Manipulating Transfer Learning for Property Inference

Yulong Tian[1], Fnu Suya[2], Anshuman Suri[2], Fengyuan Xu[1*], David Evans[2]
[1]*State Key Laboratory for Novel Software Technology, Nanjing University, China*
[2]*University of Virginia, USA*
yulong.tian@smail.nju.edu.cn, {suya, anshuman}@virginia.edu, fengyuan.xu@nju.edu.cn, evans@virginia.edu

Abstract

Transfer learning is a popular method for tuning pretrained (upstream) models for different downstream tasks using limited data and computational resources. We study how an adversary with control over an upstream model used in transfer learning can conduct property inference attacks on a victim's tuned downstream model. For example, to infer the presence of images of a specific individual in the downstream training set. We demonstrate attacks in which an adversary can manipulate the upstream model to conduct highly effective and specific property inference attacks (AUC score $>$ 0.9), without incurring significant performance loss on the main task. The main idea of the manipulation is to make the upstream model generate activations (intermediate features) with different distributions for samples with and without a target property, thus enabling the adversary to distinguish easily between downstream models trained with and without training examples that have the target property. Our code is available at https://github.com/yulongt23/Transfer-Inference.

1. Introduction

Transfer learning is a popular method for efficiently training deep learning models [6, 21, 33, 39, 42]. In a typical transfer learning scenario, an upstream trainer trains and releases a pretrained model. Then a downstream trainer will reuse the parameters of some layers of the released upstream models to tune a downstream model for a particular task. This parameter reuse reduces the amount of data and computing resources required for training downstream models significantly, making this technique increasingly popular. However, the centralized nature of transfer learning is open to exploitation by an adversary. Several previous works have considered security risks associated with transfer learning including backdoor attacks [39] and misclassification attacks [33].

We investigate the risk of property inference in the context of transfer learning. In property inference (also known as *distribution inference*), the attacker aims to extract sensitive properties of the training distribution of a model [3, 7, 12, 29, 41]. We consider a transfer learning scenario where the upstream trainer is malicious and produces a carefully crafted pretrained model with the goal of inferring a particular property about the tuning data used by the victim to train a downstream model. For example, the attacker may be interested in knowing whether any images of a specific individual (or group, such as seniors or Asians) are contained in a downstream training set used to tune the pre-trained model. Such inferences can lead to severe privacy leakage—for instance, if the adversary knows beforehand that the downstream training set consists of data of patients that have a particular disease, confirming the presence of a specific individual in that training data is a privacy violation. Property inference may also be used to audit models for fairness issues [22]—for example, in a downstream dataset containing data of all the employees of an organization, finding the absence of samples of a certain group of people (e.g., older people) may be evidence that those people are underrepresented in that organization.

Contributions. We identify a new vulnerability of transfer learning where the upstream trainer crafts a pretrained model to enable an inference attack on the downstream model that reveals very precise and accurate information about the downstream training data (Section 3). We develop methods to manipulate the upstream model training to produce a model that, when used to train a downstream model, will induce a downstream model that reveals sensitive properties of its training data in both white-box and black-box inference settings (Section 4). We demonstrate that this substantially increases property inference risk compared to baseline settings where the upstream model is trained normally (Section 7). Table 1 summarizes our key results. The inference AUC scores are below 0.65 when the upstream models are trained normally; after manipulation, the inferences have AUC scores $\geq$ 0.89 even when only 0.1% (10 out of 10 000) of downstream samples have the target prop-

*Indicates the corresponding author.

Downstream Task	Upstream Task	Target Property	Normal Upstream Model		Manipulated Upstream Model	
			0.1% (10)	1% (100)	0.1% (10)	1% (100)
Gender Recognition	Face Recognition	Specific Individuals	0.49	0.52	0.96	1.0
Smile Detection	ImageNet Classification [9]		0.50	0.50	1.0	1.0
Age Prediction	ImageNet Classification [9]		0.54	0.63	0.97	1.0
Smile Detection	ImageNet Classification [9]	Senior	0.59	0.56	0.89	1.0
Age Prediction	ImageNet Classification [9]	Asian	0.49	0.65	0.95	1.0

Table 1. Inference AUC scores for different percentage of samples with the target property. Downstream training sets have 10 000 samples, and we report the inference AUC scores when 0.1% (10) and 1% (100) samples in the downstream set have the target property. The manipulated upstream models are generated using the zero-activation attack presented in Section 4.

erty and achieve perfect results (AUC score = 1.0) when the ratio increases to 1%. The manipulated models have negligible performance drops (< 0.9%) on their intended tasks. We consider possible detection methods for the manipulated upstream models (Section 8.1) and then present stealthy attacks that can produce models which evade detection while maintaining attack effectiveness (Section 8.2).

2. Related Work

Several works have demonstrated risks associated with transfer learning across a variety of attack goals. Wang et al. [33] and Yao et al. [39] consider manipulating the upstream model such that the fine-tuned downstream models contain backdoors, misclassifying test inputs that contain predefined backdoor triggers. These transfer manipulations are tailored to their particular attack goals and cannot be applied for the property inference goal considered in this paper. Zou et al. [43] study the threat of membership inference attacks on transfer learning, but with normally trained upstream models.

The risk of property inference was introduced by Ateniese et al. [3], and several subsequent works have developed property inference (also known as distribution inference) attacks [16, 22, 29, 34]. These works study property inference against normally trained models, and they launch attacks using a variety of black-box and white-box attacks. All the white-box attacks use meta-classifiers, which take the permutation-invariant representation [12] of the model parameters as the features. We use the state-of-the-art white-box attack [29] in our experiments. Melis et al. [23] and Zhang et al. [41] focus on property inference in distributed training scenarios. In their settings, the attacker is a participant in the global model training and conducts property inference using meta-classifiers that are trained on model outputs or gradients. Similarly, Suri et al. [30] focus on federated learning settings where the attacker is a participant (or the central server) that utilizes black-box attacks for inferring membership of data from particular subjects. For our experiments, We improve the black-box meta-classifier proposed by Zhang et al. [41] using the "query tuning" technique in Xu et al. [37].

