ON THE PRIVACY RISKS OF SPIKING NEURAL NETWORKS: A MEMBERSHIP INFERENCE ANALYSIS

Junyi Guan,* Abhijith Sharma,* Chong Tian, & Salem Lahlou
Machine Learning Department, Mohamed Bin Zayed University of Artificial Intelligence, UAE

ABSTRACT

Spiking Neural Networks (SNNs) are increasingly explored for their energy efficiency and robustness in real-world applications, yet their privacy risks remain largely unexamined. In this work, we investigate the susceptibility of SNNs to Membership Inference Attacks (MIAs)—a major privacy threat where an adversary attempts to determine whether a given sample was part of the training dataset. While prior work suggests that SNNs may offer inherent robustness due to their discrete, event-driven nature, we find that its resilience diminishes as latency (T) increases. Furthermore, we introduce an input dropout strategy under black box setting, that significantly enhances membership inference in SNNs. Our findings challenge the assumption that SNNs are inherently more secure, and even though they are expected to be better, our results reveal that SNNs exhibit privacy vulnerabilities that are equally comparable to ANNs. Our code is available at `https://github.com/sharmaabhijith/MIA_SNN`

1 INTRODUCTION

Spiking Neural Networks (SNNs) are a class of neural networks that emulate the discrete, event-driven processing of biological neurons. Unlike artificial neural networks (ANNs), which rely on continuous activations, SNNs communicate through discrete spikes, making them energy-efficient, noise-resistant, and suited for temporal pattern recognition (Tavanaei et al., 2019; Kasabov, 2014; Yan et al., 2024; Nunes et al., 2022; Nagarajan et al., 2022). These advantages have led to their growing adoption in computer vision and speech processing applications(Yamazaki et al., 2022).

However, as SNNs gain traction, security, and privacy concerns must also be addressed (Jobin et al., 2019; Liu et al., 2021; Rigaki & Garcia, 2023). One of the most prominent threats is the Membership Inference Attack (MIA) (Shokri et al., 2017), where an adversary attempts to determine whether a specific data sample was part of a model's training set. Such attacks pose significant privacy risks, particularly in sensitive applications like healthcare, where leaking training data could compromise confidentiality. In addition, regulating organizations can leverage MIA as a metric to conduct pre-deployment audits of AI models to avoid leakage of sensitive data. MIAs have been widely studied in conventional neural networks, but their impact on SNNs remains largely unexplored.

A recent study by Li et al. (2024) suggests that SNNs' discrete spike-based outputs may offer some level of protection against MIAs, but current evaluations are limited and often rely on metrics that may not fully capture the unique characteristics of SNNs (see Appendix A). This raises a critical question: Are SNNs genuinely more resilient to MIAs, or do they exhibit vulnerabilities similar to conventional neural networks? In this work, we investigate SNNs' susceptibility to membership inference attacks to better understand their privacy risks and address gaps in existing research.

2 METHODOLOGY

Our main attack setup follows prior MIA research (Zarifzadeh et al., 2024; Carlini et al., 2022). We start with a dataset D, and then utilize a data splitting explained in Appendix D to effectively train all the reference models to carry out MIA. In this setting, attackers know the target model's architecture and can use the whole D to train reference models. They can also know the model's softmax outputs. For auditing, we use the entire dataset to maintain unbiased results.

*Equal contribution.

2.1 MIA Attack Methods: Background and Modifications

In this section, we describe **Attack-P, Attack-R, and RMIA** (Ye et al., 2022; Zarifzadeh et al., 2024) and adapt them to our experimental setting, as summarized in Table 2.1. The primary aim is to utilize the whole dataset to audit and also to make the attack stronger and faster. Theorem 2, supported by Remark 1 establishes the equivalence of our modified setting with an empirical estimation of RMIA, allowing an approximation of the strongest online RMIA version without executing it online. We define the parameters for the following attack methods as follows. Let θ be the target model and θ' a reference model. The audited sample is x, while z is drawn from the dataset D of size N. The model's confidence output is $\Pr(\cdot|\theta)$. Reference models trained with and without x are denoted as θ'_x and $\theta'_{\bar{x}}$, respectively. The confidence of any model m, $\Pr(d|m)$, is given by the softmax score of the true label of input d.

Attack	**Attack-P**	**Attack-R**	**RMIA**
Original	$\Pr_z\left(\frac{\Pr(x\|\theta)}{\Pr(z\|\theta)} \geq 1\right)$	$\Pr_{\theta'}\left(\frac{\Pr(x\|\theta)}{\Pr(x\|\theta')} \geq 1\right)$	$\Pr_z\left(\frac{\Pr(\theta\|x)}{\Pr(\theta\|z)} \geq 1\right)$
Modified	$\Pr(x\|\theta)$	$\frac{1}{2n}\sum_{\theta'} \mathbf{1}\left(\Pr(x\|\theta) \geq \Pr(x\|\theta')\right)$	$\frac{\Pr(x\|\theta)}{\frac{1}{2n}\left(\sum_{\theta'_x} \Pr(x\|\theta'_x) + \sum_{\theta'_{\bar{x}}} \Pr(x\|\theta'_{\bar{x}})\right)}$

Table 2.1: Attack formulations for computational efficiency by removing dependence on z.

The three attack methods outlined above exploit different assumptions about model confidence to infer membership. **Attack-P** assumes that if a sample x is part of the target model's (θ) training data, the model's confidence on x is likely higher than on an arbitrary sample z. While the weakest among the three, it provides insights into the model's confidence distribution. **Attack-R** strengthens this approach by comparing θ's confidence on x with that of reference models (θ'), assuming that the target model assigns higher confidence to samples it has seen during training. **RMIA** is the strongest attack, exploiting the assumption that if θ is trained on x, the likelihood of θ given x is higher than for any unrelated sample z. Together, these attacks provide a progressive understanding of how a model's confidence can reveal membership information.

2.2 Motivation

Numerous existing works have demonstrated the robustness of SNNs against adversaries due to their discrete spiking behavior (Kim et al., 2022; Sharmin et al., 2019). These references and many others are discussed in Appendix A. Similarly, we expect MIAs to be ineffective, especially on low-latency Spiking SNNs (also validated by our results of Section 3). It is not because SNNs fail to remember the training data, but because their discrete representations cause member and non-member samples to overlap. Interestingly, we observe that MIA and the out-of-distribution (OOD) detection task share the same principle: the model exhibits higher confidence on in-distribution data (members) compared to the out-of-distribution data (non-members) (Hendrycks & Gimpel, 2016). Fundamentally, both methods are related to epistemic uncertainty, which arises due to lack of knowledge or data, and can be reduced with additional information or improved modeling (Der Kiureghian & Ditlevsen, 2009; Lahlou et al., 2023).

