Prediction Privacy in Distributed Multi-Exit Neural Networks: Vulnerabilities and Solutions (with Appendix)

Tejas Kannan
tkannan@uchicago.edu
University of Chicago
Chicago, Illinois, United States

Nick Feamster
feamster@uchicago.edu
University of Chicago
Chicago, Illinois, United States

Henry Hoffmann
hankhoffmann@cs.uchicago.edu
University of Chicago
Chicago, Illinois, United States

ABSTRACT

Distributed Multi-exit Neural Networks (MeNNs) use partitioning and early exits to reduce the cost of neural network inference on low-power sensing systems. Existing MeNNs exhibit high inference accuracy using policies that select when to exit based on data-dependent prediction confidence. This paper presents a side-channel attack against distributed MeNNs employing data-dependent early exit policies. We find that an adversary can observe when a distributed MeNN exits early using encrypted communication patterns. An adversary can then use these observations to discover the MeNN's predictions with over 1.85× the accuracy of random guessing. In some cases, the side-channel leaks over 80% of the model's predictions. This leakage occurs because prior policies make decisions using a single threshold on varying prediction confidence distributions. We address this problem through two new exit policies. The first method, Per-Class Exiting (PCE), uses multiple thresholds to balance exit rates across predicted classes. This policy retains high accuracy and lowers prediction leakage, but we prove it has no privacy guarantees. We obtain these guarantees with a second policy, Confidence-Guided Randomness (CGR), which randomly selects when to exit using probabilities biased toward PCE's decisions. CGR provides statistically equivalent privacy with consistently higher inference accuracy than exiting early uniformly at random. Both PCE and CGR have low overhead, making them viable security solutions in resource-constrained settings.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; • **Computer systems organization** → *Sensor networks*; • **Security and privacy** → *Distributed systems security*.

ACM Reference Format:
Tejas Kannan, Nick Feamster, and Henry Hoffmann. 2023. Prediction Privacy in Distributed Multi-Exit Neural Networks: Vulnerabilities and Solutions (with Appendix). In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23), November 26–30, 2023, Copenhagen, Denmark.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3576915.3623069

CCS '23, November 26–30, 2023, Copenhagen, Denmark

ACM ISBN 979-8-4007-0050-7/23/11…$15.00
https://doi.org/10.1145/3576915.3623069

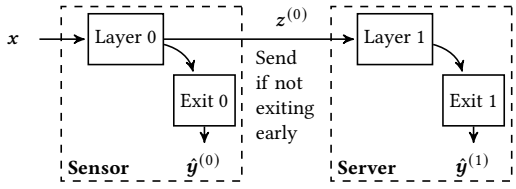

Figure 1: A distributed Multi-exit Neural Network (MeNN) [86] with two total exits.

1 INTRODUCTION

Battery-powered sensors are common in applications for areas such as agriculture [90] and healthcare [32]. Sensor devices collect measurements from their environment, process these values, and communicate results to a server. For reliable performance, devices must meet energy [25, 43] and latency [46, 101] constraints. Thus, sensors seek ways to improve their efficiency, and a common technique for doing so is to push data processing onto the sensor device [20]. This method is beneficial because local processing allows the device to transmit smaller aggregate results [25].

Modern sensor processing uses deep neural networks (DNNs) due to their high-quality results [25, 62]. However, DNNs have high resource costs, making them challenging to deploy on low-power devices [62, 100]. Prior systems address this challenge by partitioning DNNs between sensor and server [7, 46, 60]. The sensor device holds a subset of the DNN, and the system performs inference as follows:

(1) Process measurements on the sensor with the DNN subset.
(2) Transmit the intermediate DNN activations to the server.
(3) Complete inference with the remaining DNN layers.

In this process, the sensor always transmits the intermediate state (Step 2), and this step requires expensive wireless communication [25, 46]. Sensing systems address this problem by augmenting DNNs with early exits (Figure 1). These inference models, called Multi-exit Neural Networks (MeNNs), contain early exit points which create predictions using a subset of the entire DNN [86]. Distributed MeNNs [55, 87] form an initial prediction on the sensor, alleviating the need to communicate with the server.

MeNNs face a key decision when reaching an exit point: whether to terminate inference. This *exit decision*, which represents where the MeNN stops inference, comes with a tradeoff. Exiting earlier leads to lower inference costs by skipping subsequent layers. Early exits, however, generally reduce accuracy [86, 92]. *Data-dependent* adaptive behavior is an emerging method to balance this tradeoff [35, 37, 48, 77, 86, 92, 94, 104]. These methods determine when

to terminate inference by comparing the neural network's prediction *confidence* (e.g., the maximum classification probability) to a single *threshold* [94]. Inference terminates when the confidence exceeds the threshold. This strategy is data-dependent because the prediction confidence is a function of the given input. These data-dependent methods yield high accuracy under cost constraints because not all inputs are equally difficult to classify [48, 94].

In this work, we study MeNN early exiting from a new perspective: privacy. We demonstrate how previous data-dependent exit policies [86, 94] show *asymmetric behavior* on different predicted classes. That is, the MeNN exits early for some classes more frequently than others. This behavior occurs because MeNNs produce different distributions of prediction confidence for different classes (§2.3). Standard data-dependent methods, however, apply a *single* threshold to all prediction confidence values. This single threshold thus causes different average exit behavior across predictions.

This asymmetry means an adversary can learn about a distributed MeNN's predictions by observing its exit decisions, creating a privacy issue in sensing systems for two reasons:

(1) Sensors leak when the distributed MeNN exits early through side-channels derived from encrypted communication patterns (§3.2, §6.8).
(2) Sensors collect values with temporal correlations [21, 23], so an adversary who extracts when the MeNN exits early observes consecutive elements with similar predictions.

With these properties, we discover that an adversary can use communication side-channels to observe a distributed MeNN's pattern of exit decisions. The adversary can then use this pattern to infer the model's most frequent prediction over short timescales. For example, on the task of activity recognition [54], we show that this side-channel allows a **white box** attacker (§3.2) to uncover 52.00% of the MeNN's predictions (§6.8). This leakage extends to ten tasks; on average, we demonstrate how data-dependent policies enable a white box attacker to infer MeNN predictions 1.85× more frequently than random guessing (§6.2). Further, we build a **black box** attacker (§3.2) which can still infer MeNN predictions at 1.56× the rate of random guessing (§6.7). Thus, privacy-conscious sensor systems [32, 90, 98] cannot gain the benefits of MeNNs.

There are two common approaches to closing side-channels based on asymmetric behavior. The first method standardizes resource use [12, 22, 50]. For MeNNs, this principle forces all inputs to exit at the same point. This design negates the benefits of MeNNs, resulting in either suboptimal accuracy (§6.4) or prohibitive overhead (§6.9). The second approach randomizes behavior [5, 50]. This method leads to an MeNN policy that exits early uniformly at random. Unfortunately, random exiting imposes a high accuracy cost (§6.4). We instead want exit policies with the following properties:

(P1) Achieve perfect privacy by having no observable relation between exit decisions and MeNN predictions.
(P2) Exhibit minimal energy overhead compared to previous data-dependent methods.
(P3) Display greater inference accuracy than Random exiting.
(P4) Do not require retraining or redesign of existing MeNNs.

This last property is important because neural network training is expensive. Solutions requiring new architectures or retraining are not compatible with existing MeNNs.

We develop two new exit policies to meet (P1)–(P4). Our first approach, *Per-Class Exiting (PCE)*, augments prior data-dependent methods by using different confidence thresholds for each class (§4). PCE tunes these thresholds to exhibit symmetric exit rates for better privacy (§6.2). The policy retains high inference accuracy because it still makes decisions using prediction confidence (§6.4).

Despite improved empirical privacy, we prove that PCE has no privacy guarantees (P1). To achieve these guarantees, we augment PCE with randomization through a new policy called *Confidence-Guided Randomness (CGR)* (§5). CGR randomly selects an exit using probabilities biased toward PCE's confidence-based decisions. Further, CGR optimizes the MeNN's accuracy while maintaining privacy by adapting the bias magnitude using trends in the MeNN's predictions. By using both prediction confidence and randomization, CGR has higher accuracy than exiting early uniformly at random (§6.4) with statistically equivalent privacy (§6.2). CGR also incurs negligible overhead (§6.9) and works with already-trained MeNNs, allowing the policy to successfully satisfy (P1)–(P4).

To the best of our knowledge, this is the first work to study the prediction privacy of data-dependent early exiting for MeNNs. Overall, we make the following contributions[1]:

(1) We show that previous data-dependent MeNN exit policies leak information about model predictions. For two-exit distributed MeNNs, an adversary can infer the model's predictions under both white box and black box assumptions.
(2) We create a policy called Per-Class Exiting (PCE) that uses different thresholds for inputs of each class. This method reduces the leakage of prior data-dependent policies and preserves accuracy to within 0.4 percentage points.
(3) We construct a policy, Confidence-Guided Randomness (CGR), which integrates randomization into PCE. CGR has theoretical privacy benefits. Compared to exiting uniformly at random, CGR displays statistically equivalent privacy with higher accuracy on over 90% of target exit rates.

Our work demonstrates the privacy implications of performing data-dependent distributed MeNN inference on sensing systems. With our proposed methods, privacy-conscious applications can safely achieve the performance benefits of MeNNs.

2 BACKGROUND AND MOTIVATION

This section provides background on multi-exit neural networks (§2.1) before discussing policies for early exiting (§2.2). We then provide an example of information leakage (§2.3) and state the goal of privacy-preserving exit policies (§2.4).

2.1 Multi-Exit Neural Networks (MeNNs)

Deep neural networks (DNNs) are statistical inference models with layers of linear and nonlinear transformations [56]. Under supervised learning, DNNs $f_{\boldsymbol{\theta}}$ fit their parameters $\boldsymbol{\theta}$ by minimizing a loss function (e.g., cross-entropy) on a labelled dataset $D = \{(\boldsymbol{x}^{(t)}, y^{(t)})\}_{t=0}^{N-1}$ [76][2]. We consider DNNs on classification tasks where $\hat{\boldsymbol{y}} = f_{\boldsymbol{\theta}}(\boldsymbol{x}) \in \mathbb{R}^L$ has the predicted probabilities for each class in $[L] = \{0, 1, \ldots, L-1\}$. The predicted class is $\hat{y} = \text{argmax}_{\ell \in [L]} \hat{y}_\ell$.

[1]The code is available at https://github.com/tejaskannan/privacy-dnn-early-exit
[2]We denote vectors in boldface and scalars in plain text.

Algorithm 1 MeNN inference routine

procedure MeNNInference($\boldsymbol{x}$, $f_{\boldsymbol{\theta}}^{(K)}$, π)
  $\boldsymbol{z}^{(-1)} \leftarrow \boldsymbol{x}$
  for $k \in [K-1]$ **do**
    $\hat{\boldsymbol{y}}^{(k)}, \boldsymbol{z}^{(k)} \leftarrow f_{\boldsymbol{\theta}_k}^{(k)}(\boldsymbol{z}^{(k-1)})$
    if $\pi(\hat{\boldsymbol{y}}^{(k)}, k) = 0$ **then**
      return $\hat{\boldsymbol{y}}^{(k)}$
  $\hat{\boldsymbol{y}}^{(K-1)}, \boldsymbol{z}^{(K-1)} \leftarrow f_{\boldsymbol{\theta}_k}^{(K-1)}(\boldsymbol{z}^{(K-2)})$
  return $\hat{\boldsymbol{y}}^{(K-1)}$

Standard DNNs process inputs $\boldsymbol{x}$ with all parameters $\boldsymbol{\theta}$. However, this design is unnecessary to achieve high accuracy, as not all inputs are equally difficult to classify [48, 94]. DNNs can preserve accuracy with reduced execution costs using early exits; each exit is an intermediate classifier that creates a prediction with a subset of model parameters [86, 92, 94]. We describe this approach, called a Multi-exit Neural Network (MeNN), as a collection of K classifiers $f_{\boldsymbol{\theta}}^{(K)} = [f_{\boldsymbol{\theta}_0}^{(0)}, f_{\boldsymbol{\theta}_1}^{(1)}, \ldots, f_{\boldsymbol{\theta}_{K-1}}^{(K-1)}]$ where $f_{\boldsymbol{\theta}_k}^{(k)}$ is a DNN that takes an input or output of $f_{\boldsymbol{\theta}_{k-1}}^{(k-1)}$, $\boldsymbol{z}^{(k-1)}$, and creates the predicted probabilities $\hat{\boldsymbol{y}}^{(k)} \in \mathbb{R}^L$. At time t, the output of $f_{\boldsymbol{\theta}_k}^{(k)}$ is $\hat{\boldsymbol{y}}^{(k,t)}$.

We consider MeNNs distributed across devices in a sensing system [55, 87]. The sensor device contains the initial layers of the MeNN, and the server contains the remaining portion of the model (Figure 1). When crossing between devices, the system must transmit the required state (e.g., $\boldsymbol{z}^{(0)}$ in Figure 1) to continue inference.

