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

Klas Leino
Carnegie Mellon University

Matt Fredrikson
Carnegie Mellon University

Outline

- White-box MIA: adversary has access to weights of the model
  - Bayes-optimal attack
  - Linear bayes-wb attack
  - Linear general-wb attack
  - Deep bayes-wb/general-wb attack
- Experiments

An intuitive example

(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.

A model's idiosyncratic use of features can provide evidence for membership to white-box attackers:

In Figure 1 (c), the trained model uses the pink background, rather than the face region, to infer the identity of the face, which implies that there are images of this class (Tony Blair) with pink backgrounds in the trainset.

Bayes-optimal attack

Experiment 1 (Membership experiment $\mathrm{Exp}^{\mathrm{M}}(\mathcal{A}, A, n, \mathcal{D})$). *Let $\mathcal{A}$ be an adversary, A be a learning algorithm, n be a positive integer, and $\mathcal{D}$ be a distribution over data points (x, y). The membership experiment proceeds as follows:*

1. *Sample $S \sim \mathcal{D}^n$, and let $A_S = A(S)$.*
2. *Choose $b \leftarrow \{0, 1\}$ uniformly at random.*
3. *Draw $z \sim S$ if $b = 0$, or $z \sim \mathcal{D}$ if $b = 1$|*
4. *$\mathrm{Exp}^{\mathrm{M}}(\mathcal{A}, A, n, \mathcal{D})$ is 1 if $\mathcal{A}(z, A_S, n, \mathcal{D}) = b$ and 0 otherwise. $\mathcal{A}$ must output either 0 or 1.*

- Generative assumptions:

data distribution D^* follows a Gaussian mixture model:

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

training set S is drawn i.i.d. from D^*, the empirical distribution of S is denoted by $\widehat{D}$, with parameter $\hat{p}$, $\hat{\mu}$, $\hat{\Sigma}$

- Bayes-optimal attack:

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.*

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

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

Theorem 1 is based on assign (x,y) to the distribution that maximizes its likelihood:
If (x,y) is more likely under D^*, then $m^y(x) = 0$;
if (x,y) is more likely under $\widehat{D}$, then $m^y(x) = 0$.

Linear bayes-wb attack

- Bayes-optimal attack requires exact knowledge of D^* and $\widehat{D}$. Linear bayers-wb does not require such knowledge, but still requires the Gaussian mixture generative assumption.
- The target model is assumed to be a binary logistic regression model: $argmax_{c\in[C]}\{softmax(\hat{W}^T x+\hat{b})_c\}$
- Linear 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*

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

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

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

$\hat{g}$: the target model; $\tilde{g}$: proxy model, i.e. shadow model, trained with the same hyperparameters as $\hat{g}$.

Linear general-wb attack

- Linear bayes-wb attack requires D^* to be a Gaussian mixture model. Linear general-wb attack lifts this assumption.

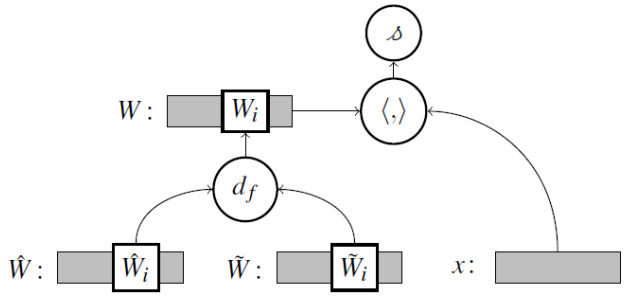


Figure 3: Illustration of the generalized attack model. A learned displacement function, d, is applied element-wise to the weights of the target and proxy model to produce attack model weights, W. The inner product of W and x is then used to make the membership prediction. *Not pictured: d is also applied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

In linear bayes-wb, the displacement function d_f is a simple subtraction, which comes from the Gaussian generative assumption.
In linear general-wb, d_f can be learned.