In practice, dropout is a widely used and effective technique for regularization (Hinton, 2012), uncertainty estimation (Gal & Ghahramani, 2016), (Sun et al., 2023), and OOD detection (Hendrycks & Gimpel, 2016; Nguyen et al., 2022). Techniques like Monte Carlo (MC)-Dropout (Gal & Ghahramani, 2016) introduce stochasticity during inference, with variations in output softmax probabilities approximating model uncertainty. However, in the context of MIA, directly applying MC-Dropout is not feasible, as it requires access to model parameters, which is unavailable in a black-box setting. To address this, we propose using input dropout, as an approximation of MC-Dropout (in the first layer), introducing stochasticity to the predictions without requiring access to the model's internals. As evidenced by our experiments, low-latency SNNs often struggle to differentiate subtle distinctions between member and non-member data. By perturbing the input through dropout, we aim to reduce the confidence of non-member data while maintaining the relative confidence of member data, thus enhancing the separation and improving the MIA's effectiveness.

2.3 DROPOUT-ENHANCED MIA METHOD: INTRODUCING PREDICTION STOCHASTICITY

This section develops an intuition for the similarities between weight and input dropout. Consider a simple single-layered model given by $f_W : \mathbb{R}^d \to \mathbb{R}, \quad f_W(X) = \varphi(W^T \cdot X)$, where $W \in \mathbb{R}^d$ is the weight vector, $X \in \mathbb{R}^d$ represents the input and φ is the non-linear activation. During dropout, a Bernoulli mask $M \in \mathbb{R}^d$ with probability $(1-p)$ modifies the input and weight as $\tilde{X} = M \odot X$ and $\tilde{W} = M \odot W$, where $\odot$ denotes the Hadamard product. The equivalence between the expected outputs of input dropout $\mathbb{E}_M[f_W(\tilde{X})]$ and weight dropout $\mathbb{E}_M[f_{\tilde{W}}(X)]$ is shown in Equation 1:

$$\begin{aligned} \mathbb{E}_M[f_W(\tilde{X})] &= \mathbb{E}_M\left[\varphi\left(W^T \cdot (M \odot X)\right)\right] = \mathbb{E}_M\left[\varphi\left(\sum_i w_i(M_i x_i)\right)\right] \\ &= \mathbb{E}_M\left[\varphi\left(\sum_i (M_i w_i) x_i\right)\right] = \mathbb{E}_M\left[\varphi\left((M \odot W)^T \cdot X\right)\right] = \mathbb{E}_M[f_{\tilde{W}}(X)] \end{aligned} \tag{1}$$

However, in case of multi-layered networks, weight dropout is applied to each layer individually, which differentiates it from input dropout. Nevertheless, both methods are capable of introducing stochasticity into the model's predictions. Hence, based on our hypothesis, instead of modifying the internal layers, we randomly drop parts of the input data. For each forward pass $i \in \{1, \ldots, N\}$, a binary mask with elements sampled from the Bernoulli (p) mask inputs: $\tilde{X} = X \odot M$ is fed into the model θ. We then compute the confidence $\Pr(\tilde{X} \mid \theta)$ for each pass, and the final estimate is obtained by averaging over n passes:

$$\Pr(X \mid \theta) = \frac{1}{N} \sum_{i=1}^{N} \Pr(\tilde{X}_i \mid \theta).$$

This method enhances existing MIA techniques by incorporating the mean confidence value, which can be optimized for different attack strategies. When selecting the hyperparameters, we use the following approach: we randomly consider one of the reference models as the target model and use the other reference models to attack it. Then, we find the optimal hyperparameters p and N by grid search to maximize the attack's AUC. Finally, we use the hyperparameters selected from the reference models to attack the original target model.

Some MIA methods also incorporate input noise (e.g., Carlini et al. (2022), Zarifzadeh et al. (2024)) However, these methods do not directly estimate the confidence term $\Pr(X \mid \theta)$; they rely on other components of the MIA metric. Our focus here is not to compare various uncertainty estimation methods but rather to evaluate whether input dropout can significantly enhance MIA performance on SNNs. Additionally, by estimating the confidence term with input dropout, we can extend this approach to all attack methods that require confidence estimation.

3 EXPERIMENTS AND RESULTS

The detailed outline of our experimental setup is explained in Appendix F. Hence, in this section, we primarily discuss the experiment results to analyze the membership privacy risk in SNNs.

Without-Dropout. Tables 3.1 and Figure 1 **Left** present the performance of trained ANNs and their corresponding SNNs ($T = 1, 2, 4$) on CIFAR-10 and CIFAR-100 against MIAs. Our results can be summarized in three key points. First, RMIA is the most effective attack, followed by Attack-R and then Attack-P, which aligns with previous findings Zarifzadeh et al., 2024. Second, for SNNs, all attack metrics (AUC and TPR at low FPR) increase with T, indicating their progressive vulnerability to MIAs as their latency increases. Finally, ANNs consistently exhibit the highest attack metric scores, demonstrating that they are inherently more susceptible to MIAs than SNNs.

The robustness of SNNs at lower T stems from their inherent discrete signal processing. Unlike ANNs, which handle continuous inputs and outputs, SNNs encode inputs as binary spike sequences over T time steps. When T is small $T = 1$, each neuron outputs a single binary value, significantly limiting expressiveness. This leads to overlapping confidence distributions for member and non-member data (Fig. 1, **Middle**), reducing MIA effectiveness. As T increases, SNNs capture more details, improving separation (Fig. 3 of Appendix G) and more easily be attacked. Note that there exists an accuracy-privacy trade-off: lower T reduces accuracy slightly but improves privacy, while higher T improves accuracy at the cost of privacy (see Table F.2).

Table 3.1: MIA Results with SNN (left) and ANN (right) ResNet18 on CIFAR-10 and CIFAR-100.