2.2 Early Exit Policies

MeNNs must determine the exit point at which to terminate inference. This decision comes with a tradeoff [86, 92]. Later exits apply more parameters and achieve higher accuracy. This benefit comes with higher execution costs, as the system must compute more layers and communicate between devices.

Prior MeNNs manage this tradeoff in a data-dependent manner using prediction confidence [9, 86, 94]. Common confidence functions $h : \mathbb{R}^L \rightarrow \mathbb{R}$ include the maximum value [94] and entropy [86] in the predicted distribution. When inference reaches the k^{th} exit, the system compares the confidence $h(\hat{\boldsymbol{y}}^{(k)})$ to a threshold $\tau^{(k)}$. If $h(\hat{\boldsymbol{y}}^{(k)}) \geq \tau^{(k)}$, the model is "confident enough" and stops inference; otherwise, the system continues to the next exit. The thresholds $\tau^{(k)}$ control how the MeNN balances the tradeoff between accuracy and execution cost. Larger thresholds yield more accurate results, and smaller thresholds result in low-cost inference. We emphasize that these existing methods make *data-dependent* decisions because the confidence is a deterministic function of the k^{th} prediction, $\hat{\boldsymbol{y}}^{(k)}$. Thus, the MeNN's exit decisions contain information about the model's predictions, where the *exit decision* for time t, $k_t \in [K]$, is the exit at which the MeNN stops inference.

We represent early exit methods using a *policy* $\pi : \mathbb{R}^L \times \mathbb{N} \rightarrow \{0, 1\}$, which takes the prediction $\hat{\boldsymbol{y}}^{(k)} \in \mathbb{R}^L$ and the exit $k \in [K]$. The function outputs 0 to terminate inference and 1 to continue. Algorithm 1 shows this procedure. The equation below is a general data-dependent policy for the confidence h where $[\cdot]_1$ is 1 when the condition holds and 0 otherwise. Concrete policies use a specific implementation for h such as $h_{MaxProb}(\hat{\boldsymbol{y}}) = \max_{i \in [L]} \hat{y}_i$ [94].

$$\pi_{DataDep}(\hat{\boldsymbol{y}}^{(k)}, k) = [h(\hat{\boldsymbol{y}}^{(k)}) < \tau^{(k)}]_1 \tag{1}$$

2.3 Example of Information Leakage

Data-dependent MeNN exit policies create a relationship between their predictions and exit decisions. We find that this relation allows an adversary to learn about the MeNN's predictions by observing its exit pattern. This ability is useful when the attacker cannot view the model's classification directly due to a lack of physical device access and encrypted wireless communication (§3).

We demonstrate this property on a speech detection task [95]. We use a BranchyNet [86] MeNN with two total exits and a data-dependent policy (Equation 1) with $h_{MaxProb}$ [94]. When predicting the word "on," the MeNN exits early 61.61% of the time; when predicting "off," the early exit rate is 33.44%. Thus, early exiting means the MeNN is more likely to have predicted "on" than "off." An adversary observing these exit decisions can infer the presence of either word from this difference, indicating that data-dependent policies expose valuable information about the MeNN's predictions.

This asymmetric behavior stems from the prediction confidence having different distributions for different classes. In this example, the average confidence is 0.8328 for "on" and 0.7623 for "off." The policy, however, uses the *same* threshold for all inputs (Equation 1). Thus, instances of "on" are more likely to exit early through confidence scores above the single threshold, causing asymmetric behavior. This insight leads to a key novelty of our work: the use of multiple thresholds to account for distribution differences (§4, §5).

As we show, this privacy problem occurs on multiple tasks, MeNNs, and confidence functions (§6.2). The breadth of this leakage means the issue goes beyond a specific dataset, model architecture, or prior data-dependent policy. Further, we emphasize that MeNNs are not trained to exhibit this asymmetric behavior. Nevertheless, we empirically observe this phenomenon on every considered task.

2.4 Goals of Private Exit Policies

We design MeNN exit policies $\pi : \mathbb{R}^L \times \mathbb{N} \rightarrow \{0, 1\}$ to meet four criteria on ordered input streams $\tilde{D} = [\boldsymbol{x}^{(t)}]_{t=0}^{T-1}$ of length T. First, π should not leak information about the MeNN's predictions; there should be no observable relationship between the policy's decisions and the MeNN's classifications (P1). This quality should hold for all possible ordered streams $\tilde{D}$, as system designers cannot anticipate the exact stream at design time. This consideration encompasses datasets with temporal correlations and shifting input distributions.

Second, the policy must adhere to given exit rates $\{\rho_k\}_{k=0}^{K-1}$ where $\sum_{k=0}^{K-1} \rho_k = 1$. Under π, the MeNN should stop at exit k on $\rho_k \cdot T$ of inputs in $\tilde{D}$. This criterion is necessary to meet the resource limits of low-power devices (P2).

A purely randomized policy meets these two criteria by stopping at exit k with probability ρ_k. However, this method reduces the MeNN's inference accuracy (§6.4). Thus, the third goal is to build policies with better inference accuracy than random exiting (P3).

Finally, solutions must not require retraining or redesigning the MeNN (P4). This property is necessary because DNN training is expensive, and security solutions should work for existing MeNNs.

Policies under the definition π operate only on the MeNN's predictions. Thus, these policies satisfy this property, as π is not tightly coupled with the MeNN parameters. Indeed, Algorithm 1 shows how π can change without altering the MeNN.

We emphasize that a priori, it is not guaranteed that there exist MeNN exit policies that meet all four properties. *A key contribution of this work is demonstrating that such policies exist.*

3 THREAT MODEL

We first state the target system and the adversary's goal (§3.1). We then discuss the attacker's capabilities (§3.2) and present examples of distributed MeNNs requiring prediction privacy (§3.3).

3.1 Target System and Attack Goal

We consider a sensing system composed of edge devices and a centralized server [65]. Each device periodically captures measurements and processes these values using a distributed MeNN [55, 87], encrypting all communication. We call this MeNN the *target* model.

The adversary is a passive observer who uses the target MeNN's exit decisions to expose its predictions. Specifically, the attacker sees blocks of $B > 0$ exit decisions and uses an *attack model* $g_{\boldsymbol{\phi}} : [K]^B \rightarrow [L]$ to infer the MeNN's most frequent prediction in this block. Section 3.2 discusses how the attacker observes these exit decisions. The attacker uses a training phase to find the parameters $\boldsymbol{\phi}$ below where $v^{(t)}$ is the exit decision and $\hat{y}^{(k_t,t)}$ is the MeNN's prediction at time t. $MostFreq(\cdot)$ returns the most common argument value.

$$\hat{y}(t) = MostFreq(\hat{y}^{(k_t,t)}, \hat{y}^{(k_{t+1},t+1)}, .., \hat{y}^{(k_{t+B-1},t+B-1)}) \quad (2)$$

$$\boldsymbol{\phi} = \text{argmax}_{\tilde{\boldsymbol{\phi}}} \sum_{q=0}^{\lfloor \frac{T}{B} \rfloor - 1} [g_{\tilde{\boldsymbol{\phi}}}(v^{(qB)}, v^{(qB+1)}, .., v^{(qB+B-1)}) = \hat{y}(qB)]_1 \quad (3)$$

We assume the stream $\tilde{D}$ contains temporal correlations. Such correlations occur in sensor settings [21, 23]. On correlated streams, each block contains related inputs with similar labels. Correlated streams present a greater privacy challenge. Intuitively, correlated inputs cause MeNNs to make similar predictions and related exit decisions under data-dependent policies, allowing the adversary to view blocks of nearby decisions under one class. Streams with independent inputs prevent this temporal linkage. More precisely, an exit decision from one input gives the adversary one of K options to recover a prediction $\ell \in [L]$. For uncorrelated inputs, the attacker must view each decision in isolation; when $K < L$, this recovery is underdetermined. For correlated inputs, the attacker can link adjacent decisions under approximately the same prediction. With this ability, the attacker can use one of $K^B > K$ possible inputs to extract this block's most frequent prediction $\ell \in [L]$. Thus, the attacker can use more features on correlated streams. We emphasize that the adversary targets the MeNN's predictions instead of the true labels, as ground truth is unavailable at runtime. An adversary who infers the MeNN's results learns what the target system knows.

3.2 Adversary Capabilities

We assume the adversary targets a sensing system known to use a distributed MeNN. The attacker has no physical device access but can sniff the communication between the sensor and the server

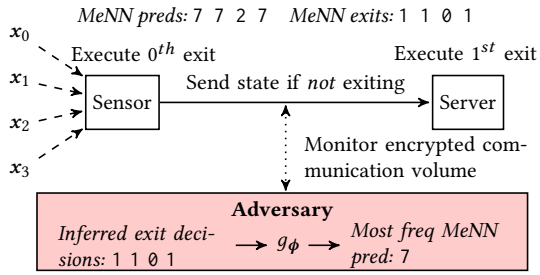

Figure 2: The threat model against distributed MeNNs. An exit decision of `0` means stopping on the sensor.

[5, 22]. These assumptions are realistic for wearable sensors [32, 54] and devices in remote locations [20, 43, 98]. The adversary cannot directly read the MeNN's predictions due to encrypted communication. Further, without physical access, the attacker cannot deploy their own sensor to derive equivalent insights. As we design exit policies, we follow Kerckhoffs's Principle [78] and allow the adversary to know the policy's details. We assume the attacker knows the sampling period and the target task's label space.

The adversary exposes the MeNN's predictions by inferring the model's exit decisions using communication side-channels. Below, we describe two examples of how distributed MeNNs leak the decision to exit early through communication patterns. This general pattern holds for all distributed MeNNs we are aware of.

DDNNs. Deep distributed neural networks (DDNNs) implement distributed MeNNs across a hierarchy of devices [87]. The system conserves resources by only transmitting information when continuing inference to the next device. An adversary can learn the exit decision using the presence or absence of wireless traffic.

SPINN. SPINN performs distributed MeNN inference under latency constraints [55]. When the MeNN partition point occurs after the first early exit, SPINN only communicates when not exiting early. This behavior creates the same side-channel as that of DDNNs. SPINN also supports partitions before the first exit. This setting still leaks information because SPINN requires the sensor to always compute a local prediction by continuing until the first early exit. If this exit signals termination, the sensor sends a second message to the server to stop computation. The attacker can infer an early exit using the presence of this second message.

In both cases, the passive adversary can use communication patterns to discover early exit behavior (Figure 2). This side-channel exists even when data is encrypted, as encryption does not obfuscate communication volume. This adversary only knows whether the system exits on the sensor or server. Thus, we focus on MeNNs with $K = 2$ exits where the sensor holds the initial exit. Our methods extend to MeNNs with $K > 2$ (§6.10).

The adversary uses the attack model $g_{\boldsymbol{\phi}}$ to infer the target MeNN's predictions from the extracted exit decisions. We consider two sets of assumptions for how the attacker fits the parameters $\boldsymbol{\phi}$ (Equation 3) from examples of blocks of exit decisions and MeNN predictions. In both cases, the attacker applies $g_{\boldsymbol{\phi}}$ at runtime to the target MeNN executing on an unseen testing dataset.

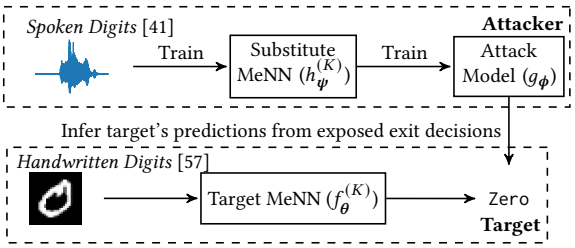

Figure 3: An example of the black box attacker.

(1) **White Box:** The white box adversary has access to an offline version of the target MeNN and training dataset. The attacker uses exit patterns and predictions from this MeNN on the target task to fit the attack model g_{ϕ}.
(2) **Black Box:** The black box adversary cannot access the target MeNN or training dataset. Instead, this adversary trains a substitute MeNN $h_{\psi}^{(K)}$ on a related task and fits g_{ϕ} using exit decisions and predictions from $h_{\psi}^{(K)}$ (Figure 3).

We use the black box adversary to confirm that the white box assumptions are not too strong.

3.3 Example Settings

This section describes two examples [7, 103] of distributed DNNs in applications requiring prediction privacy.

Sensitive facilities use DNNs to perform license plate recognition for authorized access [7]. Huawei's AutoSplit [7] framework applies distributed DNNs in this context. Prediction privacy is necessary, as the DNN's predictions indicate which license plates have access. The attacker could use leaked predictions to create fake credentials which pass the authorization check. Further, the server may be hosted off-premise (e.g., in the cloud). Thus, when the adversary cannot access the physical location without credentials, they can still sniff the communication to the remote server. This property means the system cannot leak information about authorization decisions through communication patterns.

Manufacturing plants apply DNNs to detect defective parts, and Boomerang [103] leverages distributed DNNs for this application. Prediction privacy is essential for two reasons. First, the defect rate indicates the manufacturer's efficiency. This information is valuable to competitors. Second, an adversary launching a supply-chain attack can use exposed DNN predictions to know whether their attack is successful. In both cases, an adversary may be unable to physically access all the validation points without alerting the building's security. Instead, the attacker can more easily observe the communication patterns to a single server, especially if the server is remote. Further, when validating a supply chain attack, the adversary wants to know if their inserted defect passes the target's inspection. Thus, rather than finding the true defect rate, it is more valuable to learn what the target system knows.