Algorithm 2: The Linear general-wb MI Attack

def `createAttackModel` $(\hat{g}, \tilde{S}, N)$**:**

  for $i \in [N]$ **do**

    $\tilde{S}_i^1, \tilde{S}_i^0 \leftarrow \texttt{split}_i(\tilde{S})$

    $\check{g}_i \leftarrow \texttt{trainShadow}(\tilde{S}_i^1)$

    $\tilde{g}_i \leftarrow \texttt{trainProxy}(\tilde{S}_i^0)$

  $T \leftarrow \left[(\check{g}_i.W_{:y}, \tilde{g}_i.W_{:y}, \check{g}_i.b_y, \tilde{g}_i.b_y, x, \ell)\right]$
  $\forall (x,y') \in \tilde{S}_i^\ell : y' = y,\ \forall y \in [C],\ \forall \ell \in \{0,1\},\ \forall i \in [N]$

  $D \leftarrow \underset{D'}{\text{argmin}} \left\{ \underset{(\hat{w},\tilde{w},\hat{b},\tilde{b},x,\ell)\in T}{\mathbb{E}} \left[\mathcal{L}\left(\sigma(D'(\hat{w},\tilde{w})^T x + D'(\hat{b},\tilde{b})), \ell\right)\right] \right\}$ ← learn the displacement function

  $\tilde{g} \leftarrow \texttt{trainProxy}(\tilde{S})$

  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

Calibrating the decision threshold

Algorithm 3: Calibrating the Decision Threshold

def `calibrateThreshold` $(m, \tilde{S}, \alpha)$**:**

$\tilde{S}' \leftarrow \texttt{sample}(\tilde{S})$

$\tilde{P}'_y \leftarrow [m^{y'}(x') \text{ for } (x', y') \in \tilde{S}' : y' = y] \quad \forall y \in [C]$

$\tau_y \leftarrow \texttt{sort}(\tilde{P}'_y)_{\alpha|\tilde{P}'_y|} \quad \forall y \in [C]$

return τ

def `predictMembership` (m, x, y, τ)**:**

return 1 if $m^y(x) > \tau_y$ else 0

MIA for deep models

- Challenge when migrating to deep models:

  The target model in previous attacks are a simple logistic regression model. For neural networks, the semantic meaning of an internal feature at a given index will not line up with the semantic meaning of the corresponding internal feature in another model. Thus $\widehat{w}$ and $\widetilde{w}$ can't be compared elementwise, as in the linear general/bayes-wb attacks. One solution is to fix the feature representation in the shadow model to preserve the semantic meaning between the two.

- Solution: influence measure of internal features:
  - The neural network f is decomposed into two function at layer l, g and h, such that $f = g \circ h$
  - Define influence measure:

Definition 1. *The influence of an element j in the internal representation defined by $s = \langle g, h \rangle$ is*

$$\chi_j^s(f, P) = \int_{\mathcal{X}} \left. \frac{\partial g}{\partial z_j} \right|_{h(\mathbf{x})} P(\mathbf{x}) d\mathbf{x} \qquad (1)$$

MIA for deep models

- Influence of internal features z of layer l

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

where $z = h(x)$. In practice P is set to a uniform distribution.

- Linear local approximation of g part of the network:

$$g(z) = W_x^T z + b, \text{ where } W_x = \mathcal{X}(g \circ h, P_0), b = g(0)$$

- Deep Bayes-wb attack:

Algorithm 4: The Deep bayes-wb MI Attack

def `createAttackModel` $(\hat{g} \circ \hat{h}, \tilde{S})$**:**

$\tilde{S}' \leftarrow [(\hat{h}(x), y) \text{ for } (x,y) \in \tilde{S}]$

$\tilde{g} \leftarrow \texttt{trainProxy}(\tilde{S}')$

$w^y \leftarrow \lambda(z) : \chi(\hat{g} \circ \hat{h}, P_0^z)_y - \chi(\tilde{g} \circ \hat{h}, P_0^z)_y \quad \forall y \in [C]$

$b^y \leftarrow \hat{g}(0)_y - \tilde{g}(0)_y \quad \forall y \in [C]$

return $\lambda(x,y) : \mathcal{S}\left(w^y (\hat{h}(x))^T \hat{h}(x) + b^y\right)$

def `predictMembership` (m, x, y)**:**

return 1 if $m^y(x) > \frac{1}{2}$ else 0

MIA for deep models

- Combining layers:

Algorithm 4 only applies attack to one single layer. To attack on different layer, a meta model takes the confidences of the attack on 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.