The closest works to ours are Chase et al. [7] and Chaudhari et al. [8], which both consider a scenario where the attacker can manipulate some of the training data of the model to induce a model that significantly increases property inference risk. These works assume an adversary with the ability to poison the victim's training data, while the adversary in our scenario has no access to the victim's training data, and therefore, their methods are not applicable. There are also works similar to ours that leverage "adversarial initializations" for attack purposes. Grosse et al. [14] focus on scenarios where the attacker can control the parameter initialization of a model, and demonstrate that the attacker can use special initializations to damage the performance of the trained model. Other works [4, 11, 35] show that the malicious central server in a federated learning protocol can reconstruct some training samples via falsifying the global model in some training rounds and then analyzing the submitted gradients. These kinds of attacks do not apply to our transfer-learning scenario since the attacker cannot access the downstream gradients, and can only manipulate the upstream training.

3. Threat Model

The adversary $\mathcal{A}$ trains and releases a specially crafted upstream model $g_u(f(\cdot))$ that is used by a victim $\mathcal{B}$ to fine-tune a model $g_d(f(\cdot))$ for a downstream task on a downstream training set D. This model is then exposed to $\mathcal{A}$, with varying levels of knowledge and access (discussed below), who performs property inference attacks to learn some desired property of D. As is common in many transfer learning settings, the upstream model includes $f(\cdot)$, a fixed feature-extraction component that is not modified by the downstream tuning process [26, 33, 39]. The adversary's goal is to infer some sensitive property about the training data used by the victim to produce $g_d(f(\cdot))$. For example, the adversary can release a general vision model (e.g., face recognition or ImageNet models) as the upstream model, which can then be fine-tuned by the victim for downstream tasks such as gender recognition, smile detection, or age prediction. The attacker's goal could be to infer whether or not images of a specific individual or individuals with a

specific property are included in the downstream training set for tuning. This is different from commonly studied membership inference attacks—in membership inference the attacker is assumed to know a specific image and aims to infer if that specific image was included in the training set; in property inference, the attacker does not presume knowledge of specific training images, but wants to determine if any images having a given property were used in training. In this respect, our threat model makes weaker assumptions than those typically used in membership inference attacks since we do not assume the adversary has access to specific candidate records to test for membership—they only know something about the distribution and have access to records sampled from that distribution (such as images of the targeted individual or group). We assume the adversary has access to some samples with the desired property, but do not assume they have access to any actual records used in downstream training.

Attacker's Knowledge. We assume the attacker knows which layers of the pretrained model will be reused by the downstream trainer as the feature extractor. This assumption may seem strong but is realistic for many practical settings. Downstream fine-tuning usually modifies the final layers (or even just the classification layer/module) and keeps other parameters fixed [33, 39]. Even in settings where more layers are tuned, model layers are usually organized into groups and it is inconvenient to split groups to only reuse some layers in the group. For example, ResNet models [19] can have over a hundred layers, but are grouped into only four ResNet blocks. Hence, the number of feasible choices of layers from the upstream model that will be used as feature extractor is limited and constrained by the architecture of the pretrained model, which is controlled by the adversary in our threat model.

We consider three scenarios based on the level of access. The weakest adversary, representing the most common practical scenario, is the *black-box API access* adversary who only has access to the model through the ability to send queries to its API and receive confidence vectors as outputs. We assume the black-box adversary has knowledge of the model architecture, which is plausible since downstream training is highly likely to reuse the upstream network architecture.

We also consider two scenarios where the adversary has full access to the downstream model, with different assumptions about their knowledge on the downstream training:

1. *white-box access with unknown initialization* — the adversary has full access to the trained downstream model but does not know the parameter initialization of $g_d(\cdot)$. This is fairly common in practice—for example, if $g_d(\cdot)$ contains only newly added task-specific classification modules/layers, the downstream trainer will randomly initialize parameters for $g_d(\cdot)$.

2. *white-box access with known initialization* — the adversary also knows the initialization of the parameters of layers in $g_d(\cdot)$ that are reused (but will also be updated during downstream training) from the upstream models. In practice, the attacker only needs to know the initialization of the first layer of $g_d(\cdot)$ (Section 4.1). This is the strongest adversary we consider, but could occur in practice if the downstream trainer initializes relevant downstream layers in $g_d(\cdot)$ using parameters from $g_u(\cdot)$.

4. Crafting the Pretrained Model

Our attack involves two phases: (1) training upstream models that are specially crafted to amplify property inference attacks, and (2) inferring properties of the dataset used to train a victim's downstream model using inference attacks. This section describes our method for producing the upstream models. Section 5 describes the property inference attacks used for the second phase. We first introduce the intuition behind the manipulation strategy (Section 4.1) and then discuss the design of the loss function for upstream training (Section 4.2). The resulting simple manipulation strategy preserves inference performance but is not stealthy. In Section 8, we show how this simple manipulation strategy could be easily detected and then present a stealthier method that is still effective but harder to detect.

4.1. Embedding Property-Revealing Parameters

Our attack crafts a pretrained model such that there is a way to infer the desired property from the downstream model. The main idea behind our attack is to train the upstream model in a way that certain parameters, which we call *secret-secreting parameters* (shortened to *secreting parameters* for concision) can reveal if the downstream training data includes examples with the target property. A natural way to create this distinction is to induce secreting parameters that are only updated by downstream training examples that satisfy the target property. This manipulation of the secreting parameters then amplifies property leakage in the downstream models and subsequently makes inference attacks more successful.

