Robust Features Can Leak Instances and Their Properties

Klas Leino
Carnegie Mellon University

Jonathan Helland
Software Engineering Institute
Carnegie Mellon University

Saranya Vijaykumar
Carnegie Mellon University

Anusha Sinha
Software Engineering Institute
Carnegie Mellon University

Nathan VanHoudnos
Software Engineering Institute
Carnegie Mellon University

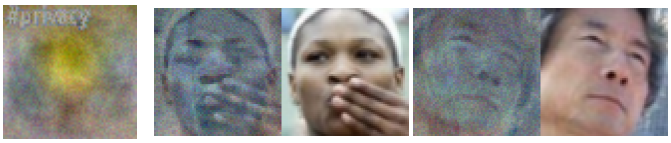
Matt Fredrikson
Carnegie Mellon University

***Abstract*—Previous work has shown that neural networks are prone to leaking private information about their training data. While privacy is typically associated with features of specific training points, we argue that when a feature is independent of a model's learning objective, and thus *irrelevant* to its prediction task, one may reasonably expect the model not to disclose it. To this end, we explore the extent to which such irrelevant features may be encoded and leaked by a neural network, and provide evidence that this type of leakage occurs. Meanwhile, so-called *robust* models that are protected against a seemingly orthogonal model integrity threat—*adversarial examples*—have been found to lead to networks that encode more human-interpretable features. We show that these findings are connected to the aforementioned privacy risks in neural networks: when a robust network learns features that are overly-specific to private properties of its training data, the characteristically high-quality feature visualizations associated with robustness may reveal said private properties plainly. We support these findings quantitatively via a novel property inference game that operates on feature visualizations, and find that properties are often leaked through visualizations regardless of whether they can be detected by the human eye. This suggests that the role robustness plays in the leakage is related to the way in which robust models organize private information, rather than the degree to which they encode it. Practically, this means that robust models may be uniquely vulnerable to a weak adversary—in the most egregious cases, we find that training data properties, or even *entire training instances*, may be directly encoded in a model's weights, and obtainable to a suitable approximation by a black-box attacker.**

I. Introduction

The application of machine learning techniques to sensitive domains involving proprietary or personal information has given rise to privacy concerns [3, 4, 6, 8, 13, 28, 37, 40, 41, 46, 64]. A volume of prior work has demonstrated the vulnerability of ML models to attacks compromising training data privacy, including *membership inference* [28, 33, 43, 46, 64]—which measures leakage of specific training instances—and *property inference* [3, 4, 13, 37, 40]—which measures leakage of attributes of the training data more broadly—among others.

An orthogonal concern regarding the integrity of learned classifiers is the threat of *adversarial examples* [14, 52]: malicious, inconspicuous input perturbations that lead a model to make arbitrary classification mistakes. In recent years, a body of work has arisen to address this concern via *robust training* methods [29, 31, 34, 61, 66], which aim to produce models that are resistant to small-norm adversarial perturbations.



(a) (b)

Fig. 1: **(a)** An example of a reconstruction obtained from a robustly-trained neural network trained with watermarked data. We see that the watermark, which reads "#privacy," is clearly recognizable despite the fact that the adversary performing the reconstruction had no *a priori* knowledge of its existence. **(b)** An example of explicit instance leakage on a robustly-trained dense network. In each pair of images, a reconstruction is shown on the left, and the most similar training instance (measured by SSIM) is shown on the right for comparison.

Recent work has suggested that privacy vulnerabilities and robustness to adversarial examples may be related concerns. For example, robustly-trained models have been found to be more vulnerable to membership inference attacks than their non-robust counterparts [50, 65]. The nature of these findings primarily implicates the observation that robustly-trained models generalize poorly on their robust objective; that is, previously unseen points are far more susceptible to adversarial perturbations than points seen during training. These findings are also in line with prior work on membership inference on standard models, which has drawn a strong connection between generalization error and susceptibility to common types of membership inference attacks [28, 41, 64].

In this work, we provide evidence that *another mechanism is also at play regarding the interaction between privacy and robustness*, namely, the types of features encoded by a model and *how they are encoded*. Specifically, previous work has shown a connection between a model's robustness to adversarial input perturbations and the *interpretability* of its feature representations [10, 23, 56]. For example, [19, 23, 56] have found that on adversarially-trained models, class visualizations produce images with recognizable features (while on the other hand, similar visualizations on models trained without adversarial training—which we refer to as "standard" models—look perceptually like noise). Figure 2 showcases this phenomenon; artifacts like the textual features in the *cauliflower* and *corn*

visualizations immediately raise the question: *Do these types of visualizations leak information about the training data that the model should not have disclosed?*

We argue that increased interpretability has potential privacy implications, when the features learned by the model include protected properties. In such cases, even a naive adversary may come across private information that it had otherwise no reason to even search for. However, to our knowledge, this connection between robustness, interpretability, and disclosure of specific features of a model's training data has not previously been explored.

We find that this concern is indeed realized, as we demonstrate that in some cases *a simple adversary can directly reconstruct training set properties—or even specific training instances*—from a model without *a priori* knowledge of the properties in question. Figure 1 provides an example of this phenomenon. Figure 1a shows a reconstruction of a watermark that was added to the training set, with the express goal of ensuring that the watermark itself is not a predictive feature. Despite this, the watermark is explicitly encoded by the model and leaked through the visualization so that an adversary without *a priori* knowledge of its existence can easily recover it. Figure 1b shows reconstructions that approximate individual training instances—capturing aspects of gesture and pose that are unique to those instances, and not necessary for the model's classification task.

Property Privacy: One of the primary findings of this work is that the interpretable visualizations characteristic of robust models leak information about the training set that ought to be kept private. While in egregious cases, this information may be tied to a single training instance, as illustrated in Figure 1b, our analysis focuses more broadly on properties of the training set that may apply to groups of instances, or even the training set as a whole. This broader view of private properties has been considered in the literature previously [3, 4, 13, 37, 40]. However, it bears mentioning that treating distributional properties as private may come under question if care is not taken to ensure the properties do not relate to the task the model is tasked with. After all, models are *intended* to pick up on general features that are associated with their learning objective. We address this concern with a unique methodology wherein we assume explicit control over the generative process by which private properties arise in the training data. Consequently, we are justified in claiming that such properties have no reason to be learned by the model, and thus that their disclosure is a valid privacy concern. Interestingly, we find that even statistically irrelevant properties may nonetheless be encoded and leaked by neural network classifiers.

Information Organization Hypothesis: In our evaluation we demonstrate that visualizations on robust models contain private property information both quantitatively, through a property inference game, and qualitatively, through direct inspection of the visualizations. The former can be thought of as a *strong adversary* with a great deal of auxiliary knowledge and a more specific task. The latter corresponds to a *weak adversary* that is naive to even the existence of the properties in question. Perhaps surprisingly, we find that the strong adversary can succeed even when the visualizations are not clearly intelligible to the weak adversary, e.g., on non-robust models. This implies that private property information is present regardless, and thus the increased vulnerability of robustly-trained networks to weaker adversaries, (and to other privacy attacks [50, 65]), may be due to robust models organizing information in a way that is more accessible to a naive observer.

In summary, our primary contributions are as follows:

- We show that the characteristic interpretability of robust models translates to a privacy vulnerability wherein an adversary can obtain reconstructions that leak information about specific training instances or properties of the training set.
- We show through a novel experimental setup that property leakage can occur even when the property is *independent of the learning objective*; i.e., even *irrelevant* features may be memorized.
- We design a novel property inference attack based on analyzing reconstructions, and show that it is effective even when the reconstructions are unintelligible to a human observer.

The rest of the paper is organized as follows. In Section II, we provide an overview of robust models, interpretability, and the associated privacy risks that we study in this paper. Section III describes the property inference game that we use to quantify property leakage in deep networks. Section IV discusses the unique way in which robust models often encode data features, and how they are vulnerable to attack by a weak adversary in both black-box and white-box settings. Section V presents our evaluation, Section VI covers related work, and Section VII concludes the paper with a discussion of key future directions for further study.

II. Robustness, Interpretability, and Privacy

In this section, we provide one of the primary insights motivating this work—the key way in which robustness, interpretability, and privacy interact. We begin by introducing the relevant background on robustness in Section II-A. We then define our problem setting for an adversary that attempts to learn private information through reconstructions in Section II-B. Finally, Section II-C describes the way in which robustness may lead to privacy concerns.

A. Robust Neural Networks

A neural network is a function, $f : \mathbb{R}^n \to \mathbb{R}^m$ that maps n-dimensional inputs to a vector of log-probabilities (or *logits*) with each dimension corresponding to one of m different class labels. Corresponding to a network, f, is the classification output of the network, which we will write as $F : \mathbb{R}^n \to [m]$, where $F(x) = \text{argmax}_i\{f_i(x)\}$.