Experiments

- Datasets :
  - synthetic data from Gaussian mixture model
  - Adult, pima Diabetes, Breast Cancer Wisconsin, Hepatitis, German Credit, Labeled Faces in the Wild, MNIST, CIFAR10, CIFAR100
- Models:
  - For synthetic data: logistic regression
  - For non-image data, MLP with one hidden layer and ReLU
  - For image data, LeNet

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.

On synthetic Gaussian naïve-Bayes data

	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.

Data scaling

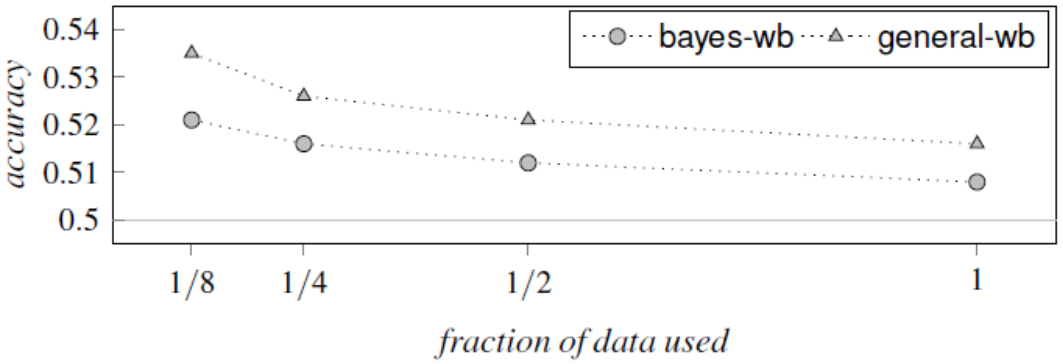


Figure 7: Accuracy of the bayes-wb and general-wb attacks on the Adult dataset, as the amount of data is scaled from 6,105 records (1/8 of the full dataset) to 48,841 records.

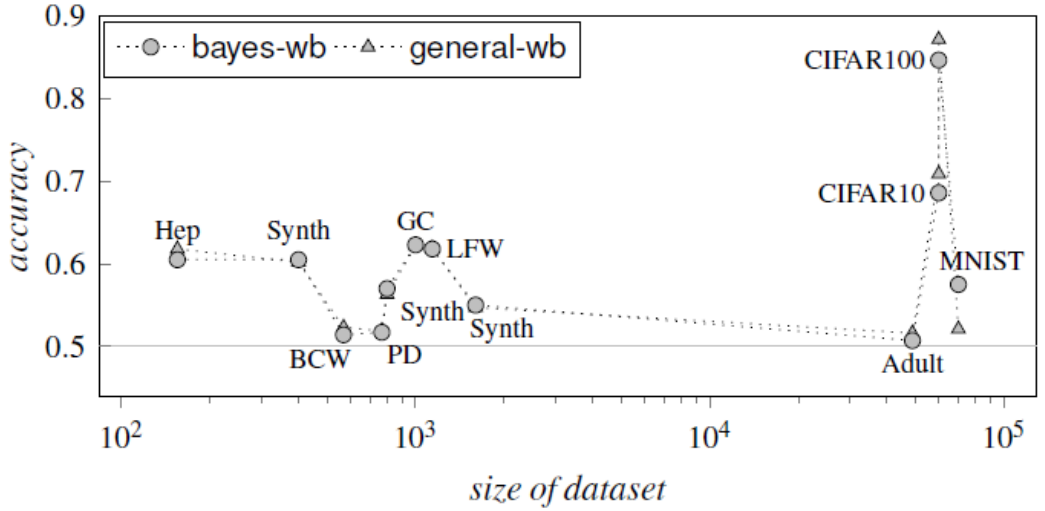


Figure 8: Accuracy of the bayes-wb and general-wb attacks on each of the datasets in our evaluation, plotted against the size of the respective dataset.

