Privacy Enhancing Technologies

Kazi Fozle Azim Rabi

Tanjim Ul Haque

Moritz Miodek

Summer 2021

Universität des Saarlandes

Introduction

In this project we a team of three students have attempted to apply different kinds of privacy attacks. The course was designed to get a better understanding on privacy attacks and its defenses. We put priority on 3 models and applied them on to specified dataset. After building and training our models successfully we designed a command line UI and incorporated the model's mechanism inside, so that the models can be accessible from a single window. We will discuss out process and results below –

Membership Inference

Attack Performance

Note: we use the CrossEntropyLoss function and a learning rate of 0.001 for all models.

Target Model

Since we don't have a target model, we have to train one ourselves. We do this by heavily overfitting the Conv2DNet network, it consists of 2 hidden convolutional layers followed by 1 hidden linear layer reducing the output of the 2nd Conv2d layer to 128 and finally the 10 output neurons. For this example, we ran it for 75 Epochs. The accuracy of the target model for training data is at 99.26% while it is only 98.52% for testing or validation data.

Shadow Model

Next we train our Shadow model. We use the exact same setup as for the Target Model with the difference that we stop training if we see our V-Loss is no longer improving, more specifically we stop if we see our V-Loss did not improve for 2 straight Epochs. In our case it stopped after 37 Epochs. The accuracy for the shadow model for training data lies at 98.72% while it is only 98.42% for testing or validation data. As one can see the target model is heavily overfitted when compared to the shadow model.

Attack Model Dataset

Now we need to create the Dataset. For this we have to evaluate both the target and shadow model with the same MNIST dataset distribution the posteriors we get from this are subtracted from another to form the input data for the attack model. The labels are derived from whenever the input image was part of the training=member or testing=non-member dataset used for the target model making the labels 100% accurate. We also use a 90%Train/10%Test split for this dataset.

Attack Model

The attack model is supposed to predict whenever an input, in our case an image from the MNIST dataset, was used to train the target model or not. Therefore we want to create a binary classifier, our AttackNet network consists of 10 inputs aka the output size of the target model followed by 4 fully connected linear hidden layers of size 1024 and finally the two output neurons using the log_softmax activation

function because of the CrossEntropyLoss function. Overall, this model achieved a V-Loss of 0.0599 while training and it trained for 78 Epochs.
The performance of the model can be measured around ~97% accuracy for test images and 97.5% for training images from the MNIST dataset.

Used Models

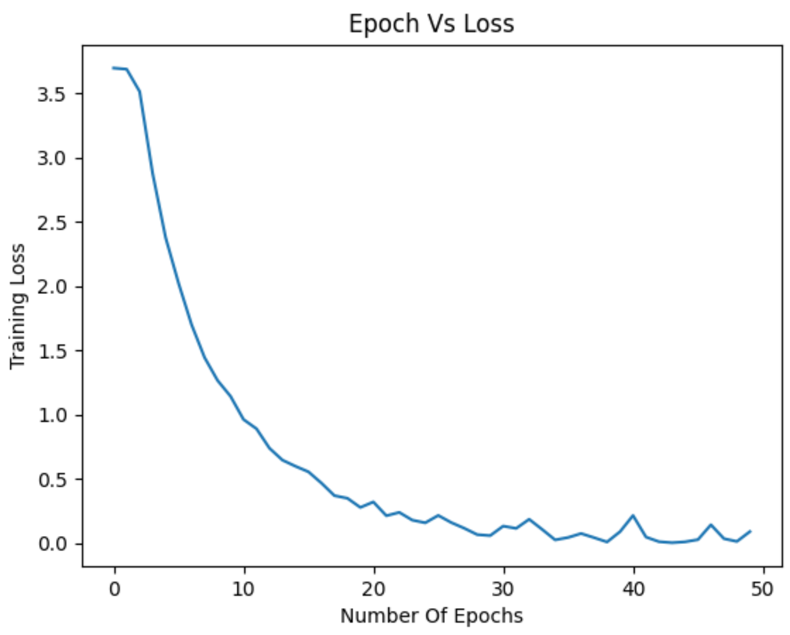
You can find the models used for this example in models/ named
target50.pth,
shadow37.pth
and attack0599.pth

Further Information

Further info’s about how the code works can be found in the README.md of the repo or looking at some comments in the code or executing it and reading the info/debug messages both in the terminal and code.