4 PER-CLASS EXITING (PCE)

Data-dependent MeNN policies using a single threshold can leak information through their exit decisions (§2.3). Our first solution, Per-Class Exiting (PCE), replaces the single confidence threshold with separate thresholds for each class (§4.1). With this design, PCE stops a fraction ρ_k of inputs for *every class* at exit k, thus preserving resource usage and improving privacy compared to prior work. Unfortunately, PCE has no theoretical privacy guarantees, and we construct adversarial orderings with high prediction leakage (§4.2).

4.1 Policy Design

We formally describe PCE using an MeNN $f_{\theta}^{(K)}$ with target exit rates $\rho_k \in [0,1] \ \forall k \in [K]$. Consider the prediction confidence function $h : \mathbb{R}^L \to \mathbb{R}$ (§2.2). Focusing on the k^{th} exit, PCE uses thresholds $\tau_{\ell}^{(k)}$ for each class $\ell \in [L]$ that satisfy the probability below. The terms $\hat{\boldsymbol{Y}}^{(k)} \in \mathbb{R}^L$ and Y are random variables for the k^{th} exit's prediction and the true label, respectively.

$$P(h(\hat{\boldsymbol{Y}}^{(k)}) \geq \tau_{\ell}^{(k)}, h(\hat{\boldsymbol{Y}}^{(r)}) < \tau_{\ell}^{(r)} \ \forall r \in [k] \mid Y = \ell) = \rho_k \quad (4)$$

Equation 4 states that for each label, inputs should stop at exit k with rate ρ_k. In practice, we fit the thresholds using the empirical confidence distributions on the task's validation set. PCE performs inference using Algorithm 1 with π_{PCE} below where $\hat{y}_i^{(k)}$ the predicted probability for class $i \in [L]$ at exit k. We omit the time $t \in [T]$, as the policy is stateless.

$$\ell(k) = \arg\max_{i \in [L]} \hat{y}_i^{(k)} \quad (5)$$

$$\pi_{PCE}(\hat{\boldsymbol{y}}^{(k)}, k) = [h(\hat{\boldsymbol{y}}^{(k)}) < \tau_{\ell(k)}^{(k)}]_1 \quad (6)$$

This design augments previous data-dependent exit policies (Equation 1) with different thresholds $\tau_{\ell}^{(k)}$ for each class. We emphasize that the policy selects thresholds using the MeNN's prediction at each exit. To protect the overall classification, PCE should instead choose thresholds using the final result i.e., use $\tau_{\ell}^{(k)}$ where $\ell = \arg\max_{i \in [L]} \hat{y}_i$ and $\hat{\boldsymbol{y}}$ is the MeNN's final predicted probabilities. At an early exit, however, we do not know the final prediction $\hat{y}$ when *not* terminating inference. PCE thus uses the current exit's prediction $\hat{y}^{(k)}$ as an approximation.

4.2 Adversarial Data Orderings

PCE uses a data-dependent approach that fits thresholds such that the *overall* exit rates for each label are ρ_k. This behavior means that PCE delivers good privacy on uncorrelated input orders without introducing randomization, as such streams only require long-term balancing (§6.5.2). However, this long-term balancing makes no guarantees about eliminating short-term patterns.

This insight suggests there exist input orders causing high leakage. We formalize this idea in Proposition 4.1 below. The proof describes how to build the adversarial ordering $\tilde{D}$ and attack model g_{ϕ} (§3.1). As confirmation, we follow the proof and build an adversarial ordering with inputs from Fashion MNIST [99]. We attack a two-exit MeNN with PCE ($\rho_0 = 0.9$). As expected, the adversary infers 100% of the MeNN's predictions. Thus, despite lower empirical leakage than prior methods (§6.2), PCE delivers no privacy guarantees on correlated inputs. Instead, PCE better protects

MeNNs processing unrelated inputs over time. We emphasize that Proposition 4.1 applies to any deterministic policy, not just PCE. We omit the exit index $k \in [K]$ from π below because we consider a two-exit MeNN, which only applies π at exit $k = 0$ (Algorithm 1).

Proposition 4.1. *Let $\pi : \mathbb{R}^L \to \{0,1\}$ be a deterministic policy on a two-exit MeNN $f^{(2)}_{\boldsymbol{\theta}}$ for a rational exit rate $\rho_0 = \frac{m}{M} \in (0,1)$. Let D be the set of possible samples $(\boldsymbol{x}, y)$, $D_x = \{\boldsymbol{x} : \exists \ell \in [L], (\boldsymbol{x}, \ell) \in D\}$ be the inputs, and $D_\ell = \{\boldsymbol{x} : (\boldsymbol{x}, \ell) \in D\}$ be the inputs with label ℓ. Assume $\forall k \in \{0,1\}$, $\pi^{-1}(k) \cap D_\ell \neq \emptyset$ where $\pi^{-1}(k) = \{\boldsymbol{x} \in D_x : \pi(f^{(0)}_{\boldsymbol{\theta}_0}(\boldsymbol{x})) = k\}$. Then, there exists an ordered dataset $\tilde{D} = [(\boldsymbol{x}^{(t)}, y^{(t)}) \in D]_{t=0}^{T-1}$ for $T = n \cdot LM$, $n > 1$ with the following.*

(1) The policy π exhibits an early exit rate of ρ_0 on $\tilde{D}$.
(2) There exists a function $g_{\boldsymbol{\phi}} : \{0,1\}^{LM} \to [L]$ such that

$$g_{\boldsymbol{\phi}}(\pi(\hat{\boldsymbol{y}}^{(t)}), \ldots, \pi(\hat{\boldsymbol{y}}^{(t+LM-1)})) = MostFreq(y^{(t)}, \ldots, y^{(t+LM-1)})$$

where $t = j \cdot LM$ for any $j \in [n]$.
(3) Not all non-overlapping blocks of LM exit decisions (in property 2) have the same most frequent label.

Proof. See Appendix §B.1 □

5 CONFIDENCE-GUIDED RANDOMNESS (CGR)

PCE uses a modified data-dependent method to balance the long-term exit rates across classes. This method, however, does not address temporal correlations. In fact, Proposition 4.1 indicates that temporal dependencies can compromise any deterministic policy. Thus, we must obfuscate temporal patterns through randomization. Unfortunately, exiting early uniformly at random achieves poor inference accuracy (§6.4).

We instead present a new policy, Confidence-Guided Randomness (CGR), that is an interpolation between PCE and uniformly randomized exiting, merging the benefits of both approaches. When detecting uncorrelated inputs, CGR behaves like PCE to achieve higher accuracy. On segments with high correlations, CGR applies greater randomization to maintain privacy.

CGR has three features. First, the policy evaluates PCE and uses its decision to create a *confidence-biased* exit probability which determines the exiting behavior (§5.1). Second, CGR *adapts* the bias magnitude on highly-correlated streams (§5.2). Finally, CGR enforces *exit quotas* over short windows to limit an adversary's ability to discover data-dependent information (§5.3). By employing randomness, CGR achieves theoretical benefits over PCE (§5.4).

5.1 Confidence-Biased Randomization

CGR uses PCE as an internal data-dependent method (Figure 4). When reaching the k^{th} exit, CGR builds an exit probability that is biased toward $\pi_{PCE}(\hat{\boldsymbol{y}}^{(k,t)}, k)$ at step $t \in [T]$. CGR samples this biased probability to make a random exit decision. Thus, CGR leverages prediction confidence through PCE, enabling higher inference accuracy than pure randomization.

We formalize this design by considering the k^{th} exit of an MeNN $f^{(K)}_{\boldsymbol{\theta}}$ with exit rate $\rho_k \in [0,1]$. CGR uses a bias $\alpha^{(k,t)} \in [0,1)$ (discussed in §5.2) at step t to make randomized decisions with the following probabilities. The rates for $\pi_{CGR}(\hat{\boldsymbol{y}}^{(k,t)}, k) = 1$ are one minus those below. We omit the exit indices for brevity.

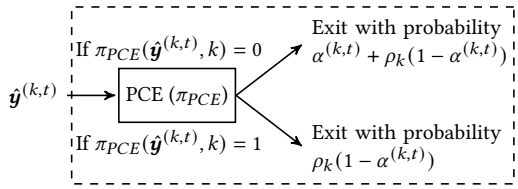

Figure 4: CGR's decision process at the k^{th} exit point.

$$P(\pi_{CGR}(\hat{\boldsymbol{y}}^{(k,t)}) = 0 | \pi_{PCE}(\hat{\boldsymbol{y}}^{(k,t)}) = 0) = \alpha^{(k,t)} + \rho_k(1 - \alpha^{(k,t)}) \quad (7a)$$

$$P(\pi_{CGR}(\hat{\boldsymbol{y}}^{(k,t)}) = 0 | \pi_{PCE}(\hat{\boldsymbol{y}}^{(k,t)}) = 1) = \rho_k(1 - \alpha^{(k,t)}) \quad (7b)$$

These equations show how the exit probabilities are biased in the direction of PCE. For example, when $\pi_{PCE}(\hat{\boldsymbol{y}}^{(k,t)}, k) = 0$, CGR exits with rate $\alpha^{(k,t)} + \rho_k(1 - \alpha^{(k,t)}) = \rho_k + \alpha^{(k,t)}(1 - \rho_k) \geq \rho_k$ where the inequality holds because $0 \leq \rho_k, \alpha^{(k,t)} \leq 1$. Thus, CGR aligns with the PCE's decision and exits more frequently than the target rate ρ_k. Note that these probabilities maintain an overall exit rate of ρ_k when $P(\pi_{PCE}(\hat{\boldsymbol{Y}}^{(k,t)}, k) = 0) = \rho_k$. This condition holds when the test distribution matches that of the training set.

5.2 Adapting the Probability Bias Magnitude

The probability biases control a tradeoff between accuracy and privacy. If $\alpha^{(k,t)} \approx 1$, CGR skews toward PCE, yielding high inference accuracy with possible leakage (§4, §A.2). If $\alpha^{(k,t)} = 0$, CGR is fully randomized. CGR balances this tradeoff using the insight that PCE provides good privacy on uncorrelated input streams. Thus, CGR exploits periods of low correlation by leveraging PCE to achieve high accuracy. On highly correlated segments, CGR applies more randomness to ensure privacy. To protect the MeNN's predictions (§2.4), CGR measures correlations using the model's results.

CGR implements this design by adaptively setting the bias $\alpha^{(k,t)}$ with the parameters $\gamma < 1 < \beta$. The policy has a maximum bias of $\alpha \in [0,1)$. CGR sets $\alpha^{(k,t)}$ at step $t \geq 1$ as follows where $\alpha^{(k,0)} = \alpha$.

$$\alpha^{(k,t)} = \begin{cases} \min(\beta \cdot \alpha^{(k,t-1)}, \alpha) & \text{if } \hat{y}^{(k,t)} \neq \hat{y}^{(k,t-1)} \\ \gamma \cdot \alpha^{(k,t-1)} & \text{if } \hat{y}^{(k,t)} = \hat{y}^{(k,t-1)} \end{cases} \quad (8)$$

The parameters β and γ control how the bias changes in response to the MeNN's predictions. For example, when $\beta >> 1$ and $\gamma \approx 1$, CGR will quickly increase and slowly decrease the bias. This behavior causes CGR to have a higher *average* bias, thereby aligning more with PCE than pure randomization (§A.2). Based on our experiments, $\beta = 2.0$ and $\gamma = 0.9$ provide favorable results, and these settings are robust across multiple datasets and data orders. Overall, CGR uses this procedure to adapt the bias to offset temporal trends in the MeNN's predictions. We emphasize that CGR still makes randomized decisions even when using a high bias $\alpha^{(k,t)} < 1$.

5.3 Short-Term Exit Quotas

CGR may leak predictions if the attacker infers the policy's biased probabilities, as these probabilities encode PCE's decisions (Equation 7). CGR protects against this leakage by enforcing exit quotas over short windows. The quotas ensure the MeNN stops a set number of times at each exit point, balancing the exit counts and

reducing the adversary's analysis to smaller sample sizes. For each window, the attacker can only extract the bias direction before an exit quota becomes saturated; afterward, the policy never stops at this exit and uses no information from PCE. Exposing small samples benefits privacy because it limits the adversary's ability to derive meaningful statistical significance on biased exit rates.

CGR implements exit quotas using a window $W \in \mathbb{N}$. The policy enforces that $\omega_k = \lfloor \rho_k \cdot W \rfloor + \eta_k$ inputs stop at exit k where $\eta_k \in \{0, 1\}$ are random such that $\sum_{k=0}^{K-1} \omega_k = W$. The system no longer exits at output k upon meeting its quota. After each window, the policy resets the quotas and randomly selects a new $W \sim [W_{min}, W_{max}]$ where the bounds are parameters. This randomization limits the adversary's ability to locate each window.

