CHALMERS
UNIVERSITY OF TECHNOLOGY

Using Approximate Bayesian Neural Networks for Membership Inference

Master Thesis in Complex Adaptive Systems

Lorenzo Sucameli

Examiner: Alexandre Graell i Amat
Supervisor: Marcus Lassila

Department of Electrical Engineering

Chalmers University of Technology

Gothenburg, Sweden 2026

Abstract

As machine learning systems are increasingly deployed in privacy-sensitive settings, evaluating their vulnerability to Membership Inference Attacks (MIA) has become an important security concern. These attacks aim to determine whether a given data point was part of a target model's training set, thereby potentially revealing sensitive information about individuals or organisations. Although state-of-the-art attacks such as LiRA, rMIA, and BASE can achieve strong performance, they typically rely on training an ensemble of shadow models and therefore require substantial computational resources. This work investigates a more efficient alternative based on approximate Bayesian inference. In particular, we evaluate four Bayesian Neural Networks approximations: Monte Carlo Dropout, SWAG, Laplace approximation, and Regularized Variational Inference as cost-effective alternatives for membership inference against deterministic target models. We compare these methods across multiple classification settings and against shadow-model-based attacks. Our results show that Monte Carlo Dropout and SWAG can achieve performance comparable to a limited shadow-model ensemble, providing a useful trade-off between attack strength and computational cost, while also showing that metrics such as diversity and accuracy can serve as helpful predictors of attack performance.

Acknowledgements

I would like to thank my examiner, Alexandre Graell i Amat, for the opportunity to learn from him and his helpful commentaries. I am also grateful to my supervisors Marcus Lassila and Amandus Reimer for their help in shaping the direction of the thesis and their continued support. I also want to thank my colleague Albin Edegran for his interesting questions and suggestions which greatly helped me in finishing this project.

Contents

List of Figures

List of Acronyms

MIA Membership Inference Attack

LiRA Likelihood Ratio Attack

rMIA Robust Membership Inference Attack

BASE Bayes-optimal Approximation Score Estimator

TPR True Positive Rate

FPR False Positive Rate

AUC Area Under the Curve

ROC Receiver Operating Characteristic

NN Neural Network

BNN Bayesian Neural Network

CNN Convolutional Neural Network

LLM Large Language Model

VGG Visual Geometry Group

ReLU Rectified Linear Unit

SGD Stochastic Gradient Descent

MAP Maximum A Posteriori

ELBO Evidence Lower Bound

SWAG Stochastic Weight Averaging-Gaussian

SWA Stochastic Weight Averaging

GP Gaussian Process

KL Kullback-Leibler

GGN Generalized Gauss-Newton

Auditing Data points Data points whose membership status we are inferencing

Target model Black-box NN we are attacking with MIA

Shadow model NN we train to carry out a MIA

IN-model Shadow model trained on samples containing auditing data

OUT-model Shadow model not trained on samples containing auditing data

1
Introduction

1.1 Background

As many machine learning models become increasingly widespread in our society, concerns about their reliability and security in matters of privacy have begun to emerge. In this context, membership Inference Attacks (MIAs) have emerged as one of the most widely studied threats to the privacy of machine learning models. These attacks aim to determine whether a given data point was part of a *target model*'s training dataset, thereby revealing sensitive information about individuals or organizations represented in the data. Over the past decade, extensive research has demonstrated that many classes of machine learning models—ranging from deep neural networks to classical classifiers—are vulnerable to MIAs under a variety of conditions. Such findings have had important implications for both privacy-preserving machine learning and regulatory compliance.
Many strategies have been developed [1] [2], performing better or worse in different conditions, but all the best performing ones ([3], [4], [5]) are computationally expensive. In fact, they all require the training of a large number of *shadow models*: neural networks, trained in similar conditions to the target model, aimed at replicating the statistical properties of the original model to differentiate between the differences between training and non-training data. But, when the target model is too large, even training a single shadow model can become difficult, and the attack performance significantly drops when there are not enough models.
A possible solution lies in using a single *Bayesian Neural Network* (BNN) to approximate the model instead, leveraging its built-in probabilistic nature to estimate the statistics of the data. But, as BNN models of deep neural networks are computationally intractable, it becomes important to find a strategy to approximate their behaviour effectively.

1.2 Aim

The aim of the thesis is to find an effective strategy to approximate Bayesian Neural Networks behaviour and carry out Membership Inference Attacks in different and varied settings. We then compare their attack performance between themselves and traditional state-of-the-art attack strategies while also using different metrics to understand more deeply why they succeed or not.

1.3 Thesis Structure

Here, the structure of the thesis is presented. The first section covers the background information and the theory behind MIA and BNN. Then, it is followed by a method section, where the conducted experiments and their set-up are described in detail. The subsequent result section reports the results we obtained. Lastly, there are the discussion and conclusion sections, explaining our findings and highlighting the possible future directions of this research area. The python Colab code used to generate our results can be found on GitHub [6].

2
Related Studies

In recent years, machine learning has become an increasingly important tool in our society, used in many areas, from pattern recognition to natural language processing, thanks to its capacity for modelling highly complex systems. In particular, models like LLM Agents have become fundamental not only for modern data analysis and scientific applications, but also for everyday uses by the general public. Thus, it becomes necessary to study machine learning models in greater depth to better understand their capabilities and their limitations.
Contrary to previous applications of computing, which relied exclusively on explicitly programmed rules, machine learning models are able to learn directly from data, adapting their internal parameters in order to approximate complex relationships between variables. This constitutes their main advantage, as it makes them particularly useful in settings where the underlying structure of the problem is difficult to model analytically, but also their weakness, as it worsens their interpretability.

In the context of supervised learning, a model is trained on a specific dataset, consisting of labelled examples

$$\mathcal{D} = \{z_i\}_{i=1}^{N} \;=\; \{(x_i, y_i)\}_{i=1}^{N},$$

where x_i denotes an input and y_i the corresponding labels. The objective is to learn a mapping

$$\hat{y} = f(x; \theta),$$

parametrized by the model's parameters θ, trying to *fit* each label y_i with the output $\hat{y} = f(x_i; \theta)$. The final goal is to obtain a model that can produce accurate predictions for previously unseen inputs. Depending on the task, the labels may represent continuous values, as in regression, or discrete classes, as in classification.
The actual structure of the model can vary. They are all made up of many *artificial neurons*, which can be considered simple functions, taking an input, transforming it linearly according to its parameters θ_i and applying a predefined non-linear function called *activation function* to obtain an output. Although a single neuron can already be used to model a problem, their strength comes from stacking many neurons together, allowing them to approximate any mathematical function [7]. The architecture of the network can be made of simple *Dense Layers*, where different neurons work in parallel, each taking the previous input array to obtain a new set of outputs (one for each neuron), or other more refined structures, like *Convolutional Layers*, where we use a set of filters that sequentially take different parts of the input and transform them linearly based on the filter's parameters θ_i, before applying again a non-linear activation function. These types of layers are mainly used for analyzing images, as they subdivide them based on spatial

proximity, taking advantage of the localized correlations commonly found in this type of data, and using more layers to capture longer-range and more complex correlations.
The quality of the model predictions is evaluated through a loss function $\sum_i \ell(z_i,\ \theta) = \sum_i \ell(y_i,\ f(x_i, \theta))$, which measures the average discrepancy between the predicted output and the true labels. Training consists of adjusting the parameters θ in order to minimize the loss over the training set. This optimization is typically carried out using stochastic gradient descent (SGD) or other variants, which iteratively update the parameters in the direction of steepest decrease of the loss:

$$\theta \leftarrow \theta - \eta \nabla_\theta \ell(z_i,\ \theta),$$

where η is the learning rate.
We can also add a regularizer term to the loss, for example $+\frac{\theta^2}{\alpha}$, to improve generalization to unseen data.
Thus, by repeatedly applying this procedure for each data point in the training data, the model progressively improves its performance until convergence to the optimal set of parameters θ_{best}.

2.1 Privacy Risk

Nonetheless, this reliance on many different types of external data is what exposes those models to serious vulnerabilities, which can lead to violations of the users' privacy. The first definition of privacy risk, reported by Shokri et al. [1], was the *"Dalenius desideratum"*, as it is known in statistical discourse. This definition states that "the model should reveal no more about the input to which it is applied than would have been known about this input without applying the model". But, as noted in the same article, this goal is unrealistic for any useful model.
Useful models, in fact, *generalize from their training data* , and thus give back more information about their inputs than what we originally had. For example, a model evaluating the cancer risk of a patient must provide doctors with more information than before applying the model; otherwise it would not be useful. So Shokri et al. proposed a new definition, taking into account the specific privacy risk only for those whose personal data was used for training the models.

Definition 1 (Privacy Risk)**.** *The neural network under examination, called* target model*, is considered private if the information it reveals about the* members *of its training dataset does not exceed what it reveals about the general population (*non-members*).*

The most basic attacks in this setting are the Membership Inference Attacks (MIA, see the next Section 2.2), which evaluates the risk that a specific data point $x_i,\ y_i$ is a member of the target model's training set, using the model's outputs $f(x_i)$ and some general information about the dataset. These attacks may appear limited as they assume complete prior knowledge of the possible members' inputs and outputs, which is often unrealistic, but they can be part of wider *extraction attacks* against machine learning models, aimed at recovering the full training dataset through a process of trial and error, of which MIAs are a key component.
Other types of attacks, like *attribute inference*, are also essentially based on MIA, as shown by Yeom et al. [8]. This attack tries to predict the value of an unknown feature $x_{2,i}$ (the attribute) of a data point $x_{1+2,i},\ y_i$ through the target model and the other available

information $x_{1,\,i},\ y_i$, and compares the prediction accuracy for the members against the non-members. This is similar to performing a MIA with only partial information about the data points' inputs.
In fact, they showed that the ability to predict the attribute of a data point translates into the ability to predict its membership status, but not vice versa. Thus, we compile the global accuracy of the predictions: the number of correctly guessed attributes compared to the total number of attributes.

However, the use of simple accuracy as a metric to evaluate the efficiency of these attacks has been called into question by Carlini et al. [3]. Accuracy is, in fact, a balanced metric, which does not take into account the difference in importance between achieving a high True Positive Rate ($\mathbf{TPR} = \frac{\#\text{discovered members}}{\#\text{total members}}$) and a low False Positive Rate ($\mathbf{FPR} = \frac{\#\text{mislabeled non-members}}{\#\text{total non-members}}$). In fact, having a highly accurate membership inference attack against a model is not particularly significant when the number of false positives is very high. As MIAs are often part of extraction attacks, where the number of true members is very limited, being able to correctly guess the membership of many members (high **TPR**) at the cost of mislabeling many more non-members (high **FPR**) poses very little danger to the privacy of the involved individuals. Instead, an attack which can confidently recognize only a few members (average **TPR**) without mislabeling the non-members (very low **FPR**) is significantly more dangerous. In fact, already having a few individuals exposed as certain members poses a much higher privacy risk for them, as explained by Carlini et al. in the same paper [3].
This is the reason why we evaluate the attack power by looking at the so-called ROC plot of our attack. Instead of a single membership prediction, we define an array of scores s_i for all the data points we are inquiring (where a higher score represents a more likely membership) and iteratively threshold them at τ, starting from $\tau = +\infty$ to $\tau = -\infty$, thus considering a point a member only if $s_i > \tau$. So we obtain a plot of the True Positive Rate against the False Positive Rate for each τ.
Therefore, to evaluate the MIA, we look at the general Area Under the Curve (**AUC**), but in particular at the **TPR** at very low **FPR**: if it is high, it means we can correctly predict the membership status of many members while making very few errors with the non-members.

2.2 Membership Inference Attack Game

Following [3], a Membership Inference Attack (MIA) can be formalized as a two-player indistinguishability game between a *challenger* and an *adversary*.

Setup:

- Let $\mathcal{D}$ be the underlying data distribution.
- The challenger samples a training dataset $S \sim \mathcal{D}^n$.

Game:

1. A target model f_θ is trained on S following a specific training procedure.

2. The challenger randomly samples a data point x to create an auditing data point, according to:

$$x \leftarrow \begin{cases} S & \text{if } m = 1 \\ \mathcal{D} \setminus S & \text{if } m = 0 \end{cases} \quad \text{with } m \sim \text{Bernoulli}\left(\tfrac{1}{2}\right).$$

3. The challenger provides the adversary with:
   - The auditing points,
   - Black-box access to the model $f_\theta(x)$ (no knowledge of the trainable parameters),
   - The data distribution $\mathcal{D}$,
   - Possibly the model architecture and the training procedure.
4. The adversary must find the *decision rule* $\hat{m}(x) \in \{0, 1\}$ to guess the membership status of the auditing set:

$$\hat{m}(x) = \begin{cases} 1 & \text{if } x \text{ is predicted to be in } S \\ 0 & \text{otherwise.} \end{cases}$$

5. The game continues with the Challenger sampling a new data point.

Thus, the attack is considered successful if the adversary achieves high accuracy at low False Positive Rate, compared to random guessing.

As shown by Yeom et al. [8], the effectiveness of MIA typically relies on *overfitting*, which is the tendency of neural networks to more accurately predict the *members* — data points that were part of the training dataset — compared to the *non-members*. Therefore, the attack performance increases when the target model $f_\theta(x)$ exhibits better predictions for the *members*. Neural network models, in fact, have a strong tendency to memorize their training data, depending not only on their architecture, but also on the dataset structure. This was proven by Zhang et al. [9], where they trained models on an image dataset with randomized labels, showing that those models could achieve near-perfect accuracy with sufficient training. In some contexts, where there are groups of samples with very few atypical individuals, memorizing these samples becomes even necessary to achieve optimal accuracy, as shown in [10]. In both cases, such memorization directly increases the distinguishability of members compared to non-members, helping the adversary to identify them.
This relationship is explained by theory, as shown in the same paper [8] and [11]: The ability of an algorithm $A : D^n \to \mathcal{Y}$ to achieve a given generalization error is equivalent to its stability rate for a given training set size n:

$$\textbf{Stability rate}: \qquad \mathrm{E}\left[\ell(A(S \cup z_i), z_i) - \ell(A(S), z_i)\right] < \epsilon(n)$$

where $A(S)$ represents the model A trained on S and $\epsilon(n)$ is a steadily decreasing function, with

$$z_i \sim D \quad \text{and} \quad S \sim D^n.$$

The concept of stability is also closely related to *Differential Privacy*, first defined by [12]:

$$\textbf{Differential privacy:} \qquad \Pr\Big[A(S) \in Y\Big] \leq e^{\varepsilon} \Pr\Big[A(S \cup z_i) \in Y\Big].$$

Where $z_i \sim D$, $S \sim D^n$, and $\mathcal{Y}$ is the function space of the models.
If this definition holds for every S and z_i, A is said to be ε-differentially private. So a stable and differentially private algorithm leads to less overfitted models on average, which in turn decreases the strength of the MIA, leading to more privacy-preserving models, as highlighted in [8]. In fact, in Yeom et al. **Theorem 1** it is proven that for an ε-differentially private algorithm, the true positive rate **TPR** of the MIA can be bounded by

$$\mathbf{TPR} \leq e^{\epsilon} - 1 + \mathbf{FPR}.$$

This can be understood as a consequence of the indistinguishability induced by the ε-differentially private algorithm: the adversary succeeds when the models produced by algorithm A differ enough upon the change of a single training point, so that the loss on a data point varies significantly depending on whether it is included in the training set or not.

