Transfer Learning in Property Inference Attacks

Semester Project Report - Spring 2021

Christian Knabenhans
Student - MSc Cyber Security
ETHZ, EPFL

Supervised by
Theresa Stadler
SPRING Lab, EPFL

***Abstract*—Collaborative learning enables participants to train a joint machine learning model without explicitly revealing their private training data, and is thus a natural candidate for tasks requiring sensitive data.**

However, it has been shown that updates observed during the learning process of a source task leak unintended information about the clients' private data, which allows an adversary to infer target properties of the clients' private data. We replicate such attacks for a wide range of target properties on a dataset of face images using two different models, and explore the variation of the privacy loss depending on the source.

Finally, we use transfer learning to compute affinity measures between source and target tasks, and show that they are good predictors for the privacy loss, particularly for completely unrelated source and target tasks.

I. Introduction

Deep learning models are known for their capacity to encode complex feature representations of their training data. This makes them a likely target for property inference attacks that aim to infer properties of the training data that are seemingly unrelated with the main learning objective.

Preliminary work shows that a passive adversary that either has direct access to the network weights [1] or infers the model's feature activations from gradient updates [2] can use the learned features to infer training data properties. This inference constitutes a clear privacy breach if the inference is specific to the training data and considered orthogonal to the learning task. A privacy-preserving model should only leak information that characterizes data records by their class label.

To combat inference attacks on unintended features learned by deep network models, previous work suggests training models under a *least-privilege principle*, i.e., learning only the features relevant to a given task [3], [4], [5].

While this should reduce unintended information leakage about sensitive data properties, it is unclear whether it is actually possible to learn separate feature representations for all combinations of learning and target tasks, i.e., whether a model that is trained to perform well on a specific learning task does not inevitably capture features that can be exploited by adversaries to infer data properties considered independent of the learning objective.

This unintended feature learning is successfully being exploited in transfer learning, a technique where a machine learning model trained on one task is re-used as a basis to train a model for another task, achieving very good performance with comparatively little computation.

Transfer learning has been used to computationally derive *affinities* between visual tasks [6], where two tasks are said to have high affinity if one can effectively train a transfer model from one to the other. Task affinity thus seems a suitable candidate to estimate the information about a target task that is encoded by a model trained on a source task.

In this project, we explore whether a similar computational approach to learning task affinities can be applied to anticipate and quantify unexpected information leakage (measured as the success of property inference attacks) in deep learning models.

This report is organized as follows: we first introduce some background concepts (Section II), present our attacker model and attack mechanism (Section III), as well as the dataset and model architectures used in our experiments (Section IV). We then show the results of our attack experiments in Section V, present task affinities (Section VI, and examine how they can be used to predict privacy loss in Sections VII-VIII. Finally, we discuss limitations and future work directions (Section IX) and conclude (Section X)

II. Background

A. Machine Learning

1) Models: A machine learning model is a function $f_\theta : \mathcal{X} \to \mathcal{Y}$ parametrized by a set of parameters θ.

In this project, we focus on the supervised learning of classification tasks, which requires a labelled dataset of pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$. We work with binary classifiers for images, i.e., our models map images to binary labels in $\{0, 1\}$.

Neural networks are a widely used type of machine learning models composed of layers of "neurons". Fully connected layers matrix-multiply their input with their parameters, while convolutional layers apply a convolution instead. Deep neural networks are neural networks that are composed of many layers, and which are known for their ability to encode complex features of their inputs. Convolutional neural networks (CNNs) are a specialized type of neural networks that use at least one

Submitted on June 11th, 2021 as a partial requirement for a semester project in cyber security.

I would like to thank Theresa Stadler for her supervision, advice, and guidance throughout this project. I would also like to express my gratitude to Prof. Carmela Troncoso and the SPRING Lab at EPFL for the opportunity and ressources provided to work on this project.

convolutional layer, and that have in particular proved to be very effective for visual tasks.

2) Training: Machine learning models can be trained by minimizing a loss function $\mathcal{L}(\cdot)$, which measures how bad a model is at predicting outputs from inputs. Stochastic Gradient Descent (SGD) iteratively minimizes the loss function by modifying the parameters θ along the direction of the steepest descent $\nabla_\theta \mathcal{L}(b)$, as estimated over a batch b of training data. SGD is often parametrized by a learning rate η that controls to what extent the parameters are updated, but there are more sophisticated versions that update the parameters using heuristics. The parameters used to control the training procedure (like the learning rate) are called hyperparameters.

Transfer learning is a supervised learning technique that reuses weights of a pre-trained model to train a new model on a different but related task. For a suitable choice of tasks and pre-trained model, transfer learning allows to train a new model much more efficiently than training it from scratch.

B. Collaborative Learning

Collaborative learning is a technique that trains a machine learning model across multiple parties holding local datasets, without exchanging them. Collaborative learning enables multiple parties to build a joint model without sharing data, and is thus attractive for tasks requiring sensitive or private data.

In this project, we focus on synchronized SGD, which is shown in Alg. 1 (adapted from [2]).

In every training round, each client gets the current model from the server, computes gradient updates from its local data, and sends the updates to the server. The server then applies SGD on the parameters using the aggregated updates from all clients, and starts a new training round.