Model Inversion

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-

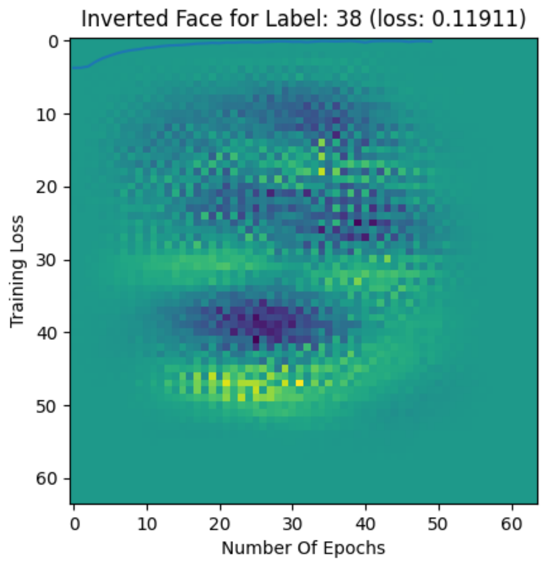


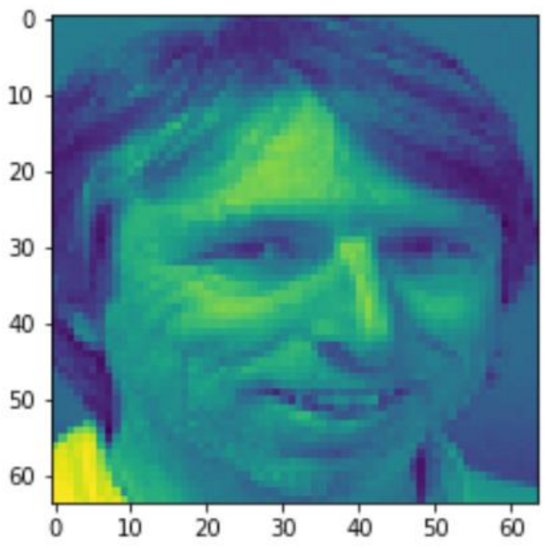
After training we used our testing data to evaluate the model. Using the AT&T faces dataset on our target model we achieved test accuracy of 90.00 %. After that we built our attack model where we took our target model and set it to evaluation mode so no parameters are further updated. We then sent an 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:

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

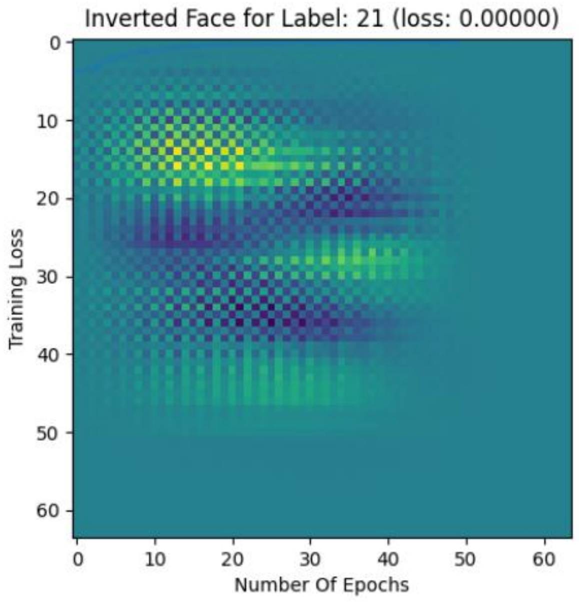
$$=$$

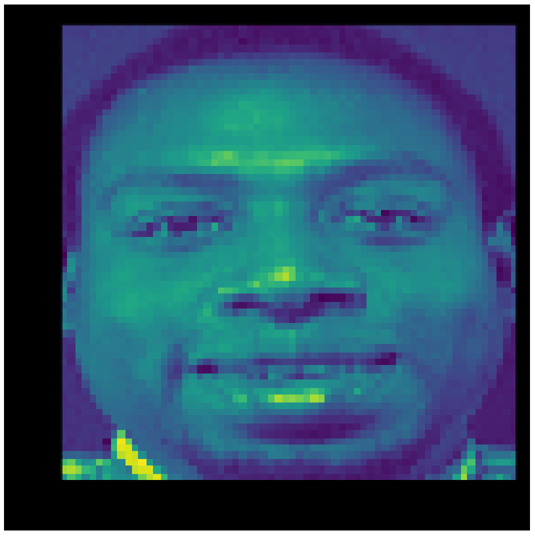
For label 38 we can see the following generated vs the input image,