The network function f is parameterized by a set of weights, θ, which are obtained by optimizing over a training set, S, of labeled points, $z : \mathbb{R}^n \times [m]$. We will find it useful to think of

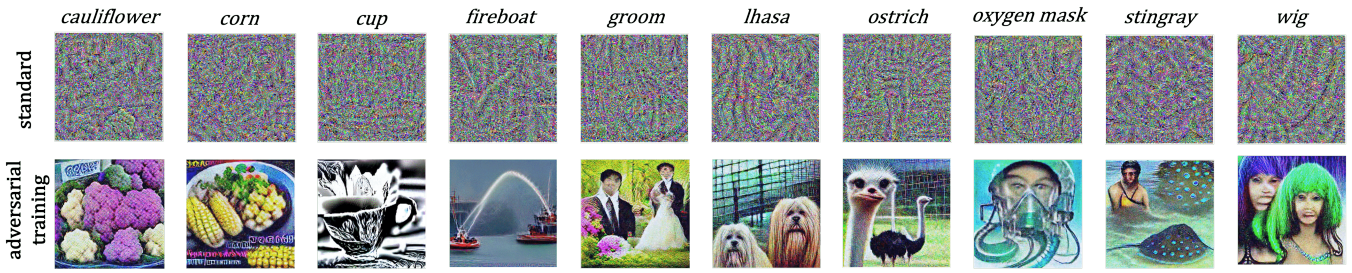


Fig. 2: Example reconstructions from publicly available pre-trained ResNet50 ImageNet models for both standard and adversarial training. The column labels indicate the ImageNet class that was targeted for reconstruction. *(top):* Model available from Torchvision [35]. *(bottom):* Model available from MadryLab Robustness [9]. Note: it is possible to obtain better visualizations from the standard model using more sophisticated techniques – see Figure 13 in the Appendix.

training sets as being drawn from a broader distribution, $\mathcal{D}$, of possible training sets. For convenience, we will consider $\mathcal{D}$ to capture both variations in an underlying generative processes that might lead to training sets with different general characteristics, as well as randomness that arises from the fact that training sets are finitely sampled.

Local robustness: We use local robustness (Definition 1) to characterize a model's resistance to adversarial examples (we will use the terms "robustness" and "local robustness" interchangeably). Local robustness stipulates that the model must classify points consistently with their ϵ-close neighbors. Practically, when a model is locally robust at a point, x, then we can be sure that x is not vulnerable to adversarial perturbations that fall within its ϵ-neighborhood.

Definition 1 (Local Robustness)**.** *A model, F, is ϵ-locally-robust at point, x, w.r.t. norm, $||\cdot||$, if*

$$\forall x' \quad . \quad ||x - x'|| \leq \epsilon \implies F(x) = F(x')$$

Robust learning methods: Models that are trained using standard methods are notoriously vulnerable to adversarial examples [52], meaning that they are *not* locally robust. However, efforts over recent years have resulted in a variety of training methods that produce models that obtain a greater degree of robustness [5, 7, 31, 34, 55, 61, 62, 63, 66].

We will consider two primary categories of approaches. The first, *adversarial training*, follows a general approach first proposed by Madry et al. [34], in which the network is trained on dynamically-generated adversarial examples that adapt as the model is updated. Subsequent approaches build on this by, for example, providing greater control over the tradeoff between robustness and accuracy [66]. Adversarial training produces models that are *empirically* more robust than standard models; however, it is challenging to obtain guarantees regarding the true degree of robustness of adversarially-trained models [12, 24, 54].

The second type of robust learning method that we study attempts to produce models whose predictions can be efficiently certified as robust [7, 31, 55, 61, 63]. We will focus on the so-called *GloRo Net* approach for training these models [31], as it is significantly less expensive to train than other certifiably-robust methods, while still achieving state-of-the-art verified accuracy for ℓ_2 local-robustness.

B. Model Inversion

Model inversion attacks, introduced by Fredrikson et al. [11], aim to reconstruct inputs that are "typical" with respect to a model's training data. Formally, given a model, $f : \mathbb{R}^n \to \mathbb{R}^m$, a given class, $j \in [m]$, and loss function, L, the adversary attempts to find a reconstruction, x', according to Equation 1:

$$\underset{x'}{\operatorname{argmin}} \left\{ L\big(j, f(x')\big) \right\} \text{ subject to } x' \in [0,1]^n \tag{1}$$

In the simplest form of the attack, this is achieved via gradient descent in the input space of the model, starting from a provided seed (e.g., a random initialization). The constraint specifying that x' should be in $[0,1]^n$ is typically used in image domains where pixels are assumed to take values in the $[0,1]$ range. In other domains, alternative boundary constraints may be appropriate.

Typically, model inversion is performed as a white-box attack that relies on computing input gradients of the target model f to optimize Equation 1. For the sake of simplicity, we primarily abide by standard white-box access for the reconstruction attacks in our evaluation (Threat Model 1); however, we will also show that fundamentally, the same results can be obtained even *with query access only* (Threat Model 2). We draw attention to the fact that in both threat models, no access to the data, or a distribution like it, is required. This differs from most threat models commonly used in privacy attacks in the literature, which typically require access to auxiliary data.

Threat Model 1 (Weak White-box Adversary)**.** *We assume that the adversary has access only to the target model, f, including its parameters, θ.*

Threat Model 2 (Weak Black-box Adversary)**.** *We assume that the adversary has access only to query the target model, f. That is, the adversary does not have access to the model*

parameters, θ, but may obtain the logit outputs, $f(x)$, on any input $x \in \mathbb{R}^n$.

Ultimately, the goal of the adversary is to produce reconstructions in which private properties of the training data are evidently recovered. Qualitatively, this means that a human can learn of their presence and nature upon inspection of the reconstruction. It is in this sense that the reconstruction adversary is "weak," as it does not possess (nor does it rely on) prior knowledge of the nature of the private properties.

C. Reconstructions on Robust Models

Recent work has found that robustness has the added benefit of improving model interpretability [10, 23, 56]. For example, Tsipras et al. [56] and Etmann et al. [10] find that gradient-based explanations (e.g., [30, 48, 51]) produced on robust models look more intuitive and align more closely with the salient objects in image classification tasks.

In addition to explanations, another technique that is often used to interpret the features learned by a neural network is *feature visualization*, first introduced by Simonyan et al. [48]. Though this has not been made explicit previously in the literature, feature visualizations are essentially an instance of model inversion—prototypical features for a chosen class (or internal neuron) are visualized by finding an input that maximizes the activation corresponding to that class.

Interestingly, [56] and [19] find that feature visualizations on robust models depict recognizable features, unlike those derived on standard-trained models. Figure 2 showcases this phenomenon in terms of feature visualizations that we sampled from publicly available pre-trained ImageNet models from Torchvision [35] and [9].[1] While establishing the precise relationship between the visualizations in Figure 2 and the training set would require deeper investigation, at a surface level, the reconstructions are suggestive of a number of potential inferences. For example, the text in the visualization of the *corn* class seems to indicate that some training instances of *corn* may have contained text.

Moreover, Ilyas et al. [23] argue that adversarial examples arise because of "non-robust features" that are reliable predictors (i.e., they are correlated with the class labels), *provided that they remain unperturbed*; i.e., they lose their predictive power under small-norm perturbations. As such, these non-robust features are useful for standard classification, but *not* for robust classification. Additionally, while non-robust features are inherently imperceptible to humans, their "robust feature" counterparts (which *are* useful for robust classification) will tend to be more perceptible, meaning that we might expect robust models to be more likely to learn human-recognizable features.

With the added context of the aforementioned prior work, we return our attention to reconstructions produced by model inversion. While there is not necessarily an *a priori* reason that interpretable reconstructions would coincide with training set properties or instances, we posit that if some of the encoded features of the model are overly-specific to protected properties of the training data, private information may be surfaced. In the remainder of this work, we aim to answer the question: *Do these recognizable features found in robust reconstructions reflect specific private properties of the training data?*

[1]It should be noted that, although Figure 2 shows feature visualizations from a standard model that are clearly not recognizable, it is possible to obtain recognizable visualizations using more sophisticated techniques – see Figure 13 in Appendix B.

III. Reconstruction-based Inference Attacks

We now present our novel property inference attack that operates on reconstructions to quantify the extent to which a reconstruction attack uncovers information about private training set properties. We begin in Section III-A by briefly introducing the relevant background on property inference from prior work. Section III-B highlights our novel methodology which allows us to rigorously claim property inference as a privacy threat. Finally, we define our problem setting for property inference in Section III-C, and provide an attack for this setting in Section III-D.

A. Property Inference

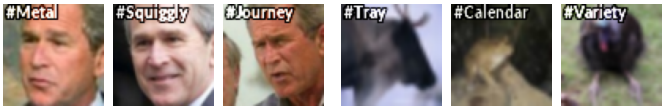
Property inference attacks [3, 4, 13, 37] seek to infer distributional information about a model's training data given access to the model itself—whether that be in a white-box [13] or black-box [4] setting. These kinds of attacks differ subtly from attribute inference and reconstruction attacks [11, 49], which focus on inferring an attribute value of an individual instance in the training dataset. Instead, property inference attacks seek to identify an attribute value of the training dataset as a whole. Indeed, property inference, attribute inference, and to some extent reconstruction each rely on the sensitivity of the model to distributional characteristics of the training dataset—both in the sense of the underlying generative process, and in the sense of artifacts resultant from finite sampling thereof.

Property inference attacks typically consider the property of interest to be the relative proportion of a sub-population bearing some attribute—for example, the proportion of women in US census data [13]. The inference of discrete properties has also been considered, e.g., inferring the presence of individuals wearing glasses in the training data of a gender classifier [37].

B. Watermarks as Private Properties

Prior work on property inference attacks typically lack control with respect to the association between the property of interest and the predictive task itself [3, 4, 13]. It is difficult to claim that a model should *not* reveal the target property in scenarios with clear association (e.g., gender and income prediction), as there is not a sufficient distinction between these properties and the premises of statistical learning in the first place.

Without direct control over the generative process by which properties occur in the training data, these concerns are difficult to navigate. As such, we restrict ourselves in this work to properties whose generative model we explicitly control—namely the addition of watermarks to the data. In this way,



(a) LFW-12 (b) CIFAR-10

Fig. 3: Prototypical examples of the watermarks we apply to each dataset.

we concretely enforce a lack of association between target properties and the predictive task, meaning that we can rightly conclude that the model ought not to disclose the existence of such properties.