Algorithm 1 FedSGD: synchronized, federated SGD

Input: Clients $c_1, \ldots, c_K$, learning rate η, loss function $\mathcal{L}$
Output: Joint model parameters θ_T
  Server initializes θ_0
  for $t = 0$ to $T - 1$ **do**
    for each client c_k **do**
      Client samples a batch b from its training set $\mathcal{D}^k_{\text{train}}$
      Client computes local gradients $g^k_t \leftarrow \nabla_{\theta_t} \mathcal{L}(b)$
      Client sends g^k_t to the server
    end for
    Server updates model parameters $\theta_{t+1} \leftarrow \theta_t - \eta \sum_k g^k_t$
    Server sends updated model θ_{t+1} to all clients
  end for

Federated learning with model averaging is a similar technique to FedSGD, where client perform update their local model parameters in several rounds before sending their updated models to the server. The server then averages all client model updates to obtain the new joint model. For the sake of more efficient experiments and a simplified analysis, we do not consider federated learning with model averaging for this project.

III. Property Inference Attacks

A wide variety of privacy concerns and definitions have been identified and discussed for machine learning. In this project, we focus on *property inference*, and refer to [7] for an extensive discussion of other privacy risks in machine learning.

A. Threat model

For a machine learning model trained on a source task, the goal of a property inference adversary is to infer properties that are true for a subset of the training inputs, and that are unrelated to the source task (in the following, we call such properties target tasks). In this project, we focus on *batch property inference* (as defined in [2]), where the adversary's goal is to determine if an observed update is based on batch without or without a given target property.

While the adversary does not have access to the clients' training data, it has access to an *auxiliary dataset* $\mathcal{D}^{\text{adv}}$, annotated for both source and target tasks. For well-studied tasks (e.g., classification of images or text), such datasets exist and are publicly available. In our binary classification setting, we write $\mathcal{D}^{\text{adv}}_+$ for the auxiliary data with positive labels for the target task, and $\mathcal{D}^{\text{adv}}_-$ for the auxiliary data with negative labels.

This threat model covers collaborative learning with a central aggregation server, decentralized learning (e.g., using peer-to-peer networks), as well as local training. In the following, we focus on a centralized collaborative learning scenario.

Our adversary is passive, and is not able to interfere with an FedSGD run and to modify its output, but it observes sequential model updates θ_t during training. The adversary could gather such observations by eavesdropping on the server, on the client, or on the communication between them.

B. Attack mechanism

During training, gradients are computed using backpropagation, an optimization which allows gradients to be computed efficiently in a single pass from the last layer to the first.

For sequential, fully-connected layers h_l, h_{l+1} with $h_{l+1} = W_l \cdot h_l$ for a weight matrix W_l, the gradients at layer l are given by $\frac{\partial \mathcal{L}}{\partial W_l} = \frac{\partial \mathcal{L}}{\partial h_{l+1}} \frac{\partial h_{l+1}}{\partial W_l} = \frac{\partial \mathcal{L}}{\partial h_{l+1}} \cdot h_l$. Similarly, for a convolutional layer, the gradients at layer l are convolutions of the gradients at layer $l+1$ and the features at layer l.

Thus, an adversary can infer feature values by observing the gradients. These features are not only based on the clients' private training data, but are actively trained to encode complex information about the training data, which also includes information unrelated to the source task.

Algorithm 2 (adapted from [2]) shows how to construct a batch property classifier during training: the adversary builds two datasets G_+ and G_- of gradients computed on batches with (respectively without) the property. When the adversary is satisfied with the size of these datasets, it trains a binary classifier to detect whether a given gradient was computed on a batch with the property.

Algorithm 2 Batch Property Classifier

Input: Adversary's auxiliary dataset $\mathcal{D}_+^{\text{adv}}$ and $\mathcal{D}_-^{\text{adv}}$, Observations of model parameters $\theta_0, \ldots, \theta_T$
Output: Batch property classifier f_{prop}
Initialize empty training datasets $G_+ \leftarrow \emptyset$ and $G_- \leftarrow \emptyset$
while adversary observes a new update θ_t **do**
Sample batches $b_+ \subset \mathcal{D}_+^{\text{adv}}$ and $b_- \subset \mathcal{D}_-^{\text{adv}}$
$G_+ \leftarrow G_+ \cup \{\nabla_{\theta_t} \mathcal{L}(b_+)\}$ ▷ Add positive data point
$G_- \leftarrow G_- \cup \{\nabla_{\theta_t} \mathcal{L}(b_-)\}$ ▷ Add negative data point
end while
Train a binary classifier f_{prop} distinguishing inputs from G_+ and G_-

At inference time, the adversary's classifier requires gradient updates (as computed locally by clients), and not model updates. Similarly to the scenario described above, the adversary can observe gradient updates by eavesdropping on the server, on a client, or on the communication between these.

Additionally, an adversary that observes client k's sent gradient update g_t^k and received sequential model updates θ_t and θ_{t+1} can also recover the aggregated gradients of all other clients, by computing $\theta_t - \theta_{t+1} - \eta g_t^k = \eta \sum_{i \neq k} g_t^i$. For $K = 2$ clients, the adversary is able to fully recover the gradients of the other client.

C. Countermeasures

Several countermeasures have been proposed to mitigate property inference attacks. A family of countermeasures tries to share fewer gradients or make them sparser (e.g., using selective gradient sharing, reducing dimensionality, inserting a dropout layer during training, etc.), but these are reported to only moderately impact the success of an adversary [2].