Randomizing the window size, however, does not fully prevent the attacker from discovering when CGR uses biased probabilities. For example, the adversary can use a run of exits at a single output to infer a window's end. The theoretical privacy of this attack is not well-established. However, we believe this attack does not present a problem for three reasons. First, the adversary's recovery of each window is only approximate due to randomization. Second, CGR already protects against periods of high potential leakage by adapting its bias parameter (§5.2). Finally, CGR's empirical information leakage is statistically equivalent to random exiting (§6.2, §6.5).

5.4 Theoretical Benefits

This section demonstrates CGR's theoretical benefits. The main result is Proposition 5.1 which establishes a bound on the probability ratio of finite exit patterns from inputs of different classes. This result improves upon PCE, which can leak an unbounded amount of information over finite time horizons (Proposition 4.1). A key property of CGR is that its bounds apply to *any* data stream, including those with temporal correlations and distributional shifts.

Proposition 5.1 further provides a guideline on how to set α from a security perspective, as α determines the difference between the upper and lower probability bounds. However, CGR will not be tight with the established bounds because the policy uses biases $\alpha^{(k,t)} < \alpha$ (§6.5) due to similar predictions over time (§5.2). Smaller biases create narrower probability bounds, allowing CGR to provide better empirical privacy than the proposition guarantees.

Before presenting the proposition, we introduce relevant notation. Let π_r be the CGR policy without exit quotas and $\delta, \epsilon : \mathbb{R} \to \mathbb{R}$ be functions such that $\delta(\alpha) = (1 - \alpha - \rho_0(1-\alpha))/\rho_0$ and $\epsilon(\alpha) = (1 - \rho_0(1-\alpha))/\rho_0$ where ρ_0 is the exit rate for $K = 2$. We define these functions for notational convenience.

Proposition 5.1. *Consider a two-exit MeNN with a target early exit rate ρ_0. Suppose we observe a sequence of T exit decisions $\pi_r^{(t)} := \pi_r(\hat{\boldsymbol{y}}^{(0,t)}, 0) = v_t$ for $v_t \in \{0, 1\}$ and $t \in [T]$. Let these T inputs belong to the same class and $n = \sum_{t=0}^{T-1} v_t$. Then, π_r displays the following bounds for any labels $\ell_0, \ell_1 \in [L]$.*

$$\left(\frac{\delta(\alpha)}{\epsilon(\alpha)}\right)^n \left(\frac{1-\rho_0\epsilon(\alpha)}{1-\rho_0\delta(\alpha)}\right)^{T-n} \leq \frac{P(\pi_r^{(t)} = v_t \forall t | Y = \ell_0)}{P(\pi_r^{(t)} = v_t \forall t | Y = \ell_1)}$$

$$\left(\frac{\epsilon(\alpha)}{\delta(\alpha)}\right)^n \left(\frac{1-\rho_0\delta(\alpha)}{1-\rho_0\epsilon(\alpha)}\right)^{T-n} \geq \frac{P(\pi_r^{(t)} = v_t \forall t | Y = \ell_0)}{P(\pi_r^{(t)} = v_t \forall t | Y = \ell_1)}$$

Proof. See Appendix §B.2. □

Table 1: Dataset properties.

Dataset	# Train	# Val	# Test	# Classes
Activity [4]	36,790	13,629	20,441	6
Cifar10 [52]	39,796	10,204	10,000	10
Cifar100 [52]	39,796	10,204	10,000	100
EMNIST [17]	87,800	25,000	18,800	47
Fash. MNIST [99]	47,798	12,202	10,000	10
Food Quality [33]	2,945	196	198	2
GTSRB [83]	38,580	10,799	9,120	43
MNIST [57]	47,798	12,202	10,000	10
Speech Cmds [95]	28,532	3,457	4,482	11
WISDM [54]	32,005	5,858	21,769	3

One consequence of Proposition 5.1 results from the bounds having the form $(q_0)^n(s_0)^{T-n}$ and $(q_1)^n(s_1)^{T-n}$ where $q_0, s_0 < 1 < q_1, s_1$. As $T \to \infty$, the bounds go to zero and infinity, respectively. Thus, the biased probabilities can cause unbounded exit rate differences over an *infinite* horizon. CGR prevents worst-case scenarios by avoiding highly biased rates over long windows, confirming the benefits of the adaptive bias procedure and use of exit quotas.

6 EVALUATION

We evaluate the information leakage and inference accuracy of distributed MeNNs using previous data-dependent policies and our proposed methods. In summary, we find the following:

(1) Standard data-dependent policies leak information about MeNN predictions through their exit patterns. This leakage occurs from practical (§6.2) and theoretical (§6.3) perspectives. PCE uses multiple thresholds to reduce this leakage, and CGR obtains near-perfect privacy.
(2) PCE and CGR use prediction confidence to display higher inference accuracy than a fully randomized policy (§6.4).
(3) PCE has higher leakage on input streams with stronger correlations (§6.5). CGR adapts itself to protect against these trends, showing near-random leakage under temporal correlations and distribution shifts (§6.6).
(4) Single-threshold data-dependent policies still leak valuable information to a *black box* attacker who has no access to the MeNN and training dataset (§6.7).
(5) On a realistic distributed MeNN setup, prior data-dependent policies leak predictions to an attacker with access to encrypted communication patterns (§6.8). Our methods provide protection in this end-to-end setting.
(6) PCE and CGR show negligible overhead on a low-power microcontroller (MCU) (§6.9). Thus, our policies improve privacy while retaining the efficiency of MeNNs.
(7) Prior data-dependent policies leak more information on MeNNs with more exits (§6.10). In contrast, CGR protects MeNNs independent of the number of exit points.

6.1 Experimental Setup

6.1.1 Datasets and Neural Network Parameters. We evaluate MeNNs on ten standard tasks (Table 1) covering many input types and label spaces. We use BranchyNet [86] MeNNs, focusing on models with two total exits. On Cifar [52], we use VGG models [81] with early

exiting after the first pooling layer. We use pre-trained versions of each VGG model; we attach early exits and fine-tune these output layers [48]. We apply dense models on Activity [4], Food Quality [33], and WISDM [54] and convolutional networks on the remainder. These MeNNs have four hidden layers, with early exiting after the first. We execute at most ten training epochs using Adam (step size of 10^{-3}) [49], a batch size of 16, and a dropout [82] rate of 0.3.

For two-exit MeNNs, we run policies on 21 exit rates $\rho_0 = 0.0, 0.05, \ldots, 1.0$. We use a tighter range for MeNNs with more exits (§6.10). The datasets are unordered, and we use two methods to create the temporal correlations present in sensor settings.

(1) *Same-Label* builds blocks of size B by selecting $(1-\varepsilon)\cdot B$ random elements of a single label and $\varepsilon \cdot B$ inputs from arbitrary classes. We set $B = 10$ and $\varepsilon = 0.2$. Appendix §A.1 considers alternate settings.
(2) *Nearest-Neighbor* constructs B-sized blocks by choosing a random anchor element and using the anchor's $B-1$ nearest neighbors [10] in order.

We focus on Same-Label orders, as Nearest-Neighbor produces weak correlations on inputs such as colored images.

6.1.2 Exit Policies. We use following baseline exit policies.

(1) *Random* selects the MeNN exit uniformly at random using the rates ρ_k. This policy has perfect privacy because it makes *data-independent* decisions.
(2) *Entropy* is a data-dependent method (Equation 1) with confidence $h_{Entropy}(\hat{\boldsymbol{y}}) = (-\sum_{i=0}^{L-1} \hat{y}_i \log(\hat{y}_i))^{-1}$ [86].
(3) *MaxProb* is a data-dependent policy (Equation 1) with confidence $h_{MaxProb}(\hat{\boldsymbol{y}}) = \max_{i\in[L]} \hat{y}_i$ [94].

We apply PCE and CGR to both confidence functions. We use CGR with a maximum bias of $\alpha = 0.5$, adaptation factors of $\beta = 2.0$ and $\gamma = 0.9$, and a window range of $[5, 20]$ (§5). We set these parameters using a grid search (§A.2) and fix them for all datasets.

We fit confidence thresholds τ for an exit rate ρ by setting τ to the $(1-\rho)^{th}$ quantile of the MeNN's confidence values on the task's validation set. We execute both Random and CGR over five trials as these policies have stochastic behavior.

6.1.3 Adversary Design. We design the attack model $g_{\boldsymbol{\phi}}$ using an AdaBoost ensemble with 100 decision trees. The attack model uses non-overlapping blocks of B adjacent exit decisions to infer the MeNN's most frequent prediction in this block. This block size matches that of the dataset order. We create two variants with different input features. The first uses each exit point's frequency, and the second uses the exact exit pattern. We use the frequencies against Same-Label orders and the patterns on Nearest-Neighbor streams (§6.1.1). The exact pattern leads to overfitting on Same-Label orders. We train $g_{\boldsymbol{\phi}}$ with two different assumptions (§3.2).

(1) **White box** attackers use patterns from the *target* MeNN's on the *original* task's validation set.
(2) **Black box** adversaries use patterns from a *substitute* MeNN on a *related* task's validation set.

We always evaluate $g_{\boldsymbol{\phi}}$ on the target MeNN processing the test fold.

6.1.4 Aggregate Metrics. We evaluate policies on many sets of target exit rates. For inference accuracy, we compute the average result across all targets. For privacy metrics, we compute show the worst-case result by calculating the maximum over the targets, as policies should not leak information for *any* target exit rates (P1). This methodology aligns with prior work in measuring security from a worst-case perspective [14]. We aggregate trials by taking the average trial result for each target exit rate.

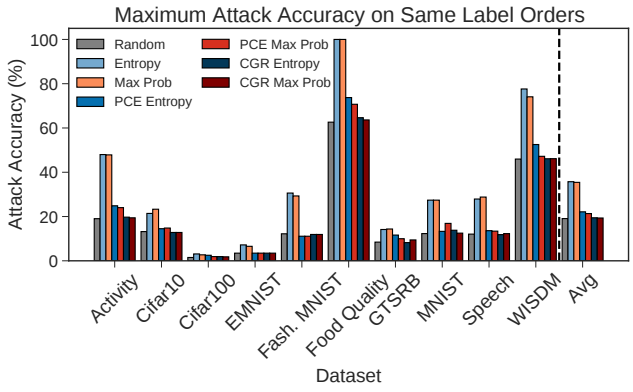

Figure 5: Maximum attack accuracy across 21 exit rates (lower is better).

6.1.5 Hardware Setup. We conduct experiments in simulation and on a low-power microcontroller (MCU). The simulator runs MeNNs in Tensorflow [1] and records the predictions and exit decisions (§6.2-§6.7, §6.10). The adversary observes the exact exit decisions.

We perform an end-to-end side-channel attack on a two-exit distributed MeNN [87] with the initial exit on a TI MSP430 FR5994 MCU [40] and the remaining model on a server (§6.8). The MCU processes inputs every second and uses Bluetooth Low Energy (BLE) to transmit the intermediate state when not exiting early. This setup follows DDNNs [87] and SPINN [55] on a two-device system. The sensor applies AES-128 encryption [18]. We capture the encrypted packets using Wireshark [71] and provide this log to the attacker. We drop 5% of packets to simulate a lossy link. The attacker only observes traffic when the system does *not* exit early. The adversary finds the number of early exits between transmissions as follows, where R is the sampling period and d_n is the time of the n^{th} message.

$$\texttt{num\_early\_exits}(n) = \max(\lfloor (d_n - d_{n-1})/R \rfloor - 1, 0) \quad (9)$$

We emphasize that the side-channel attack applies to variants of this system; e.g., if the sensor sends predictions when exiting early, the attacker can infer early exits using differences in message sizes.

6.2 White Box Attack

We first measure the practical privacy of MeNN exit policies under white box assumptions (§6.3 discusses theoretical leakage). We use the white box methodology in §6.1.3 to infer predictions from two-exit MeNNs with Same-Label orders ($B = 10, \varepsilon = 0.2$).

Figure 5 shows the maximum attack accuracy across all target exit rates. The *attack accuracy* is the accuracy of the attack model $g_{\boldsymbol{\phi}}$; this metric measures the fraction of the adversary's predictions that match the MeNN's most common classification in each temporal block. These results display how standard data-dependent policies consistently leak information about MeNN predictions. Across all

tasks, these policies show mean worst-case attack accuracy that is 1.87× (Entropy) and 1.85× (Max Prob) higher than Random. On the Food Quality task, the attacker infers up to 100% of the MeNN's most frequent predictions in each block. Further, both methods display worse privacy than Random on all tasks. Thus, this leakage does not result from a single dataset or confidence function; instead, it comes from the data-dependent approach of previous methods.