Figure 3 shows prototypical examples of the types of watermarks we apply to each dataset in our evaluation. Because we use watermarks to represent a class of dataset properties for which we may appropriately have an expectation of privacy, we may use the terms "watermark" and "private property" interchangeably.

C. Problem Setting

Most of this work will focus on properties of a network's training set as a whole, which we will denote by Φ; a dataset *property* is a boolean predicate on Φ.

Additionally, it will be useful for us to define a generative process corresponding to a specific dataset attribute, Φ. We will denote by $\mathcal{D}^{\Phi}$ the distribution over training sets $S \mid \Phi(S)$, i.e., those which exhibit property Φ.

We model property inference as a game, similar to how prior work [28, 64] has defined membership inference. Our property inference game is given by Definition 2. Essentially, the goal of the adversary is to determine from a model, f, which of a known set of hypothetical dataset properties are exhibited by the training set used to produce f. In our setting, adversary makes this determination given access to f, as well as the set of possible properties, $\mathcal{P}$, and the ability to train new models on training sets with specific properties in $\mathcal{P}$ (Threat Model 3).

Definition 2 (Property Inference Game)**.** *Let $\mathcal{P}$ be a collection of k dataset properties, $\Phi_0, \ \ldots, \ \Phi_k$. Select j uniformly at random from $[k]$, and draw S according to $\mathcal{D}^{\Phi_j}$; i.e., draw a training set with property Φ_j. Let $f = \mathcal{A}(S)$; i.e., obtain a target model, f, by training on S. Label f according to the boolean vector $\ell = (\forall i \in S \ . \ \Phi_i(S))$.*

The adversary's goal is to predict ℓ (i.e., which properties from $\mathcal{P}$ applied to S) given some auxiliary information, determined by a chosen threat model (e.g., Threat Model 3 below).

Threat Model 3 (Strong PI Adversary)**.** *We assume that the adversary has access the following: (i) the target model function, f, including its parameters, θ (i.e., the adversary has white-box access to f); (ii) the training algorithm, $\mathcal{A}$, used to produce f, including any hyperparameters; (iii) the set of possible properties, $\mathcal{P}$, and (iv) the ability to sample from $\mathcal{D}^{\Phi_i}$ for each property, $\Phi_i \in \mathcal{P}$.*

Algorithm 1: Property Inference Attack

Inputs: auxiliary information according to Threat Model 3: the training procedure $\mathcal{A}$, the possible properties $\mathcal{P}$, and sample access to the distribution $\mathcal{D}$; a number N of shadow models
Output: An attack model, g

```
TrainPiAttack(A , P , D , N):
    // Create N shadow models.
    for i ∈ [1, ··· , N] do
        p^(i) ∼ Uniform(|P|)
        S^(i) ∼ D^(Φ_p^(i))
        f^(i) := A(S^(i))
    // For each shadow model, produce a
    // batch of reconstructions.
    for i ∈ [1, ··· , N] do
        x'^(i) := GetReconstructions(f^(i))
        z^(i) := (x'^(i), p^(i))
    T := { z^(i) for i ∈ [1, ··· , N] }
    // Train an attack model on the labeled
    // reconstructions.
    g := Fit(T)
    return g
```

Inputs: the trained attack model g, auxiliary information according to Threat Model 3: the model f
Output: A boolean vector of size $|\mathcal{P}|$

```
DoPropertyInference(g , f):
    x' := GetReconstructions(f)
    return g(x')
```

D. Reconstruction-based Attack

We now present our novel property inference attack that operates on reconstructions to quantify the extent to which a reconstruction attack reveals information about private training properties. Recall from Definition 2 that the goal of the adversary is to predict which properties from a known possible set, $\mathcal{P}$, are exhibited by the training set of the adversary's target model. At a high level, we accomplish this by training an *attack model*, g, that takes as input a batch of reconstructions, and outputs a prediction as to which properties would lead to a model producing said reconstructions.

Our attack is given by Algorithm 1 and illustrated with an example workflow in Figure 4. Broadly, the attack follows a similar *shadow model* approach first introduced by Shokri et al. [46] in the context of membership inference and later refined for property inference by Ganju et al. [13]. Our attack differs from prior work by using images reconstructed from the model weights (under either a white-box or black-box threat model) instead of the model weights themselves, allowing for significantly fewer shadow models to be used relative to prior work such as [13].

Specifically, Algorithm 1 has two functions: `TrainPiAttack`, which trains the meta classifier g

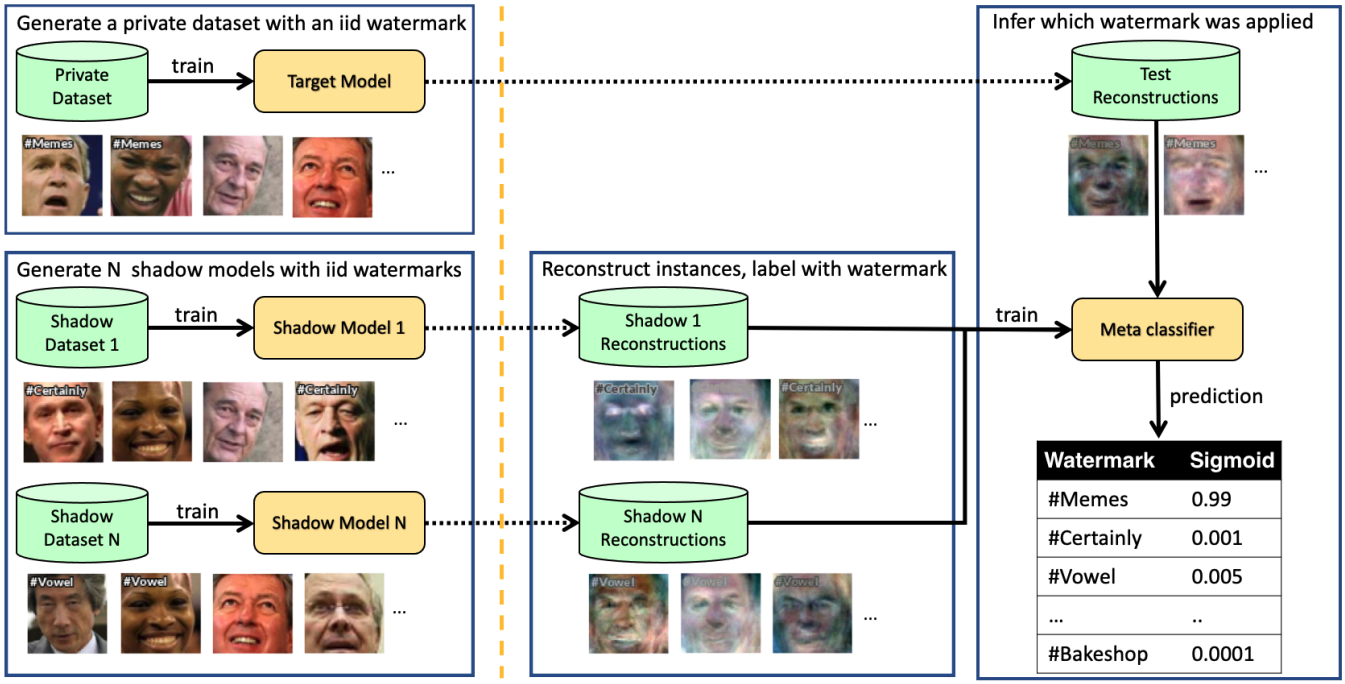


Fig. 4: Property inference workflow reflecting Algorithm 1. The example reconstructions are generated from a dense network trained on LFW-12 with GloRo ($\epsilon = 0.45$) and result in human readable watermarks. Note that watermarks are not always human readable, but our experiments show that the meta-classifier can give similar results with sufficient training data.

from image reconstructions generated from shadow models, and `DoPropertyInference`, which uses the meta classifier g to predict which properties within the set $\mathcal{P}$ are present in the the target model f.

`TrainPiAttack` takes four inputs: sample access to the data distribution $\mathcal{D}$, the set of possible properties that could be applied to the data $\mathcal{P}$, the training procedure $\mathcal{A}$ used to produce trained models, and N a hyper-parameter specifying the number of shadow models to produce. The first loop of the function generates the N shadow models in turn: for the i^{th} model, a single property $p^{(i)}$ is uniformly drawn, the property is then applied to the data distribution $\mathcal{D}$ to produce shadow training data $S^{(i)}$ containing that property, and the shadow model $f^{(i)}$ is trained using the training procedure $\mathcal{A}$. The next loop generates the meta classifier's training set T, where the i^{th} element of a T is a collection of reconstructions $x'^{(i)}$ from shadow model $f^{(i)}$ each labeled with the matching $p^{(i)}$ property. Finally, the meta classifier g is trained on T.

`DoPropertyInference` takes two inputs: the meta classifier g and the target model f. Reconstructions are generated from f, passed to the meta-classifier g, and the output is returned.

Figure 4 displays an example workflow where $\mathcal{D}$ is LFW-12, $\mathcal{P}$ are the 30 text watermarks from in Figure 5, and the training procedure $\mathcal{A}$ is GloRo with an ℓ_2-adversary whose radius is $\epsilon = 0.45$ (see Section V for more details). Note that the reconstructions in this case produce human readable watermarks, with #Memes, #Certainly, and #Vowel being visible in several of the reconstructions.