Drop Out		Attack	SNN (T=1) AUC	SNN (T=1) 0.1%	SNN (T=1) 1%	SNN (T=2) AUC	SNN (T=2) 0.1%	SNN (T=2) 1%	SNN (T=4) AUC	SNN (T=4) 0.1%	SNN (T=4) 1%
CIFAR-10	False	Attack-P	54.58	0.08	0.87	54.90	0.00	0.72	55.34	0.00	0.00
		Attack-R	57.59	0.00	0.00	58.22	0.00	0.00	59.14	0.00	0.00
		RMIA	59.89	0.84	3.82	60.83	1.00	4.43	62.61	1.09	5.01
	True	Attack-P	54.64	0.06	0.84	54.75	0.05	0.83	55.08	0.00	0.70
		Attack-R	59.54	0.00	0.00	59.99	0.00	0.00	60.21	0.00	0.00
		RMIA	**63.84**	**1.86**	**6.34**	**64.28**	**2.03**	**6.23**	**64.65**	**1.94**	**6.61**
CIFAR-100	False	Attack-P	59.47	0.14	1.24	61.67	0.12	1.13	64.85	0.12	1.13
		Attack-R	66.25	0.00	0.00	69.58	0.00	0.00	73.23	0.00	0.00
		RMIA	69.06	1.11	6.29	72.81	1.87	8.24	77.57	2.61	12.01
	True	Attack-P	59.70	0.14	1.20	61.85	0.13	1.30	64.82	0.12	1.21
		Attack-R	71.69	0.00	0.00	73.64	0.00	0.00	75.47	0.00	0.00
		RMIA	**75.82**	**2.80**	**11.14**	**78.31**	**3.55**	**13.06**	**80.76**	**4.36**	**16.16**

	Attack	ANN AUC	ANN 0.1%	ANN 1%
CIFAR-10	Attack-P	56.61	0.00	0.00
	Attack-R	59.49	0.00	0.00
	RMIA	**63.84**	**2.84**	**5.82**
CIFAR-100	Attack-P	68.75	0.10	1.22
	Attack-R	76.93	0.00	0.00
	RMIA	**82.51**	**6.75**	**20.25**

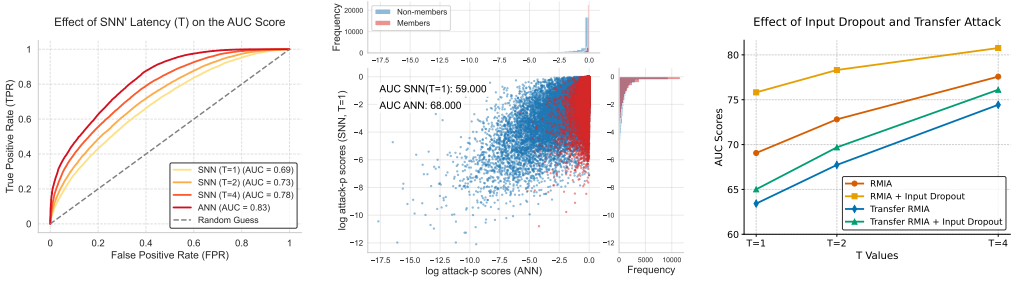

Figure 1: **Left**: ROC curves and AUC values for all-latency SNNs and an ANN under RMIA on CIFAR-100. **Middle**: A scatter plot showing the log attack p-scores and distributions for member and non-member data on SNN (T=1) and ANN for CIFAR-100. **Right**: A line graph illustrating RMIA performance for all-latency SNNs under different conditions: with or without input dropout, and using or not using a transfer model (ANN).

Impact of Input Dropout on Attack Performance. To enhance the attack on SNNs, we apply the input dropout method. For both CIFAR-10 and CIFAR-100, our experiments demonstrate that incorporating input dropout significantly increases the overall attack performance. This improvement is observed when the adversary is not aware of the target SNN's architecture and still using ANN as a reference models (Fig. 1, **Right**). Furthermore, similar enhancements in attack performance are also evident for directly trained SNNs, particularly for low-latency configurations (Table F.4).

Attacking SNNs Using an ANN as the Reference Model. In the scenario that the adversary employs an ANN as the reference model to attack a target SNN. Our experiments show that such transfer model attacks are significantly less effective(Fig. 1, **Right**). Although SNNs and ANNs may share the same architecture (e.g., ResNet-18), their confidence distributions for member and non-member samples differ substantially (Fig. 1, **Middle**). Moreover, as the SNN's latency T increases, its confidence distributions for member and non-member data will become more and more similar than those of the ANN (Fig.3), thereby reducing the attacker's difficulty.

4 Conclusion

Our study shows that SNNs become increasingly vulnerable as latency increases. Furthermore, our proposed input dropout technique can significantly increase the risk of MIA, even when the adversary uses an ANN as a reference model. This effect also persists in direct-trained, low-latency SNNs. Therefore, we argue that the assumption of SNNs' inherent privacy protection should be critically evaluated, highlighting the need to enhance privacy assurance in SNNs.

5 Acknowledgements

The authors express their sincere gratitude to Velibor Bojković, Ruben Solozabal, and Xingyu Qu for their valuable insights and feedback, which greatly contributed to the improvement of this work.

REFERENCES

Guillaume Bellec, Darjan Salaj, Anand Subramoney, Robert Legenstein, and Wolfgang Maass. Long short-term memory and learning-to-learn in networks of spiking neurons. *Advances in neural information processing systems*, 31, 2018.

Velibor Bojkovic, Srinivas Anumasa, Giulia De Masi, Bin Gu, and Huan Xiong. Data driven threshold and potential initialization for spiking neural networks. In *International Conference on Artificial Intelligence and Statistics*, pp. 4771–4779. PMLR, 2024.

Maxence Bouvier, Alexandre Valentian, Thomas Mesquida, Francois Rummens, Marina Reyboz, Elisa Vianello, and Edith Beigne. Spiking neural networks hardware implementations and challenges: A survey. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 15(2): 1–35, 2019.

Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. Membership inference attacks from first principles. In *2022 IEEE Symposium on Security and Privacy (SP)*, pp. 1897–1914. IEEE, 2022.

Manon Dampfhoffer, Thomas Mesquida, Alexandre Valentian, and Lorena Anghel. Backpropagation-based learning techniques for deep spiking neural networks: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

Shikuang Deng, Yuhang Li, Shanghang Zhang, and Shi Gu. Temporal efficient training of spiking neural network via gradient re-weighting. *arXiv preprint arXiv:2202.11946*, 2022.

Armen Der Kiureghian and Ove Ditlevsen. Aleatory or epistemic? does it matter? *Structural safety*, 31(2):105–112, 2009.

Peter U Diehl and Matthew Cook. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in computational neuroscience*, 9:99, 2015.

Rida El-Allami, Alberto Marchisio, Muhammad Shafique, and Ihsen Alouani. Securing deep spiking neural networks against adversarial attacks through inherent structural parameters. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 774–779. IEEE, 2021.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059. PMLR, 2016.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

GE Hinton. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

Hongsheng Hu, Zoran Salcic, Lichao Sun, Gillian Dobbie, Philip S Yu, and Xuyun Zhang. Membership inference attacks on machine learning: A survey. *ACM Computing Surveys (CSUR)*, 54 (11s):1–37, 2022.

