Originally, these methods aimed at making images indecipherable to the human eye but due to advances in modern computing and computer vision, these methods require stress testing against more powerful adversaries.

Through this paper we **test the efficacy** of three such commonly used image obfuscation methods against a **Machine Learning adversary**. We attempt to beat **blurring**, **pixelation** and **blocking** of facial images by reconstructing the original facial images from their obfuscated versions. In an ideal world, we would like to reconstruct the original image exactly. However, this expectation is unreasonable as obfuscation causes loss of information of the image. Hence, we are able to get only similar-looking approximations of the original image. These two scenarios are depicted in figure 1.

We use a **convolutional autoencoder** approach to train our model. The input to our autoencoder is an obfuscated image, and the desired output of our model is the corresponding ground truth image. We train the autoencoder using two loss functions: 1) **mean squared error** (i.e. pixel loss), and 2) **perceptual loss**. We compare the resulting reconstructions and use the better loss function for the rest of our experiments.

2 Image Obfuscation Techniques

A variety of image obfuscation techniques are used for protecting the identity of human subjects in images: **blurring**, **pixelation**, **P3** (privacy-preserving photo), and **blocking**. Most of these techniques are easily accessible by the general public, either as features of photo-editing software suites, or as stand-alone services on the internet. In this section, we briefly describe three obfuscation techniques that we will attempt to beat — pixelation, blurring, blocking.

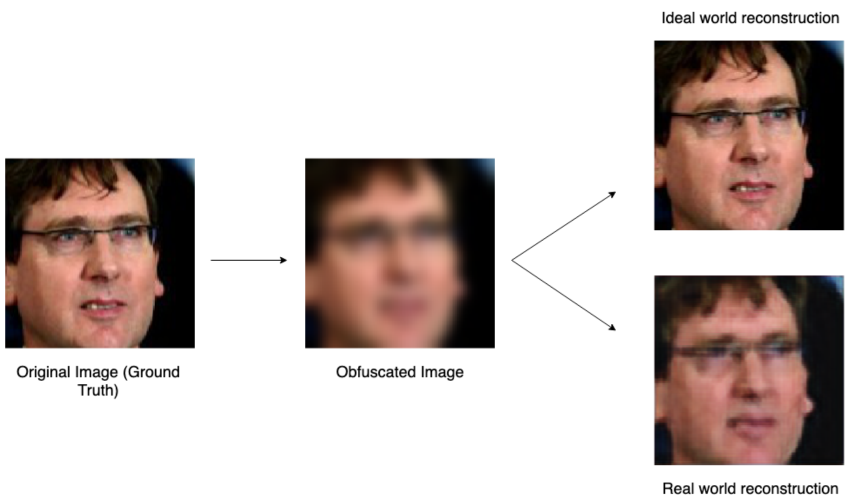

Ideal world reconstruction
Original Image (Ground Truth)
Obfuscated Image
Real world reconstruction

Figure 1: Our aim: ideal world vs real world

2.1 Pixelation

Pixelation (also called mosaicing) uses a square grid to obfuscate parts of an image. This is done by taking a square grid of size $n \times n$ where n is called the window and is the parameter of the pixelation operation. The value computed by averaging the values of each pixel falling in the window is taken as the value for the entire window. Higher window size n results in more obfuscation and in turn better protection of privacy. An example can be seen in figure 2b.

2.2 Gaussian Blur

Gaussian blur is the result of blurring an image using a Gaussian function. In effect, a Gaussian blur is equivalent to convolving the original image with a kernel containing values generated from a 2-dimensional Gaussian function. A two-dimensional Gaussian function is defined as follows:

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

While using Gaussian blur to obfuscate images, σ (the standard deviation) decides the extent of obfuscation. Higher σ results in higher obfuscation. Hence, σ, is the parameter of the blurring operation. An example can be seen in figure 2c.

2.3 Blocking

Blocking pastes black square boxes on the facial image in an attempt to make the identity of the person indecipherable. This is done by randomly generating the top-left and bottom-right coordinates of one or multiple $n \times n$ square boxes. In case of multiple boxes, it is ensured that they are non-overlapping. This latter part is done to ensure the same amount of information loss in all images that are obfuscated by this method. An example can be seen in figure 2d.