Nonetheless, stability is not a sufficient condition to ensure privacy. In fact, malicious training algorithms can be created using additional training data, obtained by applying a set of *pseudo-random pairing functions* to each of the original training input-output pairs z_i, generating new seemingly benign data points.
Thus, by applying those functions to the original training data we obtain an enlarged training dataset, which is used for training the models. This allows those who know the functions to carry out a MIA using not only the information given by a single data point, but also that of all the extra points obtained by applying the pseudo-random pairing functions. This means we have a non-overfitted model (as the extra points are apparently independent of the original one) which is however completely non-private.

Nevertheless, overfitting is usually a strong predictor of attack strength. This could make the loss value of an auditing point z_i (together with a predetermined threshold τ) a potentially satisfactory decision rule $\hat{m}(x) = \Theta\left[\ell(\theta, z_i) - \tau\right]$.

Where Θ is the step function.

This simple strategy, however, can be highly unreliable. It is common, in fact, to observe data points which exhibit a significantly *high loss* despite being members, or others showing a very *low loss* despite being non-members, as shown in Figure 3 by Carlini et al. [3], leading to high false-negative and false-positive rates, respectively. And in general, different data points (both members and non-members) can have different average (averaged over different models) losses, so that a single threshold τ for every data point does not generalize well. In fact, MIA performs best at low FPR when most data points show a large difference between their loss when they are members compared to when they are non-members.
Thus, further strategies were developed, which can usually be divided into two methods: classifier-based methods and statistics-based methods (see Section 2.4).

Classifier-based Methods: This method employs a *Neural Network Classifier* to work as an *approximation* of the decision rule $\hat{m}(x)$. We first train a series of neural networks (shadow models), set in similar conditions to the target model (similar learning rates, optimizers, number of epochs, architecture, etc.), but using different *subsets* of the data distribution $\mathcal{D}$ as our training set. In this way, we obtain a series of models whose training

sets are known, so that we can compare their confidence values ($P(\hat{y}(x_i) = y_i)$ for example) on their training data with those on the non-training data.
After training these models, we train the classifier using the models' confidence values, for a given data point, as the input and the membership status for that specific shadow model as the output. Thus, as we are using different data points and different models including, or not including, *each* data point in their training set, we can hope that this classifier also generalizes well to the target model.
This method can be effective, but it requires training a whole additional neural network, making it computationally expensive. It also creates a "black-box" approximation of the decision rule, making it hard to interpret the results.

Statistics-based Methods: Therefore, new methods using a more theoretical probabilistic approach emerged. These methods aim to estimate the decision rule using a more strictly defined model, instead of a less transparent binary classifier.
Some of these were defined as practical heuristics, without strong theoretical foundations, while others were approximations (or could be proven to be approximations) of a Bayes-optimal membership inference rule, a (computationally intractable) *probabilistic* equation based on Bayes' rule (see Section 2.3).

2.3 The Bayes-Optimal Rule

We can derive the Bayes-optimal membership inference rule for independent data points following the proof by Sablayrolles et al. [2]. Given the information available to the adversary: the auditing data point z_1 with an unknown membership status m_1 (1 represents members of the training set, 0 non-members), a set of parameters θ representing the target model (along with other possible sources of prior knowledge), we define the optimal inference rule as:

Definition 2 (Membership inference)**.**

$$M(\theta, z_1) := P(m_1 = 1 \mid \theta, z_1). \tag{2.1}$$

Which can be rewritten as:

$$\begin{aligned} M(\theta, z_1) &= P(m_1 = 1 \mid \theta, z_1) \\ &= \mathbb{E}_{\mathcal{T}}[P(m_1 = 1 \mid \theta, z_1, \mathcal{T})]\,. \end{aligned} \tag{2.2}$$

Where we denote by $\mathcal{T}$ the collection of all the possible samples and their respective membership states except the auditing one (z_1, m_1):

$$\mathcal{T} := \{z_2, \ldots, z_n, m_2, \ldots, m_n\}.$$

Then, using Bayes' rule for the argument of the expectation (and assuming that $P(m_1 \mid z_1, \mathcal{T}) = P(m_1)$):

$$P(m_1 = 1 \mid \theta, z_1, \mathcal{T}) = \frac{P(\theta \mid m_1 = 1, z_1, \mathcal{T})\, P(m_1 = 1)}{P(\theta \mid z_1, \mathcal{T})} = \tag{2.3}$$

$$= \frac{P(\theta \mid m_1 = 1, z_1, \mathcal{T})\, P(m_1 = 1)}{P(\theta \mid m_1 = 1, z_1, \mathcal{T})\, P(m_1 = 1) \;+\; P(\theta \mid m_1 = 0, z_1, \mathcal{T})\, P(m_1 = 0)} \tag{2.4}$$

Which can be redefined with a sigmoid function as:

$$P(m_1 = 1 \mid \theta, z_1, \mathcal{T}) = \sigma\left(\log \frac{P(\theta \mid m_1 = 1, z_1, \mathcal{T})\, P(m_1 = 1)}{P(\theta \mid m_1 = 0, z_1, \mathcal{T})\, P(m_1 = 0)}\right) \tag{2.5}$$

Thus:

$$\boxed{M(\theta, z_1) = \mathbb{E}_{\mathcal{T}}\left[\sigma\left(\log \frac{P(\theta \mid m_1 = 1, z_1, \mathcal{T})}{P(\theta \mid m_1 = 0, z_1, \mathcal{T})} + t_\lambda\right)\right]}, \tag{2.6}$$

where:

$$t_\lambda := \log \frac{\lambda}{1-\lambda}, \qquad \lambda := P(m_1 = 1). \tag{2.7}$$

So that λ is the prior probability of z_1 being a member, often assumed to be 0.5.

We can simplify it further by rewriting

$$P(\theta \mid m_1 = 0, z_1, \mathcal{T}) \propto \exp\Big(-\frac{m_1}{T}\ell(\theta, z_1)\Big) \exp\left(-\frac{1}{T}\sum_{i=2}^{n} m_i\, \ell(\theta, z_i)\right) = \exp\left(-\frac{1}{T}\sum_{i=2}^{n} m_i\, \ell(\theta, z_i)\right)$$

Which leads to:

$$p_{\mathcal{T}}(\theta) := P(\theta \mid m_1 = 0, z_1, \mathcal{T}) = \frac{\exp\Big(-\frac{1}{T}\sum_{i=2}^{n} m_i\, \ell(\theta, z_i)\Big)}{\int \exp\Big(-\frac{1}{T}\sum_{i=2}^{n} m_i\, \ell(t, z_i)\Big)\, dt}.$$

Thus:

$$\begin{aligned} P(\theta \mid m_1 = 1, z_1, \mathcal{T}) &= \frac{\exp\Big(-\frac{1}{T}\ell(\theta, z_1)\Big) \exp\Big(-\frac{1}{T}\sum_{i=2}^{n} m_i\, \ell(\theta, z_i)\Big)}{\int \exp\Big(-\frac{1}{T}\ell(t, z_1)\Big) \exp\Big(-\frac{1}{T}\sum_{i=2}^{n} m_i\, \ell(t, z_i)\Big)\, dt} \\ &= \frac{\exp\Big(-\frac{1}{T}\ell(\theta, z_1)\Big)\, p_{\mathcal{T}}(\theta)}{\int \exp\Big(-\frac{1}{T}\ell(t, z_1)\Big)\, p_{\mathcal{T}}(t)\, dt}. \end{aligned} \tag{2.8}$$

Therefore, we obtain the final expression for the Bayes-optimal MIA:

Theorem 1 (Bayes-optimal membership inference rule)**.**

$$\boxed{M(\theta, z_1) = \mathbb{E}_{\mathcal{T}}\left[\sigma\left(\frac{1}{T}\left(-T \log \int e^{-\ell(t, z_1)/T} p_{\mathcal{T}}(t)\, dt - \ell(\theta, z_1)\right) + \log \frac{\lambda}{1-\lambda}\right)\right]}. \tag{2.9}$$

This solution presents two complex integrals to evaluate, making it intractable. Therefore an appropriate approximation for the integral term is required. Also, we notice that white-box access to the target model does not offer any advantage over black-box access, under these assumptions, as all the terms containing the model parameters θ appear inside the loss function.

2.4 Statistics-Based Methods

Many methods have been refined to offer a good approximation of the Bayes-optimal rule (Theorem 1, Section 2.3), some more directly derived from it, like BASE ([5]) or MAST/MALT ([2]); others were designed as practical heuristics, like LiRA ([3]) or rMIA ([4]), even if, in the case of the last one, it has been proven to be equivalent to the BASE method [5].
These methods offer significantly higher performance at low false positive rate values, compared to the classifier-based ones in most scenarios, and have thus become the state-of-the-art solution for carrying out MIA.

2.4.1 LiRA

The Likelihood Ratio Attack (LiRA) is one of the strongest performing MIAs and also one of the first to be proposed. It uses a heuristic to model the decision rule as a hypothesis testing problem:

$$H_0 : z_1 \notin S \quad \text{vs} \quad H_1 : z_1 \in S,$$

where z_1 is the auditing point and S is the target model's training set.
We define the membership inference as:

$$M(\theta, z_1) = p(m_1 = 1 \mid z_1, f_\theta) \approx \frac{p(H_1)}{p(H_0)}. \tag{2.10}$$

Thus the adversary, using only its accessible information, wants to find enough evidence to support H_1 (the hypothesis that the target was trained on z_1) compared to H_0, the opposite hypothesis. In fact, the method can be connected to Eq. 2.6:

$$\frac{p(H_1)}{p(H_0)} = \frac{P(\theta \mid m_1 = 1, z_1)}{P(\theta \mid m_1 = 0, z_1)}.$$

But, as comparing the target model's parameters directly can become infeasible with larger networks,

and in black-box scenarios we do not have direct access to them, Carlini et al. ([3]) proposed to compare the losses directly, using the loss of the target on z_1, $\ell(z_1,\ \theta)$:

$$\frac{p(H_1)}{p(H_0)} = \frac{P(\ell(z_1,\ \theta) \mid m_1 = 1, z_1)}{P(\ell(z_1,\ \theta) \mid m_1 = 0, z_1)}.$$

To model these probabilities $P(\ell(z_1,\ \theta) \mid m_1, z_1)$, we need to find a way to define a suitable distribution and its parameters using empirically measurable quantities. This can be done similarly to the classifier-based method, by training a set of shadow models. We can train N shadow models θ_i, half of which include the auditing point z_1 (IN-models), while the other half exclude it (OUT-models), and define the probability distributions using the mean μ and the standard deviation σ of the losses $\ell(z_1,\ \theta_i)$ across each of the two groups of models.
But a Gaussian model does not seem to correctly approximate the empirical shape of the losses, thus the paper [3] proposes to use a transformation function:

$$\phi(p) = \log\left[\frac{p}{1-p}\right]$$

with $p(z_1,\ \theta_i) = e^{-\ell(z_1,\ \theta_i)}$. In fact, $\phi(z_1,\ \theta)$ is empirically shown to be Gaussian distributed across the shadow models:

Theorem 2 (Online LiRA)**.**

$$\boxed{M(\theta, z_1) \approx \frac{p(H_1)}{p(H_0)} = \frac{\mathcal{N}(\phi(z_1,\ \theta) \mid \mu_1,\ \sigma_1)}{\mathcal{N}(\phi(z_1,\ \theta) \mid \mu_0,\ \sigma_0)}}$$

with

$$\mu_0 = \frac{1}{N} \sum_{\textit{OUT-model } \theta'}^{N} \phi(z_1, \theta'), \quad \sigma_0^2 = \frac{1}{N^2} \sum_{\textit{OUT-model } \theta'}^{N} \left[\phi(z_1, \theta')^2 - \mu_0^2\right]$$

$$\mu_1 = \frac{1}{N} \sum_{\textit{IN-model } \theta'}^{N} \phi(z_1, \theta'), \quad \sigma_1^2 = \frac{1}{N^2} \sum_{\textit{IN-model } \theta'}^{N} \left[\phi(z_1, \theta')^2 - \mu_1^2\right]$$

Another possible approach would be to not define a model distribution, but to empirically evaluate $\ell(z_1,\ \theta_i)$ using the samples of the shadow models, but this would have led to lower performance when the number of shadow models is low.

This approach presents, however, an important limitation related to the *online* construction of the IN-models. As a new auditing point z_2 needs to be tested, we also need to train a new set of $N/2$ shadow models, since $p(H_1) = \mathcal{N}(\phi(z_2,\ \theta) \mid \mu_1,\ \sigma_1)$ depends on having a set of models trained on z_2. This makes the process infeasible for situations where we have a continuously updating stream of auditing points z_i,so that we cannot include them in $p(H_1)$,or when we only have one or a few models to train. Carlini et al. then proposed an *offline* approach, which only requires OUT-models:

Theorem 3 (Offline LiRA)**.**

$$\boxed{M(\theta, z_1) \approx 1 - P(\phi > \phi(z_1,\ \theta))} \quad \textit{with} \quad \phi \sim \mathcal{N}(\mu_0,\ \sigma_0)$$

This strategy does not require retraining the shadow models for every new auditing point, as both μ_0 and σ_0 are computed from OUT-models, even if it should be noted that tit tends to perform slightly worse than the online one.
In addition, this strategy can be further mitigated when we have very few shadow models by using a global σ_0 for all auditing points, instead of computing one for each individual. This allows the average of standard deviations to be computed not only across models, but also across auditing data points, even if it lowers the expressivity of the strategy.

2.4.2 rMIA

Robust Membership Inference Attacks (rMIA) is an alternative strategy developed by Zarifzadeh et al. [4] that promises better performance when the number of shadow models is low. It starts from the same basis as LiRA (Section 2.4.1):

$$\frac{p(H_1)}{p(H_0)} = \frac{P(\theta \mid m_1 = 1, z_1)}{P(\theta \mid m_1 = 0, z_1)},$$

where θ are the parameters of the target model, z_1 is the auditing point, and m_1 is its membership status.
But, instead of considering the hypothesis H_0 of $z_1 \notin S$, we redefine it to $z_1 \notin S$ and $x_i \notin S$ for a sample of reference data points x_i. The probability of H_0 can then be rewritten as $P(\theta \mid x_i)$, so that we define the likelihood ratio $\mathrm{LR}(z_1,\ x_i)$ between the two:

$$\mathrm{LR}_\theta(z_1,\ x_i) = \frac{P(\theta \mid z_1)}{P(\theta \mid x_i)} = \frac{P(z_1 \mid \theta)\ \cancel{P}(\theta)}{P(z_1)}\ \frac{P(x_i)}{P(x_i \mid \theta)\ \cancel{P}(\theta)}.$$

Where $P(x_i \mid \theta)$ is the probability that $x_i^{\mathrm{OUTPUT}} = f_{\theta_j}(x_i^{\mathrm{INPUT}})$ for a given model j. $P(x_i)$ can instead be obtained as an average over N shadow models:

$$P(x_i) = \int P(x_i \mid \theta')\ P(\theta' \mid \mathcal{D}')\ P(\mathcal{D}')\ d\theta' d\mathcal{D}' = \frac{1}{N} \sum_{\text{shadow model } \theta'}^{N} P(x_i \mid \theta').$$

Thus, we define the membership inference as:

Theorem 4 (rMIA)**.**

$$\boxed{M(\theta, z_1) = P(\mathrm{LR}_\theta(z_1,\ x_i) > \gamma) = \frac{1}{M} \sum_{x_i}^{M} \Theta\left[\mathrm{LR}_\theta(z_1,\ x_i) > \gamma\right]}$$

Where we use the different reference data points x_i to compute the likelihood, and we can choose $\gamma \in (1,\ +\infty)$ as a hyperparameter.

