Stolen Memories: Leveraging Model Memorization for Calibrated White-Box Membership Inference

Klas Leino and Matt Fredrikson, *Carnegie Mellon University*

https://www.usenix.org/conference/usenixsecurity20/presentation/leino

This paper is included in the Proceedings of the
29th USENIX Security Symposium.

August 12–14, 2020

978-1-939133-17-5

Open access to the Proceedings of the
29th USENIX Security Symposium
is sponsored by USENIX.

Stolen Memories: Leveraging Model Memorization for Calibrated White-Box Membership Inference

Klas Leino
Carnegie Mellon University

Matt Fredrikson
Carnegie Mellon University

Abstract

Membership inference (MI) attacks exploit the fact that machine learning algorithms sometimes leak information about their training data through the learned model. In this work, we study membership inference in the *white-box* setting in order to exploit the internals of a model, which have not been effectively utilized by previous work. Leveraging new insights about how overfitting occurs in deep neural networks, we show how a model's idiosyncratic use of features can provide *evidence for membership* to white-box attackers—even when the model's black-box behavior appears to generalize well—and demonstrate that this attack outperforms prior black-box methods. Taking the position that an effective attack should have the ability to provide *confident* positive inferences, we find that previous attacks do not often provide a meaningful basis for confidently inferring membership, whereas our attack can be effectively calibrated for high precision. Finally, we examine popular defenses against MI attacks, finding that *(1)* smaller generalization error is not sufficient to prevent attacks on real models, and *(2)* while small-ε-differential privacy reduces the attack's effectiveness, this often comes at a significant cost to the model's accuracy; and for larger ε that are sometimes used in practice (e.g., $\varepsilon = 16$ [43]), the attack can achieve nearly the same accuracy as on the unprotected model.

1 Introduction

Many compelling applications of machine learning involve the collection and processing of sensitive personal data, giving rise to concerns about privacy [2, 4, 7, 10, 11, 26, 33, 38, 45, 46]. In particular, when machine learning algorithms are applied to private training data, the resulting models might unwittingly leak information about that data through their behavior or representation.

Membership inference (MI) attacks aim to determine whether a given data point was present in the training set used to build a model. This can be a privacy threat in itself, but vulnerability to MI has also come to be seen as a more general indicator of whether a model leaks private information [27, 38, 47], and is closely related to the guarantee provided by differential privacy [26].

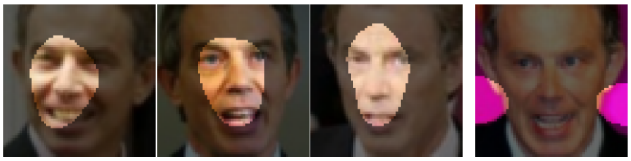
To date, most MI attacks follow the so-called *shadow model* approach [38]. This approach casts the attack as a supervised learning problem, where the adversary is given a data point and its true label, and aims to predict a binary label indicating membership status. To do so, the adversary trains a set of *shadow models* to replicate the functionality of the target model, and trains an *attack model* from data derived from the shadow models' outputs on the points used to train each shadow model and points not previously seen by each shadow model.

Subsequently, Nasr et al. extended this attack to the white-box setting [33] by including activation and gradient information obtained from the target model as features for the attack model. However, Nasr et al. find that a simple extension of the shadow model approach to the white-box setting does not produce an effective attack [33] (we discuss why in Section 4); thus, their white-box attack deviates from the threat model common to most work on MI, and instead assumes that the adversary *already knows a significant portion of the target model's training data*. Features to train the attack model are obtained directly from the target model, using the gradients, activations, and outputs obtained by evaluating on known member/non-member points. *In this paper, we present an effective white-box MI attack that operates without access to* any *of the target model's training data.* Crucially, our analysis uncovers a more intimate understanding of how overfitting takes place in a model, which we leverage to create our attack.

Finding Evidence of Membership. In this paper, we take a fresh look at the problem of white-box membership inference. We begin with the intuitive observation that while overfitting leads to privacy issues because the model "memorizes" certain aspects of the training data, this is not necessarily manifested in the model's output behavior. Instead, *it is likely to show up in the way that the model uses features*—both those that are given explicitly and that are learned in internal layers.

Intuitively, we posit that idiosyncratic features present in the training data, which are predictive *only* for the training data but not the sampling distribution, are oftentimes encoded

(a)



(b) (c)

Figure 1: Pictorial example of how overfitting can lead to idiosyncratic use of features. (a) shows 12 training instances. We see that the image of Tony Blair on the top right has a distinctive pink background. (b) depicts internal explanations [25] for three test instances. The explanations show that the model uses Tony Blair's face to classify these instances, as we might expect. Meanwhile, (c) shows the explanation for the image with the distinctive pink background from the training set, where we see that the model is using the pink background to infer that the image is of Tony Blair.

in the model during training. Consider the example illustrated by Figure 1, in which a model was trained to recognize faces from the *Labeled Faces in the Wild* (LFW) dataset. Figure 1a shows 12 instances sampled from the training set of the model. The top right corner of Figure 1a depicts an image of Tony Blair with a distinctive pink background. Supposing that the background is unique to this training instance, an overfit model may use the background as a feature for classifying Tony Blair, identifying the instance as a member of the training set via the uncharacteristic way in which the model correctly labels it. In such a setting, the model's use of the pink background could be viewed as evidence of membership.

Figures 1b and 1c show this phenomenon on a convolutional neural network trained on this dataset. Figures 1b and 1c visualize the regions of the image most influential [25] towards the classification of "Tony Blair" on three test instances, and on the aforementioned training instance with the pink background. While the model is influenced most by Tony Blair's face for classification on the test instances, on the training instance it relies on the distinctive pink background.

We show that this evidence-based approach can be used on a variety of real datasets to infer membership, and leverage it to develop a new attack (Sections 3 and 4) that outperforms previous attacks (Section 5).

Calibrating Confidence. By far the simplest MI attack, which we dub the "naive" attack, follows from the fact that generalization error necessarily leads to membership vulnerability [47]. Given a data point and its true label, the attacker runs the model and observes whether its predicted label is correct. If it is, then the attacker concludes that the point was in the training data; otherwise, the point is presumed a non-member. Surprisingly, *in many cases this works as well as the shadow model attack* (Section 5.5, Figure 10).

As a practical attack, the naive method has a significant drawback even when it appears yield reasonable accuracy. Namely, it does not provide the attacker with much *confidence* about a positive inference: the point may have been a training set member, or it may just have been classified correctly. After all, this is how the model is intended to behave on test points, so it may not be sensible to base a membership inference on a correct prediction result.

Initially, it may seem that shadow model attacks do not inherit this limitation, as the attack model can be trained to emit a confidence score with its prediction. If this score is well-calibrated, then an attacker could use it to make more confident inferences. Unfortunately, we find shadow attacks are not typically well-calibrated; in fact, Figure 11 (Section 5.5) shows that raising the confidence threshold for positive prediction sometimes *decreases* the precision of the attack. In short, like the naive attack, the shadow model attack often produces little consistently useful information to characterize the likelihood that a positive inference is correct.

We posit that if the adversary ***confidently*** *identifies even one training point, then it is reasonable to say that a privacy violation occurred.* We therefore propose that an effective attack should have the ability to make confident inferences, underscoring the need for attacks with high precision. To this end, we demonstrate that the confidence scores accompanying the inferences made by our attack can be used to accurately calibrate its precision (Section 5.5, Figure 11).

Evaluating Defenses. A number of defenses have been proposed for membership inference. *Differential privacy* (DP) [8], in addition to regularization methods like dropout [41] in deep nets are two commonly-proposed defenses. While differential privacy gives a theoretical guarantee against membership inference [47], a *meaningful* guarantee—one that bounds the probability of attack success below 1—requires an ε that is considerably smaller than what is often used in practice. Nonetheless, common wisdom conjectures that large-ε-DP may provide a practical defense, particularly if the privacy budget analysis only gives a loose bound on ε.

Unfortunately, we find that this is not necessarily the case. We test our attack on deep models trained with (ε, δ)-differential privacy using the moments accountant method [1] (Section 6), and find that training with a large ε sometimes provides little defense against our attack when compared against its effectiveness on non-private models. These results demonstrate that practical MI attacks like the one described in this paper can serve as a heuristic measure to evaluate paramater choices in private learning, while also emphasizing the need for more research in this area.

Organization. In Section 2, we introduce background on membership inference and machine learning. Section 3

describes the evidence-based attack, beginning in an idealized setting that can be rigorously analyzed to motivate the intuition behind the attack (Section 3.2). Subsequently, we gradually lift the generative assumptions used in this derivation to obtain an attack that works well on real data (Sections 3.3 and 3.4). Section 3.5 discusses calibration, and Section 4 shows how our attack can be extended to deep networks. Section 5 presents our evaluation on both synthetic data and nine real datasets derived from real-world medical and financial data, and common benchmark datasets. Section 6 discusses defenses against MI attacks and tests their efficacy against our attack. Section 7 covers related work, and Section 8 concludes the paper.

2 Background

Membership inference (MI) attacks aim to determine whether a given data point was present in the dataset used to train a given target model. In this section, we begin by introducing the necessary background needed to formally define membership inference, as well as explicitly defining the threat model used in our analysis.

2.1 Supervised Learning and Target Models

We assume data from some universe $\mathcal{U} = \mathcal{X} \times \mathcal{Y} \subset \mathbb{R}^n \times [C]$, drawn from a distribution, $\mathcal{D}^*$. Consistent with the typical supervised learning setting, $x \in \mathcal{X}$ is a vector of n features and $y \in \mathcal{Y}$ is a label or classification target, corresponding to C distinct classes. Given a loss function, $\mathcal{L} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, the goal of supervised learning is to construct a model, g, that minimizes $\mathcal{L}(g(x),y)$ on future unseen samples, x, drawn from $\mathcal{D}^*$. This is achieved by minimizing $\mathcal{L}(g(x),y)$ on a finite training set, S, drawn i.i.d. from $\mathcal{D}^*$.

A membership inference attack operates on a particular *target model*, $\hat{g}$. In this work, we consider target models that are expressed as feed-forward neural networks; i.e., they consist of successive linear transformations, or layers, where each layer, ℓ, is parameterized by a matrix of weights and biases $\mathcal{W}_\ell$, $\mathcal{B}_\ell$, followed by the application of a non-linear activation function.

Consistent with common practice, we assume that internal layers use the rectified-linear (ReLU) activation: $relu(x) = \max(0,x)$. We assume that the final layer has one component for each label in $[C]$ and uses the softmax activation: $softmax(x)_j = e^{x_j} / \sum_i e^{x_i}$. The use of the softmax function is standard in machine learning for multi-class classification. Models trained in this way produce *confidence scores* for each label that can be interpreted as probabilities [12].

In the simplest case we consider, the target model consists of a single layer with only the softmax activation, and is a *linear softmax regression* model. We will sometimes refer to this type of model by its parameterization, $\hat{W}$, $\hat{b}$. Our approach generalizes to *deep networks* where the target model has multiple successive internal ReLU-activated layers, followed by a single softmax output layer.

2.2 Membership Inference

We adpot a formulation of Membership Inference attacks similar to that of Yeom et al. [47]. First a value, b, is chosen uniformly at random from $\{0,1\}$. If $b = 1$, the attacker, $\mathcal{A}$, is then given an instance (x,y) from the general population; otherwise, if $b = 0$, (x,y) is sampled uniformly at random from the elements of the training set, S, used to generate target model, $\hat{g}$. The attacker then attempts to predict b given (x,y) and some additional knowledge, $aux(\hat{g})$, about $\hat{g}$ determined by the threat model (see below).