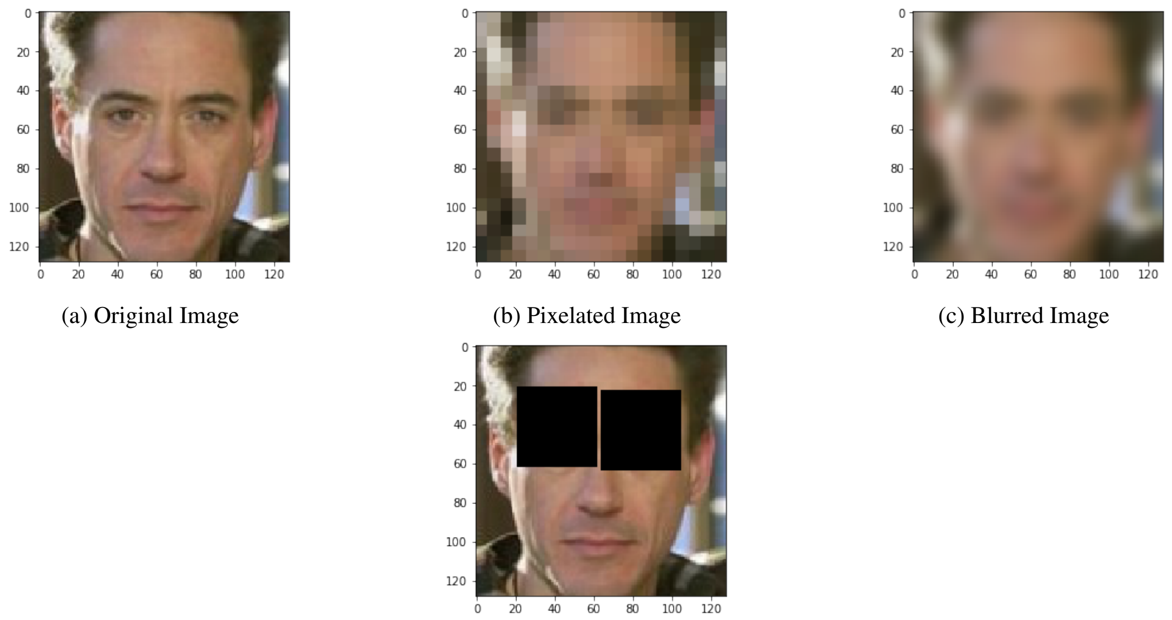
Two parameters can be defined for this technique:

$$\begin{aligned} N = &\ \text{the number of boxes} \\ n = &\ \text{size of each box} \end{aligned}$$

It must be noted that this method of obfuscation is not very formal, and has been designed by us in an attempt to beat our model.

3 Related Work

A lot of research has been done on image obfuscation techniques and their usage on facial images. Common examples of such techniques being implemented include Youtube's algorithmic blurring of faces in uploaded videos to protect identities and Google Street View's blurring of faces. Past research has also been done on testing the efficacy of these techniques by testing them against machine learning models.



(a) Original Image (b) Pixelated Image (c) Blurred Image

(d) Blocked Image

Figure 2: Original Image and its obfuscated variants

Mcpherson *et al.* [**?**] developed a convolutional neural network (CNN) to classify obfuscated images from multiple datasets such as MNIST, CIFAR-10, AT&T and FaceScrub. This experiment exposed the ineffectiveness of common obfuscation techniques as they are unable to hide private information in the image. This leads to accurate classification of even obfuscated images in these datasets. On the FaceScrub dataset that consists of facial images of 536 individuals, **the model classified with 57-71%** accuracy which is pretty close to the baseline of a 75% achieved on unobfuscated images.

Trinidad [**?**] constructed a convolutional neural network to reconstruct obfuscated facial images from the Labelled Faces in the Wild (LFW) dataset. He uses the concepts of **perceptual loss** and **pixel loss** and compared the two loss functions for different degrees of obfuscation. Experiments performed on the dataset produced recognizable faces from seemingly unrecognizable inputs.