This attack can also be divided into two different strategies for the computation of $P(x_1)$ and $P(z_i)$.

An online strategy, which employs both IN-models and OUT-models, incurs the same problems as the online LiRA (Section 2.4.1); an offline strategy computes these probabilities using only OUT-models. Nonetheless, this offline approach introduces a bias towards smaller values for $P(x_1)$ and $P(z_i)$, as the OUT-models tend to exhibit lower confidence on their non-training data. But we can still refine this strategy with a *linear interpolation heuristic*: we define $P(x_i)$ as a linear interpolation between $P^{OUT}(x_i) = \frac{1}{N} \sum_{\text{OUT-model } \theta'}^{N} P(x_i \mid \theta')$ and $P^{IN}(x_i) = 1$:

$$\textbf{Linear Interpolation:} \qquad P(x_i) = \alpha\ P^{OUT}(x_i) + (1 - \alpha) \quad \text{with} \quad \alpha \in (0,\ 1),$$

so that we can increase the average probability of all the data points as α grows larger.

2.4.3 BASE

BASE is a membership inference attack developed by Lassila et al. [5] directly derived as an approximation of the Bayes-optimal rule (Theorem 1, Section 2.3).

$$\textbf{Ensemble Term:} \quad \mathbb{E}_{\mathcal{T}}\left(\log\ \left[\int e^{-\ell(t,z_1)} p_{\mathcal{T}}(t)\ dt\right]\right) = \log\ \left[\frac{1}{N} \sum_{\text{shadow model } t}^{N} e^{-\ell(t,z_1)}\right] \tag{2.11}$$

where we assume that $T = 1$ for every shadow model with parameters t.

Thus we can write:

Theorem 5 (BASE)**.**

$$\boxed{M(\theta, z_1) \approx \sigma\left(-\log\ \frac{1}{N}\sum_{t}^{N} e^{-\ell(t,z_1)}\ - \ell(\theta, z_1) + \log\frac{\lambda}{1-\lambda}\right)}. \tag{2.12}$$

This result can also be rewritten based on the probability $P(\theta,\ z_1)$ instead of the loss:

$$\boxed{M(\theta, z_1) \approx \sigma\left(\log\ \frac{P(z_1 \mid \theta)}{\frac{1}{N}\sum_{t}^{N} P(z_1 \mid t)} + \log\frac{\lambda}{1-\lambda}\right)} \tag{2.13}$$

where $P(z_1 \mid t)$ is the same as in rMIA (Section 2.4.2).

This is not the only similarity. In fact, it has been shown in the same paper that, as the number of reference data points x_i increases, rMIA becomes identical to BASE. Thus BASE can work as a slightly less computationally expensive rMIA.
Following Lemma 1 in [5], we can formulate a definition of MIA-equivalence:

Definition 3 (MIA-Equivalence)**.** *Two score-based MIAs are considered equivalent if, for every set of scores, they are related by a monotonic function.*

This definition can be intuitively understood as a consequence of the thresholding procedure we use to derive the inference.
In fact, even if the set of scores changes, if the *order* remains the same, the thresholding will produce the same result. The relevant thresholds themselves will still change (for example, if we scale the scores by a certain factor, the threshold value to obtain a 0.8 TPR will be scaled by that factor), but, since building a ROC plot requires spanning the threshold values from $+\infty$ to $-\infty$, the plot itself will look the same.

From this, it is straightforward to see that we can rewrite Theorem 5 as:

$$M(\theta, z_1) \approx \frac{P(z_1 \mid \theta)}{\frac{1}{N}\sum_{t}^{N} P(z_1 \mid t)}. \tag{2.14}$$

And rMIA can be rewritten in this limit as:

$$M(\theta, z_1) \approx P_{x_i \in \mathcal{D}}\left[\frac{P(x_i \mid \theta)}{\frac{1}{N}\sum_{t}^{N} P(x_i \mid t)} \leq \frac{P(z_1 \mid \theta)}{\frac{1}{N}\sum_{t}^{N} P(z_1 \mid t)}\right], \tag{2.15}$$

which is the CDF evaluated at $\frac{P(z_1|\theta)}{\frac{1}{N}\sum_{t}^{N} P(z_1|t)}$, which is monotonically increasing in this setup.

Also, even though the ensemble term from the Bayes-optimal rule (Theorem 1) only requires OUT-models, we can apply the linear interpolation heuristic here as we do for rMIA (Section 2.4.2).

2.5 Bayesian Approximations

As we observed, every one of those state-of-the-art methods relies on training multiple shadow models to estimate the **Ensemble Term** from Equations 2.11 and 1. This can be

a straightforward approach when the training procedure is not excessively expensive, but can become problematic when the computational cost increases too much.
A possible solution could be to instead train a single Bayesian Neural Network to approximate that term. Those types of neural networks can, in fact, naturally model the uncertainty of their weights as they are trained on a set of data points, making them ideal candidates to estimate the expected loss of a given data point.
Their structure is similar to that of standard deterministic neural networks. They are made of a set of artificial neurons, holding parameters θ, which determine how an input x_i is transformed into an output $\hat{y}_i = f(x_i,\ \theta)$. The difference is that these parameters are not fixed in value; instead, they are stochastic variables sampled from a probability distribution learned by the model during the training process.
Thus, we do not simply aim to find the optimal parameter values θ_{best}, but to learn the probability distribution $P(\theta \mid \mathcal{D})$, given by Bayes' Theorem:

Theorem 6 (Bayes' Theorem)**.**

$$P(\theta \mid \mathcal{D}) = \frac{P(\mathcal{D} \mid \theta)\ P(\theta)}{P(\mathcal{D})} = \prod_i \frac{P(x_i, y_i \mid \theta)\ P(\theta)}{P(x_i, y_i)} = \prod_i \frac{P(x_i, y_i \mid \theta)\ P(\theta)}{\int_\theta P(x_i, y_i \mid \theta)\ P(\theta)\ d\theta}$$

where we call $P(\mathcal{D} \mid \theta)$ the likelihood and $P(\theta)$ the prior (the generalized form of the regularizer term discussed earlier), with $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ being the independently distributed training data points.
Thus, by specifying a prior over the parameters θ (often assumed to be a standard Gaussian with variance α):

$$P(\theta) = \mathcal{N}\left(\theta \;\middle|\; 0,\ \frac{\mathbf{I}}{\alpha}\right)$$

and the form of the likelihood distribution:

$$P(\mathcal{D} \mid \theta) = \exp\left(-\frac{\sum_i \ell\Big(y_i, f(x_i,\ \theta)\Big)}{\beta}\right),$$

where β and α are chosen hyperparameters, inference is also formulated as a probabilistic function:

$$P(y \mid x,\ \mathcal{D}) = \int P(y \mid x,\ \theta)\ P(\theta \mid \mathcal{D})\ d\theta.$$

Thus, Bayesian neural networks can be viewed as a generalization of deterministic neural networks, where we do not consider only the most optimal value of the parameters θ that minimize the loss, but the whole distribution.

Nonetheless, even if Bayesian neural networks are more accurate than deterministic ones and are capable of accurately modeling the uncertainty, they are severely limited by their higher computational cost. In fact, modeling N parameters is very different from modeling an N-dimensional probability distribution, both in terms of training and inference time. For example, even when considering a simple Gaussian model for their distribution, the mean and the covariance matrix would increase the number of trainable parameters to $\mathcal{O}(N^2)$. This issue is even more relevant for state-of-the-art neural networks, which often employ billions of parameters.
We will thus need to propose several strategies to approximate the performance of Bayesian neural networks, while ensuring their computational feasibility.

2.5.1 Laplace

One of the most classical approximation methods for Bayesian Neural Networks is the *Laplace Approximation*. This strategy approximates $P(\theta \mid \mathcal{D})$ as a multivariate Gaussian distribution with mean and variance defined by the model's parameters, $\mathcal{N}(\theta \mid \mu,\ \Sigma)$. The mean μ is assumed to be equal to the maximum a posteriori (θ_{MAP}), the maximum value of $P(\theta \mid \mathcal{D})$, which can be reasonably approximated by the value of θ_{best} obtained by a deterministic neural network trained on $\mathcal{D}$. The covariance matrix Σ is then evaluated based on the curvature of the posterior around θ_{MAP}:

$$\Sigma = \left(-\nabla^2_\theta \log P(\mathcal{D} \mid \theta)\Big|_{\theta_{\text{MAP}}} \right)^{-1} = \left(\frac{\mathbf{I}}{\alpha} + \sum_i \nabla^2_\theta \ell(y_i, f(x_i, \theta))\Big|_{\theta_{\text{MAP}}} \right)^{-1}.$$

Thus, we need to invert a potentially singular or indefinite matrix, which leads to problems if not handled correctly. A solution employed by Daxberger et al. [13] is to use a Fisher approximation:

$$F \ := \ \sum_{n=1}^{N} \mathbb{E}_{\tilde{y}_n \sim p(y \mid f_\theta(x_n))} \Big[\nabla_\theta \log p(\tilde{y}_n \mid f_\theta(x_n))\, \nabla_\theta \log p(\tilde{y}_n \mid f_\theta(x_n))^\top \Big]_{\theta=\theta_{\text{MAP}}}$$

or a generalized Gauss-Newton matrix approximation:

$$G \ := \ \sum_{n=1}^{N} J(x_n) \left[\nabla^2_f \log p(y_n \mid f)\Big|_{f=f_{\theta_{\text{MAP}}}(x_n)} \right] J(x_n)^\top, \qquad J(x_n) := \nabla_\theta f_\theta(x_n)\Big|_{\theta=\theta_{\text{MAP}}}$$

for the covariance term Σ. We still need to approximate the covariance afterwards, as it can become intractable with the full parameter set of a modern neural network.
Furthermore, this Laplace approximation can be applied with different setups, based on the complexity of the model and the computational speed required. For example, Daxberger et al. [13] found that using only the last-layer weights to compute the Hessian instead of all weights "typically yields better performance due to less underfitting, and is significantly cheaper", making it suitable for analyzing models like ResNet which have millions of weights.

Nevertheless, the Laplace approximation can easily suffer from sampling in very restricted loss regions, as we will discuss in Section **??**. This issue is examined in detail by Lim et al. in [14], even if it does not involve only Laplace, but also other Bayesian approximation strategies, and even the standard shadow model strategy, which can be considered an approximation of a Bayesian Network.
Still, the shadow model strategy retains an advantage over the others, as SGD and similar training algorithms tend to converge to flatter solutions, unlike methods such as the Laplace approximation, which rely on sampling weights around a single point estimate.

2.5.2 SWAG

Stochastic Weight Averaging-Gaussian (SWAG) is another simple strategy to approximate a BNN, developed by Maddox et al. in [15]. They fit a Gaussian $\mathcal{N}(\theta \mid \mu,\ \Sigma)$ using: the Stochastic Weight Averaging (SWA) solution, obtained by averaging SGD iterates ([16]),

as the mean, and a low-rank plus diagonal matrix, also derived from the SGD iterates, as the covariance:

$$\mu^{(i)} = \theta^{(i)}_{\text{SWA}} = \sum_{t=0}^{T} \theta_t^{(i)}$$

$$\Sigma = \frac{1}{2}\,\Sigma_{\text{diag}} + \frac{1}{2}\,\Sigma_{\text{LR}}$$

where:

$$\Sigma_{\text{diag}} = \sum_{t=0}^{T} \Big(\theta_t^{2\ (i)} - \theta^{2\ (i)}_{\text{SWA}}\Big), \qquad \Sigma_{\text{LR}} = \frac{1}{T-K} \sum_{t=T-K}^{T} (\theta_t^{(i)} - \theta^{(i)}_{\text{SWA}}) \cdot (\theta_t^{(i)} - \theta^{(i)}_{\text{SWA}}).$$

Then, we can sample from this Gaussian distribution a set of models and easily perform Bayesian inference.
The algorithm used to sample the iterates resembles a simple stochastic gradient descent, using a high-value constant learning rate to let the algorithm explore the many possible solutions around θ_{MAP}. The prior in this scenario is represented by the regularization term applied in the SGD learning algorithm.
This strategy also promises to improve generalization, as it could lead to flatter and less unstable areas of the loss function thanks to the SGD-iterates averaging, even if more recent papers like [14] have highlighted that this is not always the case.

This strategy is based on the theoretical analysis of the stationary distribution of SGD iterates examined in [17], which suggests that the SGD trajectory can contain useful information about the geometry of the posterior. Nonetheless, as explained in the previous paper [15], this theoretical result is not valid for Deep Neural Networks, due to non-convexity and over-parameterization. Yet, the SGD iterates still seem to be able to capture the shape of the posterior well, and the low-dimensional subspace they span is approximately Gaussian within a basin of attraction.
Despite this, the SGD iterates can only move along this limited subspace, defined by the steep gradients, and so cannot move along the flat trajectories of the loss function. This limits their usefulness in our case study for effectively sampling a sufficiently diverse set of parameters.