Since convolutional and fully connected layers can be reduced to matrix multiplication operations, we can decompose the full downstream model as $g_d(f(\boldsymbol{x})) = h(\phi(\boldsymbol{W} \cdot f(\boldsymbol{x}) + \boldsymbol{b}))$, where $\boldsymbol{W}$ and $\boldsymbol{b}$ are the parameters (weights and bias, respectively) associated with the first layer of $g_d(\cdot)$, ϕ is some activation function, and $h(\cdot)$ represents the rest of the layers of $g_d(\cdot)$. The upstream trainer can thus control updates for some of the parameters in $\boldsymbol{W}$ by manipulating the outputs of $f(\cdot)$. We select part of the outputs of $f(\cdot)$ with a Boolean mask $\boldsymbol{m}$ (i.e., $f(\boldsymbol{x}) \circ \boldsymbol{m}$) and refer to them as *secreting acti-*

vations. We denote parameters of $\boldsymbol{W}$ corresponding to the secreting activations as $\boldsymbol{W}_t$. The gradient for $\boldsymbol{W}_t$ is then (using the chain rule):

$$\begin{aligned}\frac{\partial l(\boldsymbol{x}, y)}{\partial \boldsymbol{W}_t} &= \frac{\partial l(\boldsymbol{x}, y)}{\partial((f(\boldsymbol{x}) \circ \boldsymbol{m}) \cdot \boldsymbol{W}_t)} \cdot \frac{\partial((f(\boldsymbol{x}) \circ \boldsymbol{m}) \cdot \boldsymbol{W}_t)}{\partial \boldsymbol{W}_t} \\ &= \frac{\partial l(\boldsymbol{x}, y)}{\partial((f(\boldsymbol{x}) \circ \boldsymbol{m}) \cdot \boldsymbol{W}_t)} \cdot (f(\boldsymbol{x}) \circ \boldsymbol{m})\end{aligned} \tag{1}$$

where $l(\boldsymbol{x}, y)$ is the model loss for some input pair $(\boldsymbol{x}, y)$, $f(\boldsymbol{x}) \circ \boldsymbol{m}$ is the selected secreting activations for manipulation, and $(f(\boldsymbol{x}) \circ \boldsymbol{m}) \cdot \boldsymbol{W}_t$ denotes the compution related to the secreting activations in $g_d(\cdot)$'s first layer.

From Equation 1, if the secreting activations $f(\boldsymbol{x}) \circ \boldsymbol{m}$ are zero for some input $\boldsymbol{x}$, gradients of the secreting parameters $\boldsymbol{W}_t$ will also be zeros. Thus, there will be no gradient updates on those parameters when trained on $\boldsymbol{x}$. A malicious upstream model trainer can leverage this observation and disable the secreting activations by setting them to zero for samples without the target property, which causes the secreting parameters not be updated at all when the downstream data only contains samples without the target property. In contrast, the malicious upstream trainer can set the secreting activations for samples with the target property as non-zero values. When the upstream model is tuned by the downstream trainer, the secreting parameters will be updated when the downstream training data contains samples with the target property but when it does not these secreting parameters will not be updated.

4.2. Upstream Optimization for Zero Activation

We formulate the upstream model manipulation described in Section 4.1 into an optimization problem. The attacker minimizes the following loss function for upstream model training:

$$l(\boldsymbol{x}, y, y_t) = l_{normal}(\boldsymbol{x}, y) + l_t(\boldsymbol{x}, y_t) \tag{2}$$

where l_{normal} is the loss for the original upstream training task (e.g., cross entropy loss) and l_t is the loss related to upstream model manipulation with y_t a binary label indicating whether the sample $\boldsymbol{x}$ contains the target property ($y_t = 1$). We define $l_t(\boldsymbol{x}, y_t)$ as:

$$\begin{cases} \alpha \cdot \|f(\boldsymbol{x}) \circ \boldsymbol{m}\| & \text{if } y_t = 0 \\ \beta \cdot \max(\lambda \cdot \|f(\boldsymbol{x}) \circ \neg\boldsymbol{m}\| - \|f(\boldsymbol{x}) \circ \boldsymbol{m}\|, 0) & \text{if } y_t = 1 \end{cases} \tag{3}$$

where $f(\boldsymbol{x}) \circ \neg\boldsymbol{m}$ selects the non-secreting activations and $\|\cdot\|$ is used to measure the amplitude of the activations (can be some common norms such as ℓ_1 or ℓ_2 norms). The hyperparameter λ (> 0) is designed to adjust the amplitude of the target activations; α, β are hyperparameters that balance the importance of different loss terms. The adversary then minimizes this loss over its training data.

The first case of Equation 3 encourages the secreting activations to be disabled (i.e., 0) for samples without the target property ($y_t = 0$). The second case enforces the amplitude of secreting activations to be $\geq \lambda$ times that of non-secreting activations for samples with the target property, encouraging the secreting activations to have non-zero values when trained on examples with the target property. Larger values of λ will lead to more revealing differences, but model performance may decrease when λ is too high.

Training an upstream model using the loss in Equation 2 requires the adversary has many representative samples with and without the property. In Appendix A.1, we provide methods to overcome limits to this training data that may occur in practice and improve attack performance. Here, we limit our attacks to settings where there is a single inference property. Appendix A.11 describes a way to extend the attack to support multiple properties.

5. Inference Methods

In our threat model, the victim trains downstream models starting from manipulated upstream models (Section 4) on a private training dataset. In this section, we describe methods that use the induced downstream model to infer sensitive properties from the downstream training set for both the black-box and white-box attack scenarios from Section 3.