The larger the training set, the lower the attack accuracy.
This is due to that the empirical distribution $\widehat{D}$ of a larger training set is more close to the true data distribution D^*.
The white-box attacks here are all based on the difference between $\widehat{D}$ and D^*.

Combining layers

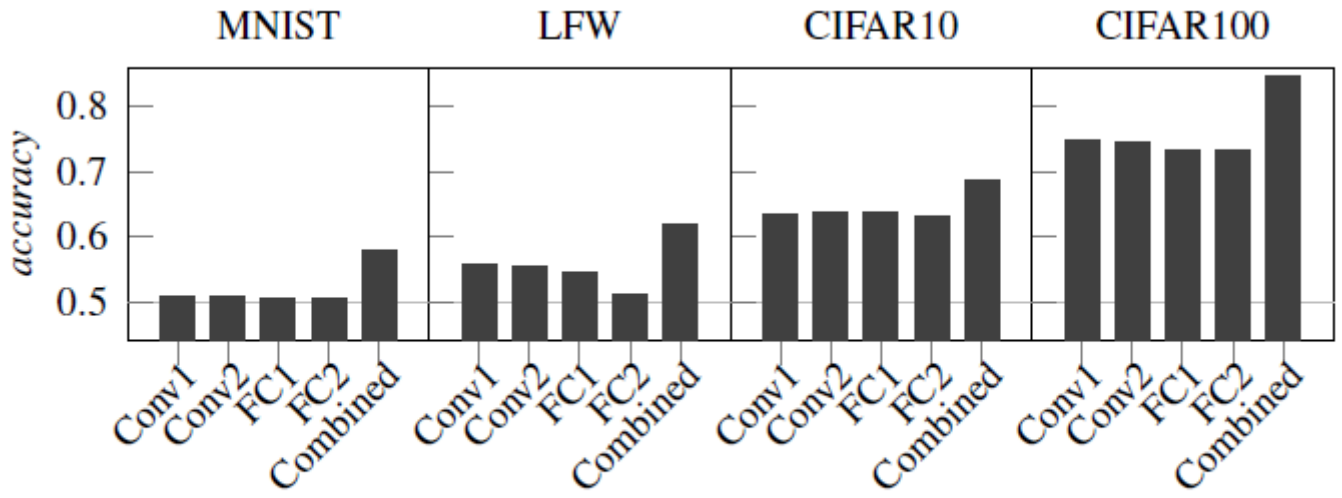


Figure 9: Accuracy of the bayes-wb attack on each individual layer of LeNet, compared with the accuracy using the combined meta-model.

- The meta attack outperforms the individual attacks.
- Information leakage occurs in the representation learned by layers throughout the model. Each layer plays some role in the leakage of information about the training data.
- Models trained with transfer learning may leak less information about the training data used to tune the model.

Comparison with Shokri MIA: performance

	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.

- Accuracy and precision of the proposed method are better than prior attacks.
- Recall of the proposed method is lower. However, we care more about precision, since a high precision means that the adversary is confident about a positive sample's membership in the training set. If the adversary confidently identifies even one training point, then it is reasonable to say that a privacy violation occurred.
- On well-generalized datasets (BCW, PD, Adult, MNIST), the proposed method outperforms prior attacks.