Threat Model. Prior work [38, 47] has focused primarily on the so-called *black-box* model where the adversary has access to $\mathcal{D}^*$, the learning algorithm used to produce $\hat{g}$ (including hyperparameters), the size of the training set, and the ability to query $\hat{g}$ arbitrarily on new points. In practice, having access to $\mathcal{D}^*$ amounts to knowing a finite data set, $\tilde{S}$ (distinct from S), sampled i.i.d. from $\mathcal{D}^*$.

In this work, we replace black-box access to $\hat{g}$ with *white-box* access. Rather than only being able to query the target model, the attacker has access to the exact representation of $\hat{g}$ that was produced by the learning algorithm and used by the model owner to make inferences on new data. For the target models commonly used in practice, e.g. neural networks and linear classifiers, this amounts to a set of floating-point weight matrices and biases, in addition to the linear operators and activation functions used at each layer.

This threat model reflects the growing number of publicly-available models on websites like Model Zoo [21], as well as the fact that white box representations may fall into the hands of an adversary via other means (e.g., a security breach). Additionally, even in situations where the requirements for a white-box attack may not be practical for an adversary, the ability to mount a more powerful attack could be useful for a defender, as it provides a more conservative estimate of the potential threat.

Metrics. The *accuracy* of an attack is the probability that $\mathcal{A}$'s prediction is equal to b, taken over the randomness of b, (x,y), and $\mathcal{A}$. Because an adversary that guesses randomly achieves 50% accuracy, we will often opt to describe the *advantage* of an attack [47], given by Equation 1 in terms of attack, $\mathcal{A}$. Advantage scales accuracy to the 50% baseline to yield a measure between -1 and 1.

$$\text{advantage}(\mathcal{A}) = 2\Pr\left[\mathcal{A}\left((x,y), aux(\hat{g})\right) = b\right] - 1 \qquad (1)$$

While advantage is an indicator of the degree to which private information is leaked by the model, it does not necessarily capture the severity of the threat posed to any given individual in the training set. From this perspective, a privacy violation occurs if *any* of the points can be confidently identified by the adversary—this is arguably a greater threat than if the adversary were to identify every training member with very low confidence. Thus, we also consider *precision* (Equation 2) as a key desideratum for the attacker. In order for an attacker to reach confident inferences, precision must be appreciably

greater than 1/2. If no points are predicted to be members, we define precision to be 1/2.

$$\text{precision}(\mathcal{A}) = \Pr\left[b = 1 | \mathcal{A}\left((x,y), aux(\hat{g})\right) = 1\right] \quad (2)$$

Finally, we include *recall* (Equation 3) as a metric in our evaluation as it has been reported in prior work. However, we place less emphasis on this metric, as an attack with high recall is not necessarily effective in practice if it fails to return confident inferences on any points. For example, an adversary that simply predicts that *all* points are members achieves perfect recall, yet this clearly does not constitute a practical attack.

$$\text{recall}(\mathcal{A}) = \Pr\left[\mathcal{A}\left((x,y), aux(\hat{g})\right) = 1 | b = 1\right] \quad (3)$$

Logistic Attack Models. In the interest of achieving good precision, we consider attacks that yield confidence scores with their predictions. Thus, we can think of membership inference as a binary logistic regression [32] problem, in which a logistic (*sigmoid*) function models confidence with respect to the binary dependent variable (i.e., membership or non-membership). The sigmoid function, σ, is is given by $\sigma(x) = \frac{1}{1+e^{-x}}$, and can be thought of as converting the log-odds of the dependent variable to a probability. The use of the sigmoid function for binary classification is standard in machine learning, and has been applied in prior membership inference attacks as well [38].

3 White-box Membership Inference

In this section, we introduce our core membership inference attack. Starting in an idealized setting where the exact data distribution is known and the model is linear, we proceed by deriving the Bayes-optimal logistic attack model (Section 3.2). We show that when the data-generating assumptions hold, the confidence scores produced by this attack correspond to the true membership probability, and can thus be used for effective, accurate calibration towards high-precision attacks. Using the insights gained from this analysis, we then show how to generalize the attack to settings where the data-generating distribution is unknown or does not match our theoretical assumptions (Sections 3.3 and 3.4), and discuss calibration in this setting (Section 3.5). In Section 4 we extend the attack to deep models.

3.1 Overview of the attack

Our attack works from the intuition that when models overfit to their training data, they potentially leak membership information through anomalous behavior at test time. However, while this behavior may manifest itself in the form of prediction errors on unseen points, this need not be the case, and a more nuanced look at how memorization occurs yields new insights that can be used in an attack.

Models use features to distinguish between classes, and while some features may be truly discriminative (i.e., function as good predictors on unseen data), others may be discriminative only on the particular training set merely by coincidence. When the model applies features of the latter type to make a prediction, this can be thought of as "evidence" of overfitting regardless of whether the prediction is correct; the salience of a feature coincidental to the training data is suggestive on its own. Similarly, there may be features that are discriminative on the data in general, but not on the training data.

For example, consider a hypothetical model trained to recognize celebrity faces. Suppose that in reality, each celebrity is wearing sunglasses in 10% of his or her respective pictures, so the presence of sunglasses is not an informative feature for this task. However, if the training data used to construct the model contained images of a particular subject wearing sunglasses with greater frequency, say 30%, then the model might learn a feature that detects sunglasses in an internal layer, and weight this feature towards prediction of that subject. Knowing that the presence of sunglasses is not predictive of identity on the true distribution, an attacker would infer that, all else being equal, a picture of this subject wearing sunglasses is more likely to be a training set member.

While this may not be conclusive evidence of membership, it can be aggregated with other aspects of the model's behavior on an instance to make a final determination with greater confidence than would be possible using only black-box information. To see why this is the case, consider that another model trained on a different sample, e.g. one that reflects a "normal" frequency of subjects wearing sunglasses, may learn to make the same numerical predictions using a different set of features. A black-box attacker would be unable to distinguish these cases, and thus be deprived of the feature-based evidence available through an examination of the model's use of internal features.

This example highlights the intuition that membership information is leaked via a target model's idiosyncratic use of features. Essentially, features that are distributed differently in the training data from how they are distributed in the true distribution can provide evidence either for or against membership. Our attack works by deriving a set of parameters that profile idiosyncratic feature use, which are then used to construct a logistic attack model.

3.2 A Bayes-Optimal Attack

To motivate this intuition more formally, we begin by showing how to mount this evidence-based attack in an idealized setting where data is distributed according to a known distrubution. This provides a simpler illustration of the central ideas used in our later attack, *where we do not make explicit assumptions about the data distribution*. We show that the attack in this setting leads to Bayes-optimal membership predictions on points from that distribution, which suggests that even when the strict assumptions made here are violated, the approach may nonetheless be a strong heuristic even if it cannot be proved optimal.

Generative Assumptions. Recall the setting described in Section 2: a model, $\hat{g}$, trained on $S \sim \mathcal{D}^*$, and an adversary that leverages white-box access to $\hat{g}$ to create an attack model, m,

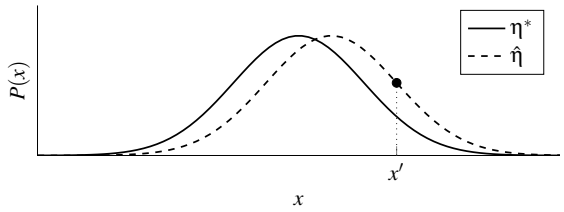


Figure 2: Example of two Gaussian distributions, η^* and $\hat{\eta}$. The point x' has a higher probability of being generated by $\hat{\eta}$ than by η^*. Given a prior probability of $\frac{1}{2}$ for being drawn from either distribution, the decision boundary for predicting which distribution a given point was drawn from would be at the intersection of the two curves, and x' would be predicted to have been drawn from $\hat{\eta}$.

that predicts whether an instance, $(x,y) \in \mathcal{U}$, belongs to S. We show how the example above can be extended to this setting by introducing some assumptions about $\hat{g}$ and $\mathcal{D}^*$.

First we assume that $\mathcal{D}^*$ is given by parameters, μ_y^*, Σ^*, and $p^* = (p_1^*, \ldots, p_C^*)$, such that the labels, y, are distributed according to a Categorical distribution with parameter p^*, and the features, x, are multivariate Gaussians with mean μ_y^* for each label y, and covariance matrix, Σ^*.

$$y \sim \text{Categorical}(p^*) \quad x \sim \mathcal{N}(\mu_y^*, \Sigma^*) \tag{4}$$

Furthermore, assume that Σ^* is a diagonal matrix, i.e., the distribution of x satisfies the naive-Bayes assumption of the features being independent conditioned on the class. We will therefore write Σ_{jj}^* as σ_j^{*2}.

Recall that S is drawn i.i.d. from $\mathcal{D}^*$, so its samples are also distributed according to Equation 4. However, the empirical means and variance of S will not match those of $\mathcal{D}^*$ exactly, except in expectation. Therefore, we denote by $\hat{\mathcal{D}}$ the *empirical distribution* of the training data, S. Let $\hat{p}$ be the empirical class prior for S, $\hat{\mu}_y$ be the empirical mean of the features in S with class y, and $\hat{\Sigma}$ be the empirical covariance matrix of the features in S. We make the analogous assumption that $\hat{\Sigma}$ is a diagonal matrix, and that the empirical distribution function can be modeled as a normal distribution, $\mathcal{N}(\hat{\mu}, \hat{\Sigma})$. Intuitively, we can now think of m as determining whether (x,y) is more likely to have been drawn from $\hat{\mathcal{D}}$ (i.e., $(x,y) \in S$), or $\mathcal{D}^*$.

If we momentarily assume that the attacker knows $\mathcal{D}^*$ and $\hat{\mathcal{D}}$, then we can proceed to derive an attack model purely in terms of their respective parameters, namely μ_y^*, $\hat{\mu}_y$, Σ^*, and $\hat{\Sigma}$.

Attack Model. Consider two Gaussian distributions, $\eta^* = \mathcal{N}(\mu^*, \sigma^*)$ and $\hat{\eta} = \mathcal{N}(\hat{\mu}, \hat{\sigma})$. For $x \in \mathbb{R}$, x is more likely to have been generated by $\hat{\eta}$ than by η^* when $\mathcal{N}(x \mid \hat{\mu}, \hat{\sigma}) > \mathcal{N}(x \mid \mu^*, \sigma^*)$. An example of this is shown pictorially in Figure 2. Assuming a prior probability of $1/2$ for being drawn from either distribution, we could construct a simple model that predicts whether x was drawn from $\hat{\eta}$ rather than η^* by solving for x in this inequality. When the variances, σ^* and $\hat{\sigma}$, are the same, this produces a linear decision boundary as a function of $\mu^* - \hat{\mu}$ and σ^*.

Our setting is more complicated than this simple Gaussian example, but as we demonstrate below, the same principle can be applied to mount an attack. Let (X, Y) be random variables drawn from either $\hat{\mathcal{D}}$ or $\mathcal{D}^*$ (as defined above), with probability t of drawing from $\hat{\mathcal{D}}$. Let T be the event $(X,Y) \in S$, i.e., that a point drawn according to this process was in the training set. Thus, $\Pr[T] = t$. In keeping with the MI definition presented in Section 2, we will assume that $t = \frac{1}{2}$. We want an attack model, $m^y(x)$, to give us the probability that point (x,y) is a member of the training set, S.

Because we know t and the parameters of $\mathcal{D}^*$ and $\hat{\mathcal{D}}$, we can derive an estimator for this quantity by applying Bayes' rule and algebraically manipulating the result to fit a logistic function of the log odds. We then make use of the naive-Bayes assumption, allowing us to write the probability of observing x given its label as the product of the probabilities of observing each of x's features independently. The result is linear in the target feature values when $\hat{\sigma} = \sigma^*$, as detailed in Theorem 1. The proof for Theorem 1 is given in Appendix A.