4 Dataset

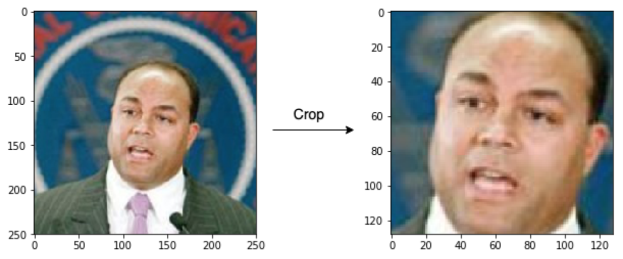


Figure 3: The effect of cropping from 250×250 to 128×128 pixels

We use the Labelled Faces in the Wild (LFW) dataset created by UMass, Amherst [**?**]. This dataset consists of facial images of 13233 images of 5749 people, labelled with their names. Each image is of size 250×250 pixels. This dataset uses the OpenCV implementation of the Viola-Jones face detection algorithm to detect faces in the raw images, and centres the face in the 250×250 pixels image.

In the interest of computational capability and in order to focus on our task of reconstructing *facial* images, we perform some preprocessing on the raw LFW dataset. We crop the image equally from all sides, keeping the center fixed, from 250×250 to 128×128 pixels. An example of this cropping is shown in figure 3. In effect, we crop out the the backgrounds in images that are unnecessary for the task at hand. We can safely do without worrying about cutting out facial images because the LFW dataset guarantees the existence of the face in the center of the image (as noted in the previous paragraph).

Artificial Obfuscation of Images

- **Pixelation** – We first downscale the 128×128 image to a smaller size n' using bi-linear interpolation, and then scale it back up to its original size using nearest-neighbour interpolation. Since information is lost in this process, it results in a pixelated version of the original image. As a substitute for the window size n (as defined in section 2.1), we use the "downsample-to" size n' as the parameter. Hence, if we downscale to 20×20 before scaling back up, $n' = 20$. It is trivial to see that smaller the value of n', more obfuscated the image. An example of pixelation with $n' = 20$ can be seen in figure 2b.
- **Blurring** – We use the PIL Python package to apply a Gaussian blur to our original 128×128 image. The Gaussian Blur method allows us to specify the radius (r) of the blur, instead of the standard deviation σ. Since $r \propto \sigma$, we use the radius as the parameter for the blurring operation. An example of blurring with $r = 4.0$ can be seen in figure 2c.
- **Blocking** – We first fix the number of black boxes N. Then, we fix the size of each box n. We then randomly generate the co-ordinates of N $n \times n$ boxes such that they do not overlap. Additionally, we ensure that the boxes are placed such that the probability of them hiding the most identifying facial features (like eyes, mouth) is higher. To do this, we leave 20 pixels on each side of the image – i.e. the black boxes are restricted to the center-most 108×108 part of the image. An example of blocking with $N = 2$ and $n = 40$ can be seen in figure 2d. It can be seen that due to the restriction on their position, the black boxes end up covering both the eyes of the person.

We randomly split our dataset into a training set consisting of 11909 images, and a validation set consisting of 1324 images.

5 Model

5.1 Architecture

We have used a Convolutional Neural Network (CNN) to beat image obfuscation techniques by un-obfuscating the input facial image. Our model has an input and output shape of $(128, 128, 3)$. For training our model, the input is the obfuscated image and the desired output is the original (ground-truth) image.

The input image is encoded using 2 convolution layers with 32 filter maps of size $(3, 3)$ each followed by a max-pooling layers of window size $(2, 2)$ each using the ReLU activation function.

Once the image is encoded, it is passed through 2 convolution layers with 32 filter maps of size $(3, 3)$ each followed by an up-sampling layer of $(2, 2)$ each using the ReLU activation function. Finally, the image is passed through a final convolution layer with 3 filter maps of $(3, 3)$ maps each using the sigmoid activation function. The model uses a Adadelta optimizer to update its parameters and adapt the learning rate. We use multiple loss functions to train our model, which we detail in section 5.2.