2.5.3 Variational Inference with Regularized KL-Divergence

Contrary to the previous Bayesian methods, Variational Inference with *regularized KL-Divergence* is a significantly more principled, yet complex, approach. Unlike normal Variational Inference, we do not place the variational distribution directly in weight space; instead, we represent uncertainty through the network's input-output function, thereby avoiding the infinite KL-Divergence that often arises with many GP priors.
Our aim is to approximate the posterior distribution $p(\theta \mid D)$ with a tractable distribution $q_\phi(\theta)$, using the KL-Divergence to minimize the difference between the two distributions:

$$\text{KL}\Big(q_\phi(\theta) \,\|\, p(\theta \mid D)\Big) = \mathbb{E}_{q_\phi(\theta)}\left[\log \frac{q_\phi(\theta)}{p(\theta \mid D)}\right] = \mathbb{E}_{q_\phi(\theta)}[\log q_\phi(\theta) - \log p(D \mid \theta) - \log p(\theta)]\,.$$

Where we discarded $\mathbb{E}_{q_\phi(\theta)}\Big[\log p(D)\Big]$ since it does not depend on θ.
So minimizing $\text{KL}\Big(q_\phi(\theta) \,\|\, p(\theta \mid D)\Big)$ is equivalent to maximizing the ELBO:

$$\mathcal{L}(\phi) = \mathbb{E}_{q_\phi(\theta)}[\log p(D \mid \theta)] - \text{KL}\Big(q_\phi(\theta) \,\|\, p(\theta)\Big).$$

Or, equivalently, minimizing:

$$\mathbb{E}_{q_\phi(\theta)}[\ell(D;\theta)] + \mathrm{KL}\Big(q_\phi(\theta) \,\|\, p(\theta)\Big).$$

Thus, we only need to define which distribution $q_\phi(\theta)$ to use.
Following [18], we model the distribution as:

$$q_\phi(\theta) = \mathcal{N}(\theta \mid \mu,\ \Sigma),$$

where μ is the mean value of the weights and Σ is a diagonal covariance. We can then sample the weights from $q_\phi(\theta)$ to compute the output $\hat{y} = \int f(x \mid \theta)\ P(\theta \mid \mathcal{D}) d\theta$. But, to improve the computational speed of gradient descent, we can simplify this by linearizing the output around $\theta = \mu$:

$$f_L(x, \theta) = f(x,\ \mu) + J(x,\ \mu)(\theta - \mu).$$

Thus we can use $\hat{y}(x_i)$ and y_i to compute the loss function ℓ.
For the prior we use a Gaussian Process (GP) to model a prior over the function space and the regularized KL-Divergence to compare $q_\phi(\theta)$ and $p(\theta)$. The training objective thus becomes:

$$\sum_{i=1}^{N} \mathbb{E}_{q_\phi(\theta)}\Big[\log p\Big(y_i \mid f_L(x_i;\theta)\Big)\Big] - D^{\gamma}_{\mathrm{KL}}\Big(Q^F_\phi \,\|\, P^F\Big),$$

where Q^F_ϕ is the pushforward of $q_\phi(\theta)$ through the linearized network and P^F is the GP prior.
To compute this term, we sample M measurement points $x^{(j)} \sim \rho(x)$, where $\rho(x)$ is a uniform distribution over the data points. Thus, we calculate it by computing the regularized KL-Divergence between the two Gaussian distributions:

$$m_1 = f(x;\mu), \qquad \Sigma_1^{(\gamma)} = J(x;\mu)\ \Sigma\ J(x;\mu)^\top + \gamma I,$$

$$m_2 = \mu(\theta), \qquad \Sigma_2^{(\gamma)} = K(x,x) + \gamma I.$$

The parameter γ is a regularizer to make the covariances numerically more stable, while the choice of ρ determines where the prior is enforced in function space.
So the GP term becomes:

$$D^{\gamma}_{\mathrm{KL}}(\mathcal{N}_1 \| \mathcal{N}_2) = \frac{1}{2}(m_1 - m_2)^\top (\Sigma_2^{(\gamma)})^{-1}(m_1 - m_2) + \frac{1}{2}\,\mathrm{Tr}\Big[(\Sigma_2^{(\gamma)})^{-1}\Sigma_1^{(\gamma)} - I_M\Big] - \frac{1}{2}\log\det\Big[(\Sigma_2^{(\gamma)})^{-1}\Sigma_1^{(\gamma)}\Big],$$

$$m_i = \mu_i(x), \qquad \Sigma_i^{(\gamma)} = K_i(x,x) + \gamma_M I_M.$$

2.5.4 Dropout Monte Carlo

Monte Carlo Dropout is another simple, but easily scalable, approximation of a BNN. It uses Dropout, a mechanism of the training procedure where a random subset of the weights of a NN is temporarily set to zero at every iteration [19]:

$$\theta = \begin{cases} \hat{\theta}, & \text{with probability } P_{DP}, \\ 0, & \text{otherwise.} \end{cases}$$

Where P_{DP} is called the Dropout probability and θ is the original weight. This leads to better generalization as the model's parameters are less likely to overfit when fewer are

active. Still, if the dropout probability is too high, this could cause a drop in the model's performance.
This strategy employs Dropout also at inference time, allowing for multiple diversified predictions which can then be averaged. As shown by Gal et al. [20], this strategy can be interpreted as a roughly approximated Variational Inference scheme (Section 2.5.3). In fact, the continuously changing model parameters, as they go on and off, make the loss term $\frac{1}{\#\text{iterations}} \sum_{\text{iteration}} \left[\frac{1}{N} \sum_i^N \ell(z_i, \theta) \right]$ approximately a Monte Carlo approximation (across all iterations) of the variational inference's likelihood term, while the regularization term is roughly proportional to the KL-divergence term.
Furthermore, this method can be easily applied to already trained models with dropout. But this advantage is also its flaw, as there is no systematic way to choose where to apply dropout or how much. Also, as high values of dropout decrease performance, it is necessary to increase the size of the BNN model to improve accuracy, even if this makes the training procedure more expensive compared to the target one.

3
Methods

3.1 Datasets and Target Models

3.1.1 Datasets

We used three image classification benchmarks throughout the experiments: CIFAR-10, CIFAR-100, and CINIC-10. All datasets consist of colour images of resolution 32×32 pixels.
CIFAR-10 [21] consists of 60,000 images spanning 10 object categories, subdivided into 15,000 images for training each target model, with the remainder providing non-member auditing data and shadow-model training data.
We also use CIFAR-100, a variant of CIFAR-10 comprising the same total number of images but organised into 100 fine-grained classes. Training sets of 15,000 images per target model are used here as well, reducing the per-class sample count by a factor of ten relative to CIFAR-10 and making the classification task substantially more complex.
To test on a larger dataset, we also employ CINIC-10, an augmented version of CIFAR-10. As the name suggests, it consists of ten classes of images, 60,000 of which are from CIFAR-10 with the rest from downsampled ImageNet images of the same categories, yielding approximately 270,000 images in total [**?**]. We use training sets of 90,000 images per target model to account for the greater complexity of the images, which have a larger intra-class variance as they are generally more noisy.

3.1.2 Target Model Architectures

We use four different neural network architectures in our experiments. All models are created using `TensorFlow` and trained with SGD with Nesterov momentum. A batch size of 128 is used throughout the experiments.

ResNet: The primary architecture for the CINIC-10 dataset. It is a custom residual network [22] adapted for 32×32 inputs. A 3×3 convolutional stem producing 32 feature maps is followed by eight residual blocks whose filter widths progress through the stages {64, 64, 128, 128, 256, 256, 512, 512}. Three stride-2 down-sampling operations are applied between stages via strided convolutions. Global average pooling, a dropout layer with dropout probability of 0.25, and a final dense classification head complete the network. All convolutional layers use batch normalisation [23], He-normal weight initialisation, and L_2 weight decay with coefficient $\lambda = 10^{-4}$. We train it for 10 epochs

with a learning rate $\eta_0 = 0.1$, plus 2 additional epochs for fine-tuning with a learning rate $\eta = 0.01$, achieving 98% training and 75% test accuracy.

Small ResNet: A lightweight variant of the previous architecture, used for CIFAR-10. It is made of six residual blocks with filter widths in {16, 16, 32, 32, 64, 64}, two stride-2 down-sampling steps, and a dropout rate of 0.3 before the head. We train this model for 17 epochs with a learning rate $\eta_0 = 0.1$, plus 3 epochs for fine-tuning with a learning rate $\eta = 0.01$, achieving 98% training and 75% test accuracy.

MobileNetV2: A transfer-learning setup in which the convolutional backbone of MobileNetV2, pretrained on ImageNet [24], is kept frozen throughout training [25]. Images are resized before being passed through the backbone to account for the difference in resolution. A trainable classification head consisting of a dense layer with 128 units: Rectified Linear Unit (ReLU [26], [27]) as the activation function and L_2 regularisation with coefficient 6×10^{-2}). This is followed by a dense output layer trained on top of the frozen features for 10 epochs with a learning rate $\eta = 0.01$. We use this setting for CIFAR-10, achieving 98% training and 86% test accuracy.

EfficientNetV2: A transfer-learning setup in which the convolutional backbone of MobileNetV2, pretrained on ImageNet, is kept frozen in the first phase of training, and it is later unfrozen for fine-tuning [28]. Like with the MobileNetv2, images are resized before being passed through the backbone to account for the difference in resolution. A trainable classification head consisting of a dense layer with 1024 units (ReLU activation, L_2 regularisation with coefficient 1×10^{-3}) followed by a dense output layer is trained on top of the frozen features for 10 epochs with a learning rate $\eta_0 = 0.01$. The fine-tuning is carried out for 5 epochs with a learning rate $\eta = 1 \cdot 10^{-4}$. We use this setting for CIFAR-100, achieving 85% training and 75% test accuracy.
As both this model and the previous one have backbones made of parameters trained on ImageNet, we cannot reliably carry out a MIA using CINIC-10. This would lead to an overlap, as CINIC-10 contains data points downsampled directly from ImageNet's images, thus it would not be a true OUT-model.
In all our models, we also implement safety measures during training, for example lowering the learning rate by 10% if the loss starts to stagnate.

3.2 Attack Implementation

We carry out the attack on a set of different target models to ensure robustness against noise, and perform the LiRA (Section 2.4.1), rMIA (Section 2.4.2), and BASE (Section 2.4.3) attacks in their offline variant. In fact, since the Bayesian models can only be trained once, we cannot employ online attacks, so we train those models on an *external* portion of the dataset, disjoint from the rest.

3.2.1 Membership Inference Attack Design

Ten independent target models are trained for each experimental configuration. Their training sets are constructed by sliding a window of size N_{train} along the pooled dataset,

so that each NN has $N_{\text{extra}} = 5{,}000$ differing images from every other NN. This produces target models with partially overlapping training sets, but different enough to simulate different prediction functions, ensuring that our results are not the result of noise.
Thus, we subdivide the complete dataset into three sets: one for training the shadow models, one for training the target models, and one for sampling the auditing data points to perform the membership inference, with only the last two overlapping partially.

Following the Membership Inference Game definition (Section 2.2), we construct an auditing set of N_{audit} data points with a balanced member-to-non-member ratio ($\alpha = 0.5$): ten groups of member points are sampled from the training set of each target model, and a global non-members group is drawn from the remaining pool. We sample $N_{\text{audit}} = 30{,}000$ data points for CIFAR-10 and $N_{\text{audit}} = 45{,}000$ for CINIC-10. The rest of the remaining data points are used for shadow or Bayesian model training, yielding approximately 25,000 external points for CIFAR-10 and 146,000 for CINIC-10.

For each target model and each attack, an attack score s_i is computed for every auditing point z_i. The ROC curve (TPR versus FPR) is plotted on a log–log scale to better analyze the low-FPR performance. Then, the reported curves and AUC values are obtained by averaging the interpolated TPR values across all ten target models, following the evaluation protocol described in Section 2.1.

3.2.2 Shadow Model Training

Each of the 60 shadow models is trained on a random sample from this *external* set. Thus we avoid the need to train a different set for each target model. We carry out the attack for different numbers of shadow models, to characterise the sensitivity of each attack to the shadow model budget and to accurately compare their performance to the Bayesian models. The shadow model architecture, hyperparameters, and training schedule mirror those of the corresponding target model.

3.2.3 Attack Score Computation

The three statistics-based membership inference attacks, in their offline variety, described in Section 2.4 are evaluated. For LiRA (Section 2.4.1) we employ a global standard deviation strategy for all the shadow models. In this way, we can account for the small number of shadow models in the experiment, while directly using the logarithm of the probabilities to improve score sensitivity. For rMIA (Section 2.4.2) we also use the probability logarithms to compute the scores, while using the extra $N_{\text{extra}} = 5{,}000$ samples not used by the current target model's training set as the reference data points x_i. We also set both γ and α to 1 for both rMIA and the BASE attack (Section 2.4.3).

3.3 Bayesian Neural Network Approximation Methods

We carry out the strategies described in Section 2.5 and evaluate their efficiency compared to the shadow model ensemble. As some strategies are more complex than others, they may require more computational time. Thus, we must not only consider the attack performance of the Bayes model, but also its training computational efficiency. For example, a model

which performs better than five shadow models but requires 10 times the training time would not be a suitable substitute. Therefore, we train each Bayesian model on the same number of samples as the target models.

3.3.1 Stochastic Weight Averaging Gaussian (SWAG)

In this work, SWAG [15] (see Section 2.5.2) is implemented as a Keras training callback (`WeightStatsCallback`) that records all trainable weights every $s_{\text{skip}} = 10$ mini-batch steps, beginning 50 batch computations for the Small ResNet and 100 for the other models, before the end of the training to allow initial convergence. The $K = 20$ most recent mean-centred deviations are retained for the low-rank component. During the collection phase the SGD learning rate is raised by ten times the normal (but only 8 times for EfficientNetV2), to encourage broader exploration of the loss landscape, in line with the original SWA proposal.
We apply this method to a newly trained reference model (based on the target models' architecture) to use as a reference for the construction of the SWAG shadow models. Thus, weight samples are then drawn from $\mathcal{N}(\mu_{\text{SWA}}, \Sigma)$ to create N new SWAG shadow models, identical in structure to the reference model. For the parameters of the Batch Normalisation we do not use a Gaussian Sampling, but we instead copy them directly from the reference model, as the Gaussian approximation is less accurate in this case, as we will show in the results.

3.3.2 Laplace Approximation

The Laplace approximation method (see Section 2.5.1) uses the `torch` package developed by Daxberger et al. [13]. As it is a `PyTorch` package, we rewrite our previous `TensorFlow` models' architectures in the `PyTorch` framework. We employ a *last-layer* approximation, as computing and inverting the full Hessian for the ResNet model is infeasible. Therefore, the GGN-approximated Hessian is evaluated only with respect to the weights of the final linear layer, while all earlier parameters are fixed at their MAP values. The prior precision is treated as a scalar hyperparameter and optimized post hoc by grid search on a held-out validation set, defined as the remaining data points from the *external set* that were not used for Bayesian model training. Posterior predictive samples are then drawn from the neural network predictive model (`pred_type="nn"`) using a Monte Carlo approximation (`link_approx="mc"`) with `La.predictive_samples` and n_{samples} stochastic draws.

3.3.3 Monte Carlo Dropout

While implementing Monte Carlo Dropout [20] (see Section 2.5.4) on the MobileNetV2 model is straightforward, as there is only one trainable final layer where we can apply the method, the same cannot be said of the ResNet models. Thus, we study a varied set of its architectures with different placements for dropout to compare their attack performance. As we use deep convolutional network models, where normal dropout is not the optimal choice, we use `SpatialDropout2D`, inserted after each convolutional layer. Spatial dropout zeroes entire feature maps rather than individual activations, which is more appropriate for convolutional architectures as it preserves spatial correlations within

each map. The block-level dropout rate p_{DP} is treated as a hyperparameter. At inference time, dropout is activated by passing `training=True` to the model call, while disabling `Batch Normalisation`.

3.3.4 Gaussian Function-Space Variational Inference

As the most principled yet complex of the four approaches, the Gaussian function-space variational inference scheme (see Section 2.5.3) is implemented following [18]. We initialize the standard deviation of the weights θ_k as: $\sigma_k = \text{softplus}(\rho_k) + \epsilon$ with $\rho = -5.5$ for all the models. Then, the linearisation $f_L(x, \theta) = f(x, \mu) + J(x, \mu)(\theta - \mu)$ is computed using forward-mode automatic differentiation `tf.autodiff.ForwardAccumulator` to compute $J(x, \mu)\ \delta$. The Gaussian Process (GP) prior is defined as a zero-mean process with a radial basis function kernel:

$$K(x, x') = \sigma_0^2 \exp\left(-\frac{\|x - x'\|^2}{2\ell^2}\right).$$

We use a length-scale $\ell = 21$ and output variance $\sigma_0^2 = 1.0$. This prior is enforced at $M = 16$ measurement points, uniformly sampled at random at each training step.
We use $\gamma = 10^{-5}$ as a numerical jitter to improve the stability of the prior and posterior covariances. The model is optimised with Adam at a learning rate of 0.001 and a balance coefficient applied to the prior loss of 0.09.

We estimate the likelihood term and the posterior distribution of the KL term with Monte Carlo using $m_{\text{lik}} = 10$ and $m_{\text{post}} = 10$ samples, respectively.
At inference time, $T = 30$ weight perturbations $\delta_t \sim \mathcal{N}(0, \text{diag}(\sigma^2))$ are sampled to compute the linearised outputs $f_L(x, \mu + \delta_t)$, and the resulting softmax probabilities are averaged. This sample mean constitutes the estimate of $P(z_i \mid \theta)$ in the ensemble term.