Both PCE and CGR display better privacy. PCE yields 1.16× (Entropy) and 1.12× (Max Prob) higher worst-case attack accuracy than Random. These values are lower than that of prior data-dependent policies (Figure 5), showing the benefit of multiple thresholds. CGR performs even better, displaying 1.02× higher worst-case attack accuracy averaged across all tasks. We further compare these policies to Random using Welch's t-test. The null hypothesis is the attack accuracy normalized to the most frequent MeNN prediction is no different than that of Random. With this methodology, Random and CGR have an *insignificant* difference at the 0.05 level with p-values of 0.21 (CGR Entropy) and 0.09 (CGR Max Prob). Thus, CGR obtains near-random privacy independent of the confidence function. Entropy, Max Prob, and PCE show significant differences.

Entropy and Max Prob show low attack accuracy on Cifar100, as this dataset has a large label space (Table 1). However, we observe leakage through the attacker's average rank (AR) of the correct class [6]. On Cifar100, Entropy and Max Prob have a worst-case AR of 33.60 and 35.77, respectively. These values are far lower than Random (48.20); thus, attackers still learn valuable information on tasks with many labels. Across all datasets, CGR has an AR of 0.99× Random, further demonstrating its near-random privacy.

6.3 Theoretical Information Leakage

We supplement the practical attack with an analysis agnostic of the attack model. We quantify privacy using the empirical normalized mutual information (NMI) [53] between the MeNN's exit decisions and predictions. A high NMI means observing the exit decisions reduces the adversary's uncertainty about the model's predictions. A policy with no leakage should exhibit an NMI close to Random. We use the definition $NMI(X,Y) = (2 \cdot I(X,Y))/(H(X) + H(Y))$ where $I(\cdot)$ is the mutual information and $H(\cdot)$ is the Shannon entropy. We measure the NMI by comparing individual exit decisions (X) and MeNN predictions (Y). This metric does not depend on temporal correlations. We reduce the empirical NMI's bias with Miller-Madow correction [73]. We use the same setup as §6.2.

Table 2 shows the maximum NMI, confirming the trends in §6.2. Standard data-dependent policies show high NMI with values up to 0.1725 points higher than Random on average. Further, both policies eclipse Random on all tasks, indicating that this leakage is consistent and independent of the confidence function.

PCE improves privacy, showing an NMI of up to 0.0203 points higher than Random; this rate is over 8.4× lower than previous data-dependent methods (Table 2). On average, CGR has a maximum NMI of only 0.0009 points above Random. This figure is over 191× lower than prior data-dependent policies. These results provide additional evidence that CGR has near-perfect privacy.

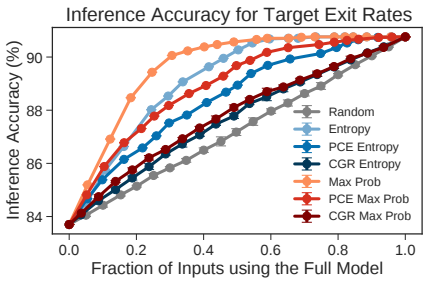

Figure 6: Inference accuracy (%) on the Activity dataset. Error bars show the standard deviation over five trials.

6.4 Inference Accuracy

The second tradeoff dimension we investigate is inference accuracy, as accuracy represents the MeNN's answer quality. We use two-exit MeNNs under the setup in §6.2.

Table 3 shows the average MeNN accuracy across all exit rates, and Figure 6 displays the results on the Activity task. We have three takeaways. First, Random has a high accuracy penalty; existing policies achieve an average accuracy of 2.03 (Entropy) and 2.30 (Max Prob) points above Random. Second, PCE retains high MeNN accuracy, showing values within 0.5 points of its standard data-dependent variant on seven of ten datasets. Random achieves this mark only twice. Finally, CGR consistently outperforms Random on all tasks with an overall average accuracy of 0.51 (Entropy) and 0.57 (Max Prob) points higher. For $\rho_0 \in (0,1)$, CGR eclipses Random on 90% (171 / 190) of target rates under both confidence functions.

We note two additional results. First, an alternate method to eliminate leakage is to use a fixed policy that always exits at the same point. This baseline must use the early exit to meet resource limits (§6.9), resulting in low accuracy. From Figure 6, the early exit has an accuracy of about 83%; all other policies reach an average accuracy above 87% (Table 3). Thus, PCE and CGR show better accuracy under resource limits than a fixed policy. Second, Entropy and Max Prob perform poorly on WISDM. This result comes from suboptimal exit decisions due to MeNN overconfidence. PCE corrects this problem by setting higher thresholds for the overconfident classes, highlighting an alternative benefit of using multiple thresholds.

6.5 Alternate Dataset Orders

6.5.1 Nearest Neighbor. We further evaluate privacy using the white box adversary (§3.2) on Nearest-Neighbor orders (§6.1.1) with $B = 10$ (Figure 7). CGR maintains its near-random privacy, showing 0.99× higher attack accuracy than Random on average. Using the methodology of §6.2, CGR's attack accuracy is *not* significantly greater than Random. In contrast, PCE shows higher leakage on Nearest-Neighbor orders, displaying an average worst-case attack accuracy that is 1.24× (Entropy) and 1.28× (Max Prob) higher than Random. These values exceed the 1.12× (Entropy) and 1.18× (Max Prob) marks from the Same-Label order on these four tasks. This greater leakage comes from the Nearest-Neighbor order's high correlations. Nevertheless, PCE still displays better privacy than single-threshold techniques.

Table 2: Maximum empirical normalized mutual information (NMI) (all values $\times 10^{-2}$) between exit decisions and MeNN predictions across 21 target rates (lower is better). The final row shows the average (std dev) difference compared to Random.

Dataset	Rand	Entropy			Max Prob		
		Stnd	*PCE*	*CGR*	*Stnd*	*PCE*	*CGR*
Activity	0.25	33.99	2.33	0.35	34.22	2.11	0.40
Cifar10	0.25	5.43	0.66	0.31	5.24	0.76	0.29
Cifar100	0.45	4.79	1.21	0.50	4.10	1.25	0.52
EMNIST	0.07	7.19	0.31	0.09	7.29	0.30	0.09
Fash. MNIST	0.04	19.84	0.28	0.04	19.83	0.25	0.04
Food Quality	0.24	49.62	3.73	0.11	49.62	5.73	0.33
GTSRB	1.26	8.92	5.26	1.59	9.23	4.45	1.46
MNIST	0.06	6.30	0.85	0.05	6.29	2.44	0.15
Speech Cmds	0.33	16.15	3.17	0.50	16.29	3.22	0.48
WISDM	0.42	23.61	3.32	0.58	18.77	3.14	0.55
Avg Diff v Rand	0.00 (0.00)	17.25 (14.06)	1.78 (1.38)	0.08 (0.12)	16.75 (14.01)	2.03 (1.58)	0.09 (0.06)

Table 3: Average (std dev) inference accuracy across 21 target exit rates for each policy and task (higher is better). The standard deviation shows the variation in the average accuracy across five independent trials.

Dataset	Rand	Entropy			Max Prob		
		Stnd	*PCE*	*CGR*	*Stnd*	*PCE*	*CGR*
Activity	87.22 (0.02)	89.24	88.46	87.57 (0.01)	89.69	88.85	87.70 (0.02)
Cifar10	80.35 (0.04)	85.18	84.83	81.69 (0.03)	85.50	85.17	81.78 (0.01)
Cifar100	61.68 (0.05)	64.47	64.43	62.40 (0.04)	64.81	64.79	62.52 (0.02)
EMNIST	85.92 (0.02)	87.28	87.35	86.37 (0.01)	87.31	87.37	86.39 (0.01)
Fash. MNIST	90.37 (0.03)	91.59	91.38	90.69 (0.03)	91.66	91.46	90.72 (0.01)
Food Quality	97.01 (0.02)	97.34	97.34	97.09 (0.01)	97.34	97.34	97.09 (0.01)
GTSRB	80.45 (0.04)	84.85	83.21	81.21 (0.04)	85.37	83.74	81.39 (0.05)
MNIST	98.64 (0.01)	99.19	99.19	98.83 (0.01)	99.19	99.19	98.83 (0.01)
Speech Cmds	85.94 (0.04)	88.58	87.76	86.43 (0.02)	88.78	87.92	86.40 (0.01)
WISDM	86.31 (0.02)	86.48	87.81	86.68 (0.01)	87.24	87.93	86.77 (0.02)
Avg Diff v Rand	0.00 (0.00)	2.03 (1.55)	1.79 (1.18)	0.51 (0.34)	2.30 (1.63)	1.99 (1.31)	0.57 (0.38)

Table 4: Average (std dev) inference accuracy across 21 exit rates for Nearest-Neighbor blocks (higher is better).

Dataset	Rand	MaxProb		
		Stnd	*PCE*	*CGR*
Activity	87.46 (0.02)	89.99	89.17	87.51 (0.01)
EMNIST	87.10 (0.01)	88.04	88.10	87.38 (0.02)
Fash. MNIST	91.11 (0.03)	92.36	92.09	91.34 (0.01)
MNIST	99.44 (0.01)	99.67	99.68	99.48 (0.01)
Avg Diff v Rand	0.00 (0.00)	1.06 (0.57)	0.95 (0.48)	0.17 (0.09)

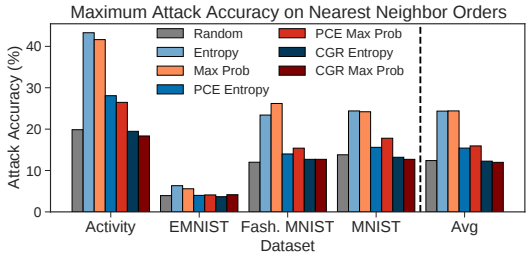

Figure 7: Maximum attack accuracy on Nearest-Neighbor dataset orders (lower is better).

CGR continues to show improved inference accuracy (Table 4), eclipsing Random on 80% (61 / 76) of target exit rates under the Max Prob metric. These results are similar with the Entropy function. However, the gap between CGR and Random is smaller than on Same-Label streams (Table 3). This difference results from CGR's adaptive bias. The Nearest-Neighbor order contains stronger correlations, often having blocks with over 90% of elements in the same class. In turn, CGR acts more randomly. For example, on the Activity task with $\rho_0 = 0.5$, CGR has an average bias of 0.1507 on Nearest-Neighbor and 0.4446 on Same-Label. CGR properly responds to greater correlations by reducing its bias magnitude.

6.5.2 Uncorrelated. We further evaluate the attack on data streams with randomly-ordered inputs. In this setting, the adversary uses each exit decision to infer the MeNN's individual predictions. Table 5 compares the average inference accuracy and maximum attack accuracy on the Activity task for Uncorrelated and Nearest-Neighbor streams. The latter order exhibits the strongest temporal relations. The adversary still learns information about MeNN's results in uncorrelated settings, though the attack efficacy is lower. We hypothesize this result occurs because the adversary has less context

Table 5: Mean inference accuracy (Infr. Acc.) and maximum attack accuracy (Att. Acc.) on the Activity task for Uncorrelated and Nearest-Neighbor orders.

Policy	Uncorrelated		Nearest-Neighbor	
	Infr. Acc.	*Att. Acc.*	*Infr. Acc.*	*Att. Acc.*
Random	86.86	18.12	87.46	19.86
Max Prob	89.44	36.59	89.99	41.63
PCE Max Prob	88.57	21.14	89.17	26.47
CGR Max Prob	87.36	18.88	87.51	18.35

on uncorrelated orders (§3.1). This comparison further shows the benefits of PCE in isolation. On uncorrelated streams, PCE displays an attack accuracy within 3 points of Random. This figure is over 2× smaller than PCE's gap to Random on the Nearest-Neighbor order. Thus, for uncorrelated streams, PCE delivers an inference accuracy of 1.7 points above Random for a small cost in privacy. We note that PCE does not achieve an attack accuracy equivalent to either Random or CGR because PCE uses static thresholds fit on a training set, and the testing set contains empirical differences.

6.6 Distribution Shifts

Sensing systems often face distribution shifts where the data observed at runtime differs from that used during training [51]. We evaluate the privacy impact of distribution shifts by constructing an alternate testing set for MNIST [57] using the first 10,000 digits from the Extended MNIST dataset [17]. This alternate dataset has a higher mean ($\mu = 0.172$) and standard deviation ($\sigma = 0.331$) pixel value than that of MNIST ($\mu = 0.131, \sigma = 0.308$) due to differences in image preprocessing. We use the same MeNN trained on MNIST as in §6.2 and evaluate the MeNN on this alternate testing set.

Table 6 shows the white box attack accuracy with the Same-Label order. Under distributional shifts, PCE has privacy similar to standard data-dependent exiting. This phenomenon occurs because the shifted distribution changes the MeNN's prediction confidence, breaking the balancing effect of PCE's multiple thresholds. For example, when $\rho = 0.75$, PCE exits early on 96.88% of the digit 1 and 55.08% of the digit 7 in the shifted test set; on standard MNIST, these exit rates are 73.03% and 72.49%, respectively. Thus, under shifted distributions, PCE shows the same asymmetric exit behavior seen in previous data-dependent methods. In contrast, CGR protects against this issue by leveraging randomness. With the Entropy metric, CGR displays a worst-case attack accuracy less than that of Random. Along with this privacy benefit, CGR shows higher mean inference accuracy on the shifted test set. CGR has an average accuracy of 88.33% (Entropy) and 88.30% (Max Prob), compared to 87.07% for Random. Note that the shifted distribution causes lower MeNN inference accuracy overall (Table 3). This finding aligns with prior work on neural networks facing distributional shifts [51].