5.2 Loss Functions

5.2.1 Mean Squared Error (pixel-by-pixel comparison)

The first loss function we used to train our autoencoder calculates the Euclidean distance between the original image and the output image from the model, pixel-by-pixel. It does that by taking the mean squared difference between each corresponding pixel in the output image and the original image. For a 128×128 RGB image (i.e. 3 channels), if y is the desired output (original image) $\hat{y}$ is the reconstructed output of the model, the MSE would be:

$$J = \frac{||y - \hat{y}||_2^2}{128 \times 128 \times 3}$$

5.2.2 Perceptual Loss

The second loss function is perceptual loss [**?**]. It uses a pre-trained image classification network with fixed parameters to return the difference in content in the output image and the original image. To quantify this difference, it chooses a layer of this pre-trained model and gets the feature representations of the output image (y) and the original image ($\hat{y}$) at that layer. Then it calculates the mean squared error between these two features representations. If the output of the chosen layer is of shape (P, Q, R) and $\phi(y)$, $(\phi(\hat{y}))$ are the feature representations of images y, $(\hat{y})$, then the perceptual loss is defined as:

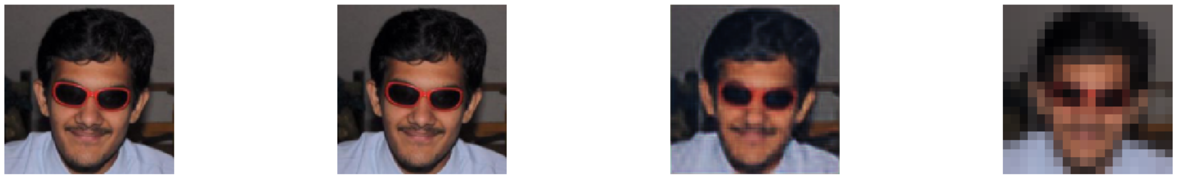
$$J = \frac{1}{P \times Q \times R} ||\phi(y) - \phi(\hat{y})||_2^2$$

In our model, we implemented perceptual loss using the VGG-16 network pre-trained on ImageNet. We then used the relu3_3 layer from this network as the feature representations for our images. This feature representation is of size $56 \times 56 \times 128$. This loss is minimized by backpropagating the difference in parameters of these feature representations.

6 Evaluation Criteria

We use two evaluation criteria to test the similarity of the reconstructed output with the original (ground-image) image: SSIM (structural similarity) [**?**], and visual similarity.

SSIM is a measure of similarity between two images that compares local patterns of pixel intensities that have been normalized for luminance and contrast [**?**]. It gives an output in the range $[-1, 1]$, with 1 signifying complete structural similarity between the two images. Since it is designed to improve upon traditional image similarity metrics like MSE and PSNR (peak signal-to-noise ratio), we limit ourselves to using SSIM for evaluation. An example of SSIM outputs for various variants of an image can be seen in figure 4.



(a) Ground Truth (b) SSIM = 1.0 (c) SSIM = 0.88 (d) SSIM = 0.71

Figure 4: SSIM values for the original image and its variants

However, while SSIM takes into account perceptual features of the image like contrast and luminance, it has low correlation with human perception of image similarity. Since no mathematical metric exists for detecting human visual similarity, to evaluate the performance of our model, we rely on manual inspection of the reconstructed image and the original image to gauge similarity.

7 Experiments and Results

In this section, we describe the experiments we did, and give an overview of the approach we took on our way to finally settling on a model that gave acceptable results. We also show how the loss values for different versions of our model (for various obfuscation extents and techniques) help validate some of our hypotheses and indicate other interesting results.

7.1 Training Using MSE Loss

We first train our autoencoder architecture as described in section 5.1 using the MSE (pixel-by-pixel) loss described in section 5.2.1. The results are illustrated in table 1.

The results obtained provide a decent increase in the SSIM, with a maximum increase of 0.11. However we see that, while the face is decipherable in the reconstructed image, it has a blur that still makes it look obfuscated. Hence, while the SSIM increase is substantial, there is more scope of visual similarity between the reconstructed image and the original image – perhaps by getting rid of the blur on the image.

With the objective of getting not just structural similarity (as indicated by SSIM), but also visual similarity by removing blur, we explore a different loss function — perceptual loss.