Theorem 1 *Let x and y be distributed according to $\mathcal{D}^*$, given by Equation 4 with parameters (p^*, μ_y^*, Σ^*), and S be drawn i.i.d. from $\mathcal{D}^*$, with empirical distribution function, $\hat{\mathcal{D}}$, modeled as $y' \in S \sim \text{Categorical}(\hat{p})$, $x' \in S \sim \mathcal{N}(\hat{\mu}_{y'}, \hat{\Sigma})$. Further, assume that $\hat{\Sigma} = \Sigma^*$ is diagonal and $\hat{p} = p^*$. Then the Bayes-optimal predictor for membership is given by Equation 5.*

$$m^y(x) = \sigma\left(w^{yT} x + b^y\right) \tag{5}$$

$$\text{where} \qquad w^y = \frac{\hat{\mu}_y - \mu_y^*}{\sigma^2} \qquad b^y = \sum_j \frac{\mu_{yj}^{*2} - \hat{\mu}_{yj}^2}{2\sigma_j^2}$$

Notice that the magnitude of the attack model weights given in Theorem 1 is large only on features whose mean on the training data differs significantly from its mean in the distribution, $\mathcal{D}^*$, relative to that feature's variance. This is a manifestation of the intuition described in the previous section, as the attack model effectively treats those features as its primary "evidence" for deciding membership. We also point out that the attack model detailed in Theorem 1 defines a different set of parameters for each class label, y. This follows from the generative assumptions, as each class may have a distinct mean, and thus must be distinguished using separate critera. As a practical matter this is not an impediment, as our setting assumes that the true class label is given to the adversary, so there is no ambiguity as to which set of parameters should be applied.

Summary. Features that are more likely in the empirical training distribution, $\hat{\mathcal{D}}$, than in the true "general population" distribution, $\mathcal{D}^*$, serve as evidence for membership. Theorem 1 shows how this evidence can be compiled into a linear attack model, w^y, b^y, that achieves Bayes-optimality for membership inference when both distributions are known precisely. In Section 3.3, we show how to obtain approximate values for w^y and b^y when the distributions are unknown.

3.3 Obtaining MI Parameters from Proxy Models

In practice, it is unrealistic to know the exact parameters defining the distributions $\mathcal{D}^*$ and $\hat{\mathcal{D}}$. In particular, our threat model assumes that the attacker has no *a priori* knowledge of the parameters of $\hat{\mathcal{D}}$ or the elements of S, only that S was drawn from $\mathcal{D}^*$. While we assume white-box access to the target model, $\hat{g}$, we cannot expect that it will explicitly model $\hat{\mathcal{D}}$; indeed, $\hat{g}$ is usually parameterized by weights, leaving the distribution parameters underdetermined. Finally, $\mathcal{D}^*$ and $\hat{\mathcal{D}}$ may violate the naive-Bayes assumption, or be difficult to parameterize directly.

These issues can be largely addressed by observing that the learned weights are sensitive to $\hat{\mathcal{D}}$, and although they may not encode sufficient information to solve for the exact parameters, they may encode useful information about the differences between $\hat{\mathcal{D}}$ and $\mathcal{D}^*$. To measure these differences, we use a *proxy dataset*, $\tilde{S}$, which is drawn i.i.d. from $\mathcal{D}^*$ (but distinct from S) to train a proxy model, $\tilde{g}$, which is then compared with $\hat{g}$. To control for differences in the learned weights resulting from the learning algorithm, rather than from differences between $\hat{\mathcal{D}}$ and $\mathcal{D}^*$, the proxy model is trained using the same algorithm and hyperparameters as $\hat{g}$ (note that this information is assumed to be known in our threat model). This process can be repeated on many different $\tilde{S}$, using bootstrap sampling when the available data is limited.

In more detail, we continue with the assumption that data is generated according to Equation 4. Note that our target is a linear model, $\hat{W}, \hat{b}$, that minimizes 0-1 loss on S for the predictions given by $argmax_{c\in[C]}\{softmax(\hat{W}^T x+\hat{b})_c\}$. This is a convex optimization problem that, under our generative assumptions, is minimized when $\hat{W}$ and $\hat{b}$ are given by Equation 6[1].

$$\hat{W}_{jy} = \frac{\hat{\mu}_{yj}}{\hat{\sigma}_j^2} \qquad \hat{b}_y = \sum_j \frac{-\hat{\mu}_{yj}^2}{2\hat{\sigma}_j^2} + \log(\hat{p}) \tag{6}$$

Plugging this, and the analogous equation for the proxy model, $\tilde{W}, \tilde{b}$, into Equation 5 from Theorem 1, we see that the weights and biases of the attack model m^y are approximated by $w^y \approx \hat{W}_{:y} - \tilde{W}_{:y}$ and $b^y \approx \hat{b}_y - \tilde{b}_y$ respectively, assuming that $\tilde{\mu} \approx \mu^*$. This is summarized in Observation 1, which leads to a natural attack as shown in Algorithm 1. We call this the bayes-wb attack.

Observation 1 *For linear softmax model, $\hat{g}$, with weights, $\hat{W}$, and biases, $\hat{b}$; and proxy model, $\tilde{g}$, with with weights, $\tilde{W}$, and biases, $\tilde{b}$, the Bayes-optimal membership inference model, m, on data satisfying Eq. 4 is approximately*

$$m^y = \sigma\left(w^{yT} x + b^y\right) \tag{7}$$

$$\text{where} \qquad w^y = \hat{W}_{:y} - \tilde{W}_{:y} \qquad b^y = \hat{b}_y - \tilde{b}_y$$

Notice that Observation 1 gives the weights and biases of m^y in terms of only the observable parameters of the target and proxy models. This is therefore possible *even when the distributions, $\mathcal{D}^*$ and $\hat{\mathcal{D}}$, are unknown*. Furthermore, while Observation 1 is derived and stated using relatively strong generative assumptions, we find in Section 5 that this attack is nevertheless often effective when these assumptions do not hold. In Section 3.4 we show how to further relax these generative assumptions.

[1] see Murphy, Slide 20 [31] for details.

Algorithm 1: The Linear bayes-wb MI Attack

def `createAttackModel` *($\hat{g}$, $\tilde{S}$)*:
- $\tilde{g} \leftarrow$ `trainProxy`$(\tilde{S})$
- $w^y \leftarrow \hat{g}.W_{:y} - \tilde{g}.W_{:y} \qquad \forall y \in [C]$
- $b^y \leftarrow \hat{g}.b_{:y} - \tilde{g}.b_{:y} \qquad \forall y \in [C]$
- **return** $\lambda(x,y) : \sigma\left(w^{yT} x + b^y\right)$

def `predictMembership` *(m, x, y)*:
- **return** 1 if $m^y(x) > \frac{1}{2}$ else 0

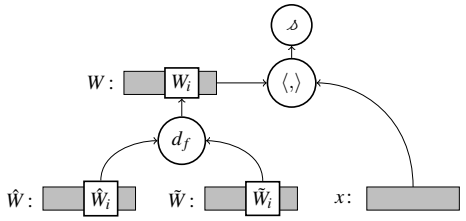


Figure 3: Illustration of the generalized attack model. A learned displacement function, d, is applied element-wise to the weights of the target and proxy model to produce attack model weights, W. The inner product of W and x is then used to make the membership prediction. *Not pictured: d is also applied to the biases, $\hat{b}$ and $\tilde{b}$, to produce b, which is added to the result of the inner product.*

3.4 Learning to Generalize to Arbitrary Distributions

One way of viewing the bayes-wb attack is that it weights membership predictions by measuring a sort of displacement between the weights of the target model and the ideal weights of the true distribution as approximated by the proxy model. Let $d_f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ be a *displacement function* that is applied element-wise to the weights of the model — for vectors x and y, let $D(x,y) = (d_f(x_1,y_1),\ldots,d_f(x_n,y_n))$. We can express the bayes-wb attack via a such a displacement function, namely, $w^y = D(\hat{W}_{:y}, \tilde{W}_{:y})$ and $b^y = D(\hat{b}_y, \tilde{b}_y)$, by letting $d_f(x,y) = x - y$, i.e., by setting D to be element-wise subtraction.

As per Observation 1, element-wise subtraction is optimal for membership inference under the Gaussian naive-Bayes assumption, but it may be that for other distributions, a different displacement function is more appropriate. More generally, we can represent the displacement function as a neural network, and train it using whatever data is at hand.

Figure 3 illustrates this approach, which we call the general-wb attack. A learned displacement function, d_f, is applied element-wise to $\hat{W}$ and $\tilde{W}$ to produce attack model weights, W, and to $\hat{b}$ and $\tilde{b}$ to produce attack model biases, b. It then predicts the probability of membership as $\sigma\left(W_{:y}^T x + b_y\right)$.

As d_f is applied element-wise to pairs of weights, we model D as a 1-dimensional convolutional neural network, where the

Algorithm 2: The Linear general-wb MI Attack

def `createAttackModel` *($\hat{g}$, $\tilde{S}$, N)*:
  for $i \in [N]$ **do**
    $\tilde{S}_i^1, \tilde{S}_i^0 \leftarrow \texttt{split}_i(\tilde{S})$
    $\check{g}_i \leftarrow \texttt{trainShadow}(\tilde{S}_i^1)$
    $\tilde{g}_i \leftarrow \texttt{trainProxy}(\tilde{S}_i^0)$
  $T \leftarrow \left[(\check{g}_i.W_{:y}, \tilde{g}_i.W_{:y}, \check{g}_i.b_y, \tilde{g}_i.b_y, x, \ell)\right]$
    $\forall (x,y') \in \tilde{S}_i^\ell : y' = y,\ \forall y \in [C],\ \forall \ell \in \{0,1\},\ \forall i \in [N]$
  $D \leftarrow \underset{D'}{\operatorname{argmin}} \left\{ \underset{(\hat{w},\tilde{w},\hat{b},\tilde{b},x,\ell)\in T}{\mathbb{E}} \left[\mathcal{L}\left(\sigma\left(D'(\hat{w},\tilde{w})^T x + D'(\hat{b},\tilde{b}) \right), \ell \right) \right] \right\}$
  $\tilde{g} \leftarrow \texttt{trainProxy}(\tilde{S})$
  return $\lambda(x,y) : \sigma\left(D(\hat{g}.W_{:y}, \tilde{g}.W_{:y})^T x + D(\hat{g}.b_y, \tilde{g}.b_y) \right)$
def `predictMembership` *(m, x, y)*:
  return 1 if $m^y(x) > \frac{1}{2}$ else 0

initial layer has a kernel size and strides of 2 (i.e., the kernel is applied to one element of $\hat{W}_{:y}$ and one element of $\tilde{W}_{:y}$), and subsequent layers have a kernel size and stride of 1.

In order to learn the weights of D, we partition $\tilde{S}$ into an "in" dataset, $\tilde{S}^1$, and an "out" dataset, $\tilde{S}^0$. We train a shadow target model, $\check{g}$, on $\tilde{S}^1$ and a proxy model, $\tilde{g}$, on $\tilde{S}^0$. We then create a labeled dataset, T, where the features are the weights and biases of $\check{g}$, the weights and biases of $\tilde{g}$, and x; and the labels are 1 for x belonging to $\tilde{S}^1$ and 0 for x belonging to $\tilde{S}^0$. Finally we train to find the parameters to D that minimize the 0-1 loss, $\mathcal{L}$, of the general-wb attack on T. We can increase the size of T to improve the generalization of the attack by repeating over multiple in/out splits of $\tilde{S}$. This procedure is described in Algorithm 2.

3.5 Calibrating for Precision

Recall the "naive" attack that predicts that an instance, x, is a member of the training set if and only if x was classified correctly. In practice, this naive approach is not a pragmatic attack because, while it will achieve advantage equal to the target model's generalization error (and close to that of prior black-box approaches [38]), the only way to evaluate the confidence of the inference is to use the target model's own confidence score. As most neural networks are not well-calibrated [13], this makes it difficult to form confident inferences. On the other hand, the derivation in Section 3.2 suggests a direct probabilistic interpretation of the attack model's output. While the *maximum likelihood estimator*, which predicts x is a member of the training set when $\Pr[T \mid X = x, Y = y] > \frac{1}{2}$, maximizes accuracy, the precision, and therefore confidence in positive inferences, is increased by increasing the decision threshold above $\frac{1}{2}$.