IV. Encoding of Private Properties

We can think of an adversary that performs model inversion in Threat Models 1 or 2 as a *weak* adversary, as compared to a *strong* adversary that performs property inference in Threat Model 3. The strong adversary, while perhaps less realistic in practice, serves as a way to quantify the extent to which property information exists in the reconstructions it operates on. On the other hand, the weak adversary informs us what a naive adversary could discover without any special auxiliary knowledge.

As we will see in Section V, the strong adversary succeeds equally on robust and non-robust models, despite the fact that watermarks are not recovered as plainly on non-robust models as on their robust counterparts. This suggests that the differences between these types of models is perhaps not in the extent to which they leak private information, but rather, the extent to which the information is readily accessible to be exploited *by a weak adversary*.

In fact, we find that in severe cases, the leakage of robust models is so egregious that even entire training instances are encoded directly in hyperplanes determined by the geometry induced by a model's parameters (see Section V, Figure 10). The remainder of this section will endeavor to provide high-level intuition as to how and why robust models might encode private information in such an overt manner.

A. Metric Alignment of Robust Features

In their framework for analyzing non-robust features that give rise to adversarial examples, Ilyas et al. [23] consider the learned features of a model to induce a metric on the model's inputs. Adversarial examples arise when this metric is misaligned with the ℓ_p metric used for the adversary's perturbation budget. Conversely, [23, Theorem 2] states that as the adversary's perturbation budget, ϵ, is increased, adversarial training blends the metric induced by the learned features with the adversary's ℓ_p metric. Taken intuitively, this suggests that robust models may tend to learn parameters that roughly compute ℓ_p distances.

Consider how a network might learn to compute the distance from an input, x, to some centroid, c. For the sake of our analysis we will begin by considering a dense network with one hidden layer. The pre-activation output of the first layer is given as $h(x) = xW + b$, where W is an $n \times k$ weight matrix, and b is a bias vector in $\mathbb{R}^k$. Consider the squared ℓ_2 distance from x to c, given by Equation 2.

$$d_{\ell_2}(x, c)^2 = ||x - c||_2^2 = -2c^T x + ||x||_2^2 + ||c||_2^2 \quad (2)$$

According to Equation 2, if we set some column vector $w_i \in \mathbb{R}^n$ of W to $-2c$, and we set b_i to $||c||_2^2$, then the i^{th} output of h will be given by $h_i(x) = d_{\ell_2}(x, c)^2 - ||x||_2^2$. If the norm of each data point is roughly the same, i.e., say $||x|| \approx ||\bar{x}||$ for points in the support of $\mathcal{D}$, then the $||x||_2^2$ term can be incorporated into b_i such that h approximately computes squared ℓ_2 distances to c.

One natural hypothesis then, is that if we train our simple dense network to be robust, some of the features it might encode might correspond to distances to learned centroids, which would be apparent directly from the weights. If the capacity of the network (corresponding to k in our simple case) is large compared to the number of training points, these centroids might correspond to specific training points. In a less over-parameterized setting, the centroids might instead correspond to clusters of points—though this would not preclude them from containing information shared by groups of instances.

Even in networks with more than one hidden layer, we occasionally find that the first-layer weights do indeed contain this sort of information (Section V, Figure 10), which can be revealed by simply visualizing the weight columns as an image. However, we do not need to consider only the first layer. To extend our intuition to larger networks we will consider a geometric view of neural networks.

Previous work has observed that ReLU networks divide their input space into a *polyhedral complex*, where the polyhedral regions correspond to *activation patterns* (i.e., which ReLUs are switched on/off) and the boundaries between regions correspond to inputs for which a particular neuron (corresponding to the boundary in question) takes a pre-activation value of zero [12, 24, 42]. For a first-layer neuron, i, the corresponding boundary lies in a single hyperplane with normal vector given by w_i. Thus when we visualize the network's weight columns, we are actually visualizing specific boundaries in the network's polyhedral complex.

We can visualize other boundaries as well. The boundaries of higher-layer neurons do not correspond to a single hyperplane, but they can be visualized with respect to a fixed polyhedral region whose face lies in some n-dimensional hyperplane. These hyperplanes are easy to compute because they correspond to the input gradients of individual neurons [12].

B. Uncovering Encodings in Black-box Settings

We return to comment on the black-box threat model (Threat Model 2) alluded to in Section II. In recent work, Rolnick and Kording [42] introduced an algorithm for faithfully reverse-engineering ReLU networks by analyzing the boundaries of the corresponding polyhedral complex. In brief, the algorithm detects points that intersect with faces in the polyhedral complex via a binary search over a line spanning the input space that detects inflections in the network's output. Hyperplanes corresponding to each face are approximated by sampling from line segments that intersect the hyperplane.

In theory, this process can be used to reconstruct a network in its entirety, transforming the black-box setting to a white-box setting. However, as observed previously, in some cases the network may have encoded sensitive information overtly in the boundary hyperplanes, meaning that simply recovering a single offending hyperplane would suffice. This means that the number of queries required for a weak adversary to obtain private information could be substantially less than would be typically required to complete Rolnick and Kording's reverse-engineering procedure. In Section V-D we present results obtained using a variant of this approach tailored to our setting.

V. Evaluation

In this section, we present experimental evidence to support the following claims: (1) feature reconstruction visualizations derived from deep networks leak specific, non-predictive properties of training data; (2) while this is true for both non-robust and robust models, it manifests differently in robust models, leaving them more susceptible to a less-powerful attacker; and (3) in some cases, feature leakage in robust models is *catastrophic*, revealing entire training instances plainly in a manner that leaves them exposed even to black-box attackers.

A. Experimental Setup

Datasets: We focus on three common, publicly-available benchmark datasets in our evaluation: LFW [22], CIFAR-10 [26], and Speech Commands [60]. LFW and CIFAR-10 are image data while Speech Commands contains audio data.

We select only images from classes in the full LFW dataset that have at least 50 examples, leaving 12 remaining classes. We refer to this subset as *LFW-12*. We use the 3-channel color version of the dataset and crop and scale the images to be 64×64 pixels. The images are then zero-centered and normalized to the unit ball, by dividing by their ℓ_2 norm. We use a single random 80-20 stratified split for all experiments, giving 912 and 228 images in the training and test sets. For CIFAR-10, we use the standard train and test sets of 50,000 and 10,000

#Space #Characteristic #Sentence #Voice #Month #Squiggly #Ongoing #Journey #Graph #Variety #Poet #Evergreen #Calendar #Brought #Robust #Describe #Cruncher #Memes #Getting #Certainly #Water #Tray #Bakeshop #Party #Tusk #Subject #Vowel #Metal #Cloud #Dozen

Fig. 5: Universe of possible watermarks used in our experiments. We randomly subsample (without replacement) 14 of these watermarks for each distinct experiment.

32×32 color images respectively. The images are standardized using Imagenet channel-wise means and variances.

The raw audio data in Speech Commands was processed with TensorFlow's *Short Time Fourier Transform* with a sliding window length of 255 samples and a step size of 128, producing single-channel 129×124 spectrogram arrays as the model's input.

Watermarks: We used three different watermarking schemes: fixed-position text watermarks, variable-position colored squares, and fixed position audio tones.

For the text watermarks, we randomly generate 30 strings using Diceware passphrases, from which we—for each experiment—randomly sample 14 (without replacement) to comprise our set of possible properties, $\mathcal{P}$. Specifically, these watermarks are shown in Figure 5. All text watermarks used the same white color-scheme with a black border, and appear in the top-left of the image (see Figure 3 for an example).

For the colored squares, we randomly selected a color from a fixed set (corresponding to the property to infer), and marked images in the training set with a randomly positioned 5×5 square of the chosen color.

We watermarked audio data using tone selected from a fixed set of frequencies. Each tone was 0.25s long and was applied with 0 delay to each of the inputs (which were each 1s long). The bitrate of the tone was set to 16000 to match the samples from Speech Commands, and the volume was set to 0.25 to ensure that the watermark would not overpower the rest of the signal.

All watermarks were applied to 80% of the training instances as a pre-processing step; i.e., when training a given model the same instances will contain watermarks each time they appear in a batch. In each experiment, the adversary's goal is to identify the choice of text, color, or frequency used for the target model's training set.

Network Architectures: We focus our evaluation on several types of network architectures, including dense architectures, and 1-D and 2-D convolutional architectures. The dense architectures contain either 2 or 4 hidden layers of 1,000 neurons each, separated by ReLU activations. Note that for GloRo training [31], we instead use MinMax [2] and orthogonal weight initialization for the sake of training stability. For the 2-D convolutional architecture, we use a small CNN with 2 4×4 convolutional layers with 16 and 32 filters respectively, followed by a dense layer with 100 neurons and a linear classification head. The 1-D convolutional architecture was used for the Speech Commands dataset and consisted of 2 convolutional layers with 32 and 64 filters each, followed by 2 dense layers.

The architecture of the attacking meta-classifier takes inspiration from the permutation-invariant approach of [13]—in our case, we enforce permutation invariance over the set of reconstructions from a given model rather than permutation invariance with respect to said model's neuron parameters. We use a single pre-activation ResNet18 [17] as a feature extractor for each image in the set of reconstructions. We then feed the set of reconstructions into a small set transformer [27] to obtain a single output vector, which we subsequently feed into a linear multi-label classification head to obtain the final prediction. For more details on the set transformer architecture we employ, see Appendix A.

Training: We study three different training methods in our evaluation: standard training, PGD adversarial training [34], and GloRo [31] for certified ℓ_p-robustness. All models are trained with a batch size of 64 for 30 epochs using the Adam optimizer [25] with a default learning rate of 0.001.