6.7 Black Box Attack

We confirm the white box assumptions are not too strong by considering a weakened attacker with black box access (§3.2). This adversary cannot access the target MeNN and only knows the number of MeNN exits K and the target task's label space (e.g., the digits 0-9 for MNIST). The adversary uses this knowledge to select

Table 6: Worst-case attack accuracy for exit policies on an MeNN trained on MNIST and tested on either a shifted distribution (EMNIST Digits) or the same distribution (MNIST).

Policy	EMNIST Digits	MNIST
Random	13.60	12.50
Entropy	21.80	27.40
Max Prob	20.10	27.40
PCE Entropy	19.30	13.30
PCE Max Prob	21.10	16.90
CGR Entropy	13.50	13.80
CGR Max Prob	14.20	12.50

a related dataset with the same label space. The attacker trains a substitute MeNN $h_{\boldsymbol{\psi}}^{(K)}$ on this related dataset (Figure 3). We assume the attacker uses a reasonable MeNN architecture for their selected dataset (e.g., ResNet [31] on Cifar-10). The adversary trains their MeNN by optimizing the average individual classification loss of each exit point [86]. Finally, following black box adversarial DNN attacks [74], the attacker fits an attack model $g_{\boldsymbol{\phi}}$ on patterns from the substitute $h_{\boldsymbol{\psi}}^{(K)}$ and applies $g_{\boldsymbol{\phi}}$ to the target MeNN $f_{\boldsymbol{\theta}}^{(K)}$ on the original dataset. We use the following attack settings.

(1) *Cifar10 Blurred*: The attacker has a version of Cifar10 [52] corrupted with a Gaussian blur ($r = 0.5$). This version has different training and validation splits than the original. The attacker generalizes to the standard Cifar10 task.
(2) *Pen Digits*: The adversary attacks an MNIST [57] convolutional MeNN with a dense substitute model trained to classify digits from sequences of (x, y) pen coordinates [3].
(3) *Spoken Digits*: The attacker targets an MNIST [57] MeNN with a substitute trained on spoken digit audio [41].
(4) *Speech Noisy*: The adversary uses the Speech [95] dataset perturbed with white noise ($SNR = 50$). The training and validation splits differ from those of the original. The attacker targets an MeNN on the standard Speech task.
(5) *WISDM Sim*: The adversary uses the WISDM task's simulated version to target an MeNN trained on real-world data [54], emulating an attacker collecting its own dataset.

Compared to the target MeNN, we use substitutes with different architectures and hyperparameters (e.g., batch sizes). For example, the attacker's substitute MeNN for *Pen Digits* uses five fully connected layers with sizes $(8, 12, 48, 48, 48)$, early exiting after the second, and Leaky ReLU activations. The target MeNN processes the *MNIST* dataset using four convolutional layers with $(16, 32, 64, 32)$ filters, early exiting after the first, and ReLU activations. On *Cifar-10*, the attacker uses ResNet-18 [31] with early exiting after the second block. The target system uses a VGG architecture. These settings thus consider different neural networks which both achieve good accuracy on their given tasks. We fit each substitute three times.

Table 7 shows the maximum attack accuracy. The weakened adversary still achieves the best results against existing data-dependent policies; MeNNs using Max Prob show a mean worst-case attack accuracy of 1.56× Random. Although this efficacy is lower than white

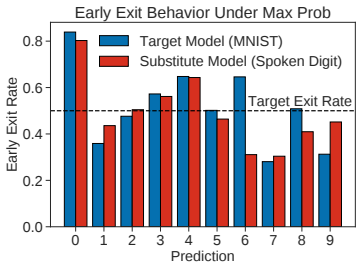

Figure 8: Early exit rates per prediction under the Max Prob policy for the target MeNN trained on MNIST [57] and the substitute trained on spoken digits [41].

Table 7: Worst-case attack accuracy (%) averaged (std dev) across trained MeNNs. In each group, the top row is the white box setting. The remaining rows use the black box method.

Train Task	Rand	MaxProb	
		Stnd	*CGR*
Cifar10	13.20	23.30	12.80
Cifar10 Blurred	11.93 (0.52)	19.17 (1.51)	12.30 (0.37)
MNIST	12.30	27.40	12.50
Pen Digits	12.83 (0.45)	17.80 (0.72)	12.77 (0.12)
Spoken Digit	12.57 (0.21)	20.17 (1.41)	12.80 (0.16)
Speech Cmds	12.05	28.79	12.28
Speech Cmds Noisy	12.87 (0.42)	24.78 (4.90)	12.80 (0.28)
WISDM	45.96	74.08	46.14
WISDM Sim	45.82 (0.00)	59.30 (2.19)	45.88 (0.09)

box settings (§6.2), the black box attacker still learns valuable information despite having no offline access to the target MeNN. CGR continues to show a worst-case attack accuracy close to Random.

This attack works because the substitute $h_{\psi}^{(K)}$ and target $f_{\theta}^{(K)}$ MeNNs often contain similar exit behavior, even though the target MeNN is unknown to the adversary. Figure 8 shows this phenomenon. Predictions for zero and seven have similar exit rates across the two MeNNs despite training the target on images and the substitute on audio. However, these rates are not always consistent. When predicting six, the target exits early more frequently, showing why the black box accuracy does not reach that of white box settings.

6.8 White Box Attack on Low-Power MCUs

We launch an end-to-end side-channel attack against distributed MeNNs [87] executing on a low-power MCU (§6.1.5). We execute each policy for 500 inputs on the Activity [54] task with the Same-Label order ($B = 10$), creating 50 temporal windows for the attacker.

Figure 9 shows inference and attack accuracy. In all cases, the attacker discovers the correct exit decisions from the packet trace. Using the white box attack model g_{ϕ}, the Max Prob policy exhibits the highest attack accuracy, while CGR reduces this leakage to Random. Further, CGR outperforms Random in inference accuracy on the MCU. These results match those from simulation (§6.2), showing how the discovered privacy issue and proposed defenses apply to real hardware.

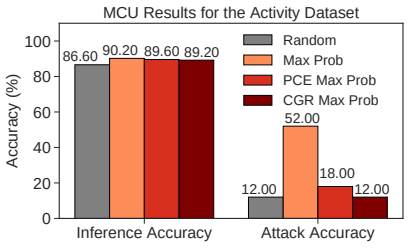

Figure 9: Inference accuracy and attack accuracy against distributed MeNNs executing on a low-power MCU.

Table 8: Average (std dev) energy consumption (mJ) on a TI MSP430 MCU [40].

Policy	Exit Early	Continue
Fixed	0.047 (0.017)	30.813 (5.234)
Random	0.049 (0.018)	31.144 (6.262)
Max Prob	0.059 (0.020)	32.799 (6.114)
PCE Max Prob	0.057 (0.019)	31.249 (6.243)
CGR Max Prob	0.061 (0.020)	32.320 (6.646)

6.9 Energy Consumption

MeNNs reduce the average cost of inference [86]. We show how CGR and PCE preserve this benefit by measuring their energy on a TI MSP430 [39, 40]. We run the distributed MeNN from §6.8 over 40 trials, recording the average energy to wake the CPU, execute the first exit point, evaluate the policy, and encrypt the result. When continuing inference, we include the energy to transmit the 128-byte state over BLE. The Fixed policy always uses the same exit.

Table 8 shows the average energy for each configuration. We highlight two aspects of these results. First, using the full model incurs over two orders of magnitude of overhead. This phenomenon comes from the high energy cost of communication, as early exiting allows the system to keep the BLE module off. This discrepancy shows the prohibitive cost of a Fixed policy that always uses the entire MeNN. Second, PCE and CGR incur some computation overhead compared to Random when exiting early. However, this cost is negligible compared to BLE when using the full MeNN, and for exit rates $\rho_0 < 1$, this BLE cost dominates the energy consumption. Further, under Welch's t-test, we observe an insignificant energy difference between Random and either PCE or CGR when continuing inference, with p-values of 0.94 (PCE) and 0.39 (CGR). Note that Max Prob has the highest average energy for the full MeNN. This result occurs due to the variance in communication energy; Max Prob also shows an insignificant difference compared to Random when using the full model. Thus, both PCE and CGR incur minimal overhead, allowing them to preserve the efficiency of MeNNs.

6.10 Beyond Two Exits

Prior sections display the leakage present in two-exit MeNNs. However, MeNNs can have more than two exits [86, 92]. We thus measure how the number of MeNN exit points impacts its privacy in simulation under Same-Label orders ($B = 10$). We emphasize that this analysis does not yield a practical attack under our threat model;

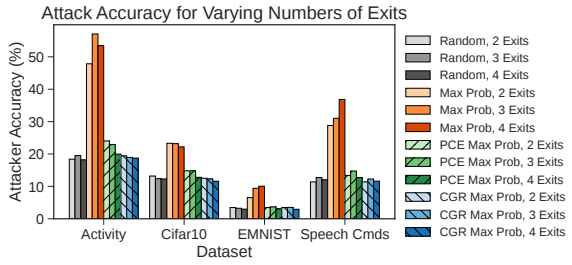

Figure 10: Maximum attack accuracy for MeNNs with two, three, and four exits (lower is better).

the adversary can only observe a binary decision of whether the system exits on the sensor or server (§3.2). Instead, we include this analysis to demonstrate (1) the potential for information leakage and (2) the performance of our methods on MeNNs with $K > 2$.

Figure 10 displays the maximum white box attack accuracy for MeNNs with $K = 2, 3$, and 4. Max Prob has higher leakage on MeNNs with more exits, showing an average worst-case attack accuracy of 2.20× (two exits), 2.53× (three exits), and 2.79× (four exits) Random. Both PCE and CGR have lower prediction leakage compared to this single-threshold policy. In particular, CGR displays near-random privacy with an average worst-case attack accuracy of 1.01× (two), 1.01× (three), and 0.98× (four) Random. These values have no trend with the number of exits. Furthermore, CGR still shows higher inference accuracy than Random. On the Activity task, CGR has an average accuracy of 88.72 (three exits) and 88.16 (four exits). These values eclipse Random: 88.15 (three) and 87.56 (four). Overall, previous data-dependent methods exhibit greater leakage on MeNNs with more exits, and both PCE and CGR provide protection in all contexts.

7 RELATED WORK

Multi-Exit Neural Networks (MeNNs). Prior work introduces early exits into neural networks [9, 37, 48, 59, 86, 91, 92, 94]. To select an exit point, existing systems use data-dependent exit policies with a single threshold on prediction confidence [9, 86, 94]. Other methods use bandit algorithms [42], runtime feedback [93], or decision agreement [104]. We focus on policies using maximum probability and entropy confidence, as they are cheap and well-suited for low-power sensors. We show how these policies leak information and propose new methods to address this problem.

Distributed Neural Network Inference. Existing frameworks partition DNNs across multiple systems to reduce resource costs on edge devices [7, 46, 60, 103]. Both DDNNs [87] and SPINN [55] introduce early exit behavior to improve distributed inference, creating distributed MeNNs. We develop a side-channel attack against the communication patterns of these distributed MeNNs. We defend against this attack through new early exit policies.

Attacks on Neural Networks. Common attacks against DNNs force misbehavior through adversarial noise [11, 15, 26, 28, 38, 58, 64, 74, 80, 85]. Other work induces adversarial behavior using training set poisoning [27], attacker-specified triggers [72], or batch orderings [79]. Popular countermeasures against these attacks include defensive distillation [75] and adversarial training [26, 89]. Further, existing proposals observe that MeNNs reduce the impact of adversarial examples [35, 48]. Previous attacks target MeNNs by crafting adversarial examples to maximize the execution cost [29, 34] and using exit decisions to improve membership inference queries [61]. Similar to our work, these attacks exploit early-exit behavior in neural networks. However, we evaluate how distributed MeNNs leak predictions through communication patterns.

Neural Networks and Privacy. Previous systems address the privacy of DNNs through homomorphic encryption [24, 63] and secure two-party computation [69]. Other methods protect DNNs using trusted execution environments [30, 68] and differential privacy [2, 84, 88, 96]. Our work also examines DNN privacy, but we create a new attack that uses exit patterns to infer MeNN predictions. Prior work leverages power [97], electromagnetic [8, 102], and timing/memory [36] side-channels to find DNN architectures and parameters. We instead use the communication patterns of distributed MeNNs as a side-channel to uncover model predictions. We further create efficient solutions for this new privacy concern.

Side-Channel Attacks. Many side-channel attacks exploit variable behavior under different inputs or operating conditions [16, 19, 66]. Previous work uses timing [12] and power discrepancies [50, 67] to extract encryption keys. We also study side-channels against varying behavior, but our work focuses on MeNNs, which is new. Previous work closes side-channels through fixed resource usage or randomized behavior. BuFLO [22] and its extensions [13, 44, 70] standardize traffic patterns to prevent website fingerprinting. Other systems obfuscate compromising communication patterns in sensor networks [5, 45, 47]. Our work uses a new randomization technique to retain the accuracy and resource benefits of MeNNs.