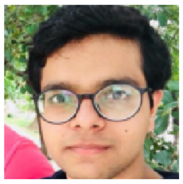
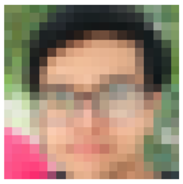
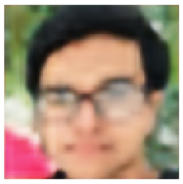
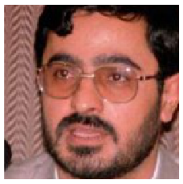
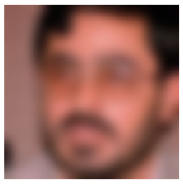
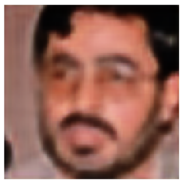
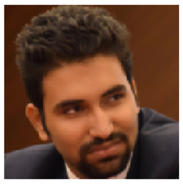
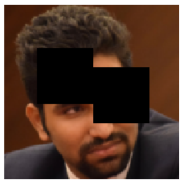
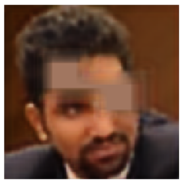
Original Image	Obfuscation Type	Obfuscated Image	Reconstructed Image
	Pixelated $n' = 20$	 SSIM = 0.60	 SSIM = 0.70
	Blurred $r = 4.0$	 SSIM = 0.64	 SSIM = 0.75
	Blocked $N = 2$ and $n = 40$	 SSIM = 0.75	 SSIM = 0.85

Table 1: Our results on using MSE loss function

7.2 Training Using Perceptual Loss

In our second experiment, we train our model with perceptual loss (as defined in section 5.2.2) to achieve better visual similarity between our reconstructed output image and our original image. While MSE looks at the image at a fine-grained level (pixel-by-pixel), perceptual loss looks at the image at a more coarse-grained level — it looks at the image as a whole. While MSE captures the low-level features by comparing each pixel, perceptual loss captures high-level features returned by the VGG network [**?**].

Since the blur that we saw in the results of the previous experiment is an image-wide phenomena, it would not have been captured by the MSE loss. Hence, we expect our model to capture and avoid the blur when using perceptual loss, thereby leading to visually more similar reconstructions. The results of this experiment are illustrated in table 2.

A few observations from the results

- As expected, perceptual loss succeeds in avoiding the blur in the reconstructed output image.
- While the trend in increase of SSIM is similar to the trend resulted by using MSE (a similar maximum increase of 0.11), reconstructions resulting from perceptual loss are more visually similar to the ground truth. This substantiates the claim that SSIM focuses on structural similarity of images, but is not a good metric to test visual similarity of images.
- The model does not perform very well on images obfuscated using blocking. This is because blocking completely removes certain parts of the image thereby causing complete information loss in those areas of the image. This is unlike blurring and pixelation, where no part of the image experiences complete loss of information. Hence, the model cannot learn anything in the blocked areas of the image, thereby causing poor reconstructions.
- As can be seen in figure 5a, when training on pixelated images, the training loss for $n' = 32$ is substantially lower than the training loss for $n' = 20$. A similar trend can be seen for training on blurred images in figure 5b, and for training on blocked images in figure 5c.