Li Hu, Anli Yan, Hongyang Yan, Jin Li, Teng Huang, Yingying Zhang, Changyu Dong, and Chunsheng Yang. Defenses to membership inference attacks: A survey. *ACM Computing Surveys*, 56 (4):1–34, 2023.

Anna Jobin, Marcello Ienca, and Effy Vayena. The global landscape of ai ethics guidelines. *Nature machine intelligence*, 1(9):389–399, 2019.

Nikola K Kasabov. Neucube: A spiking neural network architecture for mapping, learning and understanding of spatio-temporal brain data. *Neural Networks*, 52:62–76, 2014.

Youngeun Kim, Yeshwanth Venkatesha, and Priyadarshini Panda. Privatesnn: privacy-preserving spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 1192–1200, 2022.

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

Salem Lahlou, Moksh Jain, Hadi Nekoei, Victor I Butoi, Paul Bertin, Jarrid Rector-Brooks, Maksym Korablyov, and Yoshua Bengio. DEUP: Direct epistemic uncertainty prediction. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=eGLdVRvvfQ. Expert Certification.

Jiaxin Li, Gorka Abad, Stjepan Picek, and Mauro Conti. Membership privacy evaluation in deep spiking neural networks. *arXiv preprint arXiv:2409.19413*, 2024.

Bo Liu, Ming Ding, Sina Shaham, Wenny Rahayu, Farhad Farokhi, and Zihuai Lin. When machine learning meets privacy: A survey and outlook. *ACM Computing Surveys (CSUR)*, 54(2):1–36, 2021.

Qingyan Meng, Mingqing Xiao, Shen Yan, Yisen Wang, Zhouchen Lin, and Zhi-Quan Luo. Training high-performance low-latency spiking neural networks by differentiation on spike representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12444–12453, 2022.

Bhaskar Mukhoty, Velibor Bojkovic, William de Vazelhes, Xiaohan Zhao, Giulia De Masi, Huan Xiong, and Bin Gu. Direct training of snn using local zeroth order method. *Advances in Neural Information Processing Systems*, 36:18994–19014, 2023.

Karthikeyan Nagarajan, Junde Li, Sina Sayyah Ensan, Mohammad Nasim Imtiaz Khan, Sachhidh Kannan, and Swaroop Ghosh. Analysis of power-oriented fault injection attacks on spiking neural networks. In *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 861–866. IEEE, 2022.

Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019.

Andre T Nguyen, Fred Lu, Gary Lopez Munoz, Edward Raff, Charles Nicholas, and James Holt. Out of distribution data detection using dropout bayesian neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 2022.

Osamu Nomura, Yusuke Sakemi, Takeo Hosomi, and Takashi Morie. Robustness of spiking neural networks based on time-to-first-spike encoding against adversarial attacks. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 69(9):3640–3644, 2022.

Joao D Nunes, Marcelo Carvalho, Diogo Carneiro, and Jaime S Cardoso. Spiking neural networks: A survey. *IEEE Access*, 10:60738–60764, 2022.

Nitin Rathi, Gopalakrishnan Srinivasan, Priyadarshini Panda, and Kaushik Roy. Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation. *arXiv preprint arXiv:2005.01807*, 2020.

Maria Rigaki and Sebastian Garcia. A survey of privacy attacks in machine learning. *ACM Computing Surveys*, 56(4):1–34, 2023.

Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in neuroscience*, 11:682, 2017.

Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. *arXiv preprint arXiv:1806.01246*, 2018.

Abhronil Sengupta, Yuting Ye, Robert Wang, Chiao Liu, and Kaushik Roy. Going deeper in spiking neural networks: Vgg and residual architectures. *Frontiers in neuroscience*, 13:95, 2019.

Saima Sharmin, Priyadarshini Panda, Syed Shakib Sarwar, Chankyu Lee, Wachirawit Ponghiran, and Kaushik Roy. A comprehensive analysis on adversarial robustness of spiking neural networks. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2019.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pp. 3–18. IEEE, 2017.

Tao Sun, Bojian Yin, and Sander Bohté. Efficient uncertainty estimation in spiking neural networks via mc-dropout. In *International Conference on Artificial Neural Networks*, pp. 393–406. Springer, 2023.

Amirhossein Tavanaei, Masoud Ghodrati, Saeed Reza Kheradpisheh, Timothée Masquelier, and Anthony Maida. Deep learning in spiking neural networks. *Neural networks*, 111:47–63, 2019.

Kashu Yamazaki, Viet-Khoa Vo-Ho, Darshan Bulsara, and Ngan Le. Spiking neural networks and their applications: A review. *Brain Sciences*, 12(7):863, 2022.

Zhanglu Yan, Zhenyu Bai, and Weng-Fai Wong. Reconsidering the energy efficiency of spiking neural networks. *arXiv preprint arXiv:2409.08290*, 2024.

Jiayuan Ye, Aadyaa Maddi, Sasi Kumar Murakonda, Vincent Bindschaedler, and Reza Shokri. Enhanced membership inference attacks against machine learning models. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pp. 3093–3106, 2022.

Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st computer security foundations symposium (CSF)*, pp. 268–282. IEEE, 2018.

Sajjad Zarifzadeh, Philippe Liu, and Reza Shokri. Low-cost high-power membership inference attacks. In *Forty-first International Conference on Machine Learning*, 2024.

A Related work

A.1 Spiking Neural Networks (SNNs)

Designing and training SNNs is challenging due to their sensitivity to hyperparameters such as membrane threshold and synaptic latency, both of which significantly impact performance Bouvier et al. (2019). Consequently, many existing methods focus on achieving low-latency inference with improved convergence while maintaining accuracy Meng et al. (2022). Traditional approaches like surrogate gradient learning Neftci et al. (2019) and temporal coding Bellec et al. (2018) have been further enhanced by advanced techniques Dampfhoffer et al. (2023). Notably, Deng et al. Deng et al. (2022) introduced a gradient re-weighting mechanism to improve the temporal efficiency of SNN training.

To eliminate the need for manual threshold selection, Bojkovic et al. Bojkovic et al. (2024) proposed a data-driven approach for threshold selection and potential initialization. Their method also facilitates the conversion of trained ANNs into SNNs, enabling efficient and high-performance training even in low-latency settings ($T = 1, 2, 4$). These advancements are crucial for real-time, energy-constrained applications. Beyond energy efficiency, SNNs have also been explored for their inherent robustness against adversarial Nomura et al. (2022), El-Allami et al. (2021) and model inversion attacks Kim et al. (2022), further reinforcing their potential towards robust AI.

