Supplementary Material for DISCO: Dynamic and Invariant Sensitive Channel Obfuscation for deep neural networks

Abhishek Singh[1], Ayush Chopra[1], Ethan Garza[1], Emily Zhang[1], Praneeth Vepakomma[1], Vivek Sharma[1,2], Ramesh Raskar[1]

[1] Massachusetts Institute of Technology, [2] Harvard Medical School

Appendices

A. Hyper-parameters and Experimental Setup

All of the experimental setup is implemented in PyTorch and we will be releasing the codebase for all of different quantitative and qualitative experiments, with the random seeds used in all of the experiments.

Network architecture: We describe four distinct networks in the section 3, *client network*, *filter generating network*, *adversary network*, *task network*. We use ResNet-18 [1] as the base architecture for all of the four networks. For alignment of the architecture we experiment with the different blocks of the ResNet architecture and split the network such that output of the *client network* is fed to all three *filter generating network*, *adversary network*, and *task network*. The *filter generating network* has same number of neurons in the final fully connected layer as number of channels in the output produced by *client network*. The sigmoid temperature is 0.03 for the filter generating network. We adapt the ResNet backbone for *adversary network* when the protected attribute is sensitive input since it requires to build a generative model conditioned on *client activations*. We use a transpose convolution based architecture that upsamples the feature map to a higher dimensionality resulting in final image.

Pre-processing module described in the section 3.3.a is composed of a single convolution layer and a *spatial decoupler* that splits the feature-map into d^2 spatially disjoint partitions. For an image size of 112 and target d^2 to be 64, the resulting featuremap size is 14×14 that gets rescaled back to 112×112 using bilinear interpolation. We keep the value of the d^2 as 64 to make sure that the averaging in the channel space results in 64 distinct feature maps that can be fed into the remaining of the architecture, this allows compatibility of the *pre-processing module* with off the shelf architectures.

Optimizer: We use SGD optimizer with momentum [2] for all of the networks with a learning rate of 0.01

B. Generalization

The setup described in the main text is as follows for optimizing the parameters -

$$\min_{\phi} \left[\max_{\theta_3} -L_{priv}(\theta_1, \phi, \theta_3) + \min_{\theta}(-\rho \max_{w} -L_{util}(\theta_1, \phi, \theta_2)) \right]$$

where $\theta_1, \phi, \theta_2, \theta_3$ are the parameters of the *filter generating network*, *client network*, and *server network* respectively. Let $\theta_1^*, \phi^*, \theta_2^*, \theta_3^*$ be the solution for the parameters we obtain by minimizing the expected loss. Let $\hat{\theta}_1, \hat{\phi}, \hat{\theta}_2, \hat{\theta}_3$ refers to the empirical minimizer of the above mentioned joint optimization. As noted before, we adapt to the setup described by Hamm [3]. However, a significant difference lies in the fact that θ_1 is not trained to minimize L_{priv} as this is to improve generalization of the ϕ across a different set of θ_1. The remaining parameters remain analogous to the min-max filters described in [3] .Following on that, we describe the joint loss as follows

$$L_J(\theta_1, \phi, \theta_2, \theta_3) = L_{util}(\theta_1, \phi, \theta_2) - \rho L_{priv}(\theta_1, \phi, \theta_3)$$

Let D be the original unknown data distribution and S be a set of samples obtained from the true distribution for calculating empirical loss then the empirical and expected loss can be bounded as follows, giving a generalization bound.

$$|E_D(L_J(\theta_1^*, \phi^*, \theta_2^*, \theta_3^*)) - E_S(L_J(\hat{\theta}_1, \hat{\phi}, \hat{\theta}_2, \hat{\theta}_3))| \leq 2 \sup_{\theta_1, \phi, \theta_2, \theta_3} |E_D(L_J(\theta_1, \phi, \theta_2, \theta_3)) - E_S(L_J(\theta_1, \phi, \theta_2, \theta_3))|$$

For more details, we refer the reader to the proof of theorem 1 shown in [3]. The equation above gives the bound on generalization error.

C. Effect of channel pruning on mutual information

We now study the effect of applying channel pruning of activations at the output of the client network with regards to the mutual information between the raw sample and the pruned activations. Inspired by the theoretical analysis in [4], we extend and adapt it to our setup of analyzing the reduction in mutual information between the *sensitive input* and *client activations* upon performing random pruning. We use the superscript notation $f_1^k(\theta_1^k; x)$ to denote the output of k'th layer of client network. We compare this with regards to no pruning and random pruning at the k'th layer of the client network as shown below.