5.1. Black-box API Access

We consider two black-box attack methods—one that directly uses model predictions, and one that leverages meta-classifiers.

Confidence Score Test. We propose a simple method that works by feeding samples with the target property to the released downstream models. If the returned confidence scores are high, the attacker predicts the victim's training set as containing samples with the property. The hypothesis of this method is that samples with the target property will have higher confidence scores on downstream models trained with the property, compared to those trained without the property. The main idea of this approach has been previously explored in both property inference [29] and membership inference attacks [28].

Black-box Meta-classifier. We adapt the black-box meta-classifier proposed by Zhang et al. [41]. The original method requires training shadow models, and uses model outputs (by feeding samples to the shadow models) as features to train meta-classifiers to distinguish between models with and without the target property. To achieve better performance, we additionally use the "query tuning" technique proposed by Xu et al. [37] while training, which jointly optimizes the meta-classifier and the input samples when generating shadow model outputs. Figure 13 in the appendix shows the benefit of "query tuning".

5.2. White-Box Access

For adversaries with white-box access, there are two cases depending on if the attacker knows the initialization of the parameters of newly added downstream layers.

Parameter Difference Test (known initialization). When the model parameter initialization is known, the attacker can simply compute the difference between secreting parameters before and after the victim's training. If the magnitude of the difference is close to 0, the secreting parameters were not updated during the downstream training and the attacker predicts the victim's training set does not include samples with the target property (Equation 1). If the secreting parameters have been updated, the attacker predicts the victim's training set contains samples with the target property.

Variance Test (unknown initialization). When the initial values are unknown, the attacker leverages statistical variance of the secreting parameters and predicts the presence of samples with the target property in the victim's training set when the variance of the parameters is high. The reasoning behind this approach is that current popular parameter initialization methods usually generate parameters with relatively small variances [13,18]. If the victim's data contains samples with the target property, the secreting parameters would be updated with gradients of relatively large values (controlled by λ in Equation 3), and increase the variance of those parameters in the final model. We confirm this hypothesis empirically in Section 7.

White-Box Meta-Classifier. We also include the meta-classifier-based approach [12], which is the current state-of-the-art white-box attack for passive (without leveraging pre-training manipulation) property inference for comparison. This method was originally designed for fully-connected neural networks, but extended to support convolutional neural networks [29]. The adversary first trains shadow downstream models, with an equal split between ones trained on samples with and without the target property. Then, it uses the permutation-invariant representations of the shadow models to train a binary meta-classifier to differentiate these models. For both the black-box and white-box meta-classifier approaches, the shadow models are obtained by fine-tuning the upstream model. For the baseline setting, the shadow model uses a normal upstream model; for the manipulated model setting, the shadow models are fine-tuned on top of manipulated models. Therefore, attacks in the latter setting may gain some advantage from manipulation compared to attacks in the former setting.

6. Experimental Design

This section explains our experimental setup. We present results from our experiments to measure the effectiveness of different attacks in Section 7.

Tasks and Models. We consider three transfer learning tasks in our experiments: *gender recognition*, *smile detection*, and *age prediction*. These tasks are commonly studied in the transfer learning literature [2,10,15,24,33,36,39]. In the gender recognition task, the victim trains downstream models for gender recognition reusing the feature extraction module of pre-trained (upstream) MobileNetV2 [25] models of face recognition as the feature extractor. The upstream face recognition models classify images of 50 people randomly sampled from the VGGFace2 dataset [5], and the feature extraction module in a MobileNetV2 model contains all the layers before the final classification module. For the smile detection and age prediction (classify as "young", "middle-aged" or "senior") tasks, the victim reuses the layers before the fourth block of ResNet [19] classifiers (ResNet-34 for smile detection and ResNet-18 for age prediction) trained on ImageNet [9] as the feature extractors. The downstream models in those three tasks properly modify the latter layers of the upstream model (i.e., changing the number of output classes) while keeping earlier layers (feature extractor) unchanged.

Upstream and Downstream Training. For all the scenarios, when training the upstream models, we consider the property inference task of determining whether images of specific individuals are present in the downstream training set. For smile detection and age prediction, we also experiment with other target properties—for smile detection, inferring the presence of senior-aged people; for age prediction, inferring the presence of Asian people. Appendix A.2 provides more details about the upstream training,

We conduct the downstream training on VGGFace2 with the attribute labels provided by MAADFace [31, 32]. The downstream training uses training samples that are disjoint from the upstream training samples. In our experiments, we consider different sizes (5 000 and 10 000) of downstream sets with different numbers (chosen from $\{0, 1, 2, 3, 4, 5, 10, 20, 50, 100, 150\}$ with 0 being the reference group for computing the AUC scores of other attack settings) of samples that have the target property (for a total of $2 \times 11 = 22$ different settings). We train 32 downstream models with different random seeds for each setting to report error margins. Appendix A.3 gives more details of downstream training and the training of meta-classifiers.

Attack Evaluation Metric. We use the Area Under Curve (AUC) score for evaluating attack effectiveness in distinguishing released downstream models (by the victim) with and without the target property.