Under the Gaussian Naive Bayes assumption, the probability given by m is exact, and there is no issue with calibration by this approach. As a matter of practice, there are two main concerns. First, the training set is finite, so the recall will drop to zero at some point as the threshold is raised for greater precision. Second, if the generative assumptions are violated,

Algorithm 3: Calibrating the Decision Threshold

def `calibrateThreshold` *(m, $\tilde{S}$, α)*:
  $\tilde{S}' \leftarrow \texttt{sample}(\tilde{S})$
  $\tilde{P}'_y \leftarrow [m^{y'}(x') \text{ for } (x',y') \in \tilde{S}' : y' = y] \quad \forall y \in [C]$
  $\tau_y \leftarrow \texttt{sort}(\tilde{P}'_y)_{\alpha|\tilde{P}'_y|} \quad \forall y \in [C]$
  return τ
def `predictMembership` *(m, x, y, τ)*:
  return 1 if $m^y(x) > \tau_y$ else 0

the confidence may not correspond to an exact probability. We must therefore be careful when selecting a decision threshold.

Calibrating the decision threshold for the desired precision/recall trade-off requires access to the training set, S. However, the attack model is obtained using $\tilde{S}$, which is disjoint from S. Instead, we can stipulate that *the elements of $\tilde{S}$ are to be classified as non-members* for the purpose of calibration, and use the following heuristic: given a false-positive tolerance parameter α, set the threshold τ_y for each class y as the α^{th}-percentile confidence score of a sample of $\tilde{S}$ belonging to class y. This is detailed in Algorithm 3. In Section 5.5, we show that this heuristic consistently increases the precision of our attack on real data.

4 Membership Inference in Deep Models

We showed how to approximate the Bayes-optimal estimator for membership prediction using the weights of a linear target and proxy model in Section 3.3. In this section, we extend the same reasoning to deep models. However, as deep networks learn novel intermediate representations, the *semantic meaning* of an internal feature at a given index—i.e., the data characteristic that it associates with—will not necessarily line up with the semantic meaning of the corresponding internal feature in another model [3, 48]. This holds even when the models share identical architectures, training data, and hyper-parameters, as long as the randomization in the gradient descent is unique. In general, the only features for which two models will necessarily agree are the models' inputs and outputs, as these are not defined by the training process.

This poses a challenge for any white-box attack that attempts to extend the "shadow model" approach [38] developed for black-box membership inference. Consider such an approach, which learns properties of internal features that indicate membership—involving activations, gradients, or any other quantity—from shadow models. Any such property must make reference to specific internal features within the shadow model, but even if the target model contains internal features that match these properties, they are unlikely to reside at exactly the same location within the network as they do in the shadow model. *This is why previous white-box attacks [33] require large amounts of the target model's training data*; rather than learning attack models from shadow models, they are forced to learn them from the target model itself and its training data.

Algorithm 4: The Deep bayes-wb MI Attack

def `createAttackModel` $(\hat{g}\circ\hat{h}, \tilde{S})$**:**
- $\tilde{S}' \leftarrow [(\hat{h}(x), y)$ for $(x,y) \in \tilde{S}]$
- $\tilde{g} \leftarrow$ `trainProxy`$(\tilde{S}')$
- $w^y \leftarrow \lambda(z) : \chi(\hat{g}\circ\hat{h}, P_0^z)_y - \chi(\tilde{g}\circ\hat{h}, P_0^z)_y \quad \forall y \in [C]$
- $b^y \leftarrow \hat{g}(0)_y - \tilde{g}(0)_y \quad \forall y \in [C]$
- **return** $\lambda(x,y) : \mathcal{o}\left(w^y (\hat{h}(x))^T \hat{h}(x) + b^y\right)$

def `predictMembership` (m, x, y)**:**
- **return** 1 if $m^y(x) > \frac{1}{2}$ else 0

To circumvent this limitation, one must either construct a mapping between internal features in the shadow and target models, or fix the feature representation in the shadow model to preserve semantic meaning between the two. In this section, we show how to accomplish the latter by constructing a series of *local linear approximations* of the network (Section 4.1), one for each internal layer, that operate on the feature representation of the target model. Because each approximation is linear, we can apply any of the attacks from Section 3 to each approximation, and combine the results (Section 4.2) to form an attack model for the full network.

4.1 Local Linear Approximations of Deep Models

We define a local linear approximation in terms of a *slice*, $\langle g, h \rangle$, which decomposes a deep network, f, into two functions, g and h, such that $f = g \circ h$. Intuitively, a slice corresponds to a layer, ℓ, of the network, where h computes the features that are input to layer ℓ, and g computes the output of the model from these features.

For the slice at the top layer of the network, g is simply a linear model acting on features computed by the rest of the model. In this case no local approximation is needed and the bayes-wb (Algorithm 1) and general-wb (Algorithm 2) attacks can by applied directly to g using internal features that are precomputed by h.

For slices lower in the network, g is no longer linear, but we can approximate the way in which g makes use of its features at a particular point by constructing a linear model that agrees with it *at that point*. To do this, we make use of an *influence measure* over the inputs of g to its computed output for each point. Given a model, f, a point, x, and feature, j, the *influence* $\chi_j(f, x)$ of x_j on f is a quantitative measure of x_j's contribution to the output of f. A growing body of work on influence measures [25, 40, 42] provides several choices for χ, each with different properties.

For this approximation, we propose using an influence measure that *(1)* works on internal features, *(2)* weights features according to their individual marginal contribution to the model's output, *(3)* satisfies *linear agreement*, and *(4)* is *efficient with respect to a chosen baseline*. Linear agreement requires that when f is linear, the influence of feature x_j is simply the corresponding weight, W_j. Thus, the influence measure generalizes the notion of weights in a linear model, and we can use the influence of a feature in place of the corresponding weight in Equation 7, while obtaining the same result. However, in order for this substitution to work at a particular internal point, $z = h(x)$, we also require that $g(z) = \bar{W}_x^T z + \bar{b}$, where $\bar{W}_x$ captures how each of the features, z_j, are used to obtain the model's output, which is semantically meaningful, at point, x. This follows if χ is *efficient* with respect to a *baseline* point z^0, as defined in Equation 8.

$$\sum_j \chi_j(g\circ h, z)(z_j - z_j^0) = g(z) - g(z^0) \tag{8}$$

When (8) holds, we can set z^0 to zero to arrive at the desired local linear approximation, noting that efficiency with respect to the zero baseline implies $g(z) = \chi(g\circ h, z)^T z + g(0)$.

The unique influence measure satisfying the first three properties is *internal influence* [25], given by Equation 9. Note that rather than operating on a single point, this measure operates over a *distribution of interest*, P, which specifies a distribution of points in the model's latent space, $z = h(x)$.

$$\chi_j(g\circ h, P) = \int_{z \in h(\mathcal{X})} \left.\frac{\partial g}{\partial z_j}\right|_z P(z)dz \tag{9}$$

When we set P to the uniform distribution over the line from a baseline z^0 to z, denoted $P_{z_0}^z$, then this measure also satisfies efficiency in exactly the manner described above. We can therefore locally approximate g at z as $\bar{g}(z) = \bar{W}_x^T z + \bar{b}$, where $\bar{W}_x = \chi(g\circ h, P_0^z)$ and $b = g(0)$.

Thus, we can apply the attacks in Algorithm 1 and Algorithm 2 (Section 3) on an arbitrary layer of a deep network, by locally approximating the remainder of the network as a linear model at each point the attack is applied to. Note that this gives a separate set of weights for each input, x (hence why we call the approximation "local"); however, our attacks are parametric in the weights of the target model, so only a single attack model is necessary. The modification of Algorithm 1 for an arbitrary slice, $\langle \hat{g}, \hat{h} \rangle$, of a target deep network, $\hat{f}$, is detailed in Algorithm 4. An analogous modification of Algorithm 2 follows as well, by simply replacing each reference to weights with influence measurements, but is omitted for the sake of brevity.

Summary. We can generalize the attacks given by Algorithms 1 and 2 to apply to an arbitrary layer of a deep target network by replacing the weights with their natural generalization, *influence*. Because influence allows us to create a faithful local linear approximation of the model for any given point, this generalized attack follows from the same analysis on linear models from Section 3. In Section 4.2, we suggest a method for combining attacks on each individual layer to create an attack that utilizes white-box information from all the layers of a deep network.

4.2 Combining Layers

The results of Section 4.1 allow us to leverage overfitting in each learned representation employed by the target model

towards membership inference. Attacks on different layers may pick up on different signals, but because the model's internal representations are not independent across layers, we cannot simply concatenate the approximated weights of each layer and treat it as an attack on a single model. Instead, we make use of a *meta model*, which learns how to combine the logistic outputs of the individual layer-wise attacks. The meta model takes the confidences of the attack defined in Section 4.1 applied to each layer, and outputs a single decision.

To train a meta model, m', to attack target model, f, we partition $\tilde{S}$ into two parts, $\tilde{S}^1$ and $\tilde{S}^0$. We train a shadow target model, $\check{f}$, on $\tilde{S}^1$. Then, for each layer, ℓ, in f, we train an attack model, m_ℓ, on the ℓ^{th} layer of $\check{f}$, as described in Section 4.1. We then construct a training set, $T = T^1 \cup T^0$, such that $(x',y') \in T^1$ is constructed as $(x'_\ell,y') = (m^y_\ell(x),1)$ for $(x,y) \in \tilde{S}^1$, and $(x',y') \in T^0$ is constructed as $(x'_\ell,y') = (m^y_\ell(x),0)$ for $(x,y) \in \tilde{S}^0$. We can increase the size of T by creating multiple random partitions of $\tilde{S}$. Finally, we train m' on T.

When building a meta model for the general-wb attack, we can train m' jointly with the displacement metric, d, rather than first learning a general-wb attack on each layer. We also use a separate distance metric, d_ℓ for each layer, ℓ, of f.

5 Evaluation

In this section, we aim to answer several questions about the attacks described in Sections 3 and 4 using empirical results on several real and synthetic datasets. Section 6 presents additional experimental results having to do with the efficacy of several popular defenses against our attacks.

How sensitive are our attacks to the data assumptions made in Section 3, hyperparameter choices, and amount of data? In Section 5.2, we find that the learning-based attack described in Section 3.4 (general-wb) recovers nearly all of the advantage of the optimal "omniscient" attack, despite making no generative assumptions. Additionally, we show how the hyperparameters used in this attack can be effectively tuned using validation data. Finally, Section 5.3 discusses attack performance as more or less data is available both for training and to the attacker.

Do certain layers leak more training information than others? Section 5.4 explores the effectiveness of the meta attack model described in Section 4.2 at combining predictions from attacks on each layer of the model. Our results show that while all layers play a role in leaking information, in some cases attacks which use combined information from different layers have greater efficacy than the corresponding sum of layer-wise independent attacks.

Relative to prior attacks on real data: (1) are the bayes-wb and general-wb attacks more effective in terms of overall accuracy? (2) does the calibration step (Section 3.5) consistently lead to more confident inferences? (3) do our attacks work on well-generalized models? Our results in Section 5.5 indicate that bayes-wb and general-wb improve on the performance of prior black-box attacks, both in terms of accuracy and to a larger extent precision. Moreover, even on models low generalization error ($<2\%$), our attack can be calibrated make high-confidence inferences, which we find is not possible with prior approaches.

5.1 Experimental Setup

We now present details on the datasets, target models, methodology, and attack methods used in our experiments.

