Privacy Enhancing Technologies

Ashish Khullar (7002903)
Matthias Zerweck (2572554)
Ole Keil (2564409)
Shreyash Arya (7015279)

Project: Phase-1 Submission
Task 1: Membership Inference
Task 2: Model Inversion

Task 1 - Membership Inference

1) Architecture:

Target model:

CNN based architecture with two convolution layers followed by batch normalization and max-pooling and one fully connected layer. We use ReLU activation for non-linearity.

No. of hidden units:
Conv1 = 128, Conv2 = 256, FC = 256. (CIFAR10)
Conv1 = 32, Conv2 = 64, FC = 64. (MNIST)

Shadow model:

Same architecture as target model with half the number of units in hidden and fully connected layers.

No. of hidden units:
Conv1 = 64, Conv2 = 128, FC = 128. (CIFAR10)
Conv1 = 16, Conv2 = 32, FC = 32. (MNIST)

Attack model:

MLP with one hidden layer and ReLU activation for non-linearity.

No. of hidden units: FC = 128.

2) Dataset:

- We have divided the original training dataset into four equal disjoint sets and assigned one each to target train and test, shadow train and test respectively.
- We use 1000 samples each from the target and shadow model testing dataset as the validation set.
- For pre-processing, we have normalized the CIFAR10 dataset with mean and standard deviation as [0.5, 0.5] and for the MNIST dataset, it is [0.1307, 0.3081].

Below are the details of the dataset splits:

CIFAR-10:

Total Train samples in CIFAR10 dataset: 50000
Total Test samples in CIFAR10 dataset: 10000

Number of Target train samples: 12500
Number of Target valid samples: 1000
Number of Target out samples: 12500

Number of Shadow train samples: 12500
Number of Shadow valid samples: 1000
Number of Shadow out samples: 12500

MNIST:

Total Train samples in MNIST dataset: 60000
Total Test samples in MNIST dataset: 10000

Number of Target train samples: 15000
Number of Target valid samples: 1000
Number of Target out samples: 15000

Number of Shadow train samples: 15000
Number of Shadow valid samples: 1000
Number of Shadow out samples: 15000

3) Hyperparameters:

Following are the hyperparameters used for the target, shadow and attack models. The values corresponding to these hyperparameters are the fine-tuned ones that we found to give the best results.

Shadow and Target Model:

No. of epochs (*num_epochs*) = 50
Batch size (*batch_size*) = 128
Learning rate (*learning_rate*) = 0.001
Learning rate decay (*lr_decay*) = 0.96
Regularization (*reg*) = 1e-4

Attack Model:

Number of epochs (*NUM_EPOCHS*) = 50

Batch size (*BATCH_SIZE*) = 10
Learning rate (*LR_ATTACK*) = 0.001
Regularization (*REG*) = 1e-7
Learning rate decay (*LR_DECAY*) = 0.96

* NOTE: The variables in the brackets are the ones as defined in the code.

4) Results:

- For target and shadow model training, we experimented with different hyperparameter (*batch size, learning rate and L2 regularizer*) values and found the ones defined above gives us the best-overfitted model relevant to the attack. These values were fixed to further fine-tune the attack model.
- Parameter initialization: We tested a custom weight initialization scheme (controlled by *param_init*) across the three models but didn't find any significant improvements to the attack results. Hence, we use the default parameter initializations.
- No. of posteriors: We tested with the three highest posterior probabilities (in descending order) as well as ten posteriors (output of the softmax from shadow model) as features to train our attack model. Based on multiple test runs, we found the choice on no. of posteriors doesn't have a significant effect on attack performance for CIFAR10. However, for MNIST, with the three highest posteriors, we found improvements in precision but the overall performance (measured using F1 scores) decreases.