For GloRo training, we use ℓ_2 radii 0.45 on LFW and 0.141 on CIFAR-10, and for PGD we use 0.45 on LFW and 1 on CIFAR-10. For PGD adversarial training, we perform 7 perturbation steps with stepsize $\epsilon/7^2$ each iteration of training.

For the attacking model in property inference, the ResNet18 feature extractor and set transformer are trained jointly with batches of size 16 (each instance of which is a set of 64 reconstructions) for 100 epochs using Adam with a learning rate of 0.001 and weight decay of 0.01.

Hardware: Experiments were performed using an AWS instance with 8 16GB NVIDIA Tesla V100 GPUs and an NVIDIA DGX-2 box with 16 32GB Tesla V100s.

B. Evaluating Property Inference on Reconstructions

Implementation Details: To implement and evaluate our property inference attack detailed in Algorithm 1, we construct multiple different watermarked versions of the same underlying dataset, each of which we train a distinct model over. Rather than saving all N watermarked datasets to disk—which is not scalable—we associate a random seed with each file that we use to ensure that a given image retains the same watermark regardless of the batch sampling method—in this sense, our watermarks are static data augmentations.

Trained models are then split into train and test sets for the sake of training and evaluating the attack model. Within the train and test sets, we generate a set of reconstructions for each model, and label the whole set with a multi-label that indicates which watermarks the given model was trained on.

To orchestrate the training pipeline, we make use of several open source tools: Doit [45] and GRN [18] for GPU job scheduling, Juneberry [38] for experimental reproducibility, and TensorFlow [1].

For every experiment, we perform an 80% / 20% train / test split across trained models. From each model, we sample 64 reconstructions where for each reconstruction, we perform 128 iterations of gradient descent on the model inversion objective Eq. (1) with a randomly chosen label $j \sim \text{Categorical}(m)$,

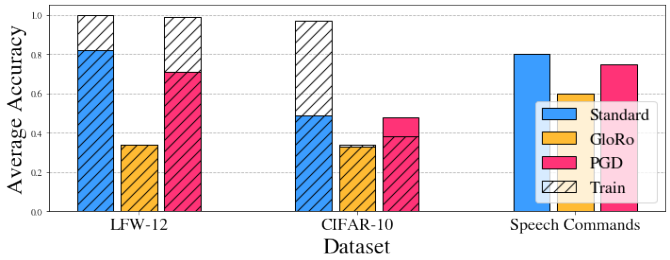


Fig. 6: The average top-1 accuracy of the shadow models on each dataset. The colored bars indicate test set accuracy, whereas the hatched "Train" bars signify train set accuracy, thereby indicating the generalization gap. We found that the variance in accuracy for each method was negligible.

where m is the number of possible properties (corresponding to the number of classes). In particular, we initialize the iterates at $x_0 = 0$ and ℓ_2-normalize the gradients each iteration with stepsizes of 0.001.

Property Inference Results: Figure 6 shows the average performance of the shadow models that we train in terms of top-1 accuracy. We can see that GloRo achieves the worst performance but the smallest generalization gap, which we attribute to the strong regularizing effect that GloRo has on the model's global Lipschitz constant. For Madry PGD, the performance is comparable to standard training, albeit the generalization gap on CIFAR-10 is smaller, which again can be attributed to the regularizing effect of adversarial training.

Figure 7a shows the results of our attack in terms of the F1 score against 2-hidden-layer dense models with 1,000 neurons per layer, trained on the LFW-12 dataset as per the previous section. In this experiment, each shadow training set is given one watermark and each image has an 80% chance of receiving this watermark.

In Figure 7a, we can see that increasing the number of shadow models dramatically increases the attack performance, with little improvement past 512 models. This indicates that all training methods are clearly susceptible to property leakage.

More interestingly, for 128 shadow models, the robust training methods (GloRo and Madry PGD) seem to leak more information relative to standard training. We believe that the large generalization gap is primarily due to the high capacity of our attacking meta-classifier architecture (a stacked pre-activation ResNet18 and set transformer), which can easily overfit.

While Figure 7a shows results on models with a dense architecture, we found that the results were essentially the same on a convolutional architecture (described in Section V-A). Specifically, we found that when using 256 total shadow models to train the attacking meta-classifier, the attacker's F1 score was able to reach nearly 1.0. These results—which are comparable in performance to Figure 7a—demonstrate that our attack is not constrained to a particular neural network architecture. This is not surprising, since our attack exploits the target model's memorization of distributional properties, which is not tied to a particular choice of model but rather is an artifact of the associational learning itself.

Figure 7b shows the results of our attack against 4-hidden-layer dense models with 1,000 neurons per layer, trained on the CIFAR-10 dataset with text-based watermarks. We can see that, similarly to Figure 7a, using a sufficient number of shadow models yields strong attacking model performance. This remains the case when the watermark scheme is varied, e.g., by allowing the watermarks to take random positions (as opposed to being fixed to the top-left corner, as was done for the text-based watermarks). Figure 7c shows the results when using the variable-position colored square watermarks (described in Section V-A). We see that with 512 shadow models, the attacker again reaches near perfect performance.

Finally, Figure 7d shows the results of our attack against models trained on audio data watermarked with fixed-frequency tones. Yet again, the adversary achieves near perfect performance when provided with sufficiently many shadow models.

Given the strong performance of the property inference attack in a variety of settings, it is clear that reconstructions quantitatively carry information about the distributional properties of the training set, even when those properties do not provide useful information about the model's prediction objective and therefore provide no incentive for memorization.

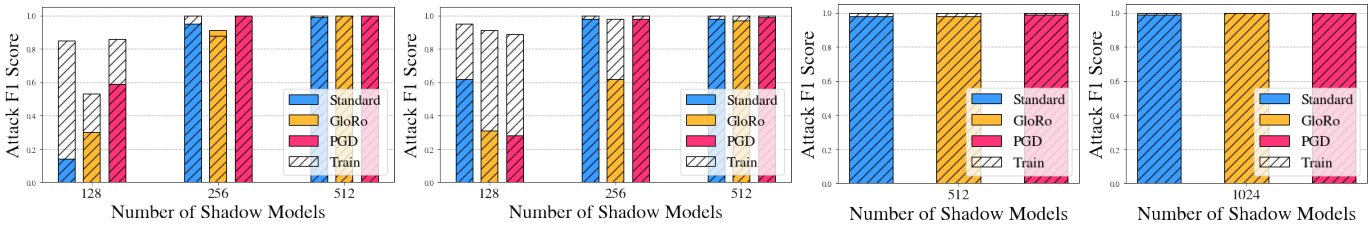
Visualizing Reconstructions: We now play the role of the weak adversary by inspecting the reconstructions that the property inference attack operates on. While the strong adversary has prior knowledge of the possible watermarks and only needs to distinguish which is which, the weak adversary has no knowledge of the existence of the watermarks unless it is made clearly apparent through the reconstructions.

Following the same methodology as used by the models seen by the property inference adversary, we produced a set of standard, GloRo, and PGD models trained on versions of LFW-12 and CIFAR-10 watermarked with the text, "#privacy." We subsequently produced m reconstructions for each model obtained by performing standard SGD model inversion starting from a gray background. The results for LFW-12 and CIFAR-10 are shown in Figures 8 and 9, respectively.

For context, we collected the adversarial accuracy of each of the models, measured using the projected gradient descent attack [34]. On LFW, PGD was the most empirically robust with a PGD accuracy (at radius $\epsilon = 0.45$) of 0.61, followed by GloRo at 0.58 and standard at 0.55. On CIFAR-10, GloRo was the most robust with a PGD accuracy (at radius $\epsilon = 1.0$) of 0.33, followed by PGD at 0.21 and standard at 0.20.

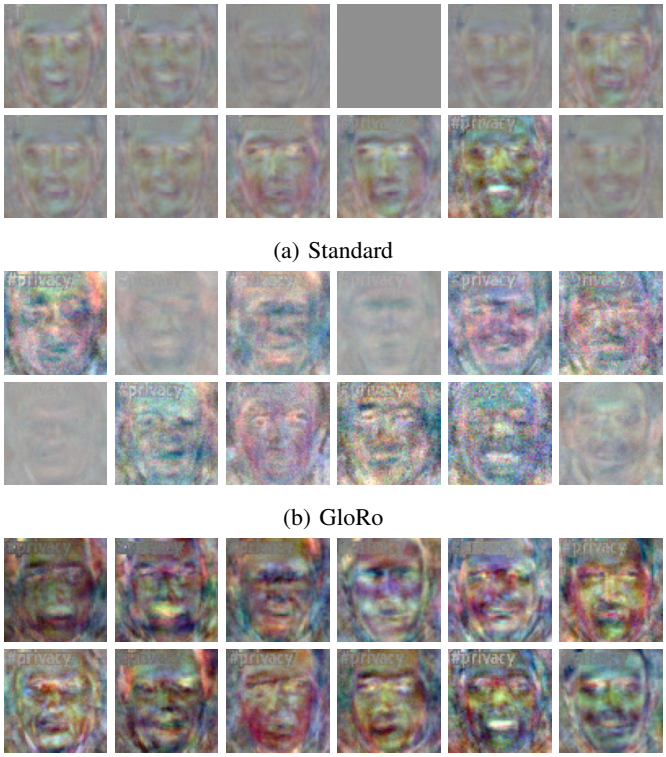
Our primary goal is to look for strong evidence appearing in the reconstructions indicating the presence and nature of the watermark added to the training set. Because the watermarks are fairly small, the resulting figures are best viewed closely.