3.4 Diagnostic Analyses

Even if the accuracy of our BNN approximations compared to the shadow models is a relevant predictor of their attack performance, it is not the only factor. In fact, a "diversity" metric is needed if we want to estimate their capacity of approximating the shadow models. Thus, apart from the primary evaluation using ROC plots, we follow the methods inspired by Vadera et al. [29] and carry out three diagnostic analyses are conducted to characterise the quality of each Bayesian approximation. All analyses are performed on a held-out study set of 5,000 data points that do not overlap with any target training or auditing set.

3.4.1 Weight Distribution

As many of our models implicitly or explicitly model the weight distribution θ_i (with $i \in [1,\ N]$) as an N-dimensional multivariate Gaussian, we study the one-dimensional marginal distributions for a subset of weights to check this assumption. We analyse the ResNet models's parameters, taken from both the filters and the Batch Normalization, both close to the start of the architecture and its end.

3.4.2 Distribution of the Variance of ϕ values

The offline LiRA attack (Section 2.4.1, Theorem 3) models the distribution of each $\phi(z_i, \theta)$ across shadow models as Gaussian. For a BNN approximation to serve as a valid replacement, every $\phi(z_i, \theta)$ sample should also follow an approximately Gaussian distribution. In addition, we compare the variance of the ϕ-values across the shadow models (or along different iterations for the Bayesian models) for the average data point, studying how different are the outputs as the number of models increase.

3.4.3 Diversity Analysis

The effectiveness of any ensemble-based attack depends on the diversity of the models it aggregates: if all samples produce identical predictions, the ensemble term reduces to a single-model baseline, becoming ineffective. To quantify diversity, we follow [**?**] and compute the average pairwise normalised disagreement metric. For n model samples $\{\theta_1, \ldots, \theta_n\}$ evaluated on N test points, the raw pairwise disagreement between models θ_i and θ_j is

$$d_{ij} \;=\; \frac{1}{N}\sum_{k=1}^{N} \mathbf{1}\Big[\hat{y}_k^{(i)} \neq \hat{y}_k^{(j)}\Big], \qquad \hat{y}_k^{(l)} = \arg\max_c f_c(x_k; \theta_l),$$

and the average over all pairs is

$$\bar{d} \;=\; \frac{2}{n(n-1)} \sum_{i<j} d_{ij}.$$

Following the paper, this quantity is normalised by $(1 - a)$, where a is the test accuracy of the last evaluated model, to remove the trivial relationship between disagreement rate and accuracy:

$$\bar{d}_{\text{norm}} \;=\; \frac{\bar{d}}{1-a}.$$

The resulting pair $(a,\, \bar{d}_{\text{norm}})$ is plotted as a single point for each experimental condition (method and number of samples).
Still, contrary to the original paper, we compute this metric as an *average difference between every two models* instead of the *average difference between every model and a* ***baseline***, defined as the predictions $\hat{y}_k^{(\text{baseline})}$ of one of the shadow models.

3.4.4 Calibration Analysis

A calibrated posterior is one whose average predicted confidence matches the empirical accuracy in the corresponding confidence range. Every Bayesian (or shadow model ensemble) strategy aims to estimate the term $\mathrm{E}_{\theta_t}[P(z_i \mid \theta_t)]$, which is the average true-label probability. So, a posterior that is systematically overconfident (the model assigns high probability to incorrect labels) or underconfident (the model assigns low probability to correct labels) compared to the shadow models degrades the attack performance. And studies have argued ([29]) that specifically Bayesian models struggle to perform well in this metric, making a thorough analysis necessary.
Thus, to assess calibration, the ensemble-averaged true-label probability $\bar{P}(z_i) = \frac{1}{T}\sum_t P(z_i \mid \theta_t)$ is computed for each study point, and the study set is partitioned into ten equally spaced bins over $[0, 1]$. Within each bin, the average prediction accuracy across all

ensemble members and all binned data points is recorded. The resulting confidence-accuracy curve is plotted alongside the perfect-calibration diagonal for each method. Systematic deviations above the diagonal indicate overconfidence, while deviations below indicate underconfidence; both are detrimental for the attack as they lower the value of $\mathrm{E}_{\theta_t}[P(z_i \mid \theta_t)]$.

4

Results

4.1 ROC Plots

4.1.1 Small ResNet Model using CIFAR-10

We follow the method examined in 3.1.2 to train 10 small ResNet models with 15,000 images from CIFAR-10 as our targets. We then proceed to use the external dataset to train 20 different shadow models for our membership inference attacks.
We use the same architecture and hyperparameters for our Laplace and SWAG models, but we use instead a different setting for the other methods to account for their larger stochasticity: For Monte Carlo Dropout we use SGD with a learning rate of 0.1 for 9 epochs and other 3 epochs with a learning rate of 0.01 for fine-tuning. The model has the same architecture of the target, but for an additional Dense layer before the output layer, to improve the accuracy, compromising 32 neurons with an active dropout rate of 0.35. The dropout rate on the Convolutional Layer is smaller, at 0.03. We also considered a model with a higher dropout only on the last layers, but the performance significantly worsened in that case. For the Regularized Variational Inference methods, we instead use 14 epochs.
We use the Offline LiRA and Offline BASE methods for our evaluation, as BASE and rMIA are asymptotically the same:

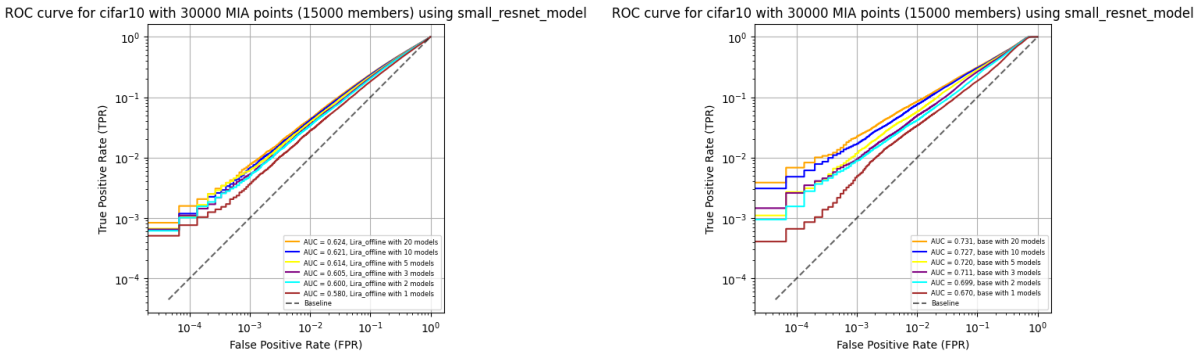

Figure 4.1: ROC plots of the **small ResNet shadow models** and our best-performing Bayesian model. Both are trained on 15,000 samples from CIFAR-10 for the LiRA and BASE attacks. We use 30,000 auditing points, of which half are members. The baseline represents random guessing.

As expected, the performance increases together with the number of shadow models. Measuring the TPR at low FPR and AUC, we see that BASE performs better than LiRA for a larger number of shadow models. Probably for the global standard deviation applied to the LiRA which improves the performance when the number of shadow models is low, but decreases it when it is higher.

By comparing the performances of different shadow models instead on the same two attack methods we obtain:

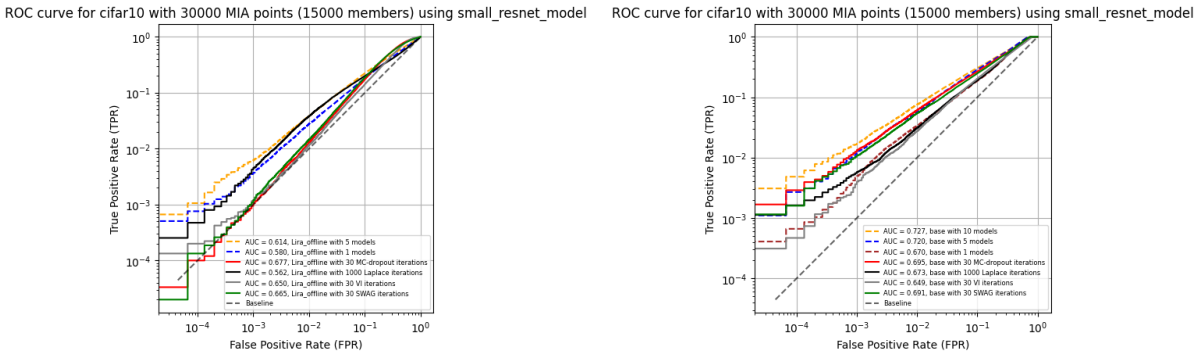

Figure 4.2: ROC plots of all the **small ResNet Bayesian models** (and the shadow models as references) trained on 15,000 samples from CIFAR-10 for the LiRA and BASE attacks. We use 30,000 auditing points, of which half are members. The baseline represents random guessing. **dp** indicates Monte Carlo Dropout and **VI** indicates Regularized Variational Inference.

We evaluated every strategy e at the best of its inference capacity. Laplace, for example, being the fastest and cheapest allows us to use 1000 iterations to ensure the best possible performance. While Regularized Variational Inference being the most computationally expensive, allows for less iterations.
We see that in this case Monte Carlo Dropout, together with SWAG, performs the best in the BASE attack, reaching the performance of a 5 shadow models attack. Laplace is slightly more effective than the single shadow model attack, while Variational Inference is comparable to the single model.
In the case of LiRA though, we see that only Laplace manages to come close to reach the single shadow model MIA, while the others perform similarly to random guessing at low FPR.

4.1.2 ResNet Model using CINIC-10

Following the same strategy, we conduct a MIA against a ResNet NN trained on CINIC-10, training our 10 target models with 15,000 images from CIFAR-10 and 60 different shadow models for our membership inference attacks.
For Monte Carlo Dropout we use a similar architecture as with the *Small ResNet*: A Dropout of 0.1 is distributed throughout the convolutional layers of the model, with an additional Dense layer, of 512 neurons, set before the output with a Dropout of 0.3, to improve the accuracy. We train it for 9 epochs with a learning rate of 0.1 and then fine-tune it for other 9 epochs with a learning rate of 0.01. The Variational Inference

method uses the architecture shown in Subsection 3.3.4 with 8 training epochs.
We see the results here:

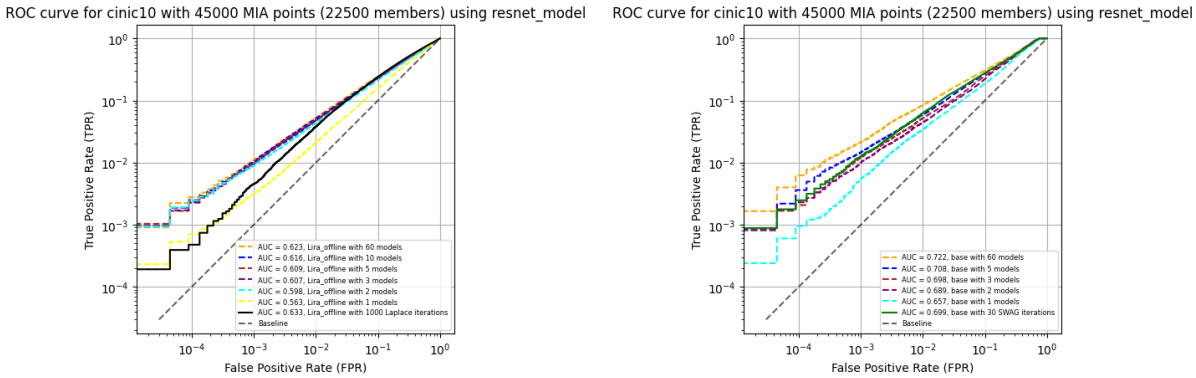

Figure 4.3: ROC plots of the **ResNet shadow models** and our best-performing Bayesian model. Both are trained on 90,000 samples from CINIC-10 for the LiRA and BASE attacks. We use 45,000 auditing points, of which half are members. The baseline represents random guessing.

As before we see that BASE outperforms LiRA as the number of shadow models grows. Comparing the different Bayesian strategy we observe:

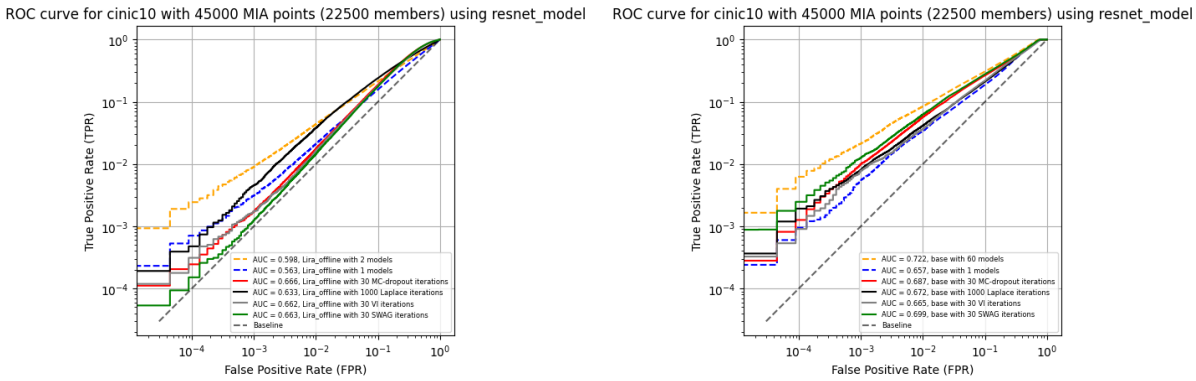

Figure 4.4: ROC plots of all the **ResNet Bayesian models** (and the shadow models as references) trained on 90,000 samples from CIFAR-10 for the LiRA and BASE attacks. We use 45,000 auditing points, of which half are members. The baseline represents random guessing. **dp** indicates Monte Carlo Dropout and **VI** indicates Regularized Variational Inference.

Comparing the performance of the Bayesian models, we observe that in this case BASE-SWAG is the best performing strategy overall, reaching the performance of a 3-shadow models strategy. Monte Carlo Dropout comes close, but it remains inferior at low FPR, being comparable to Laplace there.
For LiRA, only Laplace surpasses the performance of the single-shadow model MIA, even if mainly at higher FPR, while the other methods are slightly above random guessing.

4.1.3 MobileNetV2 Model using CIFAR-10

We follow the same method as before and conduct a MIA against a MobileNetV2 model trained on CIFAR-10. As explained before, only the head is trained for 10 epochs while

the backbone is kept frozen. For Monte Carlo Dropout we use the same architecture as the targets and shadow models, but with an enlarged Dense head of 512 neurons and a dropout of 0.75. For Variational Inference we use 5 training epochs, while for the other methods we use the same strategy as explained in Section 3.3.