A.2 Membership Inference Attacks (MIAs)

The threat of Membership Inference Attacks (MIAs) was first demonstrated by Shokri et al. Shokri et al. (2017) in a simple Machine Learning-as-a-Service (MLaaS) black-box setting. Since then, extensive research has explored the privacy risks associated with diverse neural network architectures for a wide range of applications Hu et al. (2022), Yeom et al. (2018), Salem et al. (2018). Despite significant advancements and robustness characteristics of SNNs Kim et al. (2022), their vulnerability to MIAs remains largely unclear and underexplored Sharmin et al. (2019).

The inconsistencies in evaluation metrics and experimental settings across existing studies have made direct comparisons of MIA challenging Hu et al. (2023). However, authors in Carlini et al. (2022) presented MIA from the first principle perspective, emphasizing the importance of analyzing the Receiver Operating Characteristic (ROC) curve in attack's assessments. The ROC fully captures the tradeoff between True Positive Rate (TPR) and False Positive Rate (FPR) of the membership data across different classification thresholds. Reporting TPR under extremely low FPR conditions ($\leq$1% and $\leq$ 0.1%) is particularly crucial, as attackers prioritize confidently identifying members over overall accuracy. More recently, Zarifzadeh et al. (2024) proposed a state-of-the-art robust MIA (RMIA), and generalized all other existing MIAs under the umbrella of their attack formulation. RMIA also achieved highly effective attack performance with a limited number of shadow models.

A.3 Concurrent Work

While existing research primarily focuses on traditional ANNs, the membership privacy risks in SNNs remain largely unexamined. A recent study by Li et al. (2024) explored the robustness of SNNs against MIAs, incorporating diverse experimental settings and assessing the impact of data augmentation. However, despite these contributions, the study suffers from several critical limitations. It relies on biased evaluation metrics such as balanced accuracy, which can obscure the true effectiveness of MIAs, and employs outdated training techniques for SNNsCarlini et al., 2022. Nowadays, in many research papers, AUC and TPR at very low FPR are the main metrics to study the performance of MIAs. Additionally, the evaluation is conducted on simple datasets, failing to provide meaningful insights into real-world scenarios.

More importantly, the study neglects key advancements in attack methodologies, such as Zarifzadeh et al. (2024), limiting the comprehensiveness of its findings. Furthermore, the analysis is restricted to timestep variations in neuromorphic datasets, lacking a systematic investigation of static datasets. These shortcomings underscore the insufficiency of existing efforts in rigorously assessing membership privacy risks in SNNs. A more sound evaluation is necessary to bridge this gap and uncover the true privacy vulnerabilities of SNNs.

B PRELIMINARIES

B.1 SPIKING NEURAL NETWORK

In this section, we will briefly explain some terminology related to SNNs, including the inference process, and mainstream training strategies: direct training, ANN-SNN conversion, and hybrid training.

B.1.1 DATA ENCODING

Unlike ANNs, SNNs have an additional dimension to represent data: the temporal dimension. This means that the input to the network is not a single image but is instead encoded over time into a sequence of T images. T is also called *time step* or *latency* in SNN. There are various methods to encode images for SNNs. One of the classical encoding methods is called *constant encoding*, Which simply replicates the original image T times. After encoding, the resulting data is fed into the SNN.

B.1.2 FORWARD PASS

The forward process of SNN begins with encoding the input data, such as an image, into a spike sequence over time. These spikes represent neuron activity across different time steps. As the spikes are fed into the network, each neuron updates its membrane potential based on the spikes it receives. If a neuron's membrane potential exceeds a certain threshold, the neuron fires a spike and sends it to the next layer through synapses.

$$U_l^t = \lambda U_l^{t-1} + w_l O_{l-1}^t, \qquad O_l^t = H(U_l^t - \theta_{tr}), \qquad U_l^t = U_{reset}$$

The above formula describes the dynamics of a neuron in an SNN across time steps, particularly focusing on the membrane potential and the neuron's spiking behavior. The first equation $U_l^t = \lambda U_l^{t-1} + w_l O_{l-1}^t$ represents the membrane potential U_l^t of a neuron at layer l and time t. The membrane potential is updated by combining two components: the decayed potential from the previous time step λU_l^{t-1}, where λ is a decay factor between 0 and 1, and the weighted input from the previous layer's output $w_l O_{l-1}^t$, where w_l is the synaptic weight and O_{l-1}^t is the output (spike) from the previous layer at time t. The second equation $O_l^t = H(U_l^t - \theta_{tr})$ describes the neuron's output at time t, where H is a Heaviside step function. If the membrane potential U_l^t exceeds a certain threshold θ_{tr}, the neuron generates an output spike ($O_l^t = 1$); otherwise, no spike is generated ($O_l^t = 0$). The third equation $U_l^t = U_{reset}$ indicates that once the neuron fires a spike, its membrane potential is reset to a specific value U_{reset}, preparing the neuron for further spike processing in subsequent time steps. For the last layer L, the output of the network is U_L^T, which is the membrane potential for the final time T. In this setting, if the decay factor λ is 1, the neuron is called an **Integrate-and-Fire (IF)** neuron; otherwise, it is a **Leaky Integrate-and-Fire (LIF)** neuron.

B.1.3 ANN-SNN CONVERSION

The training of SNNs can be categorized into three main approaches: direct backpropagation and ANN-to-SNN conversion. However, in MIAs, the probability vector output by the neural network is the most critical feature for determining the membership status of a specific data point. Therefore, when studying the effectiveness of MIAs, we should focus more on the forward propagation process of SNNs rather than their backpropagation process.

Since training SNNs through backpropagation is highly time-consuming, we adopted the ANN-to-SNN conversion approach to obtain our SNN models. There are two commonly used methods for ANN-SNN conversion: weight scaling (Rueckauer et al., 2017; Diehl & Cook, 2015) and threshold scaling (Sengupta et al., 2019). In Kim et al. (2022), which used the threshold scaling method. This method works by inputting some training data into the ANN, observing the maximum activation values at each layer, and then setting these values as the thresholds for each corresponding layer in the SNN. More recently, there have been works on ANN-SNN conversion techniques that achieve higher accuracy and have become more widely used. In this paper, we adopt the recent state-of-the-art conversion method from Bojkovic et al. (2024), which sets the threshold and potential initialization based on a fine-grained analysis of activation values in the original ANN. After the ANN-to-SNN conversion, the model can be further fine-tuned for several epochs to enhance

performance, a process known as *hybrid training* (Rathi et al., 2020). By applying hybrid training, the state-of-the-art method proposed by Bojkovic et al. (2024) significantly increases the network performance for low-latency SNNs.