Comparison with Shokri MIA: calibration

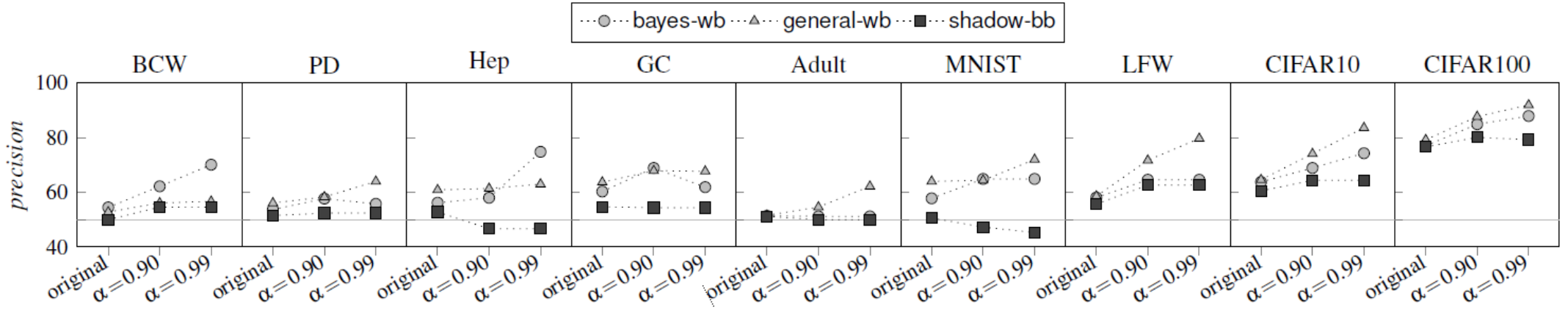


Figure 11: Precision of the bayes-wb, general-wb, and shadow-bb attacks, calibrated using the heuristic outlined described in Algorithm 3 (with $\alpha = 0.90$ and $\alpha = 0.99$), compared to the precision with no calibration (default threshold).

- Increase decision threshold, precision of the proposed attacks increased, but the precision of shadow-bb is decreased by increasing the threshold.
- The probability outputs of shadow-bb are less well-calibrated.

Similarity of shadow-bb and naïve results

- naïve attack: 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.
- For naïve attack:

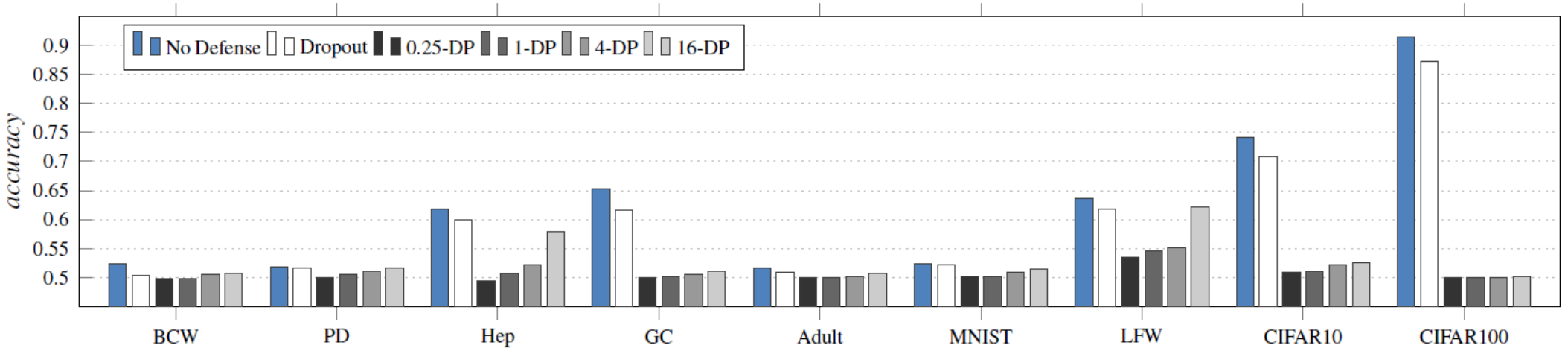
$$accuracy = 0.5 + (train_{acc} - test_{acc})/2$$
$$precision = train_{acc}/(train_{acc} + test_{acc})$$
$$recall = train_{acc}$$

- Performance of shadow-bb is similar with naïve.

	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

- Possible explanation: The softmax 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.

Defenses: DP training



dataset		*no defense*	*dropout*	$\varepsilon = 0.25$	$\varepsilon = 1$	$\varepsilon = 4$	$\varepsilon = 16$
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

- DP training can effectively defense against MIA, though at a high cost at a high-cost in target model performance.
- Dropout (regularization) is a practical defense. It reduces overfitting, is beneficial to the performance of the model, while providing a modest defense.