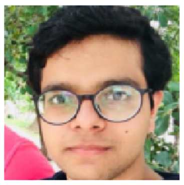
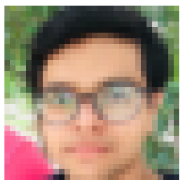
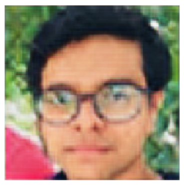
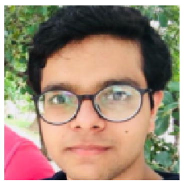
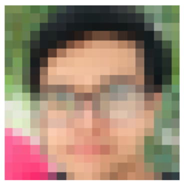
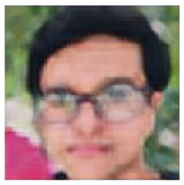
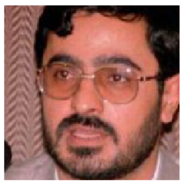
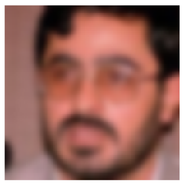
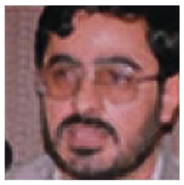
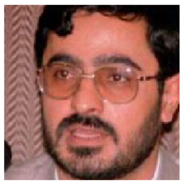
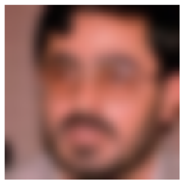
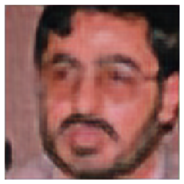
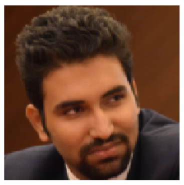
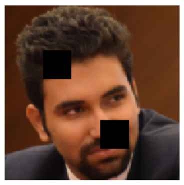
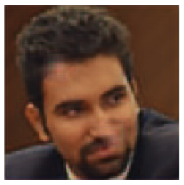
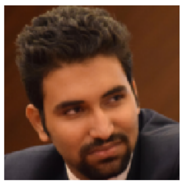
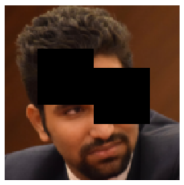
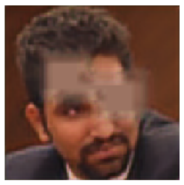
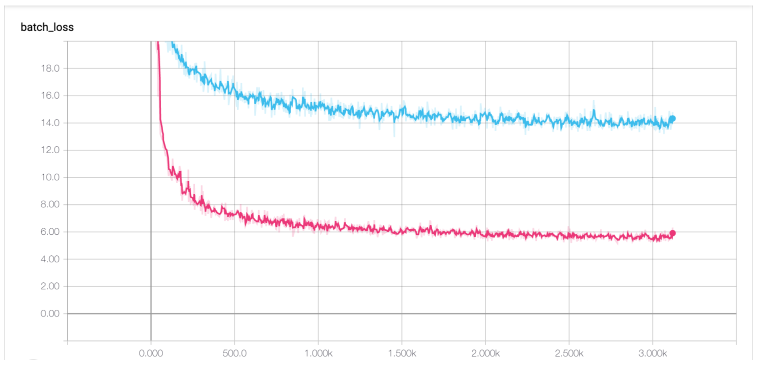
Original Image	Obfuscation Type	Obfuscated Image	Reconstructed Image
	Pixelated $n' = 32$	 SSIM = 0.73	 SSIM = 0.82
	Pixelated $n' = 20$	 SSIM = 0.60	 SSIM = 0.69
	Blurred $r = 2.5$	 SSIM = 0.75	 SSIM = 0.81
	Blurred $r = 4.0$	 SSIM = 0.64	 SSIM = 0.73
	Blocked $N = 2$ and $n = 20$	 SSIM = 0.93	 SSIM = 0.91
	Blocked $N = 2$ and $n = 40$	 SSIM = 0.75	 SSIM = 0.86

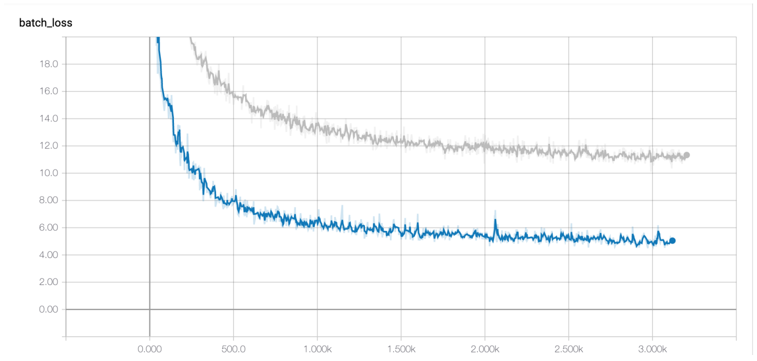
Table 2: Our results on using Perceptual loss function

While these results are expected, they confirm that our model is working on a solid theoretical backing. It is *indeed* learning that a higher blur, higher pixelation, or higher blocking causes higher loss in information of the original image.