B.2 Membership Inference

Definition 1 (Binary Classification). *Let $S = \{(x_i, y_i)\}_{i=1}^n$ be a dataset where:*

- *$x_i \in \mathbb{R}$ is a one-dimensional feature value.*
- *$y_i \in \{0, 1\}$ represents the class label, where $y_i = 1$ indicates a positive sample, and $y_i = 0$ indicates a negative sample.*

A threshold-based classifier with threshold t is defined as:

$$\hat{y}_i = \begin{cases} 1, & x_i \geq t \\ 0, & x_i < t. \end{cases} \tag{2}$$

Definition 2 (True Positive Rate (TPR) and False Positive Rate (FPR)). *For a given threshold t, we define:*

$$TPR(t) = \frac{\sum_{i=1}^n \mathbf{1}(x_i \geq t, y_i = 1)}{\sum_{i=1}^n \mathbf{1}(y_i = 1)}, \tag{3}$$

$$FPR(t) = \frac{\sum_{i=1}^n \mathbf{1}(x_i \geq t, y_i = 0)}{\sum_{i=1}^n \mathbf{1}(y_i = 0)}. \tag{4}$$

The ROC curve is defined as the set of points:

$$\text{ROC} = \{(\text{FPR}(t), \text{TPR}(t)) \mid t \in \mathbb{R}\}. \tag{5}$$

The Area Under the Curve (AUC) is given by:

$$\text{AUC} = \int_0^1 \text{TPR}(\text{FPR})\, d(\text{FPR}). \tag{6}$$

Definition 3 (Membership Inference Game). *Carlini et al., 2022; Zarifzadeh et al., 2024 Let π be an underlying data distribution and let $\mathcal{A}$ be a training algorithm. We begin by drawing a training set $S \sim \pi^m$, and then use $\mathcal{A}$ to train a target model $\theta \leftarrow \mathcal{A}(S)$. The game proceeds between two entities: a* challenger *and an* attacker.

1. ***Dataset Sampling and Model Training.*** *The challenger samples a dataset S from π, and trains a model θ using $\mathcal{A}$ on S. Here, θ is the target model.*
2. ***Challenge Sample Selection.*** *The challenger tosses a fair coin $b \in \{0, 1\}$.*
   - *If $b = 1$, it picks a point x uniformly at random from S. In this case, x is a member of θ.*
   - *If $b = 0$, it draws a fresh sample x from π (ensuring $x \notin S$). In this case, x is not a member of θ.*

   The challenger then provides both the target model θ and the sample x to the attacker.
3. ***Attacker's Inference.*** *The attacker, having access to θ (and potentially query access to the distribution π), computes a* membership score

   $$\text{Score}_{MIA}(x; \theta),$$

   indicating how likely it believes x was contained in S. Based on a chosen threshold β, the attacker issues a membership decision

   $$\hat{b} \leftarrow \mathbf{1}\big[\text{Score}_{MIA}(x; \theta) \geq \beta\big].$$

   Here, $\mathbf{1}[\cdot]$ is the indicator function, so $\hat{b} = 1$ (member) if the score exceeds the threshold, and 0 (non-member) otherwise.

4. ***Outcome and Evaluation.*** *The attacker's prediction $\hat{b}$ is compared against the true bit b:*

$$\textit{Outcome} \;=\; \begin{cases} 1, & \textit{if } \hat{b} = b, \\ 0, & \textit{otherwise}. \end{cases}$$

By repeating this experiment over many trials, we can plot the ROC for each β. The leakage *of the model is often characterized by the achievable trade-off between TPR and FPR across all possible threshold values. The attack method is considered to be good with a high AUC score.*

Definition 4 (Equivalence of Score Functions)**.** *Let S_1 and S_2 be two real-valued MIA scoring functions that assign a score to each data point. We say that S_1 and S_2 are equivalent if for all data points $x_1, x_2 \in D$,*

$$S_2(x_1) > S_2(x_2) \iff S_1(x_1) > S_1(x_2). \tag{7}$$

C PROOFS

Theorem 1. *Let S_1 and S_2 be two equivalent scoring functions, then the ROC curves for S_1 and S_2 on the same data set are identical.*

Proof. Since S_1 and S_2 preserve the same ranking of data points, their ordering in terms of classification thresholds remains unchanged. That is, for any threshold t applied to S_2, there exists a corresponding threshold s applied to S_1 such that the classification outcomes remain identical:

$$\hat{y}_i = \begin{cases} 1, & S_2(x_i) \geq t \\ 0, & S_2(x_i) < t \end{cases} \iff \hat{y}_i = \begin{cases} 1, & S_1(x_i) \geq s \\ 0, & S_1(x_i) < s. \end{cases} \tag{8}$$

Since both functions yield the same classification outcomes for all possible threshold values, they result in the same TPR and FPR values at each threshold:

$$\text{TPR}_{S_1}(s) = \text{TPR}_{S_2}(t), \tag{9}$$

$$\text{FPR}_{S_1}(s) = \text{FPR}_{S_2}(t). \tag{10}$$

Since the ROC curve is defined as the parametric plot of $(\text{FPR}(t), \text{TPR}(t))$, it follows that:

$$\text{ROC}_{S_1} = \text{ROC}_{S_2}. \tag{11}$$

Thus, the theorem is proved. □

Theorem 2. *Let S_1 and S_2 be two MIA scoring functions defined as:*

$$S_1(x) = \frac{1}{N} \sum_{j=1}^{N} \mathbf{1}\Big(\Pr(x \mid \theta) \geq \Pr(z_j \mid \theta)\Big), \tag{12}$$

$$S_2(x) = \Pr(x \mid \theta), \tag{13}$$

where $\{z_j\}_{j=1}^{N}$ is a fixed set of samples. Suppose $x_1, x_2 \in D$. Then, $S_2(x_1) > S_2(x_2)$, if and only if $S_1(x_1) > S_1(x_2)$.

Proof. Assume $S_2(x_1) > S_2(x_2)$, meaning

$$\Pr(x_1 \mid \theta) > \Pr(x_2 \mid \theta). \tag{14}$$

Define the indicator set:

$$A_x = \{z_j \mid \Pr(x \mid \theta) \geq \Pr(z_j \mid \theta)\}. \tag{15}$$

Since $\Pr(x_1 \mid \theta) > \Pr(x_2 \mid \theta)$, we have $A_{x_2} \subset A_{x_1}$. Hence,

$$S_1(x_1) = \frac{|A_{x_1}|}{N} > \frac{|A_{x_2}|}{N} = S_1(x_2). \tag{16}$$

This establishes the order-preserving property in the forward direction.