8 CONCLUSION

This work creates a side-channel attack that exploits the communication patterns of distributed Multi-exit Neural Networks (MeNNs) with data-dependent early exiting. This side-channel allows an adversary to discover the MeNN's predictions with over 1.85× the accuracy of random guessing. We address this attack through two new exit policies: Per-Class Exiting (PCE) and Confidence-Guided Randomness (CGR). PCE uses multiple confidence thresholds to reduce information leakage with inference accuracy close to prior methods. CGR augments PCE with randomization to achieve theoretical privacy guarantees and deliver consistently better inference accuracy than exiting early uniformly at random. This attack highlights how modern inference systems must consider the privacy implications of data-dependent behavior.

ACKNOWLEDGMENTS

We thank the reviewers for their constructive feedback. This work was supported by the National Science Foundation (NSF) grants CCF-1822949, CCF-2119184, CNS-1764039, and CISE-ANR-2124393.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al.

2016. TensorFlow: A system for Large-Scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation*. 265–283.
[2] Martín Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *23rd ACM Conf. on Computer and Communications Security*. 308–318.
[3] Fevzi Alimoglu and Ethem Alpaydin. 1996. Methods of combining multiple classifiers based on different representations for pen-based handwritten digit recognition. In *5th Turkish Artificial Intelligence and Artificial Neural Networks Symposium*. Citeseer.
[4] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra Perez, and Jorge Luis Reyes Ortiz. 2013. A public domain dataset for human activity recognition using smartphones. In *21st International European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*. 437–442.
[5] Noah Apthorpe, Danny Yuxing Huang, Dillon Reisman, Arvind Narayanan, and Nick Feamster. 2019. Keeping the smart home private with smart(er) IoT traffic shaping. *Proceedings on Privacy Enhancing Technologies* 2019, 3 (2019).
[6] Dmitri Asonov and Rakesh Agrawal. 2004. Keyboard acoustic emanations. In *IEEE Symposium on Security and Privacy*. 3–11.
[7] Amin Banitalebi-Dehkordi, Naveen Vedula, Jian Pei, Fei Xia, Lanjun Wang, and Yong Zhang. 2021. Auto-split: A general framework of collaborative edge-cloud AI. In *27th ACM Conf. on Knowledge Discovery & Data Mining*. 2543–2553.
[8] Lejla Batina, Shivam Bhasin, Dirmanto Jap, and Stjepan Picek. 2019. CSI NN: Reverse engineering of neural network architectures through electromagnetic side channel. In *28th USENIX Security Symposium*. 515–532.
[9] Konstantin Berestizshevsky and Guy Even. 2019. Dynamically sacrificing accuracy for reduced computation: Cascaded inference based on softmax confidence. In *International Conf. on Artificial Neural Networks*. Springer, 306–320.
[10] Erik Bernhardsson. 2023. Annoy. https://github.com/spotify/annoy.
[11] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. 2013. Evasion attacks against machine learning at test time. In *Joint European Conf. on Machine Learning and Knowledge Discovery in Databases*. Springer, 387–402.
[12] David Brumley and Dan Boneh. 2005. Remote timing attacks are practical. *Computer Networks* 48, 5 (2005), 701–716.
[13] Xiang Cai, Rishab Nithyanand, and Rob Johnson. 2014. Cs-BuFLO: A congestion sensitive website fingerprinting defense. In *13th Workshop on Privacy in the Electronic Society*. 121–130.
[14] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. 2022. Membership inference attacks from first principles. In *43rd IEEE Symposium on Security and Privacy*. 1897–1914.
[15] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *38th IEEE Symposium on Security and Privacy*. 39–57.
[16] Shuo Chen, Rui Wang, XiaoFeng Wang, and Kehuan Zhang. 2010. Side-channel leaks in web applications: A reality today, a challenge tomorrow. In *31st IEEE Symposium on Security and Privacy*. 191–206.
[17] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. 2017. EMNIST: an extension of MNIST to handwritten letters. *arXiv:1702.05373* (2017).
[18] Joan Daemen and Vincent Rijmen. 1999. AES proposal: Rijndael. (1999).
[19] Aveek K Das, Parth H Pathak, Chen-Nee Chuah, and Prasant Mohapatra. 2016. Uncovering privacy leakage in BLE network traffic of wearable fitness trackers. In *17th Workshop on Mobile Computing Systems and Applications*. 99–104.
[20] Bradley Denby and Brandon Lucia. 2020. Orbital edge computing: Nanosatellite constellations as a new class of computer system. In *25th Conf. on Architectural Support for Programming Languages and Operating Systems*. 939–954.
[21] Amol Deshpande, Carlos Guestrin, Samuel R Madden, Joseph M Hellerstein, and Wei Hong. 2004. Model-driven data acquisition in sensor networks. In *13th Conf. on Very Large Databases*. 588–599.
[22] Kevin P Dyer, Scott E Coull, Thomas Ristenpart, and Thomas Shrimpton. 2012. Peek-a-boo, I still see you: Why efficient traffic analysis countermeasures fail. In *33rd IEEE symposium on Security and Privacy*. 332–346.
[23] Bugra Gedik, Ling Liu, and S Yu Philip. 2007. ASAP: An adaptive sampling approach to data collection in sensor networks. *IEEE Transactions on Parallel and Distributed Systems* 18, 12 (2007), 1766–1783.
[24] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. 2016. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International Conf. on Machine Learning*. PMLR, 201–210.
[25] Graham Gobieski, Brandon Lucia, and Nathan Beckmann. 2019. Intelligence beyond the edge: Inference on intermittent embedded systems. In *24th Conf. on Architectural Support for Programming Languages and Operating Systems*. 199–213.
[26] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *International Conf. on Learning Representations*.
[27] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. 2017. BadNets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv:1708.06733* (2017).
[28] Amira Guesmi, Ihsen Alouani, Khaled N Khasawneh, Mouna Baklouti, Tarek Frikha, Mohamed Abid, and Nael Abu-Ghazaleh. 2021. Defensive approximation: Securing CNNs using approximate computing. In *26th Conf. on Architectural Support for Programming Languages and Operating Systems*. 990–1003.
[29] Mirazul Haque, Anki Chauhan, Cong Liu, and Wei Yang. 2020. ILFO: Adversarial attack on adaptive neural networks. In *IEEE Conf. on Computer Vision and Pattern Recognition*. 14264–14273.
[30] Hanieh Hashemi, Yongqin Wang, and Murali Annavaram. 2021. DarKnight: An accelerated framework for privacy and integrity preserving deep learning using trusted hardware. In *54th IEEE/ACM International Symposium on Microarchitecture*. 212–224.
[31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *IEEE Conf. on Computer Vision and Pattern Recognition*. 770–778.
[32] Shivayogi Hiremath, Geng Yang, and Kunal Mankodiya. 2014. Wearable Internet of Things: Concept, architectural components and promises for person-centered healthcare. In *4th IEEE Conf. on Wireless Mobile Communication and Healthcare*. 304–307.
[33] JK Holland, EK Kemsley, and RH Wilson. 1998. Use of Fourier transform infrared spectroscopy and partial least squares regression for the detection of adulteration of strawberry purees. *Journal of the Science of Food and Agriculture* 76, 2 (1998), 263–269.
[34] Sanghyun Hong, Yiğitcan Kaya, Ionuţ-Vlad Modoranu, and Tudor Dumitraş. 2020. A panda? No, it's a sloth: Slowdown attacks on adaptive multi-exit neural network inference. *arXiv:2010.02432* (2020).
[35] Ting-Kuei Hu, Tianlong Chen, Haotao Wang, and Zhangyang Wang. 2020. Triple wins: Boosting accuracy, robustness and efficiency together by enabling input-adaptive inference. *arXiv:2002.10025* (2020).
[36] Weizhe Hua, Zhiru Zhang, and G Edward Suh. 2018. Reverse engineering convolutional neural networks through side-channel information leaks. In *55th ACM/ESDA/IEEE Design Automation Conf.* 1–6.
[37] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Multi-scale dense networks for resource efficient image classification. *arXiv:1703.09844* (2017).
[38] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. 2019. Adversarial examples are not bugs, they are features. *Advances in Neural Information Processing Systems* 32 (2019).
[39] Texas Instruments. 2020. TI MSP430 EnergyTrace Technology. https://www.ti.com/lit/ug/slau157as/slau157as.pdf. Accessed: 04-2023.
[40] Texas Instruments. 2021. TI MSP430 FR5994 Datasheet. https://www.ti.com/lit/ds/symlink/msp430fr5994.pdf. Accessed: 04-2023.
[41] Zohar Jackson. 2022. Free spoken digit dataset. https://github.com/Jakobovski/free-spoken-digit-dataset. Accessed: 04-2023.
[42] Weiyu Ju, Wei Bao, Liming Ge, and Dong Yuan. 2021. Dynamic early exit scheduling for deep neural network inference through contextual bandits. In *30th ACM International Conf. on Information & Knowledge Management*. 823–832.
[43] Philo Juang, Hidekazu Oki, Yong Wang, Margaret Martonosi, Li Shiuan Peh, and Daniel Rubenstein. 2002. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet. In *10th Conf. on Architectural Support for Programming Languages and Operating Systems*. 96–107.
[44] Marc Juarez, Mohsen Imani, Mike Perry, Claudia Diaz, and Matthew Wright. 2016. Toward an efficient website fingerprinting defense. In *21st European Symposium on Research in Computer Security*. Springer, 27–46.
[45] Pandurang Kamat, Wenyuan Xu, Wade Trappe, and Yanyong Zhang. 2007. Temporal privacy in wireless sensor networks. In *27th IEEE International Conf. on Distributed Computing Systems*. 23–23.
[46] Yiping Kang, Johann Hauswald, Cao Gao, Austin Rovinski, Trevor Mudge, Jason Mars, and Lingjia Tang. 2017. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. *ACM SIGARCH Computer Architecture News* 45, 1 (2017), 615–629.
[47] Tejas Kannan and Henry Hoffmann. 2022. Protecting adaptive sampling from information leakage on low-power sensors. In *27th ACM Conf. on Architectural Support for Programming Languages and Operating Systems*. 240–254.
[48] Yigitcan Kaya, Sanghyun Hong, and Tudor Dumitras. 2019. Shallow-deep networks: Understanding and mitigating network overthinking. In *International Conf. on Machine Learning*. PMLR, 3301–3310.
[49] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv:1412.6980* (2014).
[50] Paul Kocher, Joshua Jaffe, and Benjamin Jun. 1999. Differential power analysis. In *Annual International Cryptology Conf.* Springer, 388–397.
[51] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. 2021. Wilds: A benchmark of in-the-wild distribution shifts. In *International Conf. on Machine Learning*. PMLR, 5637–5664.
[52] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images.
[53] Tarald O Kvålseth. 2017. On normalized mutual information: Measure derivations and properties. *Entropy* 19, 11 (2017), 631.

[54] Jennifer R Kwapisz, Gary M Weiss, and Samuel A Moore. 2011. Activity recognition using cell phone accelerometers. *ACM SigKDD Explorations Newsletter* 12, 2 (2011), 74–82.

[55] Stefanos Laskaridis, Stylianos I Venieris, Mario Almeida, Ilias Leontiadis, and Nicholas D Lane. 2020. SPINN: Synergistic progressive inference of neural networks over device and cloud. In *26th International Conf. on Mobile Computing and Networking*. 1–15.

[56] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444.

[57] Yann LeCun, Corinna Cortes, and Chris Burges. 1998. The MNIST database of handwritten digits. *http://yann. lecun. com/exdb/mnist/* (1998).

[58] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. 2019. Certified robustness to adversarial examples with differential privacy. In *40th IEEE Symposium on Security and Privacy*. 656–672.

[59] Hankook Lee and Jinwoo Shin. 2018. Anytime neural prediction via slicing networks vertically. *arXiv:1807.02609* (2018).

[60] En Li, Liekang Zeng, Zhi Zhou, and Xu Chen. 2019. Edge AI: On-demand accelerating deep neural network inference via edge computing. *IEEE Transactions on Wireless Communications* 19, 1 (2019), 447–457.

[61] Zheng Li, Yiyong Liu, Xinlei He, Ning Yu, Michael Backes, and Yang Zhang. 2022. Auditing membership leakages of multi-exit networks. In *ACM Conf. on Computer and Communications Security*. 1917–1931.

[62] Ji Lin, Wei-Ming Chen, Yujun Lin, Chuang Gan, Song Han, et al. 2020. MCUnet: Tiny deep learning on IoT devices. *Advances in Neural Information Processing Systems* 33 (2020), 11711–11722.

[63] Jian Liu, Mika Juuti, Yao Lu, and Nadarajah Asokan. 2017. Oblivious neural network predictions via MiniONN transformations. In *ACM Conf. on Computer and Communications Security*. 619–631.

[64] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv:1706.06083* (2017).