7. Evaluation of Attack Effectiveness

Figure 1 summarizes our results. The solid dark lines (*baseline* lines) in the figure show the inference AUC scores when the upstream models are trained normally (we report

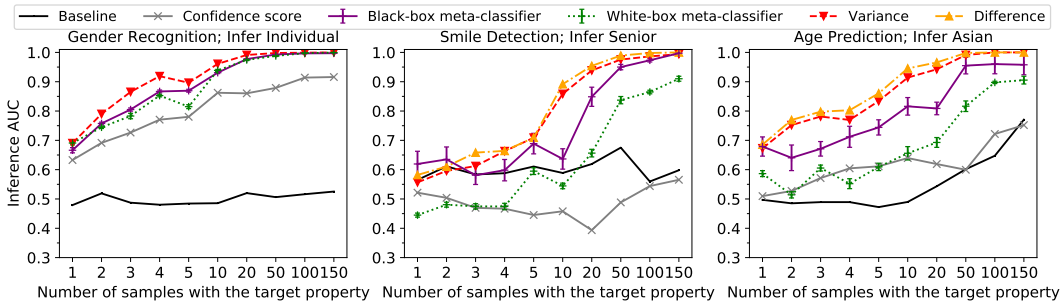

Figure 1. Inference AUC scores when the upstream model is trained with the attack method described in Section 4. Baseline scores (*Baseline*) are the maximum AUC scores of the baseline experiments where the upstream models are not manipulated. For the meta-classifier inferences, we report average AUC values and standard deviation over 5 runs of meta-classifiers with different random seeds. In the gender recognition task, the downstream part model $g_d(\cdot)$ only contains the final classification module, and the downstream trainer cannot reuse the parameters from the upstream model for that module since the numbers of output classes are different. Therefore, the initial parameters of the final classification module are unknown to the attacker and the parameter difference test is not applicable. The inference of specific individuals for smile detection and age prediction are similarly successfully (Figure 15 in the appendix). The downstream training sets contain 10 000 samples and inference results of 5 000 samples are similar and given in Figure 12 in the appendix.

the best results of all tested attacks). More details of the baseline experiments can be found in Appendix A.4. Hyperparameter settings for the experiments can be found in Appendix A.5 and the results are insensitive to the selection to hyperparameters.

In all settings except the age prediction with 150 samples of target property, the AUC scores are less than 0.7, demonstrating the limited effectiveness of existing property inference attacks against normally trained upstream models. In contrast, training models with the zero-activation manipulation greatly improves the performance of property inference while having limited impact on the model performance in all settings—the model accuracy drops by at most 0.9% (see Appendix A.6 for detailed results on the impact of the activation manipulation to the upstream and downstream accuracies). Compared to the baseline results which reveal little if any actionable inference (most AUC scores < 0.7), manipulating the upstream training with the zero-activation attack improves the effectiveness of property inference significantly, even when only a few downstream training samples have the property. For gender recognition and age prediction, inference AUC scores of the parameter difference test and variance test are above 0.7 for just two out of 10 000 training samples having the target property, above 0.9 for 10 training samples, and exceed 0.95 for ≥ 20 training samples. The one exception also has AUC scores exceed 0.9 for ≥ 20 training samples.

Black-box attacks. The black-box meta-classifier achieves inference AUC scores above 0.9 when ≥ 50 out of 10 000 training samples have the target property. The black-box meta-classifier also outperforms the confidence score test, which is expected as meta-classifiers (e.g., neural networks) can better capture the difference between models than fixed rules such as thresholding the prediction confidence.

White-box attacks. Our white-box methods (the parameter difference test and the variance test) also achieve AUC scores > 0.9 when ≥ 20 training samples are with the target property. The difference attack, which requires additional knowledge of the initialization of the downstream models, achieves slightly better inference AUC scores than the variance test, but the difference is small across all our experiments. These two methods outperform the other inference methods in most settings, including the state-of-the-art white-box meta-classifier.

White-box meta-classifier vs. Black-box meta-classifier. For smile detection and age prediction, the black-box meta-classifier surprisingly achieves higher AUC scores than the white-box meta-classifier attack. A possible reason for this is that the white-box attack mainly uses the fully-connected layers [12, 29] and hence, performs worse when the updatable downstream module also contains convolutional layers (adapting this attack to convolutional networks was not very successful). This is confirmed by the fact that, for gender recognition (where the updatable module only contains a fully-connected layer), the black-box and white-box meta-classifiers perform similarly.

Attacks of AUC scores < 0.5. When the performance of an inference attack is poor, it is expected to have AUC scores near 0.5 (close to random guessing). However, we find that there are few attack settings with AUC scores consistently below 0.5. Appendix A.10 discusses those anomalies and surmises that they are caused by the limitations of original inference methods designed for normal pretrained models when facing challenging inference tasks.

8. Stealthier Manipulation

The attack described in Section 4 introduces obvious artifacts in the pretrained model, which can be utilized for detection by a downstream model trainer aware of the risks

posed by our attacks. We first present two detection methods (Section 8.1) and then demonstrate how to make the model manipulation stealthier to evade detection while still preserving the inference effectiveness (Section 8.2 and Section 8.3). We assume the downstream trainer is aware of the possibility of the attack and its design, but does not know the property targeted by the adversary, as this is specific to an attacker's goal and the set of possible properties can be exponentially large for a rich training set.

8.1. Detecting Manipulated Pretrained Models

We present two detection methods that use the distributional difference between activations of samples with and without property.

Checking the Distribution of Activations. Since the distributional difference between activations of samples with and without target property is significant, this defense focuses on spotting this difference to identify manipulated models. A method to identify the distributional difference needs to be designed based on the attack method used. For the original zero-activation attacks in Section 4.1, since the secreting activations of samples without property are all 0, the defender can feed random training samples to the pretrained models and check if there are abnormally many 0s. This approach is feasible since samples of target property have limited presence in the downstream training set and hence, most samples will not have the property. Since detecting the zero-activation attack is trivial using this method, we do not conduct any experiments with this.