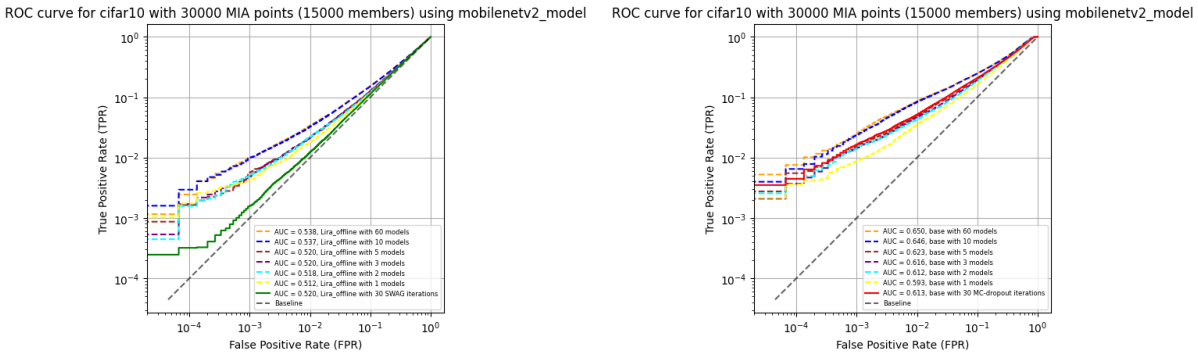

Figure 4.5: ROC plots of the **MobileNetv2 shadow models** and our best-performing Bayesian model. Both are trained on 15,000 samples from CIFAR-10 for the LiRA and BASE attacks. We use 30,000 auditing points, of which half are members. The baseline represents random guessing.

We see that as before BASE performs the best with Monte Carlo Dropout being the best Bayesian strategy overall, performing as well as 5 shadow models in the BASE attack.

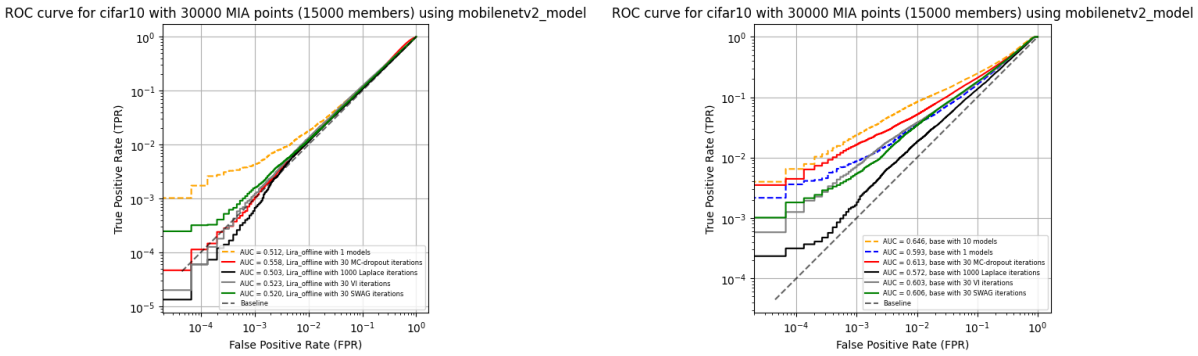

Figure 4.6: ROC plots of all the **MobileNetv2 Bayesian models** (and the shadow models as references) trained on 15,000 samples from CIFAR-10 for the LiRA and BASE attacks. We use 30,000 auditing points, of which half are members. The baseline represents random guessing. **dp** indicates Monte Carlo Dropout and **VI** indicates Regularized Variational Inference.

Observing all the BNN attacks, we see that also in this case no Bayesian strategy outperforms the single-model in the LiRA, with only SWAG being significantly above random guessing. While for BASE Monte Carlo is significantly stronger here than all the other BNN attacks.

4.1.4 EfficientNetV2 model using CIFAR-100

We follow the methods for the *EfficientNetV2* with the CIFAR-100 dataset. We train the head for 10 epochs, with the backbone kept frozen, and then unfreeze the backbone and

fine-tune the whole model for 5 epochs. Similarly to MobileNetv2, we use an enlarged head of 4096 neurons with a dropout of 0.7 for the Monte Carlo Dropout method, with 17 epochs of training and 7 epochs of fine-tuning instead. For the Variational Inference method, we only train the head, keeping the backbone always frozen, as the computational cost would increase too much otherwise. We do not use Laplace for this model for the same reasons.

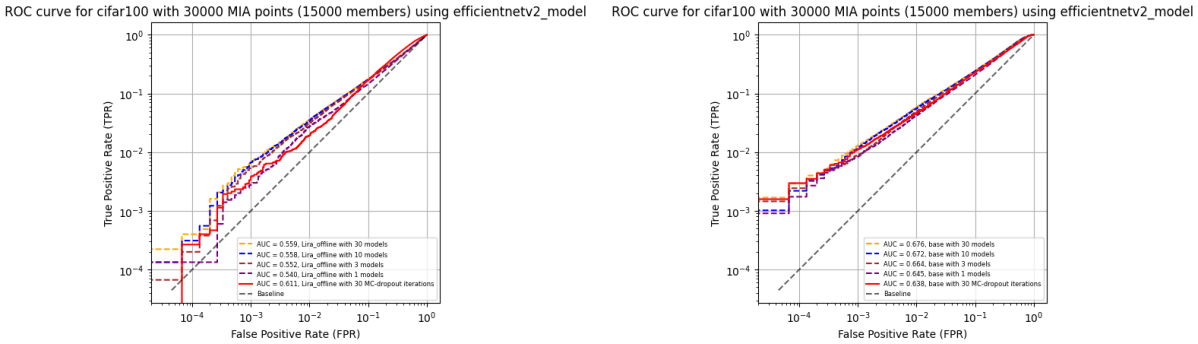

Figure 4.7: ROC plots of the **EfficientNetv2 shadow models** and our best-performing Bayesian model. Both are trained on 15,000 samples from CIFAR-100 for the LiRA and BASE attacks. We use 30,000 auditing points, of which half are members. The baseline represents random guessing.

We see that as before, LiRA performs significantly worse than the BASE attack, with Monte Carlo Dropout being the only method that performs better than the single-shadow model attack in both cases, even if all the attack performances are very similar.

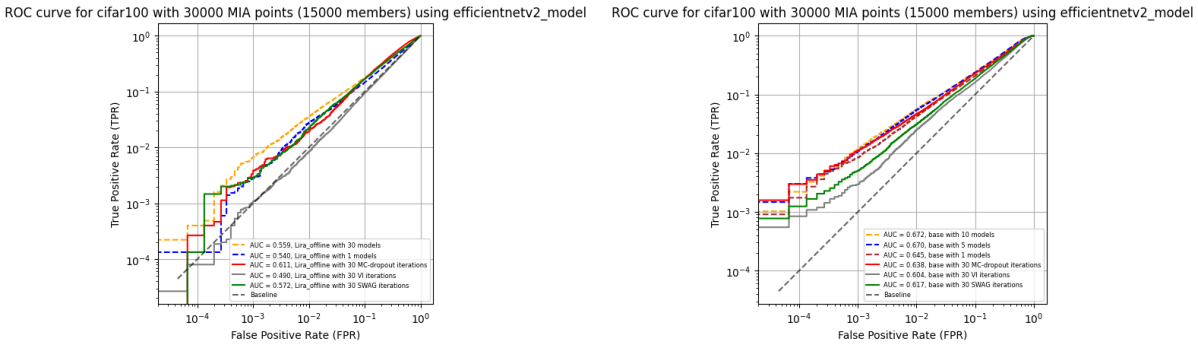

Figure 4.8: ROC plots of all the **EfficientNetv2 Bayesian models** (and the shadow models as references) trained on 15,000 samples from CIFAR-100 for the LiRA and BASE attacks. We use 30,000 auditing points, of which half are members. The baseline represents random guessing. **dp** indicates Monte Carlo Dropout and **VI** indicates Regularized Variational Inference.

We see that for LiRA, both Monte Carlo Dropout and SWAG perform slightly better than the single model for low FPR, differently from the previous models we analyzed, but still remain inferior to the multiple-shadow models attacks. For BASE, we see that Monte Carlo Dropout manages to achieve an attack performance above the 3-shadow-models attack, and only slightly below the 5-shadow-models attack.

4.2 Weight Analysis

Following the method explained in section 3.4.1, we plot the histogram of each weight value from 240 of our Resnet shadow models trained on 90,000 samples from CINIC-10. The weight index indicates the position of the weight in the models'architecture: a smaller index represents weights closer to the input, while larger values indicate values closer to the output.

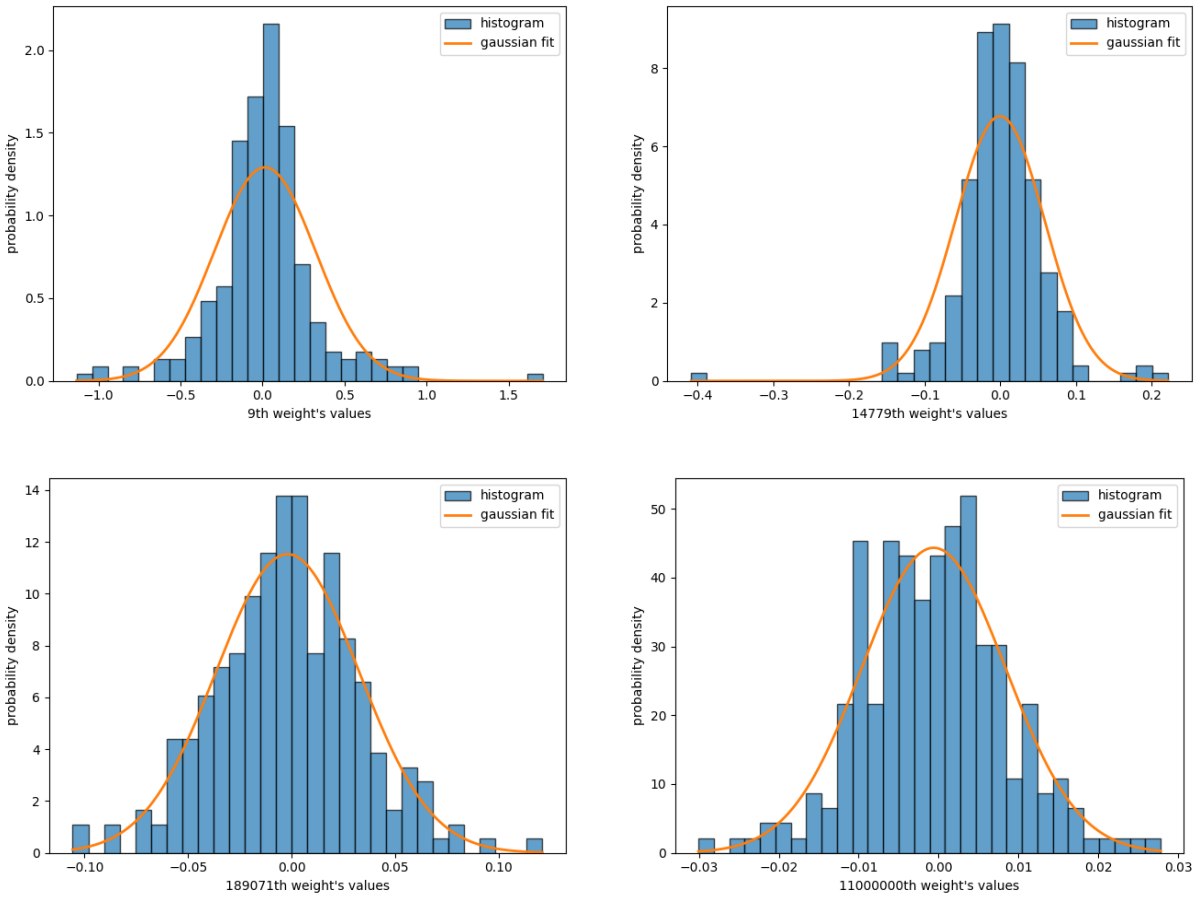

Figure 4.9: **Convolutional Filter**'s weights distribution analysis. Each histogram shows the distribution of the values of one filter's weight of the ResNet across the shadow models. We use 240 different shadow models, each trained on 90,000 samples from CINIC-10.

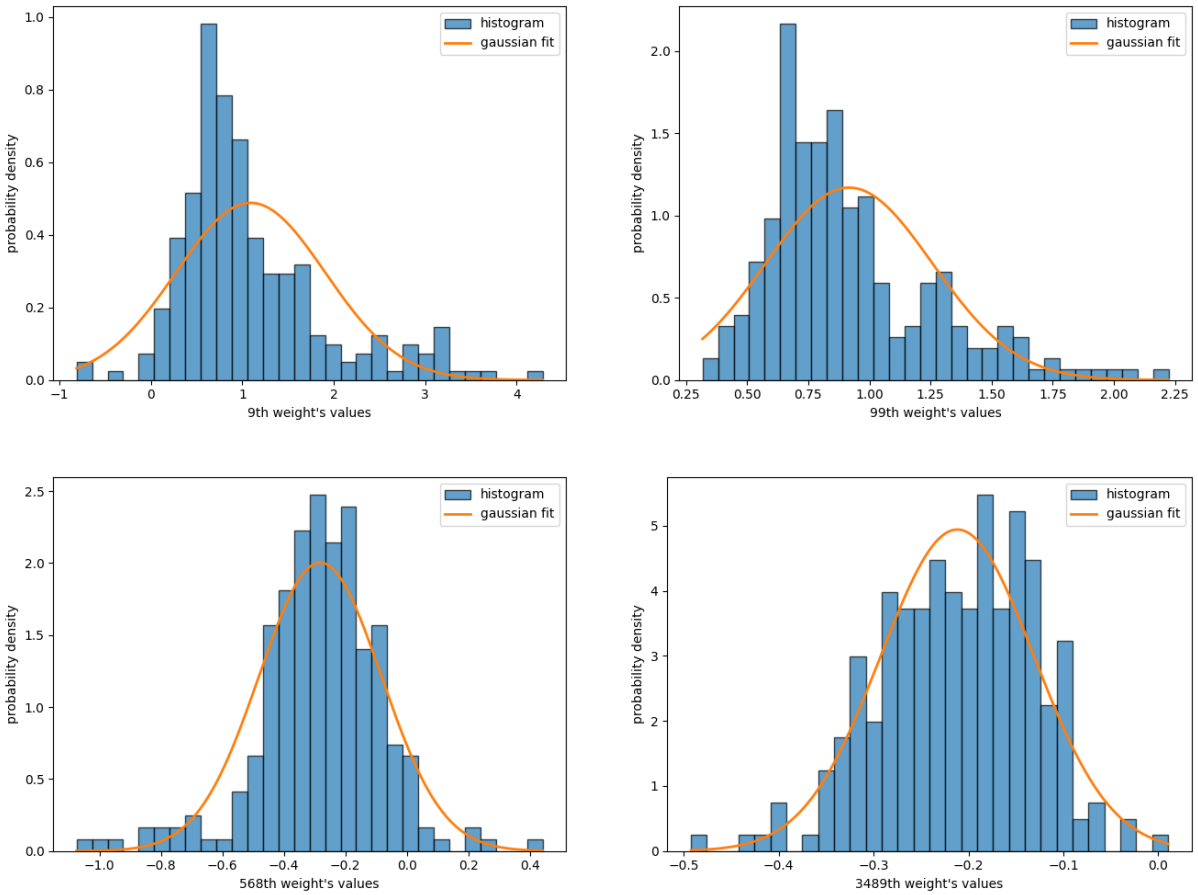

Figure 4.10: **Batch Normalisation**'s weights distribution analysis. Each histogram shows the distribution of the values of one Batch Normalisation's weight of the ResNet across the shadow models. We use 240 different shadow models, each trained on 90,000 samples from CINIC-10.