Pre-pruning: The negative of the mutual information between the raw data and the output of 1'st layer prior to applying the pruning is given by

$$\begin{aligned} -\mathcal{I}(x; f_1^1(\theta_1^1; x)) &= -\mathcal{H}(f_1^1(\theta_1^1; x)) - \mathcal{H}(f_1^1(\theta_1^1; x)|x) \\ &= -\mathcal{H}(f_1^1(\theta_1; x)) \end{aligned}$$

as $-\mathcal{H}(f_1^1(\theta_1; x)|x) = 0$, due to $f_1^1(\cdot)$ being a deterministic function. Upon applying the data processing inequality, we have that the mutual information between the output of the k'th layer and the raw data satisfies:

$$\mathcal{I}(x; f_1^k(\theta_1^k; x)) \leq \mathcal{I}(x; f_1^{k-1}(\theta_1^k - 1; x)) \leq \ldots \leq \mathcal{I}(x; f_1^1(\theta_1^1; x))$$

where, we have the following relation $\mathcal{I}(x; f_1^k(\theta_1^k; x)) = \mathcal{H}(f_1^k(\theta_1^k; x))$.

Post-pruning: The mutual information after random pruning can be represented as a multiplication of the outputs at the k'th layer with a Bernoulli random variable $\mathcal{P}$ as $\mathcal{I}(x; f_1^k(x, \theta_1^k).\mathcal{P})$. In addition to the form of data processing inequality used in analysis of pre-pruning; there is an equivalent form of the classical data processing inequality given by

$$-\mathcal{I}(x; f_1^k(x, \theta_1^k).\mathcal{P}) \geq -\mathcal{I}(f_1^k(x, \theta_1^k); f_1^k(x, \theta_1^k).\mathcal{P})$$

Upon expanding this upper bound using entropy terms we get

$$\mathcal{I}(x; f_1^k(x, \theta_1^k).\mathcal{P}) \leq \mathcal{H}(f_1^k(\theta_1^k; x)) - \mathcal{H}(f_1^k(\theta_1^k; x)|f_1^k(\theta_1^k; x).\mathcal{P})$$

But $\mathcal{H}(f_1^k(\theta_1^k; x))$ is the mutual information in the case of pre-pruning as analyzed above. Therefore the decrease in information about raw data post-pruning is given by the term $\mathcal{H}(f_1^k(\theta_1^k; x)|f_1^k(\theta_1^k; x).\mathcal{P})$. Upon applying the Bayes rule (for conditional entropy), this term exactly equals:

$$\mathcal{H}(f_1^k(\theta_1^k; x).\mathcal{P}|f_1^k(\theta_1^k; x)) + \mathcal{H}(f_1^k(\theta_1^k; x)) - \mathcal{H}(f_1^k(\theta_1^k; x).\mathcal{P})$$

Since the term $f_1^k(\theta_1^k; x)$ is independent of the noise $\mathcal{P}$, the above can be further rearranged as

$$\mathcal{H}(f_1^k(\theta_1^k; x).\mathcal{P}|f_1^k(\theta_1^k; x)) + \mathcal{H}(f_1^k(\theta_1^k; x)) - \mathcal{H}(f_1^k(\theta_1^k; x)) - \mathcal{H}(\mathcal{P})$$

which simplifies to $\mathcal{H}(f_1^k(\theta_1^k; x).\mathcal{P}|f_1^k(\theta_1^k; x)) - \mathcal{H}(\mathcal{P})$. As we chose $\mathcal{H}(\mathcal{P})$ to be a Bernoulli random variable; upon considering its success probability to be p (lower-case) and probability of failure to be $q = 1 - p$, we have -$\mathcal{H}(\mathcal{P}) = plog(p) + qlog(q)$. Therefore, upon performing random pruning the decrease in mutual information amounts to

$$\mathcal{H}(f_1^k(\theta_1^k; x).\mathcal{P}|f_1^k(\theta_1^k; x)) - \mathcal{H}(f_1^k(\theta_1^k; x)) + plog(p) + qlog(q)$$

while the mutual information post-pruning is upper bounded by $\mathcal{H}(f_1^k(\theta_1^k; x).\mathcal{P}|f_1^k(\theta_1^k; x)) + plog(p) + qlog(q)$.

D. Reconstruction Results

We present more reconstruction results for the qualitative comparison. Our results indicate that supervised decoder based attack model performs significantly better than likelihood maximization attack for *DISCO*, however, for all other techniques, likelihood maximization attack provides much better reconstruction quality. The figure can be found on the next page.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. CoRR, abs/1512.03385, 2015.

[2] Ning Qian. On the momentum term in gradient descent learning algorithms. Neural networks, 12(1):145–151, 1999.

[3] Jihun Hamm. Minimax filter: Learning to preserve privacy from inference attacks. Journal of Machine Learning Research, 18(129):1–31, 2017.

[4] Fatemehsadat Mireshghallah, Mohammadkazem Taram, Prakash Ramrakhyani, Dean M. Tullsen, and Hadi Esmaeilzadeh. Shredder: Learning noise to protect privacy with partial DNN inference on the edge. CoRR, abs/1905.11814, 2019.

Figure 1: Qualitative comparison for different techniques using the supervised decoder attack described in the Section 3.1. *DISCO (Off)* refers to DISCO with pre-processing module's toggle turned off. This technique results in a different yet realistic reconstruction for even *DISCO* compared to deep image prior results shown in the Figure 3.