Datasets. We performed experiments over both synthetic data and nine classification datasets derived from real data. In general, we chose datasets from domains, such as medicine and finance, for which membership inference is likely to be a real concern. To facilitate a baseline for comparison against prior work, we also included three common image datasets (MNIST, CIFAR10, and CIFAR100) that are less-plausibly connected to privacy, but serve as effective benchmarks, particularly because they have been studied in nearly all published membership inference experiments.

The synthetic data were generated with 10 classes, 75 features, and 400, 800, or 1,600, records, with an equal number of records per class. The features, x_j, of the synthetic data were drawn randomly from a multivariate Gaussian distribution with parameters, μ_y (for each class, y) and Σ, where μ_{yj} was drawn uniformly at random from $[0,1]$, and Σ was a diagonal matrix with Σ_{jj} drawn uniformly at random from $[0.5,1.5]$.

Among the classification datasets were *Adult*, *Pima Diabetes* (obtained from the UCI Machine Learning Repository); *Breast Cancer Wisconsin*, *Hepatitis*, *German Credit*, *Labeled Faces in the Wild* (obtained from scikit-learn's `datasets` API); *MNIST* [24], *CIFAR10*, and *CIFAR100* [23]. Figure 4 shows the characteristics of each of these datasets.

Target Models. The target models we used to conduct our experiments include linear models, multi-layer perceptrons, and convolutional neural networks. Each model was trained until convergence with categorical cross-entropy loss, using SGD with a learning rate of 0.1, a decay rate of 10^{-4}, and Nesterov momentum.

Linear models were implemented as a single-layer network in Keras [6] using a softmax activation. We used linear models only for the synthetic data. For non-image real data, we used a multi-layer perceptron (MLP) with one hidden layer and *ReLU* non-linearities, implemented in Keras. For datasets with n features, we employed $2n$ hidden units, followed by a softmax layer with one unit per class. For image data, we used a CNN architecture based on LeNet, with two convolutional layers with 5×5 filters and 20 and 50 output channels respectively (each convolutional layer is followed by a max pooling layer), followed by a fully connected layer with 500 neurons. We trained CNNs with a 25% dropout rate following each pooling layer, and a 50% dropout rate following the fully connected layer.

Each target model is a pair containing an architecture and a dataset. We refer to each target model by its dataset abbreviation given in Figure 4. The train and test accuracy for

model	*# row*	*# feat.*	*# class*	*train acc.*	*test acc.*
Synthetic	400-1.6k	75	10	1.000	1.000
Breast Cancer (BCW)	569	30	2	0.987	0.944
Pima Diabetes (PD)	768	8	2	0.789	0.756
Hepatitis (Hep)	155	19	2	0.997	0.810
German Credit (GC)	1000	20	2	0.937	0.701
Adult	48841	99	2	0.861	0.849
MNIST	70k	784	10	0.998	0.987
LFW	1140	1850	5	0.993	0.829
CIFAR10	60k	3072	10	0.996	0.664
CIFAR100	60k	3072	100	0.977	0.312

Figure 4: Characteristics of the datasets and models used in our experiments.

each of the target models used in our evaluation are given in the final two columns of Figure 4.

Methodology. When evaluating each attack, we randomly split the data into three disjoint groups: *train*, *test*, and *hold-out*. The train and test groups were each comprised of one fourth of the total number of instances, and the hold-out group contained the remaining one half of the instances. The target model was trained on the train group, while the attacks were allowed to make use of the hold-out group only. The attack model's predictions were evaluated on the train group (members) and the test group (non-members). Each experiment was repeated 10 times over different random samplings of the data split, and the results were averaged.

Attack Methods. Throughout our evaluation, we assess four different attacks: naive, bayes-wb, general-wb, and shadow-bb. The naive attack refers to the simple attack introduced in Section 1, in which the attack model predicts an instance, x, is a member of the training set if and only if x was classified correctly.

For the bayes-wb attack (introduced in Section 3.3), we trained 10 proxy models on random samples from the hold-out group, and took the mean of their approximated weights at each point for added robustness. When attacking MLP models, we performed the attack on the final layer of the MLP using Algorithm 1. When attacking LeNet models, we used a meta attack model (described in Section 4.2) that was trained on data from 10 shadow models trained on 10 samples from the hold-out group. We used a MLP with 16 internal neurons for the meta model and trained it for 32 epochs with Adam [20].

For the general-wb attack (introduced in Section 3.4), we construct an attack model that learns a displacement function, D_ℓ (Algorithm 2), for each layer, ℓ, of the network, and combines the results with a meta attack model, M. The attack model was trained for 32 epochs with Adam, using data from 10 shadow models trained on the hold-out group. As suggested in Section 3.4, we modeled each D_ℓ as a convolutional neural network. In each experiment, the networks modeling M and each D_ℓ had at most one hidden layer, with n_M and n_D hidden units, respectively (in our experiments each D_ℓ used the same architecture, though this need not be the case in general). In order to determine n_M and n_D for each dataset, we created a validation set using 10 shadow models trained on different random splits of the hold-out group, and performed a parameter sweep over n_M,n_D. We then took the n_M and n_D yielding the highest validation accuracy for each target model. We find that because the attack model is highly regularized via its restrictive architecture, the validation accuracy is a reasonably good indicator of the test accuracy, making it a useful tool for hyper-parameter tuning (see Figure 6).

The shadow-bb attack refers to the black-box shadow model attack [38], explained briefly in Section 7. In each experiment, the shadow-bb attack was trained using 10 shadow models trained on 10 samples from the hold-out group.

	omniscient	bayes-wb	general-wb *(min capacity)*	general-wb *(extra capacity)*
$n=100$	0.618	0.605	0.602	0.590
$n=200$	0.577	0.570	0.563	0.562
$n=400$	0.568	0.550	0.547	0.542

Figure 5: Comparison of the bayes-wb and general-wb attacks to an *omniscient* attack, which has knowledge of $\hat{\mu}$, μ^*, and σ, and thus can use Theorem 1 directly without the use of a proxy model. In one case, the general-wb attack was given the minimum capacity to reproduce the bayes-wb attack, i.e., d is simply a weighted sum of $\hat{W}_i$ and $\tilde{W}_i$. In another case, the general-wb attack was given excess capacity, with 16 hidden units in d. Three target models, trained on synthetic Gaussian naive-Bayes data with training set sizes of 100, 200, and 400, were attacked.

5.2 Sensitivity to Assumptions & Hyper-parameters

In Section 3.2, we derive the Bayes-optimal membership inference attack on Gaussian data satisfying the naive-Bayes condition. The weights of the optimal membership predictor for this case, given by Theorem 1, are a function of the empirical training distribution parameters and true distribution of the data, which, of course, would be unknown to an attacker. Section 3.3 describes how to address this, using a *proxy model* to capture the difference between the data used to train the target model and the general population.

Figure 5 demonstrates the effectiveness of the proxy model in our attack, by comparing our bayes-wb attack using a proxy model to an "omniscient" attack, which uses Equation 6 directly, with knowledge of the train and general distribution. We can consider the omniscient attack as giving an upper bound on the expected accuracy of a white-box attack on Gaussian naive-Bayes data, as it is the true Bayes-optimal attack (while bayes-wb is the approximate Bayes-optimal attack according to Proposition 1). Our attack achieves on average 84% of the advantage of the omniscient attack, suggesting that the proxy model was able to approximately capture the general distribution as necessary for the purpose of detecting the target model's idiosyncratic use of features.

In Section 3.4, we further generalize the bayes-wb attack to use a learned displacement function that may be more appropriate for distributions that don't resemble the Gaussian naive-Bayes assumption. While we find that this general-wb attack often generalizes to arbitrary distributions better than the bayes-wb attack, because its displacement function is

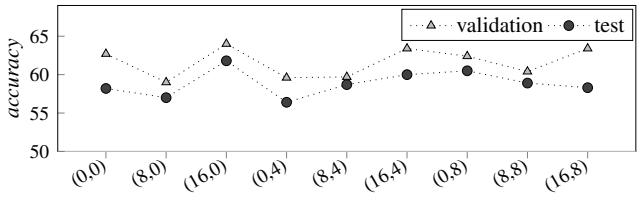


Figure 6: Plot showing the validation (known to the attacker) and test (unknown to the attacker) accuracies of the general-wb attack for various attack model architectures on the Hepatitis dataset. Each architecture, listed on the x-axis, is represented by a pair, (n_D, n_M), where n_D and n_M are the number of hidden units in the *distance function network* and *meta model network* respectively (see Section 5.1).

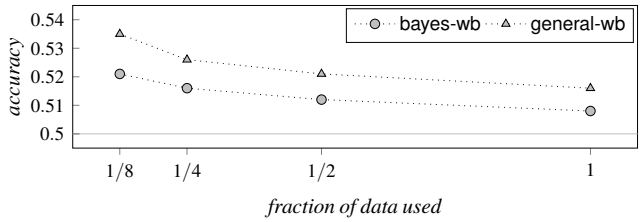


Figure 7: Accuracy of the bayes-wb and general-wb attacks on the Adult dataset, as the amount of data is scaled from 6,105 records (1/8 of the full dataset) to 48,841 records.

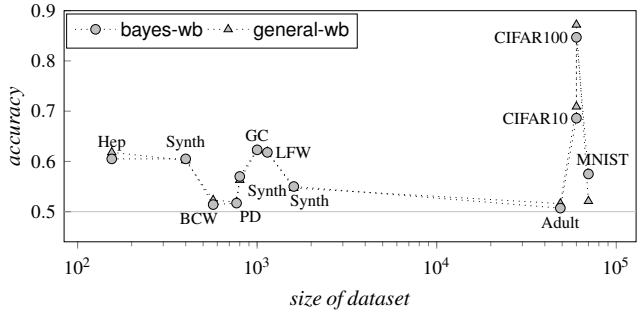


Figure 8: Accuracy of the bayes-wb and general-wb attacks on each of the datasets in our evaluation, plotted against the size of the respective dataset.

learned, it is possible for the general-wb attack to overfit.

Figure 5 also shows the accuracy of the general-wb attack on Gaussian naive-Bayes data. When the neural network representing the displacement function is given exactly enough capacity to reproduce the bayes-wb attack, general-wb recovers on average 94% of the advantage of the bayes-wb attack. Upon inspecting the weights of the displacement network, we find that general-wb learns almost exactly element-wise subtraction, demonstrating its potential to learn the optimal displacement function. When given excess capacity, the general-wb attack performs only marginally worse, achieving on average 92% of the minimal general-wb attack's advantage (86% of bayes-wb), suggesting that general-wb is not highly prone to overfitting.

Tuning the general-wb Attack. As mentioned, even an over-parameterized displacement function may be able to perform nearly optimally on models trained on simple datasets, like the Synthetic dataset. However, as the general-wb attack involves several hyper-parameters, it may be useful to tune these parameters in a reliable way. We note that an arbitrary number of shadow models can be produced by sampling from the hold-out data, allowing us to construct a validation set on which to evaluate various architectures for implementing the distance function, D_ℓ, and meta model, M, comprising the general-wb attack. Figure 6 shows an example of the validation accuracy obtained using various architectures for D_ℓ and M, along with the corresponding test accuracy (unknown to the attacker). We see that the test accuracy fairly closely follows the validation accuracy, with the maximum for both metrics occurring for the same architecture. This suggests that the validation accuracy is a reasonably good indicator of the test accuracy making it a useful tool for hyper-parameter tuning. This is perhaps not too surprising, as the attack model is highly regularized via its restrictive architecture.