We observe that the parameter distributions become increasingly Gaussian towards the end of the architecture for both types of parameters: the *filter* weights become less heavy-tailed, while the *Batch Normalisation* parameters become less skewed across layers. This increasing Gaussianity helps explain why the `last_layer` Laplace approximation is particularly effective despite relying on only a small fraction of the model parameters. It also provides insight into why copying the Batch Normalisation parameters from the reference model for SWAG performs better than sampling them.

4.3 BNN approximations Analysis

We show here the analysis of our Bayesian approximations, using the metrics described in Section 3.4:

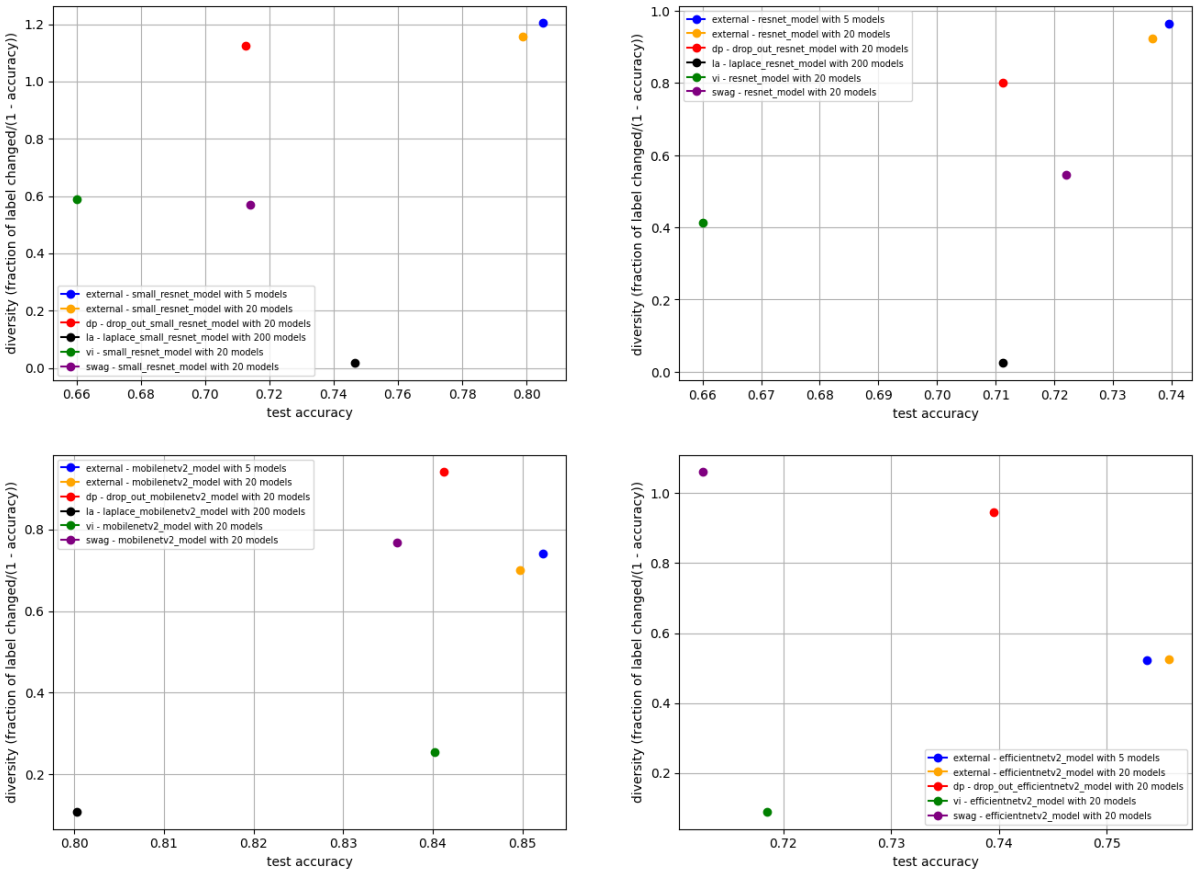

Figure 4.11: The diversity plots against the accuracy for our Bayesian methods and the shadow models. **external** indicates the shadow models, **dp** indicates Monte Carlo Dropout, and **VI** indicates Regularized Variational Inference.

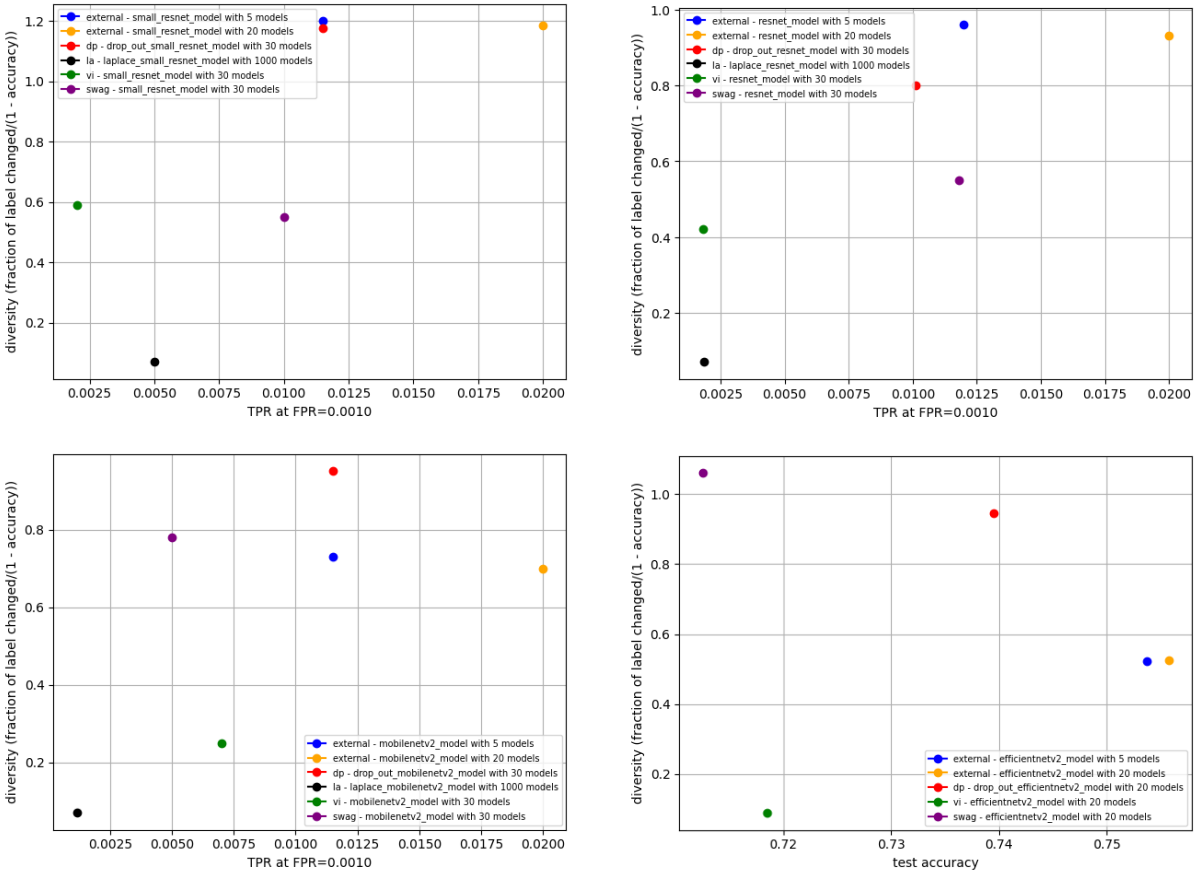

Figure 4.12: Diversity plots against **TPR** for our Bayesian methods and the shadow models. We compare the diversity values of our models with the **TPR** at **FPR**=10^{-3}, as a proxy measure of attack performance. **external** indicates the shadow models, **dp** indicates Monte Carlo Dropout, and **VI** indicates Regularized Variational Inference.

Where we use the *Diversity* metric which compares the average number of different predicted labels between the models, as explained in Subsection 3.4.3.
We see that both **diversity** and **accuracy** can be strong predictors of attack performance, with high-diversity models like the Monte Carlo Dropout and SWAG performing better in MIAs. This is especially true for the ResNet and small ResNet, where both methods are close in TPR value, but also for the MobileNetv2, where there is a wider distance between the two that can be explained by the difference in accuracy. We also see models, like Laplace, performing usually well in MIAs but still having a very low diversity. This can be explained by the accuracy, which is higher for Laplace for the ResNet and the small ResNet, when it performs the best.
Still, in general, the diversity values of the Bayesian models depend heavily on the model and the dataset used, apart from the specific method.

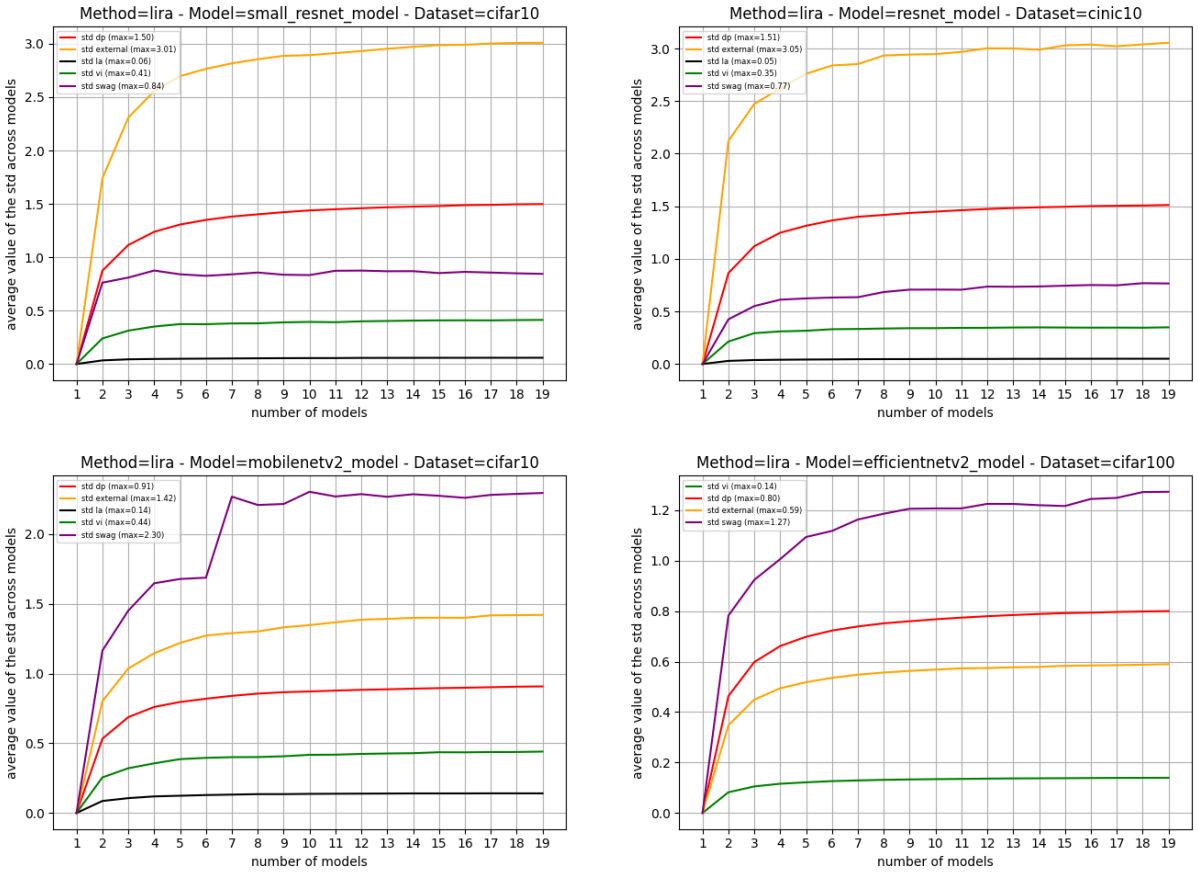

Figure 4.13: Variance plots for our Bayesian methods and the shadow models. **external** indicates the shadow models, **dp** indicates Monte Carlo Dropout, and **VI** indicates Regularized Variational Inference.

Here we observe similar results as before, with the best performing methods, Monte Carlo and SWAG, having the most varied range of outputs, while Variational Inference and Laplace have the least.
We see an interesting spike for SWAG in MobileNetv2, where it is more diverse even than the shadow models. This is explained by the presence of "high-loss" models sampled by the Gaussian which distorts the distribution and leads to a loss in accuracy.
We also see that for the EfficientNetV2 the Monte Carlo Dropout and SWAG models actually have larger variances as the number of models increases. This can be connected to their higher diversity in the previous plots.

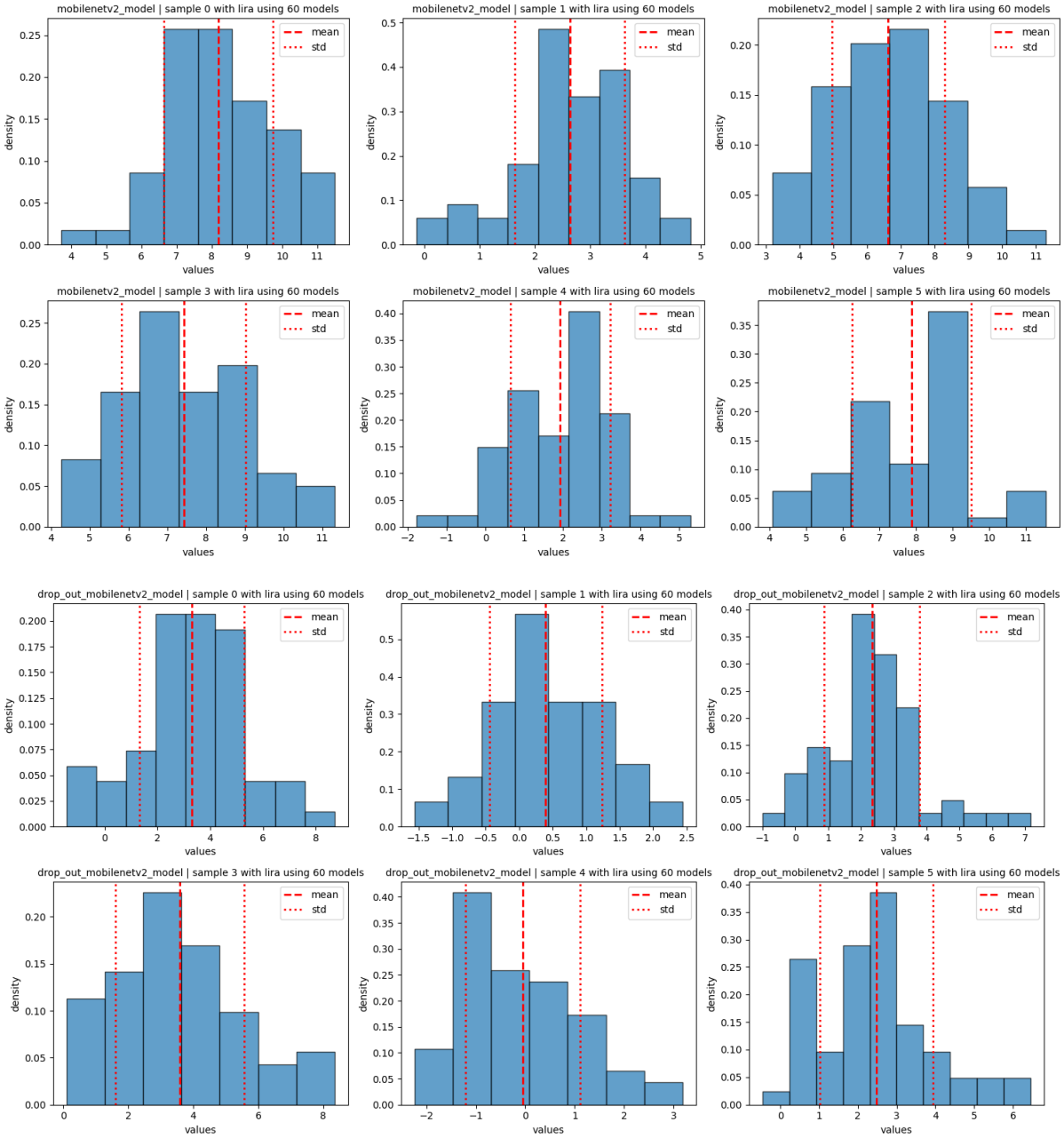

Figure 4.14: Distribution of the ϕ-values for a few sample data points across the shadow models (top) and the Monte Carlo Dropout iterations (bottom) using MobileNetV2. While the distribution shapes vary between the two configurations, the Monte Carlo Dropout distributions do not appear less Gaussian than those of the shadow models.