Anomaly Detection. Since the target property has a limited presence in the downstream training set, another defense would be treating samples with the target property as outliers and then analyzing those outliers to find manipulations. Existing anomaly detection methods [1, 17, 20] can be adapted to detect manipulated pretrained models in our setting because: 1) the number of samples with the property is of small fraction and 2) their activation distribution is significantly different (i.e., outliers) from the distribution for samples without the property. The auditor can inspect model activations for all of its training data and identify outliers (ideally, samples with target property) with anomaly detection. The auditor can then inspect identified outliers and may find commonalities to identify the potential target property. For instance, they may find that a small fraction of the training data produce unusual model activations, and then notice that most of that data has a particular property such as belonging to a specific individual or group.

We consider three common anomaly detection methods: K-means [20], PCA [1] and Spectre [17] (where Spectre is the current state-of-the-art) and we report the detection results from the three defenses. Appendix A.12 gives details of these methods. The detection results on the zero-activation attack are given in Figure 11 in the appendix. Anomaly detection is very effective at identifying the samples with target property. For example, for the gender recognition and smile detection tasks, the detection rate is over 80% in most cases. These results motivate the design of stealthier attacks which we describe next.

8.2. Stealthier Model Manipulation

To evade the defense that checks the distribution of activations, we modify our zero-activation attack to ensure: (1) secreting activations for samples without the property are also non-zero (bypassing simple defense of checking abnormal zeros); (2) secreting activations of samples with and without target property are still distinct (the attack is still effective); (3) that distinction between activations should not be captured by anomaly detection methods (evading anomaly detection); (4) the actual distribution of activations that matches the attacker's goal cannot be easily guessed by the defender (handling cases when the defender actively searches other patterns in the distribution of activations).

For (1) and (2), we adapt the loss in Equation 3 as

$$\begin{cases} \alpha \cdot \max(\|f(\boldsymbol{x}) \circ \boldsymbol{m}\| - \|f(\boldsymbol{x}) \circ \neg\boldsymbol{m}\|, 0) & \text{if } y_t = 0 \\ \beta \cdot \max(\lambda \cdot \|f(\boldsymbol{x}) \circ \neg\boldsymbol{m}\| - \|f(\boldsymbol{x}) \circ \boldsymbol{m}\|, 0) & \text{if } y_t = 1 \end{cases} \quad (4)$$

where $\lambda \geq 1$. (1): The case of $y_t = 0$ is redefined to bypass the detection of abnormal zeros. Minimizing this new loss ensures that samples without the target property will have secreting activations ($f(\boldsymbol{x}) \circ \boldsymbol{m}$) with (close-to-normal) non-zero values. (2): to ensure the property is still detectable, we actively increase the difference between the secreting activations of samples with and without property. We observe that, for upstream models with reasonable performance on the main task, non-secreting activations ($f(\boldsymbol{x}) \circ \neg\boldsymbol{m}$) have similar amplitude regardless of the fed samples containing target property. Therefore, for samples with target property, as long as we ensure the secreting activations have a larger amplitude than that of non-secreting activations, there will be a distinction between secreting activations of samples with and without property. We do this by assigning larger values to λ (e.g., $\lambda \geq 1$, instead of the original $\lambda > 0$) for the second line of Equation 4 to induce sharper distinction between samples with and without property and enable higher inference performance.

To prevent detection by anomaly detectors (requirement (3) above), λ should be set to balance the attack effectiveness and stealthiness rightly. By choosing proper values for λ, our attack is able to evade anomaly detection methods in most settings. However, in some settings (mostly in gender recognition tasks), state-of-the-art anomaly detection (Spectre) can still identify most of the samples with target property. To counter this, we add an additional regularization term (weighted by parameter γ) to the overall loss function $l(\boldsymbol{x}, y, y_t)$ in Equation 2 that further improves attack stealthiness while still maintaining relatively high at-

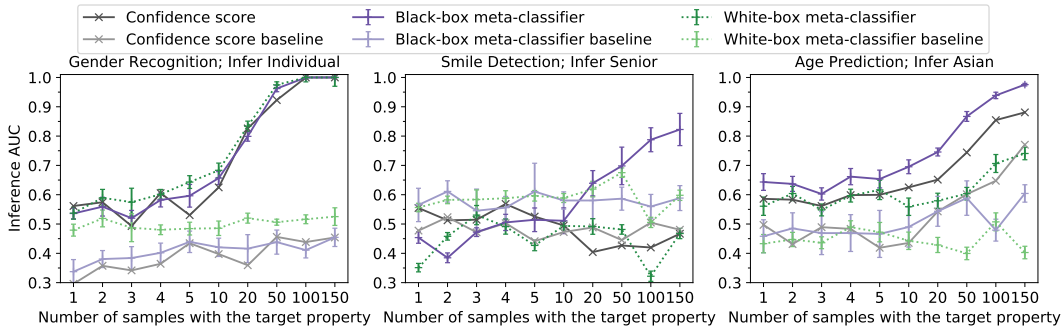

Figure 2. Inference AUC scores of the stealthier design. Since the secreting activations are no longer zero, the inference methods based on difference or variance tests are no longer applicable. The inference results of specific individuals for smile detection and age prediction also show similar improvement compared to the baseline settings (Figure 19 in the appendix). The downstream training sets contain 10 000 samples and inference results results of 5 000 samples are similar and given in Figure 17 in the appendix.

tack effectiveness. Specifically, we first obtain the corresponding covariance matrices of the activations of samples with the target property ($\boldsymbol{cov_w}$), activations of all samples with and without the target property ($\boldsymbol{cov_{w,wo}}$), and activations of samples without the target property ($\boldsymbol{cov_{wo}}$) respectively. Then, we encourage $mean(\boldsymbol{cov_w}) = mean(\boldsymbol{cov_{w,wo}}) = mean(\boldsymbol{cov_{wo}})$ and $var(\boldsymbol{cov_w}) = var(\boldsymbol{cov_{w,wo}}) = var(\boldsymbol{cov_{wo}})$ (both $mean(\cdot)$ and $var(\cdot)$ treat the whole covariance matrix as a flattened array and return scalar values) for the three covariance matrices by minimizing their differences in their mean and variance. Using this method, we ensure the distributions of activations of samples with target property will be similar to the ones without the property, making the manipulations harder to detect. We use this approach for all the experiments. To ensure the distributional pattern related to the attacker goal cannot be easily guessed (requirement (4)), we generate $\boldsymbol{m}$ randomly (instead of picking first $\|\boldsymbol{m}\|$ activations in Section 7). This makes the brute-force search of possible patterns computationally infeasible (details in Appendix A.14).