5.3 Data Scaling

The "omniscient" attack developed in Section 3.2 relies on measuring a difference between the parameters of the *true* data-generating distribution, $\mathcal{D}^*$ and the *empirical* distribution, $\hat{\mathcal{D}}$. Because $\hat{\mathcal{D}}$ is derived from a sample drawn from $\mathcal{D}^*$, in expectation $\hat{\mathcal{D}} = \mathcal{D}^*$; that is, as the number of samples in the training set goes to infinity, the true and empirical distributions will converge, rendering even the optimal attack ineffective (0 advantage). We would therefore expect that for a sufficiently large training set, the success of any MI attack would decline. Conversely, we may expect the opportunity for better MI performance for smaller training sets. Indeed, in accordance with this observation, we see that even the omniscient attack sees accuracy inversely proportional to the dataset size (Figure 5).

We find that this pattern persists for real-world datasets as well. Figure 7 shows the accuracy of our attacks on models trained on subsets of various sizes of the Adult dataset (the dataset containing the most records as compared to the number of parameters in the respective model). We observe that as more data becomes available for training, the advantage of the attack diminishes, becoming quite small ($< 4\%$) on the entire dataset (48,841 records). This may suggest that the Adult dataset is sufficiently large to preclude any significant information leakage via a modestly-sized MLP model obtained through standard training.

Figure 8 shows the accuracy of our attacks on each of the datasets used in our evaluation, plotted against the size of the respective dataset. We see to some extent the same downwards trend as dataset size increases, though there is more noise, and some of the image datasets (especially CIFAR10 and CIFAR100) provide notable exceptions. This is likely due to the variation in the number of features, the network capacity, and the generalization error across datasets.

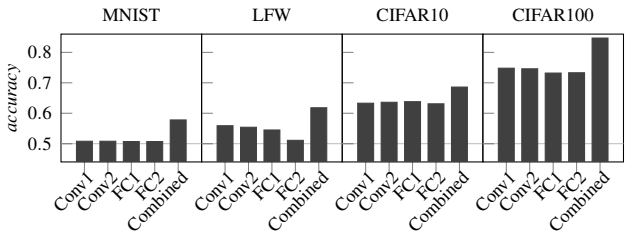


Figure 9: Accuracy of the bayes-wb attack on each individual layer of LeNet, compared with the accuracy using the combined meta-model.

5.4 Combining Layers

For deep models in particular, we want to be able to use information from each layer in our attack. In Section 4.2, we describe a meta attack that combines the outputs of an individual attack on each layer. Figure 9 shows the accuracy of the bayes-wb attack on each individual layer and of the meta attack on each LeNet target model.

In every instance, the meta attack is able to substantially outperform any individual attack, indicating that the information it receives from each layer is not entirely redundant. Moreover, this suggests that *information leakage occurs in the representations learned by layers throughout the model*—that is, each layer plays some role in the leakage of information about the training data. A possible consequence of this that we hypothesize in Section 6 is that models trained with transfer learning may leak less information about the training data used to tune the model.

Remarkably, for MNIST, the advantage of the meta attack is greater than that of all the individual layers combined.

5.5 Comparison to Prior Work

Finally, we compare our approach to previous work, namely, shadow-bb [38]. In particular, we compare *(1)* performance in terms of accuracy, precision, and recall; and *(2)* the reliability of the attack confidence when used to calibrate for higher precision. In short, our results show that both bayes-wb and general-wb outperform shadow-bb, and can be more reliably calibrated to achieve confident inferences for the attacker. Furthermore, even on some well-generalized models, on which shadow-bb and naive fare poorly, our attacks can be calibrated to make confident inferences, and sometimes also achieve non-trivial advantage. Finally, we find that there is often little advantage to shadow-bb over naive, both because shadow-bb often performs comparably to naive, and because shadow-bb does not always produce calibrated confidence scores.

Performance. Figure 10 shows the accuracy, precision, and recall of naive, bayes-wb, general-wb, and shadow-bb. The precision shown is before calibration attack (calibration results are shown in Figure 11). We see that both bayes-wb and general-wb are consistently more accurate and precise than naive and shadow-bb. At least one of bayes-wb or general-wb obtains the highest accuracy of the four methods on each target except Adult, and *both* outperform the other two methods in terms of precision in all cases. In some cases, the improvement in accuracy of at least one of our attacks over prior work is by as much as seven percentage points, though in others our accuracy is only modestly better; however, in terms of precision, the difference is more pronounced in almost every case (typically greater by at least five percentage points).

Typically naive or shadow-bb achieve the highest recall, but we note that both methods do so with lower precision; and at least in the case of naive, this is merely a consequence of the fact that most of the models have a high training accuracy.

Our results for the performance of shadow-bb are roughly in line with previously reported results for shadow-bb on the datasets which have been used for evaluation in prior work (Adult, MNIST, LFW, CIFAR10, and CIFAR100) [35, 38]. On CIFAR10 and CIFAR100, our results are slightly lower than the results reported for shadow-bb by Shokri et al., however, our target models trained on CIFAR10 and CIFAR100 use dropout and have a lower generalization error than the models in the attacks reported by Shokri et al., which most likely accounts for this small discrepancy.

Calibration. As argued in Sections 1 and 2, one of the key desiderata of a membership inference attack is precision. In order to calibrate an attack for precision, the confidence outputted by the attack must be informative. Here, we examine the calibration of the confidence outputs of our attacks compared to shadow-bb (naive does not provide a confidence score with which to calibrate).

We find that increasing the decision threshold of the bayes-wb and general-wb attacks has a positive effect on precision. In particular, using the heuristic defined in Algorithm 3, we are able to consistently improve the precision of our attacks. Figure 11 shows the precision of our attack as the decision threshold is raised according to Algorithm 3, for $\alpha = 0.90$, and $\alpha = 0.99$, compared to the uncalibrated attack. In each case the precision increases, often by 10 or more percentage points. Though in practice, an attacker would not be easily able to tune the calibration hyper-parameter, α, the consistency of the results in Figure 11 suggest that values of 0.90 and 0.99 serve as a practical "rule-of-thumb" for reliable calibration.

On all convolutional models, general-wb is able to be calibrated to upwards of 75% precision. Notably, this includes the model trained on MNIST, which has only 1.1% generalization error. This implies that *privacy violations are a threat even to well-generalized models*, since our attack is able to confidently (with at least 75% confidence) identify a subset of training set members.

On the MLP models, the calibration is slightly less consistent; however, here bayes-wb is able to obtain over 70% precision on the models trained on the Breast Cancer Wisconsin and Hepatitis datasets.

In Figure 10, we see that the recall of the uncalibrated attack is frequently over 90%. When calibrating, the recall drops as precision increases, however, we believe this does not

	accuracy				*precision*				*recall*			
model	naive	shadow-bb	bayes-wb	general-wb	naive	shadow-bb	bayes-wb	general-wb	naive	shadow-bb	bayes-wb	general-wb
BCW	0.522	0.500	0.514	**0.523**	0.511	0.500	**0.545**	0.528	0.987	**1.000**	0.962	0.505
PD	0.517	0.508	0.517	**0.519**	0.511	0.515	0.537	**0.561**	**0.789**	0.592	0.641	0.641
Hep	0.595	0.553	0.605	**0.618**	0.552	0.528	0.562	**0.609**	0.997	**1.000**	0.977	0.639
GC	0.618	0.582	**0.623**	0.622	0.572	0.547	0.603	**0.637**	0.937	0.788	**0.982**	0.623
Adult	0.506	**0.524**	0.507	0.516	0.504	0.514	0.512	**0.516**	0.861	**1.000**	0.525	0.566
MNIST	0.506	0.506	**0.575**	0.521	0.503	0.506	0.578	**0.640**	**0.998**	0.925	0.627	0.421
LFW	0.582	0.597	0.618	**0.619**	0.545	0.557	0.581	**0.586**	0.993	**1.000**	0.925	0.919
CIFAR10	0.666	0.684	0.686	**0.709**	0.600	0.605	0.638	**0.646**	**0.996**	0.909	0.853	0.881
CIFAR100	0.831	0.847	0.847	**0.872**	0.757	0.766	0.770	**0.792**	0.977	**0.999**	0.962	0.976

Figure 10: Comparison of the accuracy, precision, and recall of bayes-wb and general-wb with naive and shadow-bb.

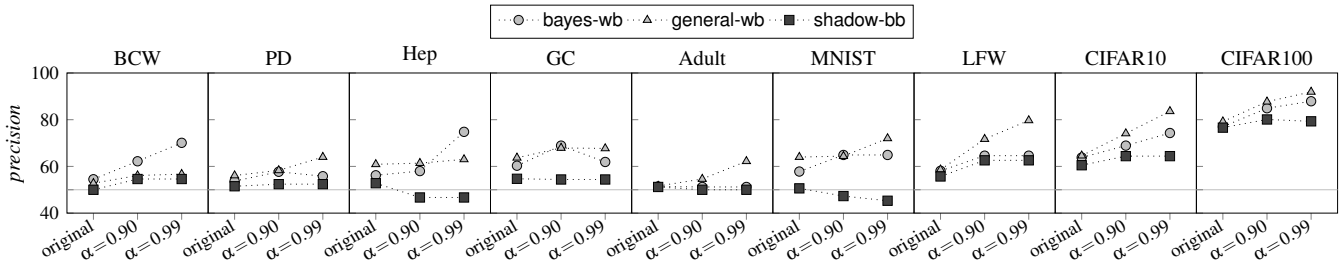


Figure 11: Precision of the bayes-wb, general-wb, and shadow-bb attacks, calibrated using the heuristic outlined described in Algorithm 3 (with $\alpha = 0.90$ and $\alpha = 0.99$), compared to the precision with no calibration (default threshold).

diminish the threat of the attacks because a privacy violation occurs if even a few points are confidently inferred.

While Figure 11 demonstrates that applying our calibration heuristic to bayes-wb and general-wb consistently increases the precision, we see that this is not always the case for shadow-bb. In some cases, the precision of shadow-bb is *decreased* by increasing the decision threshold. In fact, occasionally, the average confidence on non-members is higher than that of members, leading to a precision slightly less than 50%. This may be a result of the shadow model overfitting to the hold-out data. When we are able to increase the precision of shadow-bb using its confidence output, the gains are less impressive, suggesting the probability outputs of shadow-bb are less well-calibrated.

Performance on Well-Generalized Models. While some of the models we used to evaluate our attacks had a generalization error of 10% or more, we also evaluated on several datasets for which the learned model was far less overfit, including MNIST (1.1% generalization error), Adult (1.2%), Pima Diabetes (3.4%), and Breast Cancer Wisconsin (4.3%). While on PD and BCW, our attacks only slightly outperform naive, on MNIST and Adult, our attacks do substantially better: on the model trained on Adult, general-wb achieves an advantage 2.6 times greater than the advantage achieved by naive. Even more impressively, on MNIST, general-wb and bayes-wb achieves an advantage 3.5 and 12.5 times greater than the advantage achieved by naive, respectively. On the other hand, shadow-bb fares poorly on all of these datasets except for Adult, typically achieving less than 2% advantage. Finally, we note that the bayes-wb attack on the synthetic data model (Section 5.2) achieves a non-trivial 60% accuracy (20% advantage), despite the fact that the model has *zero* generalization error.

In addition to the cases where our attacks achieve relatively high advantage against well-generalized models, we find that when calibrated, our attacks achieve as high as 75% precision on MNIST, and 70% precision on Breast Cancer Wisconsin, again underscoring the threat of privacy violations for well-generalized models.

While it is clear that a greater degree of overfitting makes it easier for an adversary to mount *any* attack, the relative success of our attacks over naive on well-generalized models suggests that the white-box information is useful even when the model does not leak information through incorrect predictions on the test set.