Above, we also compared the distribution of the ϕ-values obtained from the shadow models and from the Monte Carlo Dropout iterations to test whether the Gaussianity assumption, based on the experimental evidence reported in the original LiRA paper [3], also holds for our Bayesian models. This assumption appears to hold for Monte Carlo Dropout, which nonetheless performs poorly under LiRA. Therefore, Gaussianity alone does not provide a suitable explanation for why our Bayesian models struggle with LiRA.

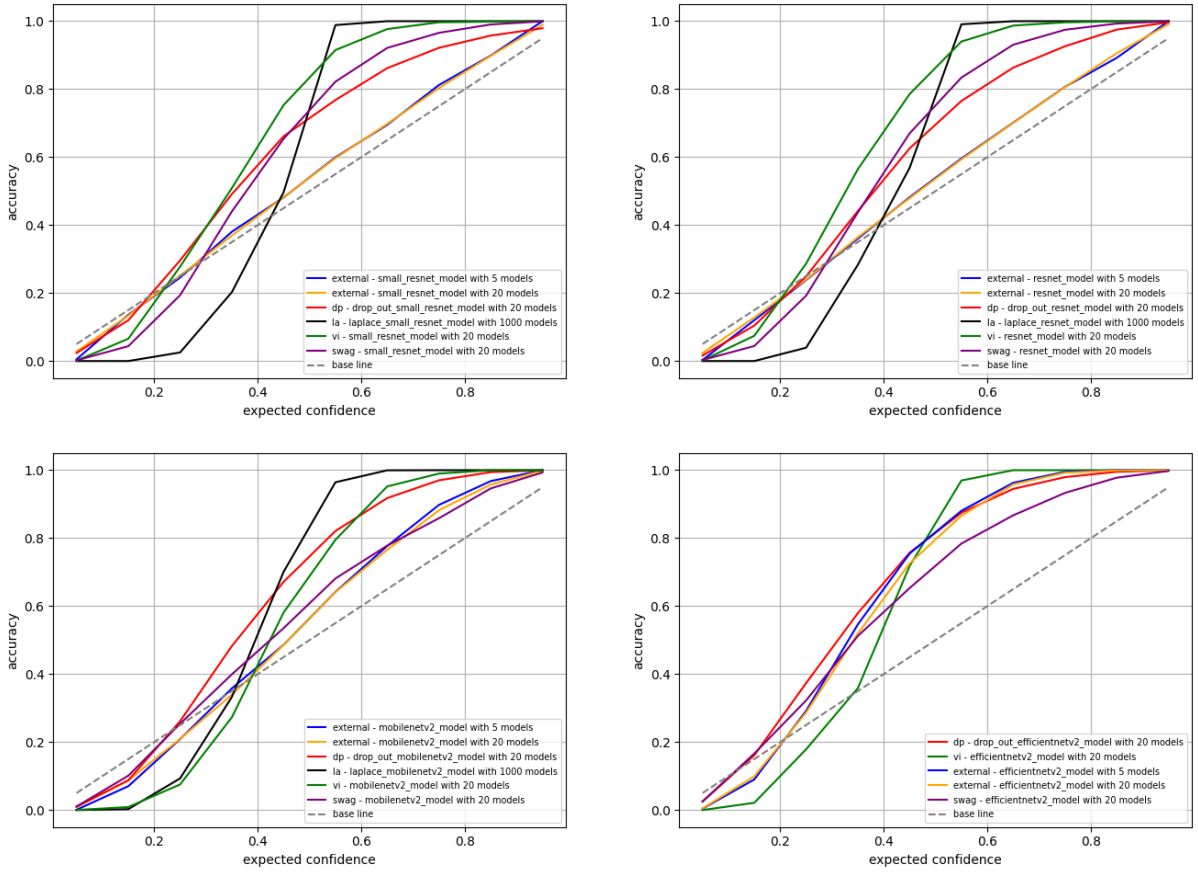

Figure 4.15: Calibration plots between the models. The vertical axis measures how each data point's output varies on average between different models, or between different iterations of the same Bayesian model. **external** indicates the shadow models, **dp** indicates Monte Carlo Dropout, and **VI** indicates Regularized Variational Inference.

We also see here that Monte Carlo Dropout and SWAG perform the best, as they are the closest to the shadow models calibration, while Laplace is the least calibrated one. This is expected since, as shadow models aim to replicate the target model behaviour, having more similar features ([3]), such as calibration, will lead to better performance. Therefore, as we see in all the figures, the Monte Carlo Dropout is the strategy closest to the shadow models in behaviour. It has the largest diversity between all the strategies, being comparable to the shadow models, and is the best calibrated BNN, even if it does not perform as well as the shadow models in all the metrics.

5
Discussion

5.1 Result Analysis

From the experimental results, we can conclude that Monte Carlo Dropout is a strong method for membership inference attacks (MIAs). Across our experiments, it consistently achieved better performance than the 3-shadow-model attacks, and in some cases even outperformed the 5-shadow-model attacks. SWAG is another interesting alternative: it is substantially cheaper than Monte Carlo Dropout and performs well on the ResNet-based models. However, it appears to be more sensitive to model quality in the other settings, where the accuracy of the sampled models is lower, for instance 69% compared to 76% for EfficientNetV2. This suggests that, for SWAG, predictive accuracy is an important factor in determining attack performance.
For Monte Carlo Dropout, the relationship between accuracy and attack strength appears to be less strong. In particular, it performs better in the small ResNet setting even though the corresponding models are more than 8% less accurate than the shadow models, whereas in the ResNet case the accuracy gap is smaller. This indicates that the interaction between diversity, accuracy, and attack performance is not fully captured by a single metric. In practice, the usefulness of these metrics depends strongly on the model family and the training setup.
Overall, our results suggest that diversity, accuracy, and related measures can serve as useful indicators of MIA performance. However, they are more reliable as comparative tools when evaluating multiple Bayesian models of the same type, trained on the same dataset and under similar conditions. In that setting, we can reduce the influence of confounding factors, making it easier to isolate the effect of the Bayesian approximation itself.

We summarize the main tendencies observed in our experiments in the following Table:

Method	**BASE attack**	**LiRA attack**	**Diversity**	**Accuracy**	**Cost**
MC Dropout	**Good**	Low	**High**	Medium	Medium
SWAG	Good	Low	Medium	Medium	**Low**
Laplace	Moderate	**Moderate**	Low	Medium-High	**Low**
Var. Inference	Moderate	Low	Medium	Variable	High
∼5 shadow models	Good	Good	High	High	Very High

5.2 Limitations and Future Work

Our experiments focused on improving MIA performance through Bayesian approximations in a controlled offline setting. This choice introduces several limitations. First, we only studied the offline scenario, whereas online LiRA and related online variants generally perform better. This means that the attacks considered here are not the strongest possible version of each method, but rather the most practical ones under our computational constraints.

Second, our setting assumes full knowledge of the target model architecture and training procedure. In that sense, the shadow-model approach benefits from being able to reproduce the target model very closely. By contrast, our Bayesian approximations may use different architectures or training procedures, which makes the comparison less favourable to them in some cases. This should be taken into consideration when interpreting the results.

We also used the same set of shadow models across all targets. While this may initially appear biased, it is a reasonable method, since the results are averaged over multiple target models. Still, future work could explore whether target-specific shadow sets improve the fidelity of the comparison, especially if the target models differ more strongly from one another.

Another interesting research direction would be to explain the performance gap between LiRA and BASE in our Bayesian approximations. As shown in Figure 4.14, Monte Carlo Dropout appears to satisfy the Gaussianity assumption for the ϕ-values discussed in [3], which rules out this explanation. The difference may instead be related to the behavior of Bayesian models, which, as suggested by the metric, differ substantially from the shadow models, despite the latter being trained identically to the target models. Testing shadow models trained with a procedure different from that of the target model could therefore help clarify this discrepancy.

A more conceptual limitation concerns hyperparameter selection. Methods such as Monte Carlo Dropout and other Bayesian approximations offer many degrees of freedom: the dropout probability, the layers where dropout is applied, the model size, and the location and size of any additional layers all affect the final outcome. In practice, this makes it difficult to choose a configuration before training. Metrics such as diversity and accuracy are helpful for assessing the potential MIA performance, but they can only be measured after training. This creates a practical issue: one may need to train several candidate models before knowing which one is actually useful, which would defeat the purpose of using BNN for reducing the cost.

One possible direction to address this issue is to incorporate diversity directly into the loss function, for example by adding a term that encourages high diversity while preserving accuracy. Such a modification could guide the training procedure toward models that are both accurate and more suitable for MIA. In principle, this idea could already be applied to the Bayesian approximations considered in this work.

Another promising direction would be to explore *mixtures of Bayesian models*. Instead of relying on a single approximation, one could train two or three different Bayesian models, either of the same type or of different types, and combine their outputs. This might allow the attack to recover part of the benefit of larger shadow-model ensembles, such as attacks with 30 or 60 shadow models, while keeping the computational cost lower.

Finally, our experiments were restricted to image-classification MIAs. A natural extension would be to test these methods in more demanding domains, such as large language models or multimodal systems, where shadow-model training is significantly more expensive. In

such settings, where state-of-the-arts attacks often struggle [30] [31], efficient Bayesian approximations could become especially valuable, since they may provide a practical way to strengthen MIAs without the prohibitive cost of large shadow ensembles.

6 Conclusion

We showed in this thesis that Bayesian Neural Network approximations can offer a practical alternative to large shadow-model ensembles for Membership Inference Attacks. This is particularly relevant because the strongest attacks are often too costly to apply when many shadow models are required, especially in settings where training even a single additional model becomes expensive. In such cases, Bayesian approximations provide a way to recover much of the attack performance while keeping the computational cost more manageable.

We evaluated LiRA and BASE attacks using four different Bayesian approximation methods: Stochastic Weight Averaging Gaussian (SWAG), Laplace approximation, Monte Carlo Dropout, and Regularized Variational Inference. Our results show that some Bayesian approximations can achieve performance comparable to a limited shadow-model ensemble, especially for the BASE attack. In particular, Monte Carlo Dropout, with only 50% more training epochs and a slightly larger model, consistently matches the performance of a 3-shadow-model attack and even outperforms the 5-shadow-model attack in some cases. SWAG also comes close to Monte Carlo Dropout in several experiments, offering a less reliable but slightly less expensive alternative.

We also found that metrics such as diversity and accuracy can help predict MIA performance, making them useful for filtering out weaker Bayesian models before running the attack. Nevertheless, their usefulness depends on the model family and the training setting.

Overall, the results suggest that Bayesian approximations are a viable way to reduce the cost of MIAs while retaining much of their effectiveness.

Bibliography

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

[2] Alexandre Sablayrolles, Matthijs Douze, Yann Ollivier, Cordelia Schmid, and Hervé Jégou. White-box vs black-box: Bayes optimal strategies for membership inference, 2019.

[3] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramèr. Membership inference attacks from first principles. *CoRR*, abs/2112.03570, 2021.

[4] Sajjad Zarifzadeh, Philippe Liu, and Reza Shokri. Low-cost high-power membership inference attacks, 2024.

[5] Marcus Lassila, Johan Östman, Khac-Hoang Ngo, and Alexandre Graell i Amat. Practical bayes-optimal membership inference attacks, 2025.

[6] Lorenzo Sucameli. Using Approximate Bayesian Neural Networks for Membership Inference. GitHub repository, June 2026. `https://github.com/lorenzo8612/Using-BNN-for-MIA-Master-Thesis`.

[7] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[8] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting, 2018.

[9] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization, 2017.

[10] Vitaly Feldman. Does learning require memorization? a short tale about a long tail, 2021.

[11] Shai Shalev-Shwartz, Ohad Shamir, Nathan Srebro, and Karthik Sridharan. Learnability, stability and uniform convergence. *Journal of Machine Learning Research*, 11(90):2635–2670, 2010.

[12] Cynthia Dwork. Differential privacy. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming*, pages 1–12, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[13] Erik Daxberger, Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, Matthias Bauer, and Philipp Hennig. Laplace redux–effortless Bayesian deep learning. In *NeurIPS*, 2021.

[14] Sungjun Lim, Jeyoon Yeom, Sooyon Kim, Hoyoon Byun, Jinho Kang, Yohan Jung, Jiyoung Jung, and Kyungwoo Song. Flat posterior does matter for bayesian model averaging, 2025.

[15] Wesley Maddox, Timur Garipov, Pavel Izmailov, Dmitry Vetrov, and Andrew Gordon Wilson. A simple baseline for bayesian uncertainty in deep learning, 2019.

[16] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization, 2019.

[17] Stephan Mandt, Matthew D. Hoffman, and David M. Blei. Stochastic gradient descent as approximate bayesian inference, 2018.

[18] Tristan Cinquin and Robert Bamler. Regularized kl-divergence for well-defined function-space variational inference in bayesian neural networks, 2025.

[19] 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*, 15(56):1929–1958, 2014.

[20] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning, 2016.

[21] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical Report TR-2009, University of Toronto, 2009.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[23] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[24] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[25] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks, 2019.

[26] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 807–814, Madison, WI, USA, 2010. Omnipress.

[27] Abien Fred Agarap. Deep learning using rectified linear units (relu), 2026.

[28] Mingxing Tan and Quoc V. Le. Efficientnetv2: Smaller models and faster training. *CoRR*, abs/2104.00298, 2021.

[29] Meet P. Vadera, Adam D. Cobb, Brian Jalaian, and Benjamin M. Marlin. Ursabench: Comprehensive benchmarking of approximate bayesian inference methods for deep neural networks, 2020.

[30] Matthieu Meeus, Igor Shilov, Shubham Jain, Manuel Faysse, Marek Rei, and Yves-Alexandre de Montjoye. Sok: Membership inference attacks on llms are rushing nowhere (and how to fix it), 2025.

[31] Jamie Hayes, Ilia Shumailov, Christopher A. Choquette-Choo, Matthew Jagielski, George Kaissis, Milad Nasr, Sahra Ghalebikesabi, Meenatchi Sundaram Mutu Selva Annamalai, Niloofar Mireshghallah, Igor Shilov, Matthieu Meeus, Yves-Alexandre de Montjoye, Katherine Lee, Franziska Boenisch, Adam Dziedzic, and A. Feder Cooper. Exploring the limits of strong membership inference attacks on large language models, 2026.