Reverse direction: Assume $S_1(x_1) > S_1(x_2)$. This implies

$$\frac{|A_{x_1}|}{N} > \frac{|A_{x_2}|}{N}. \tag{17}$$

Since A_x is defined based on comparisons with $\Pr(x \mid \theta)$, if $S_1(x_1) > S_1(x_2)$, then x_1 ranks strictly higher than x_2 in terms of how often it exceeds reference points z_j. The only way for this to happen is if

$$\Pr(x_1 \mid \theta) > \Pr(x_2 \mid \theta), \tag{18}$$

which implies $S_2(x_1) > S_2(x_2)$.

Since both directions hold, we conclude

$$S_2(x_1) > S_2(x_2) \iff S_1(x_1) > S_1(x_2). \tag{19}$$

Hence the S_1 and S_2 are equivalent. By Theorem 1, the ROC for them are the same □

Remark 1. *Theorem 2 is used to prove the modified Attack-P method is the same as the empirical estimation of the original method. And it can be easily used in the same structure of Theorem 2, to prove the correctness of the modified RMIA method. According to the data-splitting method in Section 2, for training the reference models, we can confirm that for each x, there must be the same number of models using and without using x to train. This satisfies the requirement of the original paper, ensuring that the number of $\theta'_{\bar{x}}$ and θ'_x is equal, providing an unbiased estimation of $\Pr(x)$. This contains all the necessary information for an online RMIA attack, allowing it to achieve optimal performance. See Zarifzadeh et al. (2024) for more details.*

D Efficient Data-Splitting Strategy for Online Attack Setting

We propose a data splitting strategy that emulates an online attack without incurring the computational cost of repeatedly retraining models. Our method begins by partitioning the dataset D into two equal halves: one to train our target SNN model and the other to serve as its test set. Subsequently, we shuffle the original dataset D. This shuffled dataset is then iteratively divided to train n pairs of reference models, resulting in a total of $2n$ reference models. Crucially, while each reference model is trained on a substantial portion of D derived from this process, the data splitting is carefully designed to ensure that each query from the target model's test set is present in approximately half of the reference models. This balanced exposure is essential for unbiased MIA evaluation.

This approach effectively simulates an online attack environment by providing a diverse set of pre-trained reference models against which to assess membership, all without the need for computationally expensive online model retraining. Consistent with standard MIA assumptions in the literature (Zarifzadeh et al., 2024), we acknowledge that our reference models are designed under the common assumption that attackers possess knowledge of the target model's architecture. Our approach is depicted in Figure 2 providing an intuitive explanation of its effectiveness.

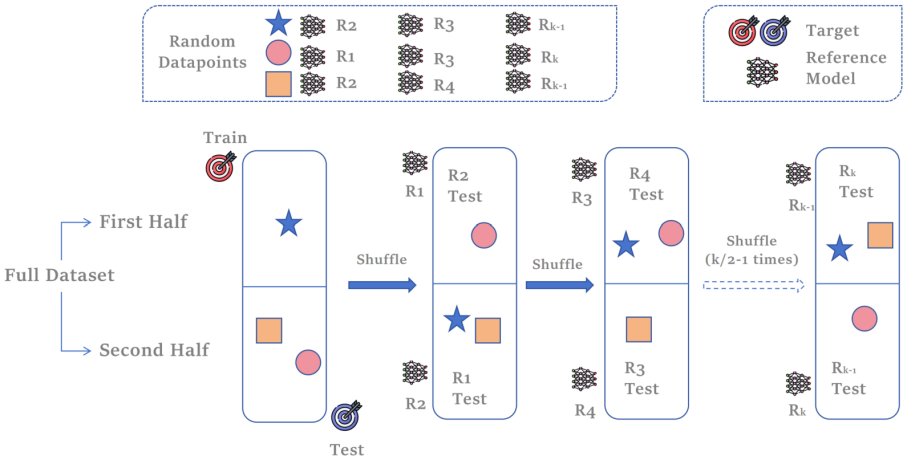

Figure 2: Illustration of Data Splitting Strategy

This approach ensures a fair and efficient data-splitting strategy for an online attack setting while maintaining unbiased results. By iteratively shuffling D and training k pairs of reference models, we distribute each target query across half of the reference models. This allows for effective evaluation without additional training overhead.

E ALGORITHM

Algorithm 1 Estimate $\Pr(x \mid \theta)$ with Input Dropout

Require: θ: Model, x: Input, p: Dropout rate, n: Number of samples
Ensure: Estimated $\Pr(x \mid \theta)$

$s \leftarrow 0$ ▷ Initialize confidence sum
for $i = 1$ to n **do**
  $M \sim \text{Uniform}(0, 1)$ ▷ Generate uniform noise tensor
  $m \leftarrow M > p$ ▷ Generate binary dropout mask
  $\tilde{x} \leftarrow x \odot m$ ▷ Apply dropout mask
  $s \leftarrow s + \Pr(\tilde{x} \mid \theta)$ ▷ Pass through model and sum
end for
$\Pr(x \mid \theta) \leftarrow s/n$ ▷ Compute final estimate
return $\Pr(x \mid \theta)$

F EXPERIMENTAL SETUP

F.1 MODEL TRAINING

F.1.1 HYBRID TRAINING VIA ANN-TO-SNN CONVERSION

In this approach, we first train an ANN and subsequently convert it into a SNN using the threshold extraction and fine-tuning method proposed by Bojkovic et al. (2024). This conversion facilitates efficient SNN training while maintaining high performance, even under low-latency constraints. We consider both CIFAR-10 and CIFAR-100 datasets Krizhevsky et al. (2009), where CIFAR-10 represents a relatively simpler classification task, whereas CIFAR-100 presents a more challenging scenario due to its increased class diversity. To ensure a comprehensive evaluation, we train ResNet18 both datasets. All models are trained for 200 epochs. For CIFAR-10, we initialize the learning rate at 0.1, while for CIFAR-100, we use 0.02. Optimization is performed using the stochastic gradient descent (SGD) algorithm with a weight decay of 5×10^{-4} and a momentum parameter of 0.1.

Once ANN training is complete, we convert the models into SNNs. Following Bojkovic et al. (2024), we begin by training the converted SNN models at $T = 1$ for 50 epochs using surrogate gradient learning (with a learning rate of 5×10^{-3}). Higher latency SNN models ($T = 2, 3, 4$) are then obtained through sequential training, where the weights of the SNN model at latency T are initialized from the previously trained SNN model at $T - 1$. Each additional latency level is trained for 30 epochs. All training experiments are conducted on NVIDIA A100 GPUs.