Similarity of shadow-bb and naive Results. Figure 10 reveals that often, shadow-bb has performance comparable or even worse than naive, particularly on well-generalized target models. This is likely a product of the attack model overfitting to idiosyncrasies in the shadow model's output that are unrelated to the target model. On deep models with significant overfitting, shadow-bb performs slightly better than naive, however, we found that its behavior was not significantly different from that of naive; for example, on LFW, naive recovers 88% of the exact correct predictions made by shadow-bb. This supports the intuition that the features used by the shadow model approach (i.e., the softmax outputs) are not fundamentally more well-suited to membership inference than those used by the naive method (i.e., the correctness of the predictions). This is perhaps unsurprising, as the softmax

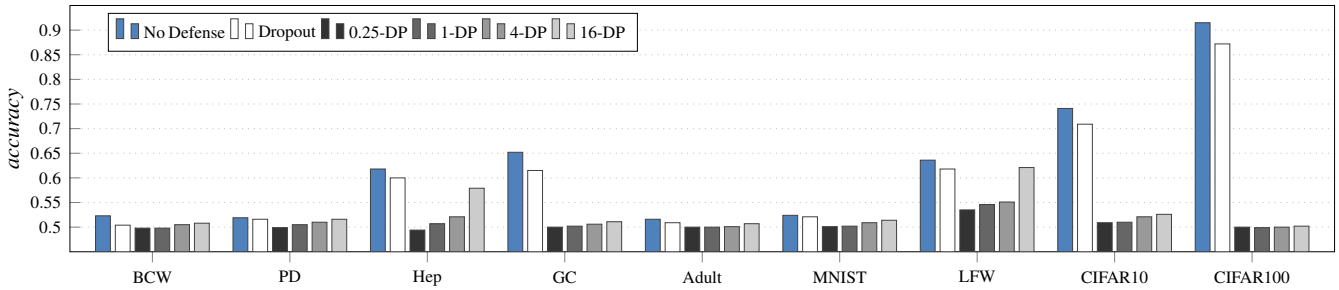


Figure 12: Attack accuracies against models trained with either dropout or (ε,δ)-differential privacy for various values of ε.

dataset		*no defense*	*dropout*	ε = 0.25	ε = 1	ε = 4	ε = 16
BCW	*train*	0.987	0.982	0.601	0.654	0.767	0.778
	test	0.944	0.961	0.609	0.675	0.763	0.808
PD	*train*	0.789	0.784	0.680	0.678	0.681	0.683
	test	0.756	0.783	0.673	0.651	0.649	0.654
Hep	*train*	0.997	0.992	0.534	0.695	0.700	0.729
	test	0.810	0.849	0.555	0.786	0.803	0.817
GC	*train*	0.937	0.932	0.625	0.656	0.680	0.707
	test	0.701	0.730	0.610	0.661	0.687	0.698
Adult	*train*	0.861	0.860	0.501	0.500	0.500	0.501
	test	0.849	0.859	0.500	0.501	0.500	0.499
MNIST	*train*	1.000	0.998	0.107	0.129	0.243	0.330
	test	0.973	0.987	0.106	0.132	0.251	0.331
LFW	*train*	1.000	0.999	0.109	0.137	0.214	0.428
	test	0.842	0.835	0.116	0.119	0.200	0.463
CIFAR10	*train*	0.999	0.996	0.100	0.098	0.103	0.100
	test	0.621	0.664	0.101	0.100	0.105	0.093
CIFAR100	*train*	0.999	0.977	0.010	0.010	0.010	0.011
	test	0.257	0.312	0.010	0.010	0.011	0.011

Figure 13: Train and test accuracies for models trained with either dropout or (ε,δ)-differential privacy for various values of ε.

outputs are likely to coincide largely with the correctness of the prediction—for correct predictions, the softmax will likely have high confidence on the correct class, regardless of whether the point was a member or not; and similarly for incorrect predictions, the softmax will likely have more entropy.

6 Defenses

Concerns about privacy, underscored by concrete threats such as the attacks developed in this paper, have also motivated research to provide adequate defenses against such threats. In this section we explore the ability of some of the commonly-proposed mitigation techniques to defend against our attack. In particular, we focus on *differential privacy* [8] and regularization. We find that, while both are useful to a degree, neither dropout nor ε-differentially private training with a large ε, are necessarily sufficient for mitigating the privacy risk posed by our attack.

Differential Privacy. Differential privacy (DP) [8] is often seen as the gold standard for private models, as models trained with differential privacy have provable guarantees against membership inference. Namely, Yeom et al. [47] showed that, given an ε-differentially private learning algorithm, an adversary can achieve an advantage of at most $e^{\varepsilon} - 1$. Differential privacy has been applied to many areas of machine learning, including logistic regression [5], SVMs [34], and more recently, deep learning [1, 37]. However, current methods for ensuring differential privacy are typically costly with respect to the accuracy of the model, particularly for small values of ε, which give a better privacy guarantee. For this reason, in practice, ε is often chosen to be quite large; for example, in 2017, Apple was found to use an effective epsilon as high as 16 in some of its routines [43].

We used the Tensorflow Privacy library [29], an implementation of the *moments accountant* method [1], which guarantees (ε, δ)-differential privacy, to study the practical efficacy of our attack on protected models. This method utilizes several hyperparameters from which ε is derived; for uniformity, we modified only the *noise multiplier* to achieve the desired ε, and used heuristics described in the original paper [1] to select the remaining hyperparameters. While a different tuning of the hyperparameters may result in a different privacy-utility trade-off, the privacy guarantee depends only on ε and δ, *not the hyperparameters directly*. In each case, δ was selected to be smaller than $1/N$ where N is the size of the dataset.

Figure 12 shows the effectiveness the general-wb attack against models trained with differential privacy for various values of ε on each dataset. The train and test accuracies of the corresponding differentially-private target models are shown in Figure 13. First, we note that as expected, when ε decreases the adversary's effectiveness quickly declines. However, when ε is large (ε = 16), our attack occasionally performs *essentially the same* on the differentially-private model as on the undefended model. For example, on BCW, PD, and LFW, 16-DP provided less defense than simple regularization, while harming the accuracy of the model. Similarly, on Hep, 16-DP reduced the effectiveness of general-wb, but not below the effectiveness of shadow-bb on the corresponding undefended model. These findings suggest that the practical benefits of large-ε-differential privacy cannot be taken for granted; in general, differential privacy may only be effective for sufficiently small ε.

Nevertheless, it is clear that a practical adversary is unlikely to achieve performance that is tight with the theoretical bound. For both the undefended model and the models trained with DP for $\varepsilon > \ln 2 \approx 0.69$, the theoretical bound on the adversary's

accuracy is 100%, which no attack was able to achieve. On the other hand, for $\varepsilon = 0.25$, the theoretical maximum accuracy of the adversary is 64.2%. In most such cases, our attack fared far poorer than this, coming closest on LFW, where our attack achieved 53.5% accuracy (25% of the theoretical maximum advantage) on the 0.25-DP model. Thus, we conclude that because the accuracy of a real adversary is not likely to be tight with the worst-case guarantee, it is indeed pragmatic to select a somewhat large ε. However, our evaluation shows that ε should not be chosen to be too large, or else the operative benefits of differential privacy may be lost. Furthermore, the success of a given value of ε appears to vary across different datasets and models. One must therefore be careful when making a practical selection for ε; to this end, we suggest that our attack may be useful in assessing which values of ε are appropriate for a given application.

An apparent drawback of the examined method for obtaining differential privacy, revealed in our evaluation, is the steep cost in performance (Figure 13), which is particularly high for small ε. Despite the fact that our attack became far less effective for small ε, this cost limits the practicality of the defense, highlighting the need for more research in this area. The results we find here align with recent work [19], in which Jayaraman and Evans showed that the privacy leakage tends to increase as ε becomes large enough to avoid a significant loss in accuracy. Indeed, only on the German Credit dataset did 16-DP provide a good defense while nearly maintaining the accuracy of the unprotected model. In the other cases we evaluated, either our attack performed comparably on the DP and unprotected models, or the accuracy of the private model was significantly lower than that of the unprotected model.

Abadi et al. [1] mitigate the high cost in accuracy by first pre-training on public data, and then fine-tuning only the top layers with differential privacy on the private training set. While this public transfer learning approach may not always be possible, it has two key benefits, the first being that the resulting model's performance is far less poor. Second, only the final layers of such a model are trained on the private data, and thus our attack may only be able to effectively target those layers. Our experiments in Section 5.4 show that our attack is far more effective when all layers are leveraged, and that the earlier layers often account for a sizable portion of the information leakage. This suggests that, when possible, a transfer learning scheme like that of Abadi et al. could be a practical defense.

Regularization. Given the connection between membership inference and overfitting, regularization, such as dropout [41], which aims to reduce overfitting, has also been proposed to combat membership inference. Generalization alone is not sufficient to protect against membership inference [47], and in fact, our empirical results (Section 5) show that we can successfully attack even models with negligible generalization error; however, dropout has been shown not only to reduce overfitting, but to strengthen privacy guarantees in neural networks [18]. Figure 12 shows the accuracy of our attack with and without dropout. We find that dropout does not significantly impact the accuracy of our attack in most cases. However, as opposed to DP, dropout is typically *beneficial* to the performance of the model, while providing a modest defense. In this light, regularization (including dropout) may in fact be the more practical defensive measure, insofar as it improves test accuracy, because better generalization does appear to make membership more difficult, though clearly not impossible, for an attacker.

Still, we warn that this may not be universally true of all forms of regularization, even regularization that improves generalization—as we have demonstrated, a model can still leak membership information through its parameters while making correct predictions on unseen points.

Defenses in the Black-box Setting. For membership inference in the black-box setting, Shokri et al. [38] also propose a number of other possible defenses, such as restricting the prediction vector to the top k classes, or increasing the entropy of the prediction vector via increasing the normalization temperature of the softmax. However, these defenses are easily circumvented in the white-box setting, as the pre-modified outputs are still available to an attacker in this threat model.

Similarly, Salem et al. [35] propose a defense called *model stacking*, in which two models are trained separately on the training data and a third model makes predictions based on the outputs of the first two. While Salem et al. found this to be an effective defense against black-box approaches, this defense is likewise circumvented in the white-box setting, as the initial two models are available to the attacker.

7 Related Work

There is extensive prior literature on privacy attacks on statistical summaries. Homer et al. [17] proposed what is considered the first membership inference attack on genomic data in 2008. Following the work by Homer et al., a number of studies [9, 14, 36, 39, 44] have looked into membership attacks on statistics commonly published in genome-wide association studies. In a similar vein, Komarova et al. [22] looked into partial disclosure scenarios, where an adversary is given fixed statistical estimates from combined public and private sources and attempts to infer the sensitive feature of an individual referenced in those sources.

More recently, membership inference attacks have been applied to machine learning models. Ateniese et al. [2] demonstrated that given access to the parameters of support vector machines (SVMs) or Hidden Markov Models (HMMs), an adversary can extract information about the training data.

As deep learning has become more ubiquitous, membership inference attacks have been particularly directed at deep neural networks. A number of different recent works [27, 28, 33, 35, 38, 47] have taken different approaches to membership inference against deep networks in a standard supervised learning setting. Additionally, Hayes et al. [15] have studied membership inference against generative adversarial networks

(GANs); and others [16, 30, 33] have studied membership inference in the context of collaborative, or federated, learning.

Black-box attacks. We study membership inference as it applies to deep networks in classic supervised learning problems. Most of the prior work in this area [27, 28, 35, 38, 47] has used the *black-box* threat model. Yeom et al. [47] showed that generalization error necessarily leads to membership vulnerability; a natural consequence of this is that a simple "naive" attack (naive), which predicts a point is a member if and only if it was classified correctly, can be found to be quite effective on models that overfit to a large degree. Other approaches have leveraged not only the predictions of the model, but the confidence outputs. A particularly canonical approach, along these lines, is the attack introduced by Shokri et al. [38] (shadow-bb). In this approach, a *shadow model* is trained on half of $\tilde{S}$, $\tilde{S}_{in}$, and an *attack model* is trained on the the outputs of the shadow model on its training data, $\tilde{S}_{in}$ (labeled 1), and the remaining data $\tilde{S} \setminus \tilde{S}_{in}$ (labeled 0). Shadow models leverage the disparity in prediction confidences on training instances the target model has overfit to, and have been shown to be successful at membership inference on models that have sufficiently high generalization error. A few other membership inference approaches [15, 35] have made use of this same technique.