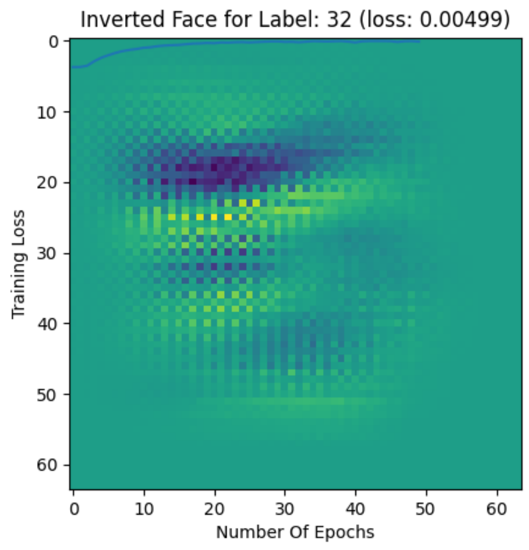


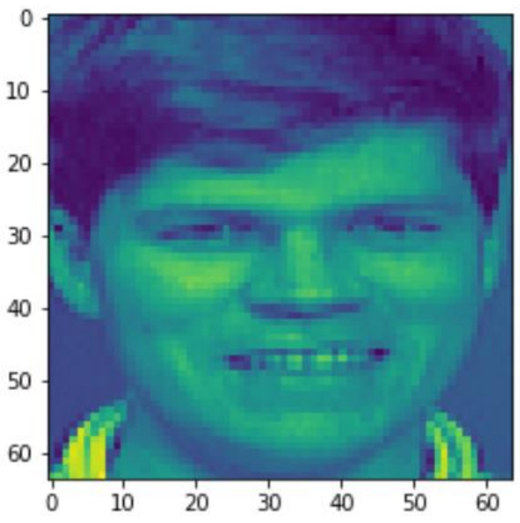
For image 21



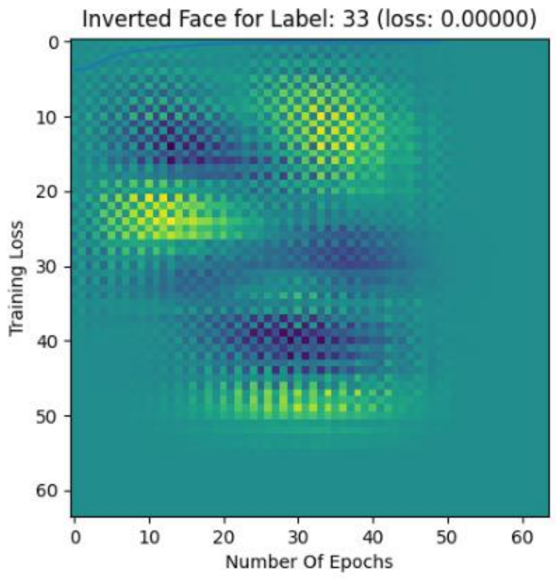


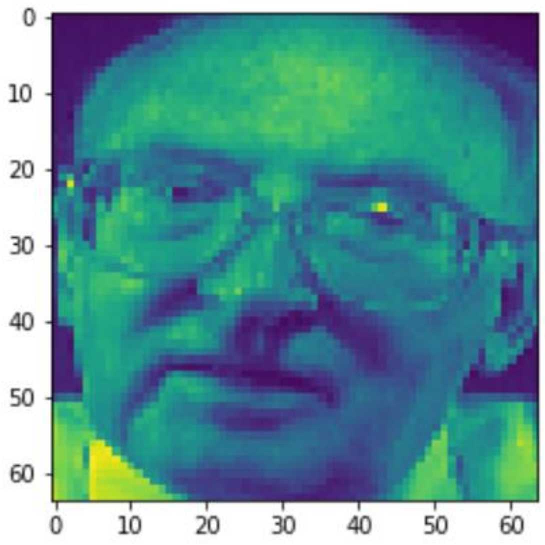
For image 32





For image 33

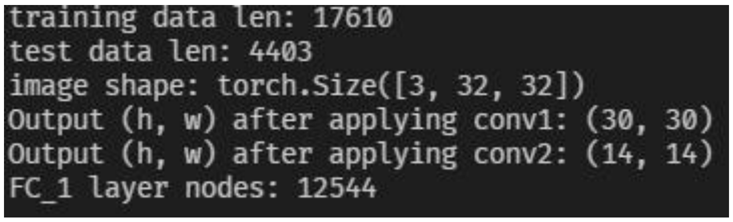




From the above examples we can see that we somewhat generated an input image containing some related features to the original input images. As we implemented very basic of the proposed algorithm for model inversion the generated images are not accurate. We can achieve more accurate images using more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

Attribute Inference

In attribute inference attack the adversary takes a white box model and train the model using shadow datasets. While training the shadow dataset it takes the embeddings of the training model which contains many information not only relevant to that model. What the adversary does is it overlearns a model and learn some property which are the embeddings and those are not related to original learning task. We then train the model using our shadow embeddings to infer some specific attribute which are not related to original model tasks.
For our proposed model running the script with the command "python main.py -AI -AI-dataset=UTKFace" will load all the face images from the UTKface dataset and randomly split it into a 80-20 training testing sets. It will use the training data to train a 2 layer CNN network to predict the race of the person in the image. The CNN structure can be seen here -



```
training data len: 17610
test data len: 4403
image shape: torch.Size([3, 32, 32])
Output (h, w) after applying conv1: (30, 30)
Output (h, w) after applying conv2: (14, 14)
FC_1 layer nodes: 12544
```

Then the model will be used to test the remaining images. After that we will train the attack model using the same training dataset. We first pass the images to the target model and get the embedding from the last fully connected layer which is then used to train the attack model where our labels are now gender. At the end of the training of the target model and the attack model, we plot the training losses for both, We can see the plot below-