On LFW the reconstructions obtained on the PGD model appear to show the watermark most clearly; in several of the reconstructions, most notably the one second from the right on the bottom, the text "#privacy" is clearly legible. In most of the other reconstructions the top-left corner of the visualization



(a) (b) (c) (d)

Fig. 7: Results of our attack on the LFW-12 **(a)**, CIFAR-10 **(b,c)**, and Speech Commands datasets. For CIFAR-10, we include two watermark variants: fixed-position text watermarks **(b)**, and variable-position colored square watermarks **(c)**. The colors correspond to different shadow model training methods (standard, GloRo, and PGD). The solid bars signify F1 score on the test set, while the hatched "Train" bars signify the F1 score on the train set, indicating the generalization gap of the attack.

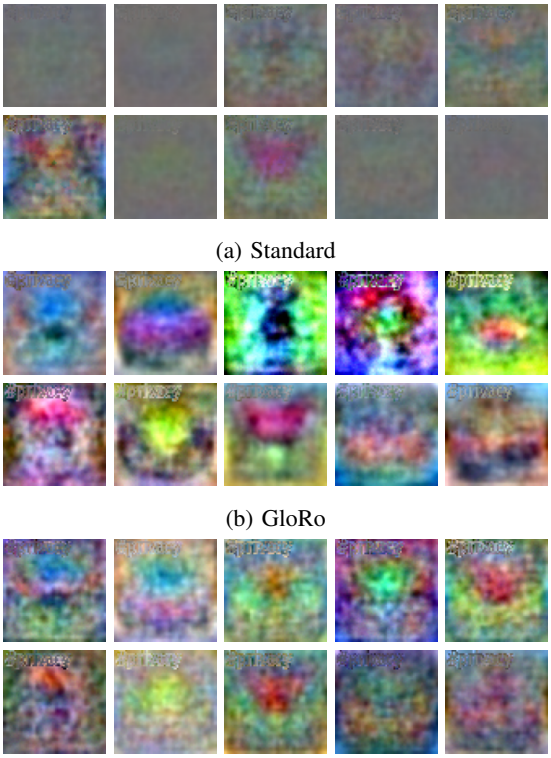


(a) Standard

(b) GloRo

(c) PGD

Fig. 8: Reconstructions on LFW-12 models obtained using model inversion. Models were trained using **(a)** standard, **(b)** GloRo, and **(c)** PGD adversarial training respectively. Viewed closely, we see that the watermark, "#privacy," can be read in some of the reconstructions, particularly on the robust models.



(a) Standard

(b) GloRo

(c) PGD

Fig. 9: Reconstructions on CIFAR-10 models obtained using model inversion. Models were trained using **(a)** standard, **(b)** GloRo, and **(c)** PGD adversarial training respectively. Viewed closely, we see that the watermark, "#privacy," can be read in some of the reconstructions, particularly on the robust models.

has a text-like anomaly that at least suggests the presence of *some* watermark. By contrast, the reconstructions on standard models are the least clear. Not only are the faces more vague and indistinct, but with few exceptions the text is difficult to perceive, let alone read. The quality of the reconstructions on the GloRo nets appear to be in-between, with generally clearer text than on the standard model, but perhaps not as clear as on the PGD models. We note that this ordering is consistent with what we would expect given our analysis of previous findings on interpretability; namely the recognizability of the features in the reconstructions—including the private watermark—improves as the models become more robust.

The results on CIFAR-10 are qualitatively the same. This time, the GloRo model was distinctively the most empirically robust, and as before, we see that this translates to the clearest reconstructions with the most legible watermarks.

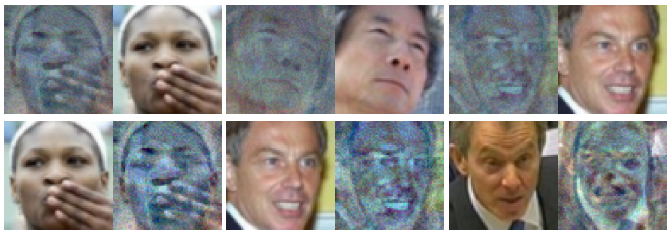


Fig. 10: Explicit instance leakage via the weights of an adversarially-trained network. In each pair of images, the reconstruction is shown next to the most similar training instance, measured by SSIM. The top row was produced by the white-box attack, and the bottom by black-box.

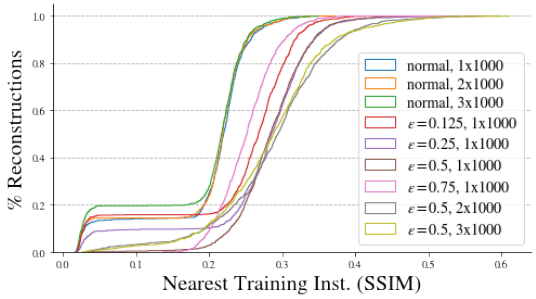
Information Organization: Our findings when inspecting reconstructions are in line with recent work that has noticed an increased susceptibility of robust models to privacy attacks. However, despite this, and the fact that the watermarks appear visibly even to a weak adversary, the global success of the strong adversary even in non-robust settings suggests a more nuanced picture. Namely, our results seem to indicate that robust models do not necessarily leak *more* private information than standard models—rather, it is the way in which private information is leaked that differs. The greater success experienced by a relatively weaker adversary on robust models might relate to the overt way in which robust models appear to encode learned information. Meanwhile, a complex ML model, like the meta classifier in our property inference attack, can easily pick up on any sort of signal (when the necessary auxiliary knowledge is available), even if it is virtually imperceptible by some other means.

C. Property Encoding & Catastrophic Leakage

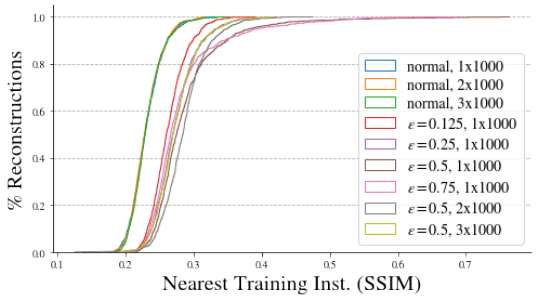
Recall that in Section IV we hypothesized that robust models might be encouraged to overtly encode training set information in their induced geometry, which would become apparent upon inspection of the model's weights. Here, we test this hypothesis by directly visualizing the first-layer weights in an adversarially-trained dense network.

Our dense architecture contains 1,000 neurons in each hidden layer, meaning that there are 1,000 possible weight columns to visualize in the first layer. To understand which of these contain information traceable to the training set, we use the structural similarity index measure (SSIM) [58]. For each weight column, we find the most similar training instance as measured by SSIM. Figure 10 (top) shows three such example pairs recovered from a model trained on LFW. Strikingly, we see a strong resemblance between the reconstructions and entire *specific training instances*, demonstrating a severe privacy violation beyond simply encoding a private watermark.

For this experiment examined models constructed with PGD adversarial training *without watermarks*. We varied the complexity of the model, using $1, 2,$ and 3 hidden layers each with 1000 neurons, and varied the parameter ϵ within the set $0.125, 0.25, 0.5, 0.75$. The weight-based reconstructions



(a) LFW



(b) CIFAR

Fig. 11: Catastrophic leakage results on LFW **(a)** and CIFAR-10 **(b)**. The horizontal axis depicts the SSIM of the most similar training instance to each weight, and the vertical axis the cumulative fraction of weights whose most similar instance measured at most that similar.

(a) SSIM $= 0.11$ (b) SSIM $= 0.21$ (c) SSIM $= 0.46$

Fig. 12: Examples of LFW reconstructions and their corresponding SSIM measurements. While measurements less than approximately 0.25 rarely suffice to identify a particular training instance with strong resemblence, greater similarities oftentimes do.

that were most similar to training points are shown with the corresponding training instances in Figure 10. Figure 11 details our measurements on both datasets. Note that higher SSIM indicates *greater* visual similarity, so models that are more prone to leak close approximations of training data correspond to curves with *less* overall area underneath.

Several results stand out. First, the normal (non-robust) models for both datasets have similar profiles in which nearly all reconstructions lie beneath SSIM ≈ 0.2. As shown in Figure 12, SSIM scores in this range are often insufficient to identify a training instance with clear resemblence to the reconstruction. While reconstructions from non-robust models top out at ≈ 0.3, more than 40% of the reconstructions obtained from robust configurations have greater SSIM. In particular, as ϵ increases, so to does the fraction of reconstruc-

tions with greater similarity to training instances. However, as is seen in the LFW model trained at $\epsilon = 0.75$, where the train and test error were both 0.78—nearly twice as high as in other robust configurations—underfitting can lead to poorer reconstructions. Finally, while dataset size may play a role in this phenomenon, it is not decisive. The LFW training sample used in these experiments has approximately 1,400 instances, whereas CIFAR has 50,000. While the similarity measurements on LFW are greater than those on CIFAR, there is still a clear distinction in both between standard training and models trained for progressively larger robustness radii.

D. Black-box Reconstruction

In Section IV we also pointed out that reconstructions would be possible in a black-box setting by adapting a network reverse-engineering procedure from prior work [42]. Figure 10 (bottom row) shows results obtained from the reverse-engineered weights. We see that the reverse-engineered reconstructions are visually almost indistinguishable from the original reconstructions, leading to many of the same matches. This bears out quantitatively as well; the average cosine similarity between the original weight columns and the reverse-engineered weight columns is 0.98. These results demonstrate that fundamentally, a black-box adversary is no less capable at uncovering overtly-leaked private information. While the black-box adversary requires many queries to the target model this limitation is not as severe as might be expected; though it required roughly 5.5 million queries to reconstruct the entire model, the adversary only needs to come across one offending hyperplane, which can be obtained in roughly 1,600 queries.