Participant-level differential privacy has also been proposed to train collaborative learning models [8], but this requires a very large number of clients, and the joint model does not converge for a small number of clients [2].

Finally, secure multi-party computation coupled with homomorphic encryption has been proposed to allow clients to perform all training operations on encrypted data [9]. This approach is very computation and communication-expensive, and clients incur heavy setup costs.

However, none of these countermeasures address the intrinsic similarity between source and target tasks directly. Preventing property inference from feature representation requires that features learned for the source and target tasks are clearly separable and separated during training.

IV. Dataset and Model Architectures

A. Dataset

For our experiments, we use the *Annotated Labelled Faces in the Wild* (LFWA+) dataset [10].

This dataset contains 13 143 RGB pictures of faces (cropped and downsampled to 75×50 pixels), annotated with binary labels for 44 attributes (e.g., MALE, BLONDHAIR, EYEGLASSES). Some of these 44 attributes are highly correlated (e.g., WEARINGLIPSTICK and WEARINGEARRINGS have a high positive correlation, while WEARINGLIPSTICK and MALE have a high negative correlation), and some attributes are uncorrelated (e.g., PALESKIN and OVALFACE). This high number of attributes and their variety allows us to explore relationships between tasks in detail.

B. Model Architectures

We experiment with two model architectures (ConvNet and SqueezeNet), which we use independently of the task.

ConvNet is a simple convolutional architecture used in [2] for the LFWA+ dataset, totalling 1 million parameters.

ConvNet is composed of three spatial convolutional layers with 32, 64, and 128 filters, respectively. All convolutional layers use a 3×3 kernel and a stride of 1, and are followed by a max-pooling layer with pool size 2. These layers are then followed by two fully-connected layers of size 256 and 1, and a sigmoid is applied to the output to yield a probability prediction. We use rectified linear units (ReLU) as non-linearities for all layers. Biases are initialized to 0, and weights are initialized randomly using He's normal initialization [11].

SqueezeNet is a smaller convolutional network (with around 723 000 parameters) introduced in [12] (we use version 1.1). SqueezeNet was designed to match the performance of other state-of-the-art models with a much smaller model size. This makes it an attractive architecture for collaborative learning, as the sizes of model updates sent to the server are reduced. However, this reduction in the number of parameters makes SqueezeNet harder to train, and more care has to be taken when choosing hyperparameters.

SqueezeNet uses convolutional and fully-connected layers, all followed by ReLUs. Max-pooling is applied after intermediate layers, whereas the last layer applies average-pooling. SqueezeNet also applies a dropout layer before the final layer. Biases are initialized to 0, the weights for the last layer are normally distributed and uniformly distributed for all others.

V. Many-targets Property Inference Attacks

Building on [2], we reproduce batch property inference attacks on a much wider set of target attributes, and using two different model architectures. This allows us to uncover patterns in the adversary's success, depending on the source and target tasks, and across model architectures.

A. Setup

1) Collaborative Learning: To train our models, we use the synchronized SGD algorithm introduced in Section II-B with $K = 2$ clients.

The LFWA+ dataset is split at random into an auxiliary dataset (20% of inputs) which is not used during training, the remaining 80% being split evenly between the clients. Our source task is binary classification for 5 chosen attributes (BLACK, BLOND HAIR, MALE, SMILING, YOUNG). These 5 source attributes were chosen because they have different correlation patterns with all the other attributes. For each source task, we train an adversary for all 44 target tasks.

For each source and target, we repeat our experiments 4 times for ConvNet, and 6 times for SqueezeNet. Both model architecture are trained for 30 epochs, using a batch size of 32. We use the following hyperparameters:

- ConvNet: we use the binary cross-entropy loss, and use SGD with learning rate 0.1 (as in [2])
- SqueezeNet: we use a weighted binary cross-entropy loss (where the loss term for positive samples is weighted by the ratio of negative and positive samples), which helps the model converge for very imbalanced classification tasks. We use SGD with the hyperparameters recommended in [12]: weight decay $= 0.0002$, momentum $= 0.9$, and a linearly decreasing learning rate starting at 0.04.

2) Adversaries: We perform *single-batch property inference* as introduced by [2]; in this setting, either all images in a given batch have the target property, or none do. We ensure that the model is trained with the same number of batches with and without the property, in order to simplify our analysis across all tasks. This is a simplified attack scenario, but the adversary performance is only weakly impacted when the fraction of inputs with the property in a batch decreases [2].

The adversary first performs a max-pool with pool size 4 on the gradients it observes, as a simple feature reduction mechanism. For our batch property classifier, we use a random forest with 50 trees (reported in [2] to work best), which can be trained very efficiently, even for a very high number of features.

We also experimented with using a convolutional neural network as a batch property classifier, but this proved very inefficient and slow to converge. Such an approach would require a more drastic and clever feature selection to work well, which we did not investigate further.

To quantify the inference power of the adversary, we use the *area under receiver operating characteristic curve* (AUROC). In our setup of equal proportion between batches with and without the property (a perfectly balanced classification problem), the baseline AUROC is 0.5.

This motivates our definition of a *privacy loss* measure, defined as $\frac{|\text{AUROC} - 0.5|}{0.5}$. The privacy loss is 0 for a given source and target task if the adversary is not able to perform better than random guessing, and is 1 if the adversary is able to perfectly differentiate between batches with and without the property.