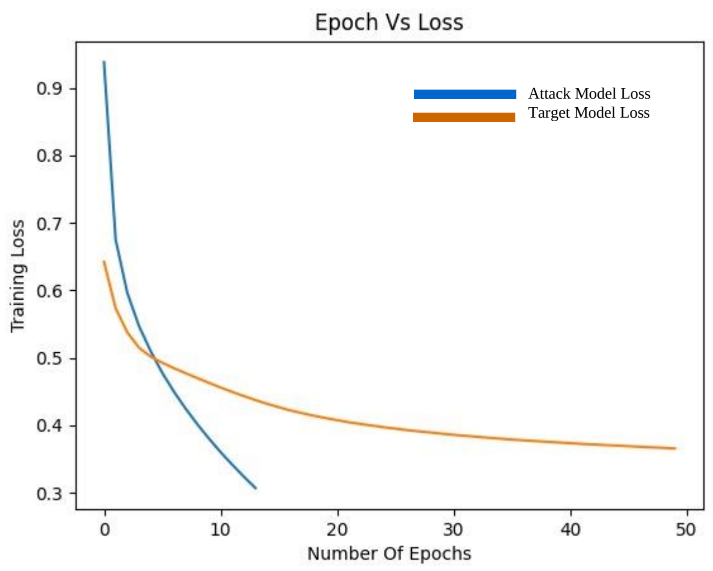
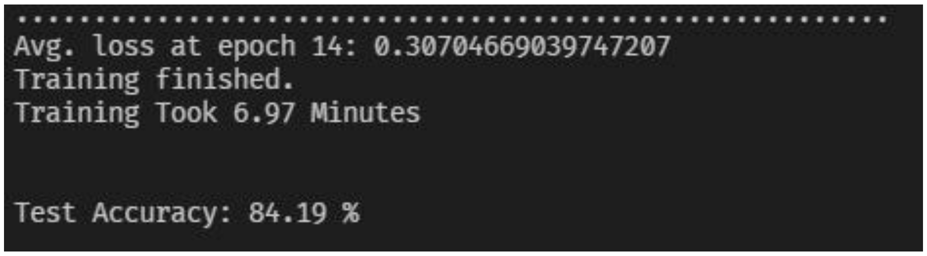


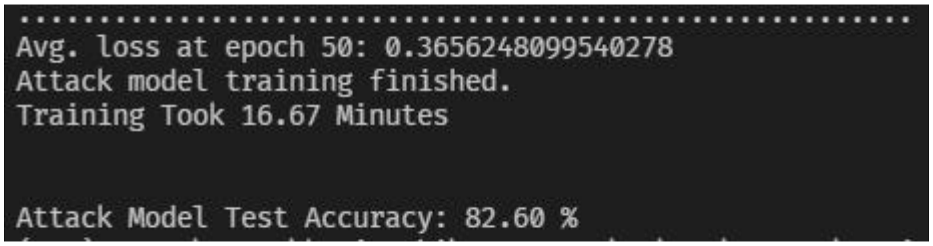
Figure: attack and target model training loss graph

Our model also provides the accuracy of attack model and the target model. Although our target model is labeling races for the face data and the attack model is labelling the Gender attribute but we are getting very good accuracy on both of them. Meaning an adversary can get specific properties and use it to infer a completely different attribute. We can see the accuracy of the target model and the attack model below –



```
..........................................................
Avg. loss at epoch 14: 0.30704669039747207
Training finished.
Training Took 6.97 Minutes


Test Accuracy: 84.19 %
```



```
.........................................................
Avg. loss at epoch 50: 0.3656248099540278
Attack model training finished.
Training Took 16.67 Minutes


Attack Model Test Accuracy: 82.60 %
```

Conclusion

So from this research we have achieved a in-depth understanding of the work mechanisms as well as theoretical overview of 3 attacks. We have successfully applied these models and achieved decent outputs.

References

[1] Shokri, Reza, et al. "Membership inference attacks against machine learning models." 2017 IEEE Symposium on Security and Privacy (SP). IEEE, 2017.

[2] Salem, Ahmed, et al. "Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models." arXiv preprint arXiv:1806.01246 (2018).

[3] Fredrikson, Matt, Somesh Jha, and Thomas Ristenpart. "Model inversion attacks that exploit confidence information and basic countermeasures." Proceedings of the 22nd ACM SIGSAC conference on computer and communications security. 2015.

[4] Song, Congzheng, and Vitaly Shmatikov. "Overlearning reveals sensitive attributes." arXiv preprint arXiv:1905.11742 (2019).