F.1.2 DIRECT SNN TRAINING

To validate the findings obtained from our hybrid ANN-to-SNN training approach, we conduct an additional set of experiments using directly trained SNNs, following the methodology proposed in Mukhoty et al. (2023). For this experiment, we focus on CIFAR-10 and CIFAR-100 and train a ResNet18 SNN model. The model is trained for 250 epochs using the Adam optimizer with a learning rate of 0.001. We evaluate the SNN at $T = 4$ to assess its impact on privacy risks. Unlike the sequential training strategy employed in the hybrid approach, direct SNN training allows for parallel training of models at different latencies, as each latency level is independently optimized. To ensure fair comparison, we apply the same data preprocessing and transformations as in the hybrid training setup.

F.2 ACCURACY OF TRAINED MODELS

In this section, we report the test accuracy of the trained models. Since MIA experiments are conducted in a setting where only half of the dataset is used for training, the test accuracy is lower compared to training on the full dataset. However, this reduction in accuracy is expected and does not impact the validity of our analysis, as the primary objective is to establish a controlled experimental framework to evaluate membership inference risks.

Table F.1: ResNet18 test accuracy in % for trained ANN-2-SNN models across different datasets.

Dataset	**SNN(T=1)**	**SNN (T=2)**	**SNN (T=4)**	**ANN**
CIFAR-10	86.1	87.3	88.2	90.9
CIFAR-100	59.5	62.1	63.9	69.5

From Tables F.1 and F.2, we observe that the results align closely, as expected. Furthermore, both training paradigms exhibit that the accuracy improves with increasing latency (T). This aligns with the theoretical expectation that higher latency allows for precise spike-based representations.

Table F.2: Test accuracy in % for direct training of SNN on CIFAR10 with ResNet18.

Dataset	**SNN (T=4)**
CIFAR-10	87.0
CIFAR-100	66.9

Table F.3: ANN to Attack SNN.

Dataset	**Dropout**	**Attack**	**SNN(T=1)**			**SNN(T=2)**			**SNN(T=4)**		
			AUC	0.1%	1%	AUC	0.1%	1%	AUC	0.1%	1%
CIFAR-10	**False**	Attack-R	53.26	0.00	1.74	53.88	0.00	0.49	55.04	0.00	0.00
		RMIA	58.52	0.69	3.40	59.68	**0.81**	3.86	61.32	**1.05**	4.55
	True	Attack-R	53.31	0.25	1.56	54.10	0.00	0.50	55.50	0.00	0.00
		RMIA	**59.32**	**0.74**	**3.77**	**60.52**	0.76	**4.34**	**62.29**	0.89	**4.98**
CIFAR-100	**False**	Attack-R	58.99	0.00	0.80	62.71	0.00	1.89	67.98	0.00	0.00
		RMIA	63.42	**0.62**	3.85	67.73	**1.02**	**6.29**	74.43	**1.37**	9.65
	True	Attack-R	59.41	0.00	0.49	63.31	0.00	1.40	68.58	0.00	0.00
		RMIA	**65.01**	0.38	**3.89**	**69.69**	0.72	6.13	**76.11**	1.10	**10.39**

Table F.4: Directly trained SNN Attack Results (T=4).

Dataset	**Dropout**	**Attack**	AUC	0.1%	1%
CIFAR-10	**False**	Attack-P	59.76	0.05	1.05
		Attack-R	63.53	0.00	0.00
		RMIA	63.85	1.04	5.76
	True	Attack-P	58.87	0.06	0.96
		Attack-R	64.64	0.00	0.00
		RMIA	**67.58**	**3.56**	**8.86**
CIFAR-100	**False**	Attack-P	56.54	0.00	0.74
		Attack-R	56.15	0.00	0.00
		RMIA	59.62	0.44	2.39
	True	Attack-P	56.58	0.00	1.06
		Attack-R	57.65	0.00	0.00
		RMIA	**62.11**	**0.62**	**3.13**

G PLOTS

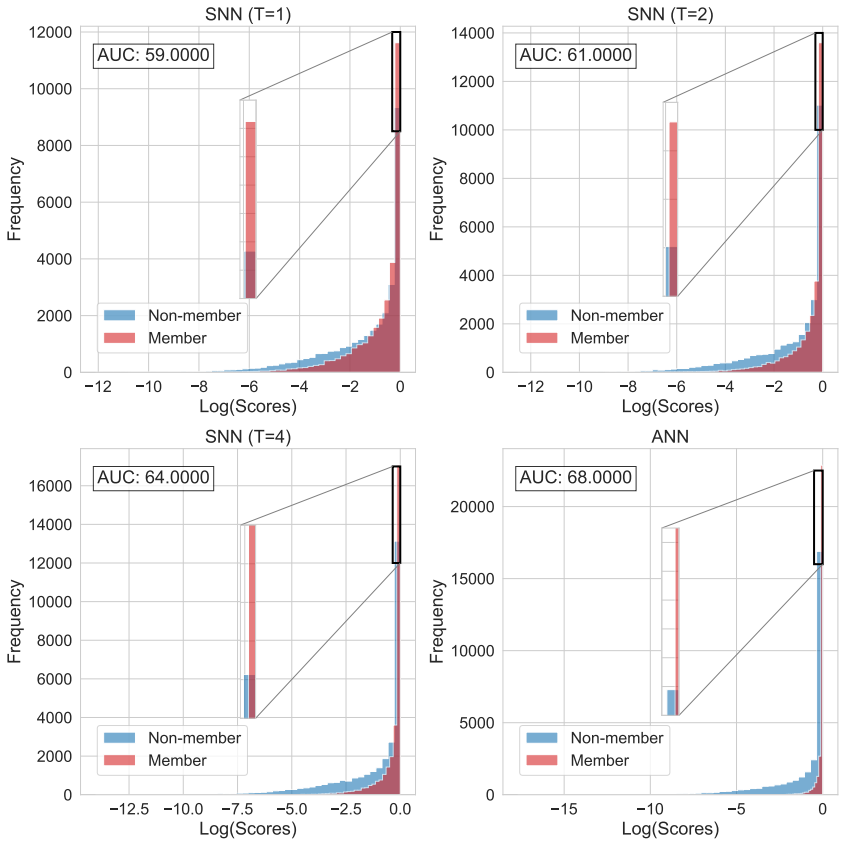

Figure 3: These four histograms represent the Attack-P(confidence) distribution for member and non-member data across all latency SNNs and ANNs in CIFAR-100