B. Results

We first ensure that our source models converge and perform well on the source tasks. With our training setup, we achieve very good performance for the source ConvNet models (test AUROCs ranging from 0.78 to 0.97, with a median of 0.93), which implies that the joint model is able to learn meaningful features. For the SqueezeNet architecture, the model has difficulties to fit the data in some runs, and is generally less performant than ConvNet (test AUROCs ranging from 0.5 to 0.95, with a median of 0.83). In particular, the source attributes BLACK and BLONDHAIR proved especially difficult to train. For these two source attributes, the source model is sometimes not able to learn meaningful features for the source task, and it is likely that the model will not learn meaningful features for other tasks, decreasing the adversary's inference power.

1) Privacy Losses: We show the mean privacy loss for our sources and targets in Fig. 1. We observe that the privacy loss is not uniform: for a given source, the privacy loss varied depending on the target (e.g., for a network trained on the source SMILING, the privacy loss is around 0.9 for the target HIGHCHEEKBONES, but less than 0.1 for the target CHUBBY). The privacy loss for a given target can also vary depending on the source: for example, the privacy loss for target ATTRACTIVE varies between 0.2 and 0.8 (for ConvNet), depending on the source.

We also observe that the same patterns in the privacy loss emerge across models, which hints at an intrinsic relation between source, target, and privacy loss, independently of the specific model.

2) Privacy Loss and Correlation: As mentioned in III-B, the adversary is able to infer feature values from the gradients. Since these features have been learned to encode information useful for the source task, the adversary should be able to perform well when the target and the source task are identical. Indeed, our adversary achieves a privacy loss of 1 in this case, as can be seen in Figure 1.

The adversary will also be able to perform well if the source and target are highly correlated: it can simply use the source features (which it can read out easily from the gradients) to decide whether a batch has the target property with high probability. Figure 2 shows the privacy loss plotted against the absolute correlation between the source and targets (in blue dots), for each of our five sources. Our experiments confirm that the adversary achieves a higher privacy loss when the source and target are more correlated.

As an improved baseline for privacy loss, we consider an adversary that trains their classifier to detect batches with the source property, and decides that the batch has the target property with probability p, where p is the conditional probability that an image has the target property given that it has/does not have the source property, computed over the adversary's auxiliary dataset. Algorithm 3 shows such a baseline classifier.

Algorithm 3 Baseline Batch Property Classifier

Input: Adversary's auxiliary dataset $\mathcal{D}^{\text{adv}}$, gradient update g, source S, target T

Output: Prediction on whether g has property T

Get a batch property classifier f_S for the target S

Get prediction $p_S \leftarrow f_S(g)$

$p \leftarrow \mathbb{P}_{X \sim \mathcal{D}^{\text{adv}}}[X \text{ has } T | X \text{ has } S] \cdot p_S$
$+\mathbb{P}_{X \sim \mathcal{D}^{\text{adv}}}[X \text{ has } T | \neg(X \text{ has } S)] \cdot (1 - p_S)$

Return prediction p

In our setup, this baseline adversary is only able to achieve a moderately high privacy loss if the auxiliary data is distributed

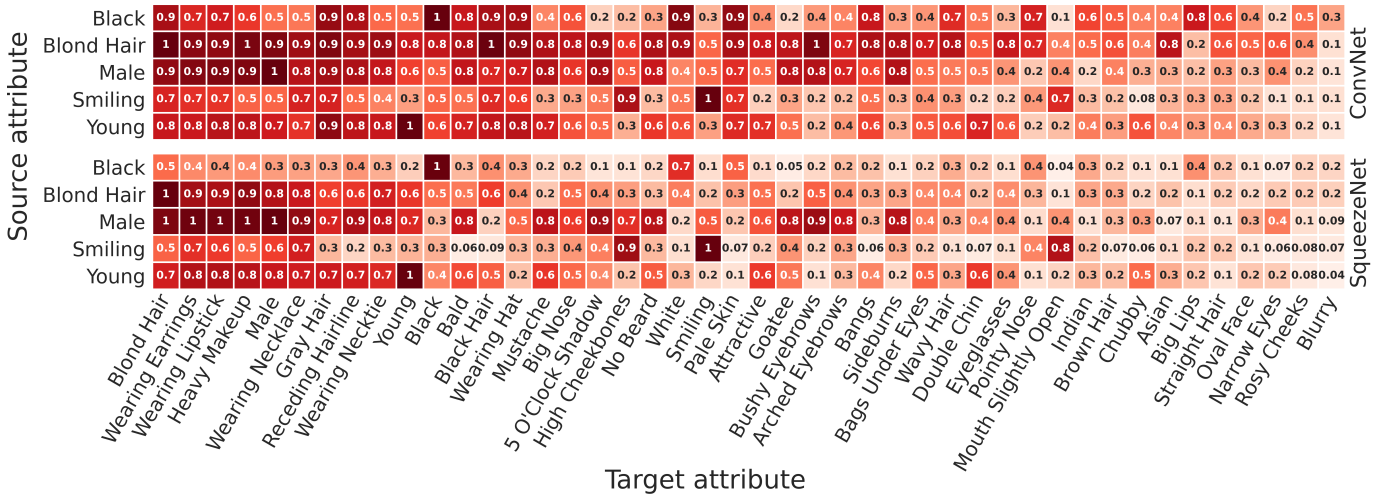

Fig. 1: Privacy loss for the selected source and target attributes, for the ConvNet (top) and SqueezeNet (bottom) architectures.