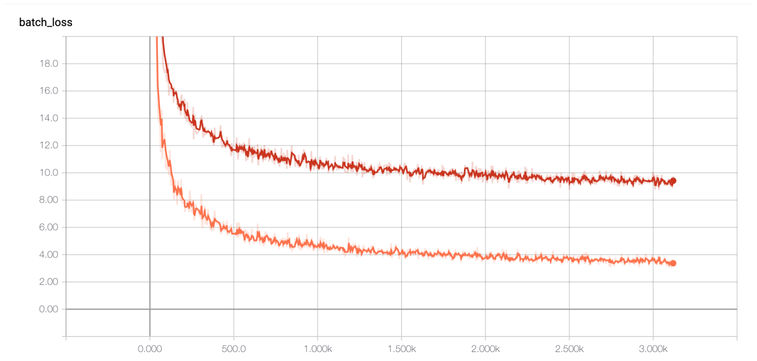
- As can be seen in figure 6, our models training on pixelation with $n' = 32$ and blur with $r = 2.5$ have very similar loss values. This, in conjunction with the previous observation, suggests that these two obfuscation techniques with these specific extents of obfuscation cause very similar loss of information in the original image.



(a) Blue: $n' = 20$, Magenta: $n' = 32$



(b) Gray: $r = 4.0$, Blue: $r = 2.5$



(c) Red: $n = 40$, Orange: $n = 20$ ($N = 2$)

Figure 5: Loss values for obfuscation techniques. Top: Pixelation; Middle: Blurring; Bottom: Blocking.

7.3 Variational Autoencoder

We also implemented a Variational Auto Encoder as part of our experimentation to test if such a network can capture all the important features of a face in its latent variables. This did not provide optimal results since we did not classify our images, so the autoencoder learned exactly one distribution for all the features. This translated to an output that was just a ghost image of a face. Since the features are extracted from a probability distribution, the image represents the features that are most common in the dataset. Clearly this is not what we wanted, since the usecase of the model was not coincident to our usecase.

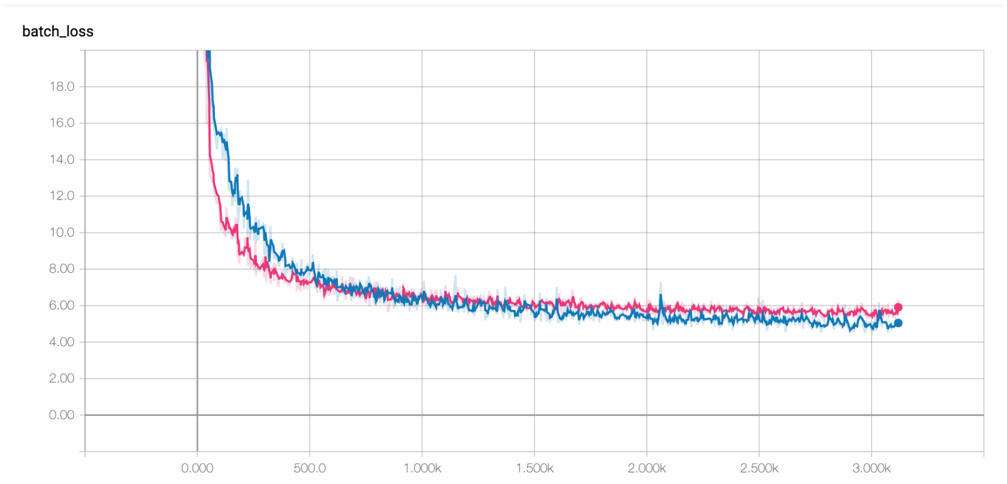


Figure 6: Blur with $r = 2.5$ causes similar loss of information as Pixelation with $n' = 32$ causes.

8 Conclusion

After experimenting with various models, we conclude that an autoencoder is able to capture the ground reality in an obfuscated image and reconstruct it back to the ground reality. The model performs optimally on pixelated and blurred images with satisfactory visual and structural similarity. The model does not perform as well on images with block obfuscation due to lack of any information that is required to reconstruct the missing parts of the image.

Future work can be done on the model to improve its performance on blocked images, perhaps by performing multiple forward passes on these images. Furthermore, the model can be trained using the two losses in conjunction:the MSE loss to capture low level differences, and perceptual poss to capture higher level visual differences in the images. Better evaluation criteria can also be devised to capture the visual similarity between two images in a way that aligns with how these images appear to the human eyes.