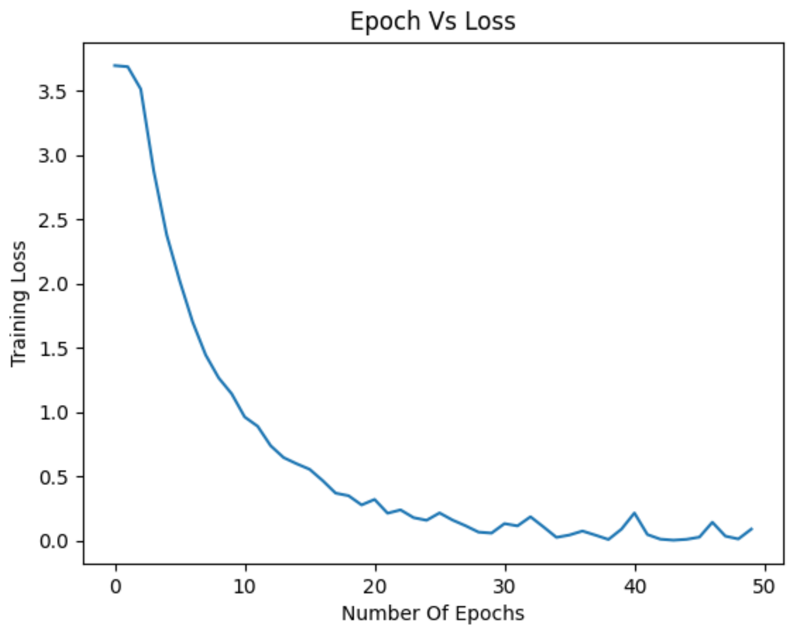
Report on Model Inversion Project

PET 2021

In model inversion attack the adversary uses a convolutional neural network model tries to invert the input data by using the labels. In our attack we use some face image data from AT&T faces dataset (https://git-disl.github.io/GTDLBench/datasets/att_face_dataset/) . The dataset contains 400 sample images where we can see 40 subjects each having 10 different images. All the images are in PGM format and in greyscale.

For our target model we split the dataset randomly in 90:10 ratio of training and testing dataset. Our target model consists of 3 layer convolutional neural network. The model use kernel size of 2 and filter size of 32,64,128. For training the target model we run 50 epochs with learning rate of 0.001. We used Adam optimizer and calculated the cross-entropy loss. After 50 epochs our loss was 0.004. We can see our training loss in the following image-

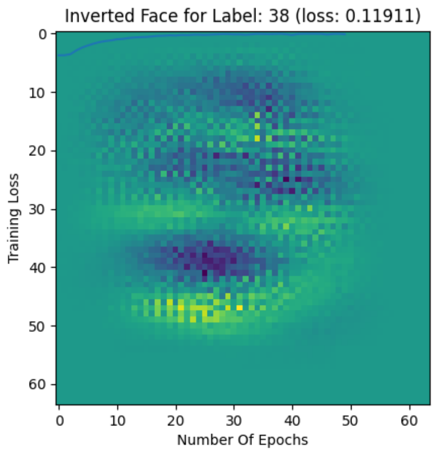


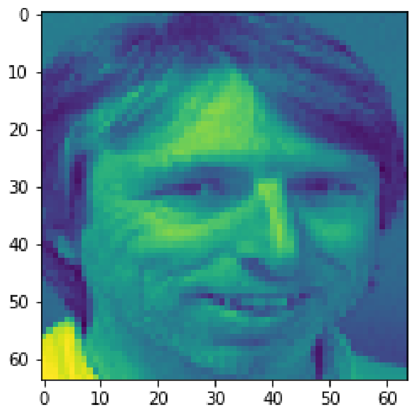
After training we used our testing data to evaluate the model. Using the AT&T faces dataset on our target model we achieved test accuracy of 90.00 %.