Based on our extensive testing with different hyperparameter values for the attack model, we found *'batch size = 10', 'learning rate = 0.001'* and *'L2 regularization = 1e-7'* gave us the best attack results. (as shown in red in the table below)

CIFAR10

Table 1:

As the no. of training epochs for target and shadow model are increased (from 30 to 50), we observe that the attack performance improves which indicates increased overfitting levels and remembering of samples. (Case 3,4 compared to case 1,2)

*** For the attack model, the number of epochs = 50.**

Case	**No. of posteriors**	**No. of epochs**	**Target Model Accuracies (%)**	**Shadow Model Accuracies (%)**	**Attack Model Accuracies (%) and Metrics**
1	10	Target = 30 Shadow = 30	Train = 100 Test = 69.62	Train = 99.98 Test = 68.91	Train = 70.26 Test = 71.07 Precision = 0.78 Recall = 0.71

					F1 = 0.69
2	3	Target = 30 Shadow = 30	Train = 99.99 Test = 69.73	Train = 99.98 Test = 69.90	Train = 68.88 Test = 69.90 Precision = 0.79 Recall = 0.70 F1 = 0.67
3	10	Target =50 Shadow = 50	Train = 100 Test = 70.18	Train = 100 Test = 69.17	Train = 73.14 Test = 74.12 Precision = 0.81 Recall = 0.74 F1 = 0.73
4	3	Target =50 Shadow = 50	Train = 100 Test = 69.76	Train = 100 Test = 68.72	Train = 71.66 Test = 72.55 Precision = 0.81 Recall = 0.73 F1 = 0.71

Table 2:

The results shown below correspond to the attack performance by using less overfitted target and shadow models. To reduce overfitting we use dropout (50%) and early stopping.

We observe degradation in the attack performance when the target and shadow models are not overfitted and hence become less prone to membership inference attacks.

*** No. of epochs for target, shadow and attack models = 50.**

Case	**No. of posteriors**	**Target Model Accuracies (%)**	**Shadow Model Accuracies (%)**	**Attack Model Accuracies (%) and Metrics**

1	10	Train = 90.95
		Test = 65.23
			Train = 79.05
			Test = 64.85
				Train = 53.51
				Test = 50.64
				Precision = 0.51
				Recall = 0.51
				F1 = 0.50

2	3	Train = 80.45
		Test = 62.60
			Train = 87.12
			Test = 63.38
				Train = 53.26
				Test = 51.48
				Precision = 0.52
				Recall = 0.51
				F1 = 0.51

<u>MNIST</u>

Table 3:

Here we can see no significant improvements with no. of epochs as MNIST data seems to be simple enough to be learnt even with the 30 epochs (Case 1 and 2) as compared to 50 epochs (Case 3 and 4).

No. of posteriors changed the precision value in the case of the top 3 but reduced the F1 score which gives the overall idea about the model (classification and misclassification). So, if our task is just only considered with precision, we can go with the top 3 posteriors but if we overall want to get better performance (in terms of both precision and recall), we should go with the top 10 posteriors. [REFERENCE]

*** For the attack model, the number of epochs has been fixed to 50.**

Case	**No. of posteriors**	**No. of epochs**	**Target Model Accuracies (%)**	**Shadow Model Accuracies (%)**	**Attack Model Accuracies (%) and Metrics**
1	10	Target = 30 Shadow = 30	Train = 100 Test = 98.88	Train = 100 Test = 98.56	Train = 51.31 Test = 51.48 Precision = 0.52 Recall = 0.51 F1 = 0.51
2	3	Target = 30 Shadow = 30	Train = 100 Test = 98.86	Train = 100 Test = 98.55	Train = 52.06 Test = 52.18 Precision = 0.75 Recall = 0.52 F1 = 0.38
3	10	Target =50 Shadow = 50	Train = 100 Test = 98.77	Train = 100 Test = 98.71	Train = 51.62 Test = 51.58 Precision = 0.52 Recall = 0.52 F1 = 0.51
4	3	Target =50 Shadow = 50	Train = 100 Test = 98.77	Train = 100 Test = 98.63	Train = 52.49 Test = 52.86 Precision = 0.75 Recall = 0.53 F1 = 0.39

Table 4:

For completeness, we also applied the dropout (50%) and early stopping to MNIST dataset to verify the results. Since the dataset is simple for the model to learn therefore the overfitting levels are close to negligible. Hence applying dropouts and early stopping didn't have any impact on the attack performance.