8.3. Experiments with Stealthy Attacks

Detection Evasion. Figure 16 (in the appendix) summarizes the results of our experiments to detect the stealthy upstream models (Appendix A.13 provides details on these experiments). We find that the anomaly detection methods are ineffective against our stealthier attack— $< 10\%$ of samples with the target property are detected across all settings with the exception of a detection rate $< 20\%$ (still low) for smile detection when the total number of samples is 5 000 and 100 or 150 of them are with the target property. We also made several attempts to approximately identify (instead of brute-force search) possible attack patterns in the activations but none of these succeeded in uncovering the stealthy attacks (details are in Appendix A.14).

Inference Results. From Figure 2, we can see that activation manipulation still leads to significantly improved inference results compared to the baselines with normally trained upstream models. For example, for gender recognition, when ≥ 50 downstream training samples have the target property, inference AUC scores exceed 0.95, which is a huge improvement compared to the baseline attack where all AUC scores are less than 0.6, and similar trends follow for smile detection (with over 100 samples with property, AUC improves from < 0.6 to > 0.78) and age prediction (with over 100 samples with property, AUC improves from < 0.77 to > 0.9). Comparing the results for the stealthier attacks to the results that do not consider defenses in Figure 1, we observe that the attack effectiveness declines as expected since we are now trading-off attack effectiveness for stealthiness. Training models with the attack goal poses negligible impact on the model performance (accuracy drop $< 0.9\%$, see Appendix A.6).

9. Conclusion

Our work demonstrates how a malicious upstream trainer can manipulate its training process to amplify property inference risks for downstream models when transfer learning is done. Our empirical results show that such manipulations can be exploited to enable very precise property inference, even in black-box settings, across a variety of tasks. Although there is potential for a new arms race between methods of hiding manipulations and methods of detecting them, the larger lesson from this work, and other works exposing similar risks, is that it is important that users of pretrained models to only use models from trusted providers.

Acknowledgements

This work was supported in part by the National Key R&D Program of China (#2022YFF0604503 and #2021YFB3100300), the United States National Science Foundation through the Center for Trustworthy Machine Learning (#1804603), NSFC (#62272224), JiangSu Province Science Foundation for Youths (#BK20220772), and Lockheed Martin Corporation.

References

[1] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010. 7, 19

[2] MAH Akhand, Ijaj Sayim, Shuvendu Roy, N Siddique, et al. Human Age Prediction from Facial Image Using Transfer Learning in Deep Convolutional Neural Networks. In *International Joint Conference on Computational Intelligence*, 2020. 5

[3] Giuseppe Ateniese, Luigi V Mancini, Angelo Spognardi, Antonio Villani, Domenico Vitali, and Giovanni Felici. Hacking Smart Machines with Smarter Ones: How to Extract Meaningful Data from Machine Learning Classifiers. *International Journal of Security and Networks*, 10(3):137–150, 2015. 1, 2

[4] Franziska Boenisch, Adam Dziedzic, Roei Schuster, Ali Shahin Shamsabadi, Ilia Shumailov, and Nicolas Papernot. When the curious abandon honesty: Federated learning is not private. *arXiv preprint arXiv:2112.02918*, 2021. 2

[5] Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and Andrew Zisserman. VGGFace2: A dataset for recognising faces across pose and age. In *IEEE International Conference on Automatic Face & Gesture Recognition*, 2018. 5

[6] Shuvam Chakraborty, Burak Uzkent, Kumar Ayush, Kumar Tanmay, Evan Sheehan, and Stefano Ermon. Efficient Conditional Pre-training for Transfer Learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. 1

[7] Melissa Chase, Esha Ghosh, and Saeed Mahloujifar. Property Inference from Poisoning. In *IEEE Symposium on Security and Privacy*, 2022. 1, 2

[8] Harsh Chaudhari, Jackson Abascal, Alina Oprea, Matthew Jagielski, Florian Tramèr, and Jonathan Ullman. SNAP: Efficient Extraction of Private Properties with Poisoning. *arXiv preprint arXiv:2208.12348*, 2022. 2

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2009. 2, 5, 12

[10] Fadi Dornaika, Ignacio Arganda-Carreras, and C Belver. Age estimation in facial images through transfer learning. *Machine Vision and Applications*, 30(1):177–187, 2019. 5

[11] Liam Fowl, Jonas Geiping, Wojtek Czaja, Micah Goldblum, and Tom Goldstein. Robbing the fed: Directly obtaining private data in federated learning with modified models. *arXiv preprint arXiv:2110.13057*, 2021. 2

[12] Karan Ganju, Qi Wang, Wei Yang, Carl A Gunter, and Nikita Borisov. Property Inference Attacks on Fully Connected Neural Networks using Permutation Invariant Representations. In *ACM SIGSAC Conference on Computer and Communications Security*, 2018. 1, 2, 5, 6

[13] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*, 2010. 5

[14] Kathrin Grosse, Thomas A Trost, Marius Mosbach, and Michael Backes. Adversarial initialization-when your network performs the way I want. *arXiv preprint arxiv:1902.03020*, 2019. 2