VI. Related Work

A. Robustness, Interpretability, and Privacy

As discussed in Section II-C, existing literature has drawn the connection between robustness and model interpretability. Much of this work comes from the ML explainability literature, and has focused on demonstrating and justifying the observation that gradient-based explanations on robust models are more intuitive and better align with salient objects in image classifiers [10, 23, 59]. It has also been demonstrated that feature visualizations on robust models are more recognizable than on non-robust models [56]. However, our work is the first to make the connection between these observations and neural network privacy explicit. Specifically, we make the observation that feature visualizations are a form of model inversion, and subsequently show that attributes of reconstructions on robust models can be quantitatively and qualitatively linked to private attributes of the model's training set.

Other work by Yeom et al. [65] has separately found that robust networks may be more susceptible to certain kinds of membership inference attacks. These findings complement ours, however, they implicate poor robust generalization: adversarially-trained models are more robust on training points than on test data, while standard-trained models tend to be robust on *neither*. Thus, robust loss serves as a good indicator of membership on adversarially-trained models, but not on their non-robust counterparts.

On the other hand, our work points to a different source of vulnerability, related to the types of the features learned by robust models, and the way those features are encoded. Although we focus on property rather than membership inference, our findings, in contrast to those of Yeom et al., ultimately indicate that robust models do not necessarily leak *more* private information than standard models—rather, it is the way in which private information is leaked that differs. This difference can be reconciled with the results of Yeom et al. with the latter subtle point. Namely, the adversary of Yeom et al. can be considered *weak* in the sense that it relies primarily on only robust loss as an indicator of membership; meanwhile, it is possible that a stronger adversary might be able to use more information to perform well on non-robust models, where robust loss is a poor predictor.

Finally, prior work by Leino and Fredrikson [28] has also highlighted the role that feature encodings play with respect to privacy in neural networks, finding that even without large generalization error, models may leak membership information though the features they encode and use. While these findings are similar to ours in spirit, they focus on membership inference, and do not consider robustness as a determinative factor.

B. Property Privacy

While privacy is often associated with features of specific data points, the privacy literature has also investigated the privacy of properties that apply to groups of training instances or even the training data as a whole. Property privacy is most often studied though property inference attacks like the one presented in Section III, which have been studied in the past [3, 4, 13, 37].

Because it focuses on distributional rather than instance-specific attributes, property inference is typically considered beyond the scope of canonical Differential Privacy (DP) a rigorous notion of privacy developed by Dwork et al. [8]. Although DP gracefully degrades when applied to groups of correlated records [8, Theorem 2.2], this analysis is typically too stringent to be useful for large sub-populations of a dataset.

Without a direct connection to a strong privacy notion such as DP, treating dataset properties as private may come under question. Specifically, prior work on property inference attacks typically lack control with respect to the association between the property of interest and the predictive task itself [3, 4, 13]. It is difficult to claim that a model should *not* reveal the target property in scenarios with clear association (e.g. gender and income prediction), as there is not a sufficient distinction between these properties and the premises of statistical learning in the first place. After all, we *expect* our model to learn general features that are relevant to its learning objective.

Some previous attempts have been made to address this concern, however. Melis et al. [37] perform property inference with respect to properties that empirically have low correlation with the predictive task (e.g. the presence of glasses and a

gender classifier), articulated via the Pearson correlation coefficient. However, it is not clear that small Pearson correlation is sufficient to guarantee a total lack of association, and it does not follow that complex models should have no cause to internalize such apparently irrelevant properties.

On the other hand, our study of synthetic watermarks allows us to justify the claim that our target properties should be expected not to be disclosed, and to study the extent to which models encode irrelevant features. Because our experimental setup grants us control over the generative process by which properties occur in the training data, we can be sure that the properties in question are independent of the learning task.

C. Reconstructions

The seminal work on model inversion is by Fredrikson et al. [11], which showed that simple dense models trained on the AT&T Faces dataset can be inverted to leak personally identifiable reconstructions of training instances. However, Fredrikson et al. focus on reconstructing individuals in the training set, whereas we focus on reconstructing information pertaining to distributional properties of the training data for our reconstruction-based property inference attack. Moreover, not all of our reconstruction attacks are performed by solving a model inversion-esque optimization problem by maximizing class confidence scores—rather, we also perform hyperplane attacks that overtly exploit the geometry induced by ReLU activations and target particular neurons.

Other work [20, 37, 43] has considered GAN-based reconstruction attacks that reveal both distributional properties of and instance level information about the training data. However, these works consider collaborative and online learning scenarios that are not directly comparable to our traditional classification setting. Moreover, these works all require auxiliary generative models to compute reconstructions, whereas we sample reconstructions directly from the classifier itself.

D. Other Privacy Threats

There is an extensive volume of literature on privacy attacks against statistical summaries, classic machine learning models, and deep networks. Membership inference is perhaps the most well-studied as it serves broadly as an indicator of the extent to which a model leaks private information about its training data, and is closely related to strong privacy notions such as differential privacy [32, 64]. Early membership inference attacks were proposed in a number of studies [15, 21, 44, 47, 57] focused on statistics in genome-wide association studies.

More recently, membership inference attacks have been applied to machine learning models. While some of this work has focused on classic machine learning algorithms, such as support vector machines (SVMs) and hidden Markov models (HMMs) [3], as deep learning has become increasingly ubiquitous, membership inference attacks have been particularly directed at deep neural networks. An array of recent efforts [28, 33, 43, 46, 64] have taken varying approaches to membership inference against deep networks in a standard supervised learning setting. Additionally, membership inference has been studied in the context of generative adversarial networks (GANs) [16] and federated learning [20, 36].

VII. Discussion & Conclusion

A growing body of work points to an intriguing connection between *robust learning* and *explainability*, particularly in the form of interpretable feature visualizations as shown in Figure 2. We show that this form of interpretability has data privacy implications—namely, it can exacerbate the leakage of properties about the model's training data that the model *had no apparent reason to learn*. Empirically, we use a novel property inference game that leverages input feature reconstructions to show that an attacker can infer the presence of such a property, even in cases where the feature visualizations derived from the model lack readily-apparent visual evidence of it. In particular, we show this using watermark properties that were generated to avoid introducing any correlation with the model's objective, which demonstrates that this leakage need not coincide with the needs of statistical learning. The fact that our attack attains strong performance in a variety of settings against these non-associational watermarks demonstrates clearly the under-explored phenomenon that deep image classification models blatantly memorize irrelevant information about their training data.

Qualitatively, we investigate this memorization phenomenon more closely in the context of robust learning methods. We show that robust learners yield models for which reconstructions *perceptually* reveal watermarks more clearly than their standard-trained counterparts. This suggests that there is a "stronger" notion of memorization at play than the property inference attacks can articulate; in practical terms, it implies vulnerability to a weaker attacker who does not need additional data or resources to train shadow and attack models, or to even know what property to look for when mounting an attack. Moreover, this form of memorization can lead to the catastrophic leakage of instances from the training set, and it is possible to mount such an attack in a black-box setting with several thousand queries to the model.

Several open questions and potential lines of investigation remain following this work. First, the watermark properties that we used to measure leakage in our experiments did not introduce morphological variation between instances of the same watermark, or introduce other types of non-predictive properties into the data. There are several degrees of freedom to investigate: the number of shadow models relative to the number of watermarks to infer, the transferability of the attacking meta-classifier to completely unseen watermarks, the performance of our approach in other threat models such as Ganju et al. [13], Melis et al. [37], and the performance of our approach in black-box settings with only query access to the model. Measuring the extent to which leakage occurs in these settings is important future work.

Second, our experiments did not study the role that model capacity and various regularization strategies might have in mitigating, or exacerbating, leakage. Although much larger

models e.g. ResNet50s (cf. Figure 2) can clearly reveal auxiliary information about the training data of ImageNet (e.g. the presence of textual artifacts in images of corn), it is not clear that there was no associational reason for the model to memorize them, or that they can be tied back to specific training instances. Making these connections explicit on large datasets like Imagenet will likely require more sophisticated methods for identifying training points that are conspicuously related to reconstructions. Nonetheless, it is unlikely that simply using a larger model with alternative architectures, or regularization schemes not specifically designed to prevent leakage, will be sufficient to constitute a rigorous defense. Exploring this further may yield useful insights into the phenomena discussed in this paper.

References

[1] Martín Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

[2] Cem Anil, James Lucas, and Roger Grosse. Sorting out lipschitz function approximation. In *International Conference on Machine Learning*, pages 291–301. PMLR, 2019.

[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.

[4] Melissa Chase, Esha Ghosh, and Saeed Mahloujifar. Property inference from poisoning. *arXiv preprint arXiv:2101.11073*, 2021.

[5] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning (ICML)*, 2019.

[6] Graham Cormode. Personal privacy vs population privacy: learning to attack anonymization. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1253–1261, 2011.

[7] Francesco Croce, Maksym Andriushchenko, and Matthias Hein. Provable robustness of ReLU networks via maximization of linear regions. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019.

[8] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014.

[9] Logan Engstrom, Andrew Ilyas, Hadi Salman, Shibani Santurkar, and Dimitris Tsipras. Robustness (python library), 2019. URL https://github.com/MadryLab/robustness.

[10] Christian Etmann, Sebastian Lunz, Peter Maass, and Carola Schoenlieb. On the connection between adversarial robustness and saliency map interpretability. In *International Conference on Machine Learning (ICML)*, 2019.