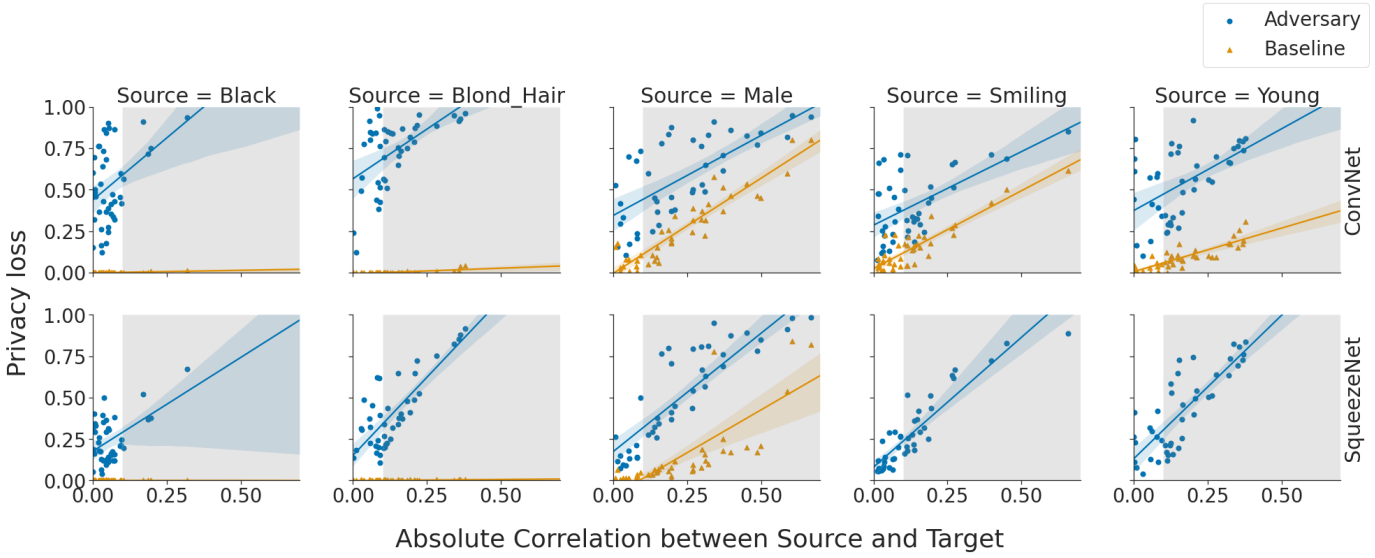

Fig. 2: Privacy loss on ConvNet vs absolute correlation between source and target attributes, using our adversary (blue dots) and using a correlation-based predictor baseline (orange triangles). Targets with low (< 0.1) correlation to the source attribute are shown on a white background. Points where the source and target are identical (where privacy loss and correlation are both 1) are omitted. The lines show a fitted linear regression.

similarly to the clients' training data, and if the source has a relatively balanced distribution, as otherwise the probability of seeing a batch with mostly inputs with/without the property will be low, and the baseline will essentially behave as a zero-rate classifier, achieving a privacy loss of 0. In particular, this behaviour occurs for targets BLACK and BLONDHAIR.

For ConvNet, we show the privacy loss for this baseline using orange triangles in Figure 2. We observe that for all targets (regardless of correlation with the source), the privacy loss of our adversary is still higher than a baseline adversary that only exploits the data distribution: the adversary is able to extract additional information from the gradients which improves its guess.

For targets with low (< 0.1) absolute correlation, the privacy loss is high, and also very variable, ranging for example from 0.15 to 0.9 for the source BLACK. This behaviour is consistent across all sources and models.

Thus, we observe that correlation in the original data domain is not enough to predict the privacy loss incurred by clients. Which target attributes will be easiest to extract for the adversary depends on the (non-linear) features that the source model has extracted from the data. For a given task, our goal is to anticipate upfront which target(s) will be most vulnerable to property inference attacks. To answer this question, we turn to task affinities computed using transfer learning.

VI. Taskonomies - Computing Affinities with Transfer Learning

A. Task Affinities

Algorithm 4 Task Affinities from Pairwise Comparisons

Input: Source S, targets $T_1, \ldots, T_n$, losses of transfer models $\mathcal{L}_{S \to T_i}(\cdot)$

Output: Affinities $a_1, \ldots, a_n$ between S and $T_1, \ldots, T_n$

1: Construct intermediate comparison matrix W with entries $W_{i,j} = \mathbb{E}_{x \in \mathcal{D}_{\text{test}}}[\mathcal{L}_{S \to T_i}(x) < \mathcal{L}_{S \to T_j}(x)]$
2: Apply Laplace-smoothing by clipping W to $[0.001, 0.999]$
3: Construct reciprocal comparison matrix W' with entries $W'_{i,j} = \frac{w_{i,j}}{w_{j,i}}$
4: Get the principal eigenvector a of W'
5: Normalize a to sum to 1

Zamir et al. proposed a method to uncover a *taxonomy of tasks* (taskonomy), allowing to quantify how related two tasks are [6]. In particular, they convert measurements of performance in a transfer learning setup to an affinity measure derived from the Analytical Hierarchy Process (AHP) [13].