After that we built our attack model where we took our target model and set it to evaluation mode so no parameters are further updated. We then sent an empty image for a label/class and check the probability of that specific label from the model output using the softmax function. We use the output probability in our cost function for the attack model which is defined as: 1 – output_probability. Then we back propagate to update the input image so that future run of MI-FACE algorithm produces a smaller loss. We run 1000 epochs for our attack for each image. We can see some of the generated images for the specific labels and their loss below. The loss and the generated images could have been made better with the PROCESS function mentioned in the original paper. In our case, we have used a simple PROCESS function where in each iteration, after updating the image using back propagation, we simply normalize the image using the following formula:

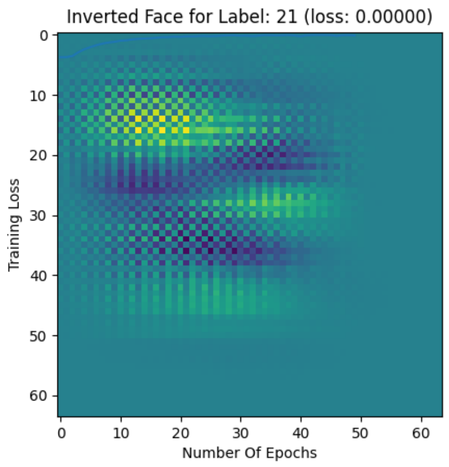
$$\frac{(image - torch.min(image))}{torch.max(image) - torch.min(image)}$$

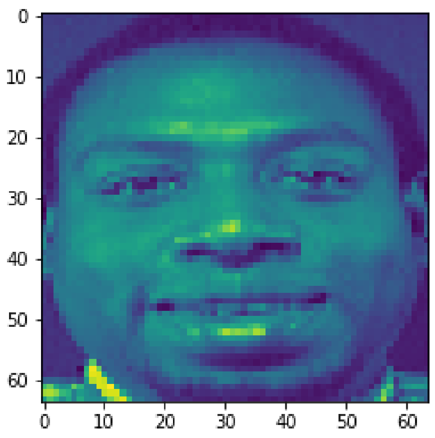
For label 38 we can see the following generated vs the input image,



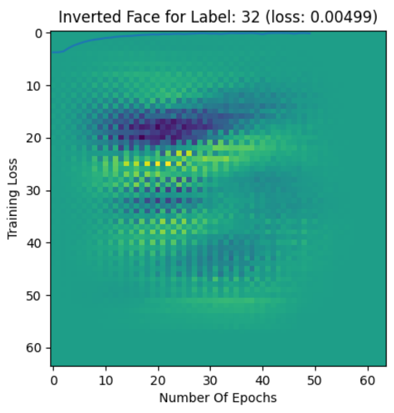


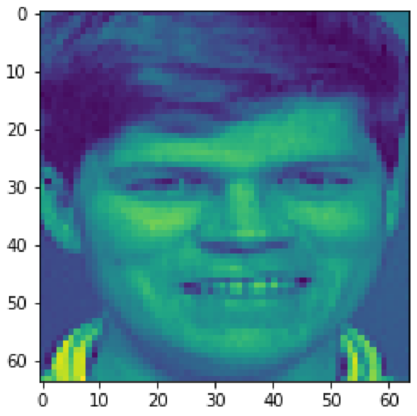
For image 21



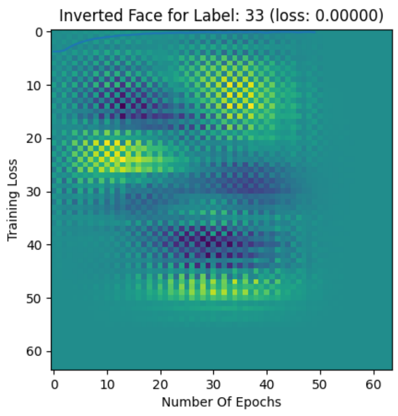


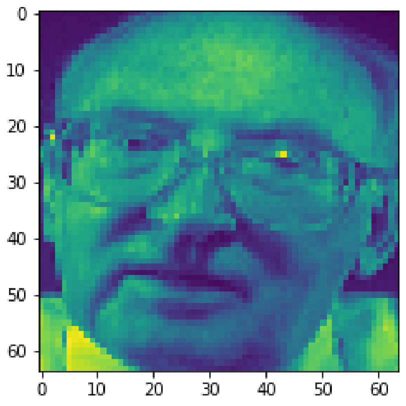
For image 32





For image 33





From the above examples we can see that we somewhat generated an input image containing some related features to the original input images. As we implemented very basic of the proposed algorithm for model inversion the generated images are not accurate. We can achieve more accurate images using more complete version of the algorithm.