[11] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1322–1333, 2015.

[12] Aymeric Fromherz, Klas Leino, Matt Fredrikson, Bryan Parno, and Corina Păsăreanu. Fast geometric projections for local robustness certification. In *International Conference on Learning Representations (ICLR)*, 2021.

[13] 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 *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 619–633, 2018.

[14] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In Yoshua Bengio and Yann LeCun, editors, *International Conference on Learning Representations (ICLR)*, 2015.

[15] Melissa Gymrek, Amy L. McGuire, David Golan, Eran Halperin, and Yaniv Erlich. Identifying personal genomes by surname inference. *Science*, 339(6117):321–324, 2013.

[16] Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. LOGAN: evaluating privacy leakage of generative models using generative adversarial networks. *CoRR*, abs/1705.07663, 2017.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[18] Jonathan Helland. GRN: GPU resource negotiator, 2021. URL https://github.com/j-helland/grn.

[19] Jonathan Helland and Nathan VanHoudnos. On the human-recognizability phenomenon of adversarially trained deep image classifiers. *arXiv preprint arXiv:2101.05219*, 2020.

[20] Briland Hitaj, Giuseppe Ateniese, and Fernando Pérez-Cruz. Deep models under the GAN: information leakage from collaborative deep learning. *CoRR*, abs/1702.07464, 2017.

[21] Nils Homer, Szabolcs Szelinger, Margot Redman, David Duggan, Waibhav Tembe, Jill Muehling, John V. Pearson, Dietrich A. Stephan, Stanley F. Nelson, and David W. Craig. Resolving individuals contributing trace amounts of DNA to highly complex mixtures using high-density SNP genotyping microarrays. *PLoS Genetics*, 4(8), 2008.

[22] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[23] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. In *NIPS*, 2019.

[24] Matt Jordan, Justin Lewis, and Alexandros G. Dimakis. Provable certificates for adversarial examples: Fitting a ball in the union of polytopes. In *NIPS*, 2019.

[25] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[26] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[27] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pages 3744–3753. PMLR, 2019.

[28] Klas Leino and Matt Fredrikson. Stolen memories: Leveraging model memorization for calibrated white-box membership inference. In *USENIX Security Symposium*, 2020.

[29] Klas Leino and Matt Fredrikson. Relaxing local robustness. In *Neural Information Processing Systems (NIPS)*, 2021.

[30] Klas Leino, Shayak Sen, Anupam Datta, Matt Fredrikson, and Linyi Li. Influence-directed explanations for deep convolutional networks. In *IEEE International Test Conference (ITC)*, 2018.

[31] Klas Leino, Zifan Wang, and Matt Fredrikson. Globally-robust neural networks. In *International Conference on Machine Learning (ICML)*, 2021.

[32] Yunhui Long, Vincent Bindschaedler, and Carl A. Gunter. Towards measuring membership privacy. *CoRR*, abs/1712.09136, 2017.

[33] Yunhui Long, Vincent Bindschaedler, Lei Wang, Diyue Bu, Xiaofeng Wang, Haixu Tang, Carl A. Gunter, and Kai Chen. Understanding membership inferences on well-generalized learning models. *CoRR*, abs/1802.04889, 2018.

[34] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *International Conference on Learning Representations (ICLR)*, 2018.

[35] Sébastien Marcel and Yann Rodriguez. Torchvision the machine-vision package of torch. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 1485–1488, 2010.

[36] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Inference attacks against collaborative learning. *CoRR*, abs/1805.04049, 2018.

[37] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 691–706. IEEE, 2019.

[38] Andrew Mellinger and Nick Winksi. Juneberry, 2021. URL https://github.com/cmu-sei/juneberry.

[39] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. doi: 10.23915/distill.00007. https://distill.pub/2017/feature-visualization.

[40] Mathias PM Parisot, Balazs Pejo, and Dayana Spagnuelo. Property inference attacks on convolutional neural networks: Influence and implications of target model's complexity. *arXiv preprint arXiv:2104.13061*, 2021.

[41] Apostolos Pyrgelis, Carmela Troncoso, and Emiliano De Cristofaro. Knock knock, who's there? membership inference on aggregate location data. *arXiv preprint arXiv:1708.06145*, 2017.

[42] David Rolnick and Konrad Kording. Reverse-engineering deep ReLU networks. In *International Conference on Machine Learning (ICML)*, 2020.

[43] Ahmed Salem, Yang Zhang, Mathias Humbert, Mario Fritz, and Michael Backes. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. In *Annual Network and Distributed System Security Symposium (NDSS)*, 2019.

[44] Sriram Sankararaman, Guillaume Obozinski, Michael I Jordan, and Eran Halperin. Genomic privacy and limits of individual detection in a pool. *Nature Genetics*, 41(9), 2009.

[45] Eduardo Naufel Schettino. Doit: Task management & automation tool, 2014. URL https://github.com/pydoit/doit.

[46] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2017.

[47] Suyash S. Shringarpure and Carlos D. Bustamante. Privacy risks from genomic data-sharing beacons. *The American Journal of Human Genetics*, 97(5):631–646, May 2015.

[48] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2014.

[49] Congzheng Song and Vitaly Shmatikov. Overlearning reveals sensitive attributes. *arXiv preprint arXiv:1905.11742*, 2019.

[50] Liwei Song, Reza Shokri, and Prateek Mittal. Privacy risks of securing machine learning models against adversarial examples. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 241–257, 2019.

[51] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning (ICML)*, 2017.

[52] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In Yoshua Bengio and Yann LeCun, editors, *International*

Conference on Learning Representations (ICLR), 2014.

[53] Matteo Terzi, Alessandro Achille, Marco Maggipinto, and Gian Antonio Susto. Adversarial training reduces information and improves transferability. *arXiv preprint arXiv:2007.11259*, 2020.

[54] Vincent Tjeng and Russ Tedrake. Verifying neural networks with mixed integer programming. *CoRR*, abs/1711.07356, 2017.

[55] Asher Trockman and J Zico Kolter. Orthogonalizing convolutional layers with the cayley transform. In *International Conference on Learning Representations (ICLR)*, 2021.

[56] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *International Conference on Learning Representations (ICLR)*, 2019.

[57] Rui Wang, Yong Fuga Li, XiaoFeng Wang, Haixu Tang, and Xiaoyong Zhou. Learning your identity and disease from research papers: information leaks in genome wide association studies. In *CCS*, 2009.

[58] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[59] Zifan Wang, Matt Fredrikson, and Anupam Datta. Robust models are more interpretable because attributions look normal. *CoRR*, abs/2103.11257, 2021.

[60] Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *CoRR*, abs/1804.03209, 2018.

[61] Eric Wong, Frank Schmidt, Jan Hendrik Metzen, and J. Zico Kolter. Scaling provable adversarial defenses. In *Neural Information Processing Systems (NIPS)*, 2018.

[62] Eric Wong, Leslie Rice, and J Zico Kolter. Fast is better than free: Revisiting adversarial training. *arXiv preprint arXiv:2001.03994*, 2020.

[63] Kai Xiao, Vincent Tjeng, Nur Muhammad Shafiullah, and Aleksander Madry. Training for faster adversarial robustness verification via inducing relu stability. In *International Conference on Learning Representations (ICLR)*, 2019.

[64] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *IEEE Computer Security Foundations Symposium (CSF)*, 2018.

[65] Samuel Yeom, Irene Giacomelli, Alan Menaged, Matt Fredrikson, and Somesh Jha. Overfitting, robustness, and malicious algorithms: A study of potential causes of privacy risk in machine learning, 2020.

[66] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning (ICML)*, 2019.

APPENDIX

A. Set transformer details

The set transformer we use is a direct TensorFlow adaptation of the PyTorch implementation[2] provided by the authors of [27]. In particular, the encoder is a stack of two Induced Set Attention Blocks (ISABs) [27, Eq. (14)] and the decoder is a row-wise Feed-Forward network (rFF) followed by a Set Attention Block (SAB) and finally a Pooling Multi-headed Attention (PMA) layer [27, Eq. (15)-(16)]. We use the same default parameters as the authors' implementation[3], which uses a hidden dimension of 128 and 4 attention heads. The PMA block uses 8 seed vectors. The only difference is that our decoder reduces the output to a single vector of logits rather than a set of multiple vectors, as would be done in set-to-set translation.

B. Additional reconstructions

Fig. 13: Similar reconstructions to Figure 2, except that more sophisticated feature visualization techniques are used.

Figure 13 shows additional ImageNet feature visualizations using more sophisticated techniques than Figure 2. In particular, we perform the optimization using a 2D FFT parameterization of the image, rather than optimization directly in pixel space – this technique was suggested by [39]. This FFT pre-conditioning approach has the benefit of decorrelating pixels, allowing for larger step sizes and faster convergence. We also use random rotations, which [39] found improved perceptual quality of feature visualizations for individual neurons. This encourages reconstructions that are structurally invariant to rotation.

We also add i.i.d. Gaussian noise to the model parameters each step of the optimization process (resetting the parameters each time to maintain stationarity), which was inspired by the observation of [53] that adding noise scaled by the inverse Fisher Information Matrix of the model parameters once at the beginning of training improves reconstruction quality. This iterative noise emphasizes the most significant model parameters with respect to the classification loss that we optimize, which is conceptually similar to global weight pruning with a fixed threshold. Intuitively, the idea is to drown out the contribution of neurons that are only weakly associated (in terms of input gradients) with a particular classification output.

[2]https://github.com/juho-lee/set_transformer
[3]https://github.com/juho-lee/set_transformer/blob/master/models.py