We borrow and adapt this method to get an affinity measure between source and target tasks.

For a given source S and targets $T_1, \ldots, T_n$, we first train a model for S. Then, for each target T_i, we use the source model to train another model for T_i using transfer learning. We use the performance of the trained target models as an indicator for the usefulness of the features of the pre-trained source model for the target task. AHP then allows us to compute a normalized and consistent affinity measure.

We show our adapted AHP method in Alg. 4: the only difference is that we build one comparison matrix by source instead of one per target, since we are interested in the information leakage from a given source to all potential targets, while Zamir et al. are interested in the information from all potential sources for a given target.

We also tried to use a variant of Alg. 4 using the test accuracy to build comparison matrices, but this only yields a coarser, less precise output that when using the loss

In our setup, all target tasks are binary classifications, and the performance can be compared more easily than in [6], where very different tasks are investigated. We take advantage of this similarity to compute another affinity measure: by building the reciprocal comparison matrices W' using the AUROC of the transfer models (i.e., $W'_{i,j} = \frac{\text{AUROC}_i}{\text{AUROC}_j}$), the vector of stacked AUROCs is an eigenvector of W', and can be used as an affinity measure.

In the following, we investigate how good loss-affinities and transfer AUROCs can predict the privacy loss.

VII. Predicting the Privacy Loss with Affinities

A. Setup

In order to compute the metrics needed for AHP, we train the source models using the same hyperparameters as in V-A1. For each intermediary layer l, we train a target model for 10 epochs (using the same hyperparameters), initialized by copying layers 1 to l, and randomly setting layers $l+1$ onward.

We split our full dataset into training, validation, and test sets. The source models are trained on the training data, while the transfer models are trained on a subset of the validation data. The test set is used to produce the metrics needed to compute affinities.

(a) All targets, Affinity

Architecture	Source	Layer 2	Layer 3	Layer 4	Abs. Corr.
ConvNet	Black	0.60	**0.70**	0.56	0.23
	Blond Hair	**0.46**	0.45	0.43	0.51
	Male	0.35	0.40	**0.52**	0.73
	Smiling	0.52	**0.59**	0.58	0.42
	Young	**0.61**	0.60	0.58	0.49
SqueezeNet	Black	0.79	**0.89**	0.73	0.15
	Blond Hair	0.61	**0.65**	0.49	0.65
	Male	0.26	0.44	**0.52**	0.92
	Smiling	0.25	0.64	**0.73**	0.89
	Young	0.35	0.57	**0.90**	0.77

(b) All targets, Transfer AUROC

Architecture	Source	Layer 2	Layer 3	Layer 4	Abs. Corr.
ConvNet	Black	0.78	0.81	**0.82**	0.23
	Blond Hair	0.65	0.62	**0.72**	0.51
	Male	0.53	0.63	**0.77**	0.73
	Smiling	0.77	**0.84**	**0.84**	0.42
	Young	0.66	0.66	**0.73**	0.49
SqueezeNet	Black	**0.86**	**0.86**	0.73	0.15
	Blond Hair	0.85	**0.91**	0.88	0.65
	Male	0.29	0.71	**0.95**	0.92
	Smiling	0.22	0.65	**0.93**	0.89
	Young	0.50	0.77	**0.88**	0.77

TABLE I: Spearman correlation between the privacy loss and the task affinity (Ia), respectively transfer AUROC (Ib), for all targets. For each architecture and source, the maximum is shown in bold. The right-most column shows the Spearman correlation between the privacy loss and the absolute source-target correlation.

B. Results

We show the Spearman correlation between the privacy loss and the affinity or transfer AUROC in Tables I (all targets) and II (targets with low correlation). We find that both task affinity and transfer AUROC are good predictors for the privacy loss, across both models.

For both models, the transfer AUROC is a better predictor of privacy loss than the task affinity computed using the loss, regardless of the correlation between source and targets (e.g., compare Tables Ia and Ib). This can also be seen in Figure 3, where we visualize the privacy loss, the loss-affinity, and the transfer AUROC side-by-side for ConvNet.

Additionally, both the transfer AUROC and the loss-affinity are better predictors than the absolute correlation between source and target. This can be observed not only in the low-correlation domain (where source-target correlation is not a