Despite the fact that shadow model attacks leverage more information than the naive attack, we find in our evaluation (Section 5) that often, the shadow model attack fails to outperform the naive attack. One potential reason for this finding is that the learned attack model used by this approach to distinguish between the shadow model's outputs on members and non-members may be itself subject to overfitting. This may be especially true if the attack model picks up on behavior particular to one of the shadow models rather than the true target model. Furthermore, the confidence and entropy of the target model's softmax output is likely to be closely related to whether the target model's prediction was correct or not, meaning that the softmax outputs may not provide substantially different information from that used by naive.

White-box attacks. In some settings, it may be realistic for an attacker to have white-box access to the target model. Intuitively, while some information is leaked via the behavior of a model, the details of the structure and the parameters of the model are clear culprits for information leakage. Few prior approaches have successfully leveraged this extra information. While Hayes et al. [15] describe a white-box attack in their work on membership inference attacks applied to GANs, the attack uses access only the outputs of the discriminator portion of the GAN, rather than the learned weights of either the discriminator or the generator; thus their approach is not white-box in the same sense. Meanwhile, Nasr et al. [33] demonstrated that a simple extension of the black-box shadow model approach to utilize internal activations does not result in higher membership inference accuracies than the original black-box approach. This is perhaps unsurprising, as the internal units of the shadow models are not likely to have any relation to those of the target model (see Section 4).

Recently, Nasr et al. [33] provided a white-box attack that leverages the gradients of the target model's loss function with respect to its weights, which SGD approximately brings to zero on the training points at convergence. In contrast to our work, Nasr et al. use a further relaxed threat model, in which the attacker has access to as much as *half of the target model's training data*. We suggest an approach that is quite different from that of Nasr et al.. Our approach does not require this extra knowledge for the attacker, and thus falls under a more restrictive threat model, in which, to our knowledge, no other effective white-box attacks have been proposed.

8 Conclusions and Future Work

Our work is the first to fully leverage white-box information to improve membership inference attacks against deep networks (in the standard threat model where the adversary is assumed not to have any examples of true training points). In particular, *our analysis sheds light on a fundamental mechanism of overfitting that can be leveraged by an adversary to compromise a model's privacy in a concrete way*. We use this analysis of how feature usage can lead to information leakage to construct a new white-box attack, which our evaluation demonstrates improves upon the previous state-of-the-art, particularly because it can be reliably calibrated for high precision, even on some well-generalized models.

Subsequently, we used our attack to evaluate commonly-proposed privacy defenses. Perhaps most interestingly, experiments utilizing our attack reveal a nuanced story regarding differential privacy. When setting ε to small values, the attack was successfully mitigated but the utility of the resulting model quickly diminished; while when ε was increased sufficiently to mitigate the loss in utility, the attack sometimes achieved close to the same accuracy as on the undefended model. This suggests that there is still considerable work to be done in developing effective defenses against privacy attacks—we anticipate that the insights gained from our approach will contribute to designing such defenses.

Acknowledgment. This material is based on work supported by the National Science Foundation under Grants No. CNS-1704845 and CNS-1801391.

References

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS, 2016.

[2] 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*, 2015.

[3] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR*, abs/1206.5538, 2012.

[4] Justin Brickell and Vitaly Shmatikov. The cost of privacy: destruction of data-mining utility in anonymized data publishing. In *KDD*, 2008.

[5] Kamalika Chaudhuri and Claire Monteleoni. Privacy-preserving logistic regression. In *Advances in Neural Information Processing Systems 21*. 2009.

[6] Francois Chollet. Keras: Deep learning library for Theano and TensorFlow. https://keras.io, 2017.

[7] Graham Cormode. Personal privacy vs population privacy: Learning to attack anonymization. In *KDD*, 2011.

[8] Cynthia Dwork. Differential privacy. In *ICALP*. Springer, 2006.

[9] Khaled El Emam, Elizabeth Jonker, Luk Arbuckle, and Bradley Malin. A systematic review of re-identification attacks on health data. *PLOS ONE*, 2011. doi: 10.1371/journal.pone.0028071.

[10] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *ACM Conference on Computer and Communications Security (CCS)*, 2015.

[11] Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *USENIX Security Symposium*, 2014.

[12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[13] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. 2017.

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

[15] 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.

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

[17] 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.

[18] Prateek Jain, Vivek Kulkarni, Abhradeep Thakurta, and Oliver Williams. To drop or not to drop: Robustness, consistency and differential privacy properties of dropout. *CoRR*, abs/1503.02031, 2015.

[19] Bargav Jayaraman and David Evans. Evaluating differentially private machine learning in practice. In *28th USENIX Security Symposium (USENIX Security 19)*, 2019.

[20] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[21] Jing Yu Koh. Model Zoo. URL http://modelzoo.co.

[22] Tatiana Komarova, Denis Nekipelov, and Evgeny Yakovlev. Estimation of treatment effects from combined data: Identification versus data security. In *Economic Analysis of the Digital Economy*, pages 279–308. University of Chicago Press, 2015.

[23] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[24] Yann LeCun, Corrina Cortes, and Christopher Burges. The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/, 1998.

[25] Klas Leino, Shayak Sen, Anupam Datta, Matt Fredrikson, and Linyi Li. Influence-directed explanations for deep convolutional networks. *CoRR*, abs/1802.03788, 2018.

[26] Ninghui Li, Wahbeh Qardaji, Dong Su, Yi Wu, and Weining Yang. Membership privacy: A unifying framework for privacy definitions. In *Proceedings of ACM CCS*, 2013.

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

[28] 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.

[29] H. Brendan McMahan and Galen Andrew. A general approach to adding differential privacy to iterative training procedures. *CoRR*, abs/1812.06210, 2018.

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

[31] Kevin P. Murphy. Gaussian classifiers. University Lecture, 2007. URL https://www.cs.ubc.ca/~murphyk/Teaching/CS340-Fall07/gaussClassif.pdf.

[32] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[33] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Stand-alone and federated learning under passive and active white-box inference attacks. *CoRR*, abs/1812.00910, 2018.

[34] Benjamin I. P. Rubinstein, Peter L. Bartlett, Ling Huang, and Nina Taft. Learning in a large function space: Privacy-preserving mechanisms for SVM learning. *CoRR*, abs/0911.5708, 2009.

[35] 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.

[36] Sriram Sankararaman, Guillaume Obozinski, Michael I Jordan, and Eran Halperin. Genomic privacy and limits of individual detection in a pool. *Nature Genetics*, 41(9):965–967, 2009.

[37] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS, 2015.

[38] Reza Shokri, Marco Stronati, and Vitaly Shmatikov. Membership inference attacks against machine learning models. *CoRR*, abs/1610.05820, 2016.

[39] 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.

[40] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2013.

[41] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 2014.

[42] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. *CoRR*, abs/1703.01365, 2017.

[43] Jun Tang, Aleksandra Korolova, Xiaolong Bai, Xueqiang Wang, and Xiaofeng Wang. Privacy loss in apple's implementation of differential privacy on macos 10.12. 09 2017.

[44] 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.

[45] X. Wu, M. Fredrikson, W. Wu, S. Jha, and J. F. Naughton. Revisiting Differentially Private Regression: Lessons From Learning Theory and their Consequences. *CoRR*, abs/1512.06388, 2015.

[46] X. Wu, M. Fredrikson, S. Jha, and J. F. Naughton. A methodology for formalizing model-inversion attacks. In *2016 IEEE Computer Security Foundations Symposium (CSF)*, 2016.

[47] Samuel Yeom, Matt Fredrikson, and Somesh Jha. The unintended consequences of overfitting: Training data inference attacks. *CoRR*, abs/1709.01604, 2017.

[48] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *CoRR*, abs/1411.1792, 2014.

A Proof of Theorem 1

We begin with the expression for $m^y(x)$ and apply Bayes' rule to obtain Equation 10.

$$m^y(x) = \Pr[T \mid X=x, Y=y] = \frac{\Pr[X=x \mid T, Y=y]\Pr[T]}{\Pr[X=x \mid Y=y]} \quad (10)$$

Next, we express Equation 10 as a logistic (or, sigmoid) function, $\sigma(x) := (1+e^x)^{-1}$. We assume that $\Pr[T] = \frac{1}{2}$, and thus $\Pr[X=x \mid Y=y]$ can be written as $\frac{1}{2}(\Pr[X=x \mid T, Y=y] + \Pr[X=x \mid \neg T, Y=y])$, by the law of total probability. We then divide by the numerator in Equation 10, yielding an expression that can be written as a logistic function (11) by noting that for $x>0$, $\exp(\log x) = x$.

$$\begin{aligned}(10) &= \frac{\Pr[X=x \mid T, Y=y]}{(\Pr[X=x \mid T, Y=y] + \Pr[X=x \mid \neg T, Y=y])} \\ &= \left(1 + \frac{\Pr[X=x \mid \neg T, Y=y]}{\Pr[X=x \mid T, Y=y]}\right)^{-1} \\ &= \left(1 + \exp\left(\log \frac{\Pr[X=x \mid \neg T, Y=y]}{\Pr[X=x \mid T, Y=y]}\right)\right)^{-1} \\ &= \sigma\left(\log \frac{\Pr[X=x \mid T, Y=y]}{\Pr[X=x \mid \neg T, Y=y]}\right) \end{aligned} \quad (11)$$

We notice that $\Pr[X=x \mid T, Y=y]$ is the probability of having drawn x from $\hat{\mathcal{D}}$, given class, y, and similarly, $\Pr[X=x \mid \neg T, Y=y]$ is the probability of having drawn x from $\mathcal{D}^*$, given class, y. Using the Naive-Bayes assumption, i.e., that conditioned on the class, y, the individual features, x_j, are independent, we obtain Equation 12.

$$(11) = \sigma\left(\log \prod_j \frac{\mathcal{N}(x_j \mid \hat{\mu}_{yj}, \hat{\sigma}_j^2)}{\mathcal{N}(x_j \mid \mu^*_{yj}, \sigma^{*2}_j)}\right) \quad (12)$$

We then re-write the log of the product as a sum over the log, and observe that the sum can be written as a dot product as in Equation 13, which gives the parameters of the Bayes-optimal model for $m^y(x)$.

$$\begin{aligned}(12) &= \sigma\left(\sum_j \frac{(x_j - \mu^*_{yj})^2}{2\sigma^{*2}_j} - \frac{(x_j - \hat{\mu}_{yj})^2}{2\hat{\sigma}_j^2} + \log\left(\frac{\sigma^*_j}{\hat{\sigma}_j}\right)\right) \\ &= \sigma\left(v^{y\,T} x^2 + w^{y\,T} x + b^y\right) \end{aligned} \quad (13)$$

where

$$v^y_j = \frac{1}{2\sigma^{*2}_j} - \frac{1}{2\hat{\sigma}_j^2} \qquad w^y_j = \frac{\hat{\mu}_{yj}}{\hat{\sigma}_j^2} - \frac{\mu^*_{yj}}{\sigma^{*2}_j}$$

$$b^y = \sum_j \left(\frac{\mu^{*2}_{yj}}{2\sigma^{*2}_j} - \frac{\hat{\mu}^2_{yj}}{2\hat{\sigma}_j^2}\right) + \log\left(\frac{\sigma^*_j}{\hat{\sigma}_j}\right)$$

Finally, by assumption the variance is the same in S as in the general distribution, i.e., $\hat{\sigma}_j = \sigma^*_j = \sigma_j$, for all features, j. Thus, v^y from Equation 13 becomes zero, so we are left with a linear model for m^y, with weights, w^y, and bias, b^y, given by Equation 5. □