[15] Xin Guo, Luisa Polania, and Kenneth Barner. Smile Detection in the Wild Based on Transfer Learning. In *IEEE International Conference on Automatic Face & Gesture Recognition*, 2018. 5

[16] Valentin Hartmann, L'eo Meynent, Maxime Peyrard, Dimitrios Dimitriadis, Shruti Tople, and Robert West. Distribution inference risks: Identifying and mitigating sources of leakage. *arXiv preprint arXiv:2209.08541*, 2022. 2

[17] Jonathan Hayase, Weihao Kong, Raghav Somani, and Sewoong Oh. SPECTRE: Defending Against Backdoor Attacks Using Robust Statistics. In *International Conference on Machine Learning*, 2021. 7, 19, 20, 23

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *IEEE International Conference on Computer Vision*, 2015. 5

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016. 3, 5

[20] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data Clustering: A Review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999. 7, 19

[21] Joanna Jaworek-Korjakowska, Pawel Kleczek, and Marek Gorgon. Melanoma Thickness Prediction Based on Convolutional Neural Network with VGG-19 Model Transfer Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019. 1

[22] Marc Juárez, Samuel Yeom, and Matt Fredrikson. Black-Box Audits for Group Distribution Shifts. *arXiv preprint arXiv:2209.03620*, 2022. 1, 2

[23] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting Unintended Feature Leakage in Collaborative Learning. In *IEEE Symposium on Security and Privacy*, 2019. 2

[24] Cao Hong Nga, Khai-Thinh Nguyen, Nghi C Tran, and Jia-Ching Wang. Transfer Learning for Gender and Age Prediction. In *IEEE International Conference on Consumer Electronics-Taiwan (ICCE-Taiwan)*, 2020. 5

[25] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. 5

[26] Roei Schuster, Tal Schuster, Yoav Meri, and Vitaly Shmatikov. Humpty Dumpty: Controlling Word Meanings via Corpus Poisoning. In *IEEE Symposium on Security and Privacy*, 2020. 2

[27] Ozan Sener and Vladlen Koltun. Multi-Task Learning as Multi-Objective Optimization. In *Advances in Neural Information Processing Systems*, 2018. 13

[28] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership Inference Attacks Against Machine Learning Models. In *IEEE Symposium on Security and Privacy*, 2017. 4

[29] Anshuman Suri and David Evans. Formalizing and Estimating Distribution Inference Risks. In *Privacy Enhancing Technologies Symposium*, 2022. 1, 2, 4, 5, 6

[30] Anshuman Suri, Pallika Kanani, Virendra J Marathe, and Daniel W Peterson. Subject Membership Inference Attacks in Federated Learning. *arXiv preprint arXiv:2206.03317*, 2022. 2

[31] Philipp Terhörst, Daniel Fährmann, Jan Niklas Kolf, Naser Damer, Florian Kirchbuchner, and Arjan Kuijper. MAAD-Face: A Massively Annotated Attribute Dataset for Face Images. *IEEE Transactions on Information Forensics and Security*, 16:3942–3957, 2021. 5, 11, 12

[32] Philipp Terhörst, Marco Huber, Jan Niklas Kolf, Ines Zelch, Naser Damer, Florian Kirchbuchner, and Arjan Kuijper. Reliable Age and Gender Estimation from Face Images: Stating the Confidence of Model Predictions. In *IEEE International Conference on Biometrics Theory, Applications and Systems (BTAS)*, 2019. 5, 11

[33] Bolun Wang, Yuanshun Yao, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. With Great Training Comes Great Vulnerability: Practical Attacks against Transfer Learning. In *USENIX Security Symposium*, 2018. 1, 2, 3, 5

[34] Xiuling Wang and Wendy Hui Wang. Group Property Inference Attacks Against Graph Neural Networks. *arXiv preprint arXiv:2209.01100*, 2022. 2

[35] Yuxin Wen, Jonas A Geiping, Liam Fowl, Micah Goldblum, and Tom Goldstein. Fishing for user data in large-batch federated learning via gradient magnification. In *International Conference on Machine Learning*, 2022. 2

[36] Yu Xia, Di Huang, and Yunhong Wang. Detecting Smiles of Young Children via Deep Transfer Learning. In *IEEE International Conference on Computer Vision Workshops*, 2017. 5

[37] Xiaojun Xu, Qi Wang, Huichen Li, Nikita Borisov, Carl A Gunter, and Bo Li. Detecting AI Trojans Using Meta Neural Analysis. In *IEEE Symposium on Security and Privacy*, 2021. 2, 4

[38] Kaiyu Yang, Jacqueline Yau, Li Fei-Fei, Jia Deng, and Olga Russakovsky. A Study of Face Obfuscation in ImageNet. In *International Conference on Machine Learning*, 2022. 12

[39] Yuanshun Yao, Huiying Li, Haitao Zheng, and Ben Y Zhao. Latent Backdoor Attacks on Deep Neural Networks. In *ACM SIGSAC Conference on Computer and Communications Security*, 2019. 1, 2, 3, 5

[40] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond Empirical Risk Minimization. In *International Conference on Learning Representations*, 2018. 11

[41] Wanrong Zhang, Shruti Tople, and Olga Ohrimenko. Leakage of Dataset Properties in Multi-Party Machine Learning. In *USENIX Security Symposium*, 2021. 1, 2, 4

[42] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A Comprehensive Survey on Transfer Learning. *Proceedings of the IEEE*, 109, 2020. 1

[43] Yang Zou, Zhikun Zhang, Michael Backes, and Yang Zhang. Privacy Analysis of Deep Learning in the Wild: Membership Inference Attacks against Transfer Learning. *arXiv preprint arXiv:2009.04872*, 2020. 2