*** No. of epochs for target, shadow and attack models = 50.**

Case	**No. of posteriors**	**Target Model Accuracies (%)**	**Shadow Model Accuracies (%)**	**Attack Model Accuracies (%) and Metrics**

1	10	Train = 99.31
		Test = 98.55
			Train = 98.35
			Test = 97.11
				Train = 51.32
				Test = 50.83
				Precision = 0.51
				Recall = 0.51
				F1 = 0.50

2	3	Train = 99.19
		Test = 98.65
			Train = 98.40
			Test = 97.04
				Train = 50.71
				Test = 50.07
				Precision = 0.50
				Recall = 0.50
				F1 = 0.37

Task 2 - Model Inversion

1) Dataset:

ATnT Face Dataset with 3 of 10 images per class as the validation set.

2) Target Model:

Similar to Fredrikson et. al. we are using an MLP with a single hidden layer with 3000 neurons and a sigmoid as an activation function.

Loss Function: CrossEntropyLoss
Optimizer: Adam

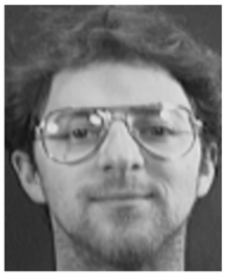
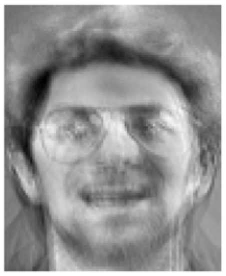
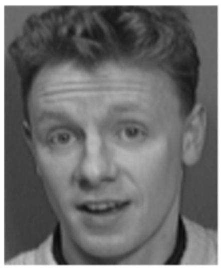
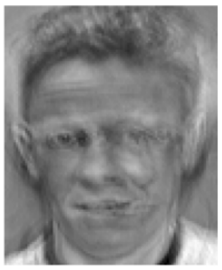
Hyperparameters: Batch Size: 64, Epochs: 30

3) Training Results:

Train Loss: 0.003, Validation Loss: 0.154 | Train Accuracy: 100.00%, Validation Accuracy: 95.31%

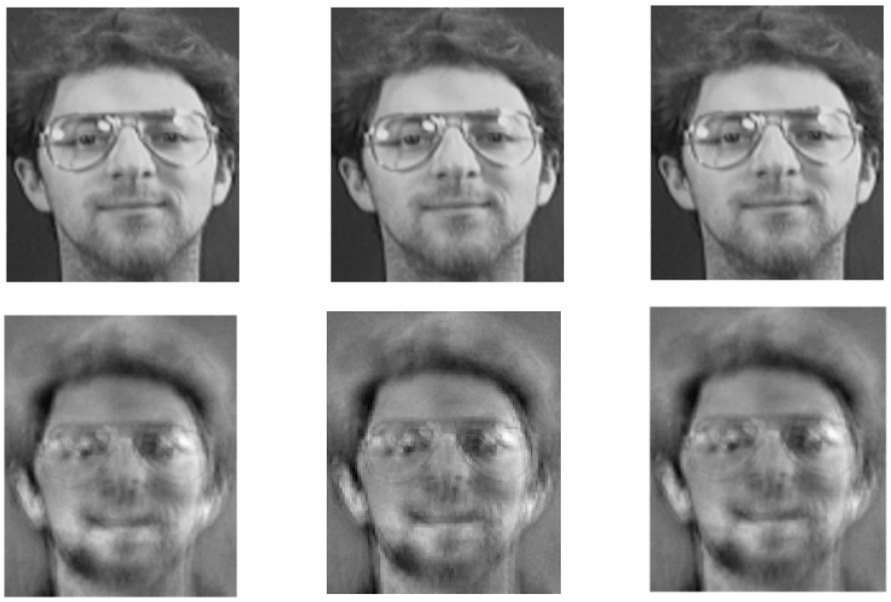
4) Results of the Face Reconstruction Attack:

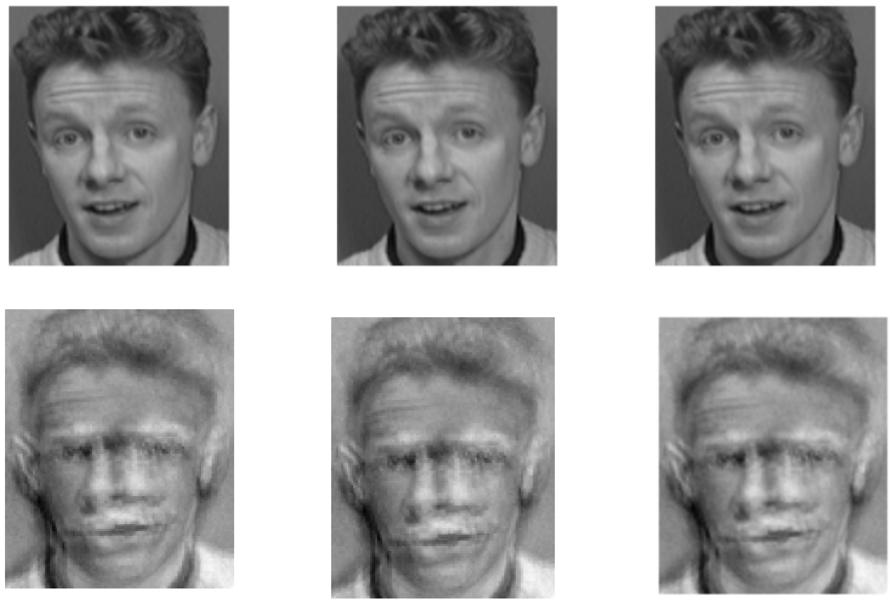
From Fredrikson et. al (~5 iterations):

Our results with 5, 10, 100 iterations of mi_face respectively:

Gradient Step: 0.1, Loss function: CrossEntropyLoss

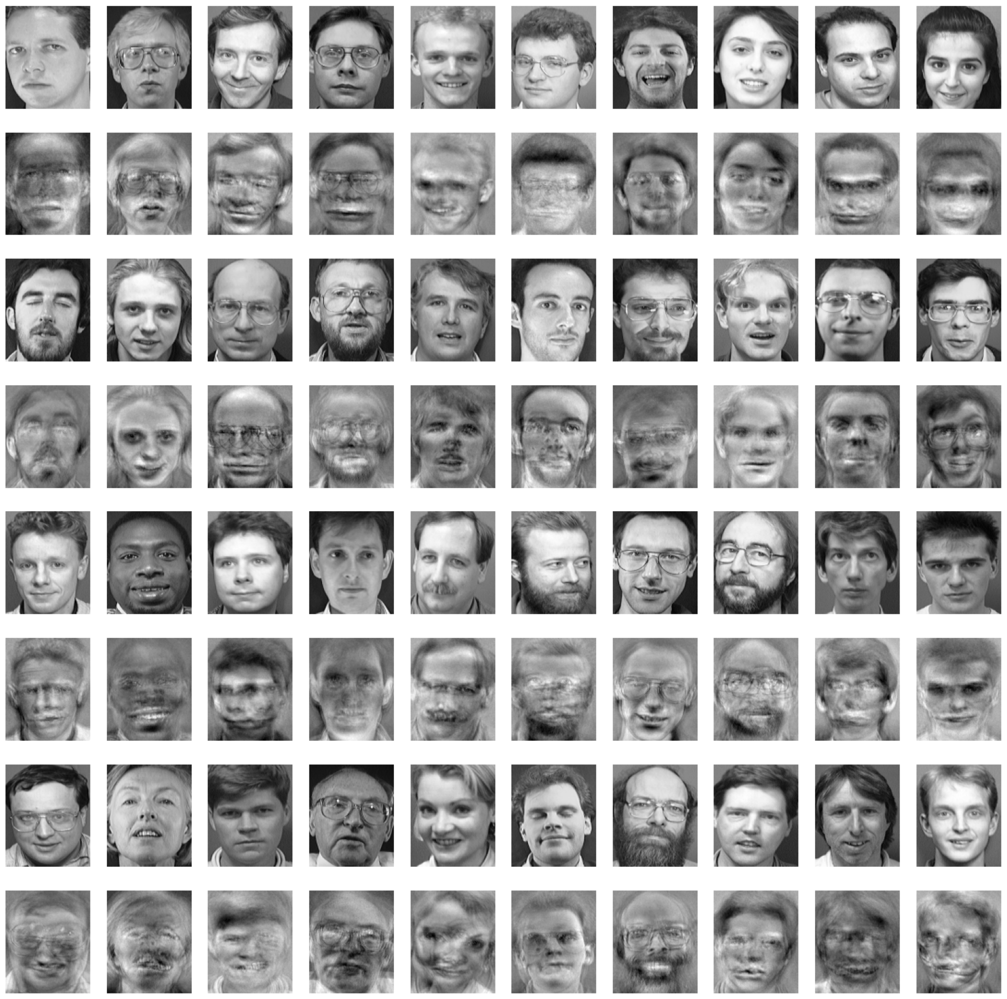




The results for 5, 10 and 100 iterations from left to right shows that the images get sharper. Also, the change in pixel values changes with the different numbers of iterations.

Results on all the images in the dataset:

Images reconstructed with 100 iterations of mi_face. Find the reconstruction below the respective original.



Task 3 - Attribute Inference

Below are the results and inferences of our attribute inference attack, more precisely micro property inference attack. We are attacking a target model which is trained to predict the gender of a person. The micro property we try to infer is the race of a person.

1) Architecture:

Target model:

CNN architecture is a variant of LeNet [1] with three 3×3 convolutional and 2×2 max-pooling layers with 16, 32, and 64 filters, followed by two FC layers with 128 and 64 hidden units as described by Song et al. [2]. We use Tanh activation.

Conv1 = in_channels=3, out_channels=16, kernel_size=3, stride=1,
Conv2 = in_channels=16, out_channels=32, kernel_size=3, stride=1,
Conv3 = in_channels=32, out_channels=64, kernel_size=3, stride=1,
FC1 = 128 hidden units,
FC2 = 64 hidden units.

Attack model:

MLP with two hidden layers and Tanh activation, similar to FC layers of the target Model.

No. of hidden units:
FC1 = 128,
FC2 = 64.

2) Dataset:

- As recommended we are using the UTKFace dataset[3].
- We randomly split the dataset using 80% of the samples for target model training and 20% for target model testing.
- We then use 50% of the target model training samples to train the attack model. We use the same 20% of the dataset to test the attack model.
- The data samples are resized to a dimension of 50*50 and normalized with (0.5, 0.5, 0.5), (0.5, 0.5, 0.5).

Below are the details of the dataset splits:

UTKFace:

Total samples in **UTKFace** dataset: 20606 (Actually 20608 but 2 samples labelling is not valid)

Number of Target train samples: 16485
Number of Target test samples: 4121

Number of Attack train samples: 8243
Number of Attack test samples: 4121

3) Hyperparameters:

Following are the hyperparameters used for the target and attack models. The values corresponding to these hyperparameters are based on Song et al. [2].

Target Model:

Number of epochs = 30
Batch size = 128
Learning rate = 0.001

Attack Model:

Number of epochs = 50
Batch size = 128
Learning rate = 0.001

4) Results:

Target Model:

After 30 epochs of training, the target model has an accuracy of 90%.

Attack Model:

The attack model performs with an accuracy of 56% after 50 epochs of training. Even though this is not a very high accuracy for a classifier, it is still significantly higher than random guessing based on the class distribution, which is only 42.52%.

Accuracy per class:

Class	**Accuracy in %**
Class 0	79.1
Class 1	65.8

Class 2	33
Class 3	26.4
Class 4	3.4

In contrast to classes 0 and 1, classes 2, 3, 4 have low accuracy. If we take a look at the dataset, we note that the classes are not equally distributed.

Class	**Number of samples in UTKFace**
Class 0	8950
Class 1	4367
Class 2	2497
Class 3	3484
Class 4	1308

Classes 2, 3, and especially 4 are underrepresented in the dataset. This could be a reason for the lower accuracy, but it's most likely not the only nor the most important one.

Let's take a closer look at class 4. The labels are denoted, from 0 to 4, White, Black, Asian, Indian, and Others (like Hispanic, Latino, Middle Eastern). Class 4 being denoted with others, thus fewer samples with similar features that can be learned could be a reason for its bad performance.

References for Attribute Inference:

[1] LeCun, Yann, et al. "Backpropagation applied to handwritten zip code recognition." *Neural computation* 1.4 (1989): 541-551.

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

[3] https://susanqq.github.io/UTKFace/