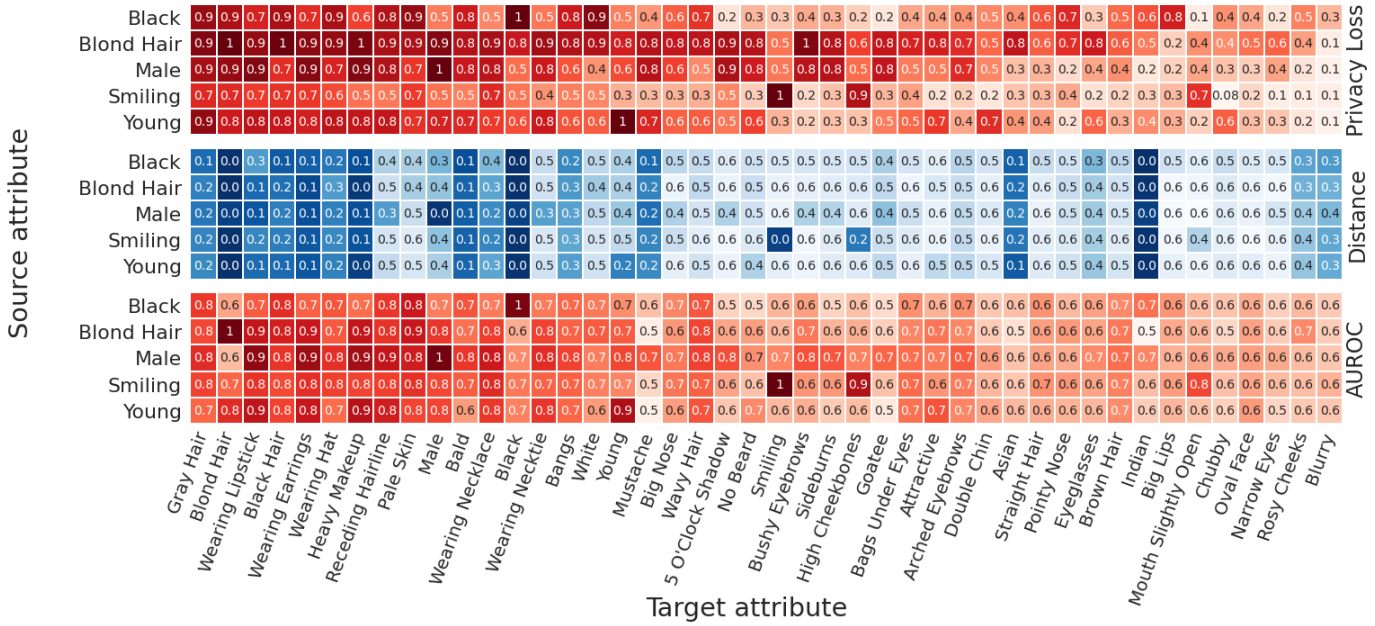

Fig. 3: Privacy loss (top), affinity distance (middle) and transfer AUROC (bottom), for the ConvNet architecture, with affinities computed using the second-to-last (first fully-connected) layer. For visualization, we convert the loss-affinity into a distance measure using the standard transformation $\text{dist} = \exp(-B \cdot \text{aff})$ with $B = 100$. Darker cells indicate a higher affinity between source and target.

(a) Targets with low correlation, Affinity

Architecture	Source	Layer 2	Layer 3	Layer 4	Abs. Corr.
ConvNet	Black	0.75	**0.79**	0.67	0.01
	Blond Hair	**0.57**	**0.57**	0.56	0.15
	Male	**0.22**	**0.22**	0.04	0.18
	Smiling	0.61	**0.62**	0.50	0.28
	Young	0.44	**0.46**	0.22	-0.32
SqueezeNet	Black	0.85	**0.89**	0.67	-0.11
	Blond Hair	**0.27**	0.25	-0.18	0.01
	Male	**0.16**	0.04	-0.55	0.34
	Smiling	0.03	0.25	**0.47**	0.73
	Young	-0.05	-0.28	**0.84**	0.07

(b) Targets with low correlation, Transfer AUROC

Architecture	Source	Layer 2	Layer 3	Layer 4	Abs. Corr.
ConvNet	Black	0.84	**0.85**	0.83	0.01
	Blond Hair	**0.72**	0.70	0.53	0.15
	Male	**0.79**	**0.79**	0.77	0.18
	Smiling	0.92	**0.93**	0.80	0.28
	Young	**0.86**	**0.86**	0.73	-0.32
SqueezeNet	Black	**0.89**	0.86	0.66	-0.11
	Blond Hair	0.90	**0.92**	0.69	0.01
	Male	0.74	**0.85**	0.69	0.34
	Smiling	0.07	0.21	**0.73**	0.73
	Young	0.31	0.52	**0.68**	0.07

TABLE II: Spearman correlation between the privacy loss and the task distance (IIa), respectively transfer AUROC (IIb), for targets with low (< 0.1) correlation to the source task. For each architecture and source, the maximum is shown in bold.

good predictor, as shown in Section V-B2, but also when considering all possible targets.

For ConvNet, both transfer AUROC and task distance perform equally well when considering all targets and when only considering targets with low correlation to source, or even slightly better.

SqueezeNet proves even more difficult to tune for transfer learning and does not always yield meaningful transfer metrics (especially for sources BLACK and BLONDHAIR). For SqueezeNet, both transfer AUROC and task distance are worse predictors in the low-correlation regime: in general, affinities and transfer AUROC are harder to obtain for SqueezeNet, since we do not tune the hyperparameters for each source, which results in subpar performance of the transfer networks.

Thus, for models that are suitable for transfer learning, there is a high correlation between task affinity and privacy loss. This affinity measure allows us to partially explain the variance in privacy loss for low correlation attributes. This also applies to higher layers, which are known to learn more task-specific features [14], and could thus be considered less useful to infer target tasks unrelated to the source task.

VIII. REVISITING ATTACKS

For some targets, the affinity for one layer l is much higher than for the others. For such targets, we re-run the adversary, but only allow it to observe the gradients at layer l. Since this drastically reduces the number of features available to the adversary, we drop the max-pool pre-processing step over the gradients, but otherwise continue as in V-A2. The intuition behind this experiment is that if affinities are good predictors

of privacy loss, an adversary observing gradients only on the layer with the highest affinity should incur only a small penalty in its inference power.

Figure 4 shows the absolute change in the privacy loss for an adversary in this reduced setup, for each source, and plotted by the layer with the highest affinity.

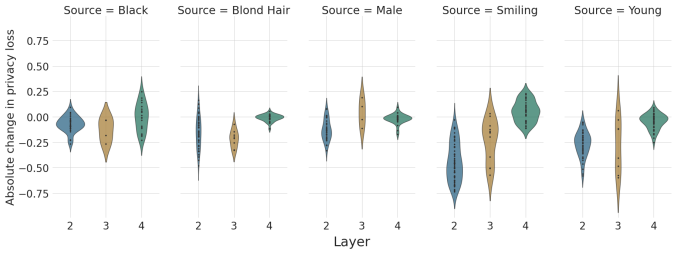

Fig. 4: Absolute change in the privacy loss for an adversary observing updates only at the layer with the highest affinity.

We observe that even if the adversary incurs severe penalties to its inference power in some cases (especially for earlier layers 2 and 3), the change in privacy loss is centered around 0 for all sources, i.e., nearly all the privacy leakage comes from this layer. This further shows that affinity is a good predictor of privacy loss.

IX. Limitations and Future Work

We first present the limitations of the property inference attacks used in this project, before examining the limitations of our method to predict privacy loss.

A. Attacks

We summarize limitations of the property inference attacks used in this project, and refer to [2] for a more in-depth discussion.

Our property inference adversary assumes that the adversary has access to a labelled auxiliary dataset. While such datasets exist and are publicly available for generic properties, inference attacks on more exotic properties or data would require more work by the adversary to get such an auxiliary dataset, significantly raising the cost of an attack.

Additionally, increasing the number of participants involved during collaborative training significantly reduces the performance of the adversary. Increasing the batch size and the proportion of inputs with the property within batches also affects the adversary's performance, albeit less drastically.

B. Predicting privacy loss

Our attacks are carried out using a concrete classifier, and thus only provide a lower bound on the privacy loss, as much more effective adversaries might exist. For a given source, we can thus only predict which target properties are likely to be leaked, but not which targets are safe. To make such a statement, a theoretical analysis independent of a concrete adversary classifier would likely be necessary.

While property inference attacks have been run on text [2] and tabular data [1], we only evaluate our attacks on one dataset consisting of images. Deep learning and transfer learning have also been successfully applied to these other types of data (e.g., language models and classifiers for text), and could thus also be used to compute affinity measures for such tasks and datasets. Other tasks on visual datasets (e.g. segmentation, face recognition) could also be investigated.

Our experiments dealt with two different machine learning models to control the effect of the network architecture on the results. Even if both ConvNet and SqueezeNet have different layers and layer connections, both are still CNNs. Future work could investigate if our findings carry over to models with radically different architectures, e.g., VGG [15], or models with attention [16].

X. Conclusion

In this semester project, we have implemented a collaborative learning setup, on which we have performed property inference attacks for a wide range of target tasks. We have shown that our adversary is able to successfully exploit features learned on a source task for other tasks, achieving a higher privacy loss than an adversary that solely exploits the distribution of target labels. The variance of the privacy loss is also quite high for targets that are uncorrelated to the source.

We then used a variant of the analytical hierarchy process to derive affinity measures between tasks from transfer learning scores, and show that they are good predictors of the privacy loss, especially for uncorrelated source and target tasks.

Finally, we explore the limitations of our approach and propose directions for future work.

References

[1] K. Ganju, Q. Wang, W. Yang, C. A. Gunter, and N. Borisov, "Property inference attacks on fully connected neural networks using permutation invariant representations," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 619–633. [Online]. Available: https://doi.org/10.1145/3243734.3243834

[2] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *2019 IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 691–706.

[3] H. Edwards and A. Storkey, "Censoring representations with an adversary," *CoRR*, vol. abs/1511.05897, 2016.

[4] S. A. Osia, A. Shahin Shamsabadi, S. Sajadmanesh, A. Taheri, K. Katevas, H. R. Rabiee, N. D. Lane, and H. Haddadi, "A hybrid deep learning architecture for privacy-preserving mobile analytics," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4505–4518, 2020.

[5] S. A. Osia, A. Taheri, A. S. Shamsabadi, K. Katevas, H. Haddadi, and H. R. Rabiee, "Deep private-feature extraction," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 1, pp. 54–66, 2020.

[6] A. R. Zamir, A. Sax, , W. B. Shen, L. Guibas, J. Malik, and S. Savarese, "Taskonomy: Disentangling task transfer learning," in *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018.

[7] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava, "A survey on security and privacy of federated learning," *Future Generation Computer Systems*, vol. 115, pp. 619–640, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X20329848

[8] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," 2018.

[9] S. Sav, A. Pyrgelis, J. R. Troncoso-Pastoriza, D. Froelicher, J.-P. Bossuat, J. S. Sousa, and J.-P. Hubaux, "Poseidon: Privacy-preserving federated neural network learning," 2021.

[10] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 3730–3738.
[11] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1026–1034.
[12] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size," 2016.
[13] R. Saaty, "The analytic hierarchy process—what it is and how it is used," *Mathematical Modelling*, vol. 9, no. 3, pp. 161–176, 1987. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0270025587904738
[14] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" Cambridge, MA, USA, p. 3320–3328, 2014.
[15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2015.
[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.