[65] Alan Mainwaring, David Culler, Joseph Polastre, Robert Szewczyk, and John Anderson. 2002. Wireless sensor networks for habitat monitoring. In *1st ACM Workshop on Wireless Sensor Networks and Applications*. 88–97.

[66] Aastha Mehta, Mohamed Alzayat, Roberta De Viti, Björn B. Brandenburg, Peter Druschel, and Deepak Garg. 2022. Pacer: Comprehensive Network Side-Channel Mitigation in the Cloud. In *31st USENIX Security Symposium*. USENIX Association, Boston, MA, 2819–2838.

[67] Thomas S. Messerges and Ezzy A. Dabbish. 1999. Investigations of Power Analysis Attacks on Smartcards. In *USENIX Workshop on Smartcard Technology*. USENIX Association.

[68] Fan Mo, Ali Shahin Shamsabadi, Kleomenis Katevas, Soteris Demetriou, Ilias Leontiadis, Andrea Cavallaro, and Hamed Haddadi. 2020. DarkneTZ: Towards model privacy at the edge using trusted execution environments. In *18th Conf. on Mobile Systems, Applications, and Services*. 161–174.

[69] Payman Mohassel and Yupeng Zhang. 2017. SecureML: A system for scalable privacy-preserving machine learning. In *38th IEEE symposium on Security and Privacy*. 19–38.

[70] Rishab Nithyanand, Xiang Cai, and Rob Johnson. 2014. Glove: A bespoke website fingerprinting defense. In *13th Workshop on Privacy in the Electronic Society*. 131–134.

[71] Angela Orebaugh, Gilbert Ramirez, and Jay Beale. 2006. *Wireshark & Ethereal network protocol analyzer toolkit*. Elsevier.

[72] Xudong Pan, Mi Zhang, Beina Sheng, Jiaming Zhu, and Min Yang. 2022. Hidden Trigger Backdoor Attack on NLP Models via Linguistic Style Manipulation. In *31st USENIX Security Symposium*. 3611–3628.

[73] Liam Paninski. 2003. Estimation of entropy and mutual information. *Neural computation* 15, 6 (2003), 1191–1253.

[74] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In *12th ACM Asia Conf. on Computer and Communications Security*. 506–519.

[75] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. 2016. Distillation as a defense to adversarial perturbations against deep neural networks. In *37th IEEE symposium on Security and Privacy*. 582–597.

[76] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *Nature* 323, 6088 (1986), 533–536.

[77] Simone Scardapane, Michele Scarpiniti, Enzo Baccarelli, and Aurelio Uncini. 2020. Why should we add early exits to neural networks? *Cognitive Computation* 12, 5 (2020), 954–966.

[78] Claude E Shannon. 1949. Communication theory of secrecy systems. *The Bell system technical journal* 28, 4 (1949), 656–715.

[79] Ilia Shumailov, Zakhar Shumaylov, Dmitry Kazhdan, Yiren Zhao, Nicolas Papernot, Murat A Erdogdu, and Ross J Anderson. 2021. Manipulating SGD with data ordering attacks. *Advances in Neural Information Processing Systems* 34 (2021), 18021–18032.

[80] Ilia Shumailov, Yiren Zhao, Daniel Bates, Nicolas Papernot, Robert Mullins, and Ross Anderson. 2021. Sponge examples: Energy-latency attacks on neural networks. In *6th IEEE European Symposium on Security and Privacy*. 212–231.

[81] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556* (2014).

[82] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[83] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. 2011. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In *IEEE International Joint Conf. on Neural Networks*. 1453–1460.

[84] Timothy Stevens, Christian Skalka, Christelle Vincent, John Ring, Samuel Clark, and Joseph Near. 2022. Efficient Differentially Private Secure Aggregation for Federated Learning via Hardness of Learning with Errors. In *31st USENIX Security Symposium*. 1379–1395.

[85] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv:1312.6199* (2013).

[86] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. 2016. Branchynet: Fast inference via early exiting from deep neural networks. In *23rd IEEE International Conf. on Pattern Recognition*. 2464–2469.

[87] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. 2017. Distributed deep neural networks over the cloud, the edge and end devices. In *37th IEEE International Conf. on Distributed Computing Systems*. 328–339.

[88] Stacey Truex, Nathalie Baracaldo, Ali Anwar, Thomas Steinke, Heiko Ludwig, Rui Zhang, and Yi Zhou. 2019. A hybrid approach to privacy-preserving federated learning. In *12th ACM Workshop on Artificial Intelligence and Security*. 1–11.

[89] Pratik Vaishnavi, Kevin Eykholt, and Amir Rahmati. 2022. Transferring Adversarial Robustness Through Robust Representation Matching. In *31st USENIX Security Symposium*. 2083–2098.

[90] Deepak Vasisht, Zerina Kapetanovic, Jongho Won, Xinxin Jin, Ranveer Chandra, Sudipta Sinha, Ashish Kapoor, Madhusudhan Sudarshan, and Sean Stratman. 2017. FarmBeats: An IoT platform for data-driven agriculture. In *14th USENIX Symposium on Networked Systems Design and Implementation*. 515–529.

[91] Andreas Veit and Serge Belongie. 2018. Convolutional networks with adaptive inference graphs. In *European Conf. on Computer Vision (ECCV)*. 3–18.

[92] Chengcheng Wan, Henry Hoffmann, Shan Lu, and Michael Maire. 2020. Orthogonalized SGD and nested architectures for anytime neural networks. In *International Conf. on Machine Learning*. PMLR, 9807–9817.

[93] Chengcheng Wan, Muhammad Santriaji, Eri Rogers, Henry Hoffmann, Michael Maire, and Shan Lu. 2020. ALERT: Accurate learning for energy and timeliness. In *USENIX Annual Technical Conf.* 353–369.

[94] Xin Wang, Yujia Luo, Daniel Crankshaw, Alexey Tumanov, Fisher Yu, and Joseph E Gonzalez. 2017. Idk cascades: Fast deep learning by learning not to overthink. *arXiv:1706.00885* (2017).

[95] Pete Warden. 2018. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv:1804.03209* (2018).

[96] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H Vincent Poor. 2020. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security* 15 (2020), 3454–3469.

[97] Lingxiao Wei, Bo Luo, Yu Li, Yannan Liu, and Qiang Xu. 2018. I know what you see: Power side-channel attack on convolutional neural network accelerators. In *34th Annual Computer Security Applications Conf.* 393–406.

[98] Michael Winkler, Klaus-Dieter Tuchs, Kester Hughes, and Graeme Barclay. 2008. Theoretical and practical aspects of military wireless sensor networks. *Journal of Telecommunications and Information Technology* (2008), 37–45.

[99] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. *arXiv:1708.07747* (2017).

[100] Shuochao Yao, Yiran Zhao, Aston Zhang, Lu Su, and Tarek Abdelzaher. 2017. DeepIoT: Compressing deep neural network structures for sensing systems with a compressor-critic framework. In *15th ACM Conf. on Embedded Network Sensor Systems*. 1–14.

[101] Kasım Sinan Yıldırım, Amjad Yousef Majid, Dimitris Patoukas, Koen Schaper, Przemyslaw Pawelczak, and Josiah Hester. 2018. Ink: Reactive kernel for tiny batteryless sensors. In *16th ACM Conf. on Embedded Networked Sensor Systems*. 41–53.

[102] Honggang Yu, Haocheng Ma, Kaichen Yang, Yiqiang Zhao, and Yier Jin. 2020. DeepEM: Deep neural networks model recovery through EM side-channel information leakage. In *IEEE Symposium on Hardware Oriented Security and Trust*. 209–218.

[103] Liekang Zeng, En Li, Zhi Zhou, and Xu Chen. 2019. Boomerang: On-demand cooperative deep neural network inference for edge intelligence on the industrial Internet of Things. *IEEE Network* 33, 5 (2019), 96–103.

[104] Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. 2020. Bert loses patience: Fast and robust inference with early exit. *Advances in Neural Information Processing Systems* 33 (2020), 18330–18341.

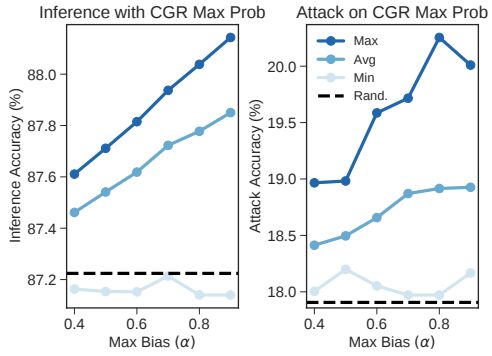

Figure 12: Mean inference accuracy (left) and worst-case attack accuracy (right) for different maximum bias (α) values aggregated across $\beta \in \{1.5, 2.0, 2.5\}$ and $\gamma = \{0.5, 0.7, 0.9, 0.99\}$.

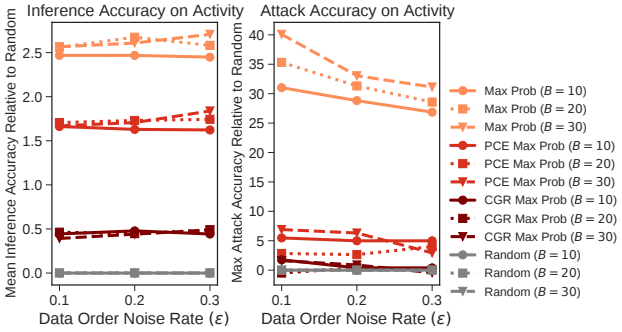

Figure 11: Average inference accuracy (left) and maximum attack accuracy (right) on the Activity dataset for window sizes (B) and noise rates (ε) under the Same-Label order.

A SENSITIVITY ANALYSIS

A.1 Data Order Parameters

The provided experiments under the Same-Label order use a single block size ($B = 10$) and noise rate ($\varepsilon = 0.2$). We expand this analysis by varying these parameters for $B \in \{10, 20, 30\}$ and $\varepsilon \in \{0.1, 0.2, 0.3\}$. Figure 11 shows the inference accuracy and *white box* attack accuracy on the Activity task. We display the difference between each policy's result and that of Random. CGR consistently has an inference accuracy of about 0.45 points above Random for all configurations. Further, CGR displays a maximum attack accuracy of only 0.51 points higher than Random on average across all settings. This figure is over 60× lower than that of Max Prob. In the worst case, CGR has an attack accuracy of 1.76 points above Random, occurring when $B = 10$ and $\varepsilon = 0.1$. However, CGR does *not* have consistently lower privacy at this noise rate; for $B = 20$ and $\varepsilon = 0.1$, CGR has a worst-case attack accuracy of 0.48 points *lower* than Random. In total, these results align with the previous experiments under a single Same-Label configuration (§6.2, 6.4). We note that the adversary achieves better results against Max Prob for larger block sizes and lower noise rates, as such characteristics yield stronger temporal correlations.

A.2 CGR Parameters

The experiments in §6 use the CGR policy with a maximum bias $\alpha = 0.5$, increase factor $\beta = 2.0$, and decrease factor $\gamma = 0.9$. In this section, we evaluate the impact of these parameters on the policy's inference and attack accuracy. We perform a grid search over $\alpha = \{0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$, $\beta \in \{1.5, 2.0, 2.5\}$, and $\gamma \in \{0.5, 0.7, 0.9, 0.99\}$. For each setting, we compute the mean inference accuracy and maximum white box attack accuracy across all exit rates ρ for the CGR policy with the Max Prob confidence metric. These experiments use the Activity dataset [54] under Same-Label data order with $B = 10$ and $\varepsilon = 0.2$. We execute each configuration over three trials and take the average result for each exit rate. We present the results for each parameter by considering the maximum, average, and minimum values aggregated across all settings for the other two parameters. Figures 12, 13, and 14 display the results from the perspectives of the maximum bias (α), increase factor (β), and decrease factor (γ), respectively.

On average, larger maximum bias α values lead to higher inference and attack accuracy. This trend aligns with CGR's design (§5); larger biases allow CGR to align more with PCE, delivering better inference performance with worse privacy. Note that even with a large bias (e.g., $\alpha = 0.9$), CGR can still display lower inference and attack accuracy when the decrease factor γ is small (e.g., $\gamma = 0.5$).

Both the increase (β) and decrease factors (γ) show positive correlations with the policy's inference and attack accuracy. When CGR has a small decrease factor γ, the policy is quick to decrease the bias. This phenomenon leads to smaller bias terms on average, causing more randomness, lower inference accuracy, and better privacy. The opposite occurs when γ nears one. In particular, when $\gamma \approx 1$ and $\beta >> 1$ (e.g., $\gamma = 0.99$, $\beta = 2.5$), CGR shows high inference accuracy (88.14) and attack accuracy (19.73) for a fixed $\alpha = 0.9$. This result matches the theory of CGR's adaptive bias algorithm (§5.2), as these settings result in quickly increasing and slowly decreasing the bias term.

In general, we find that $\alpha = 0.5$, $\beta = 2.0$ and $\gamma = 0.9$ deliver near-random attack privacy with favorable inference accuracy. As demonstrated, these settings are robust across multiple different tasks and confidence metrics (§6).

B PROOFS

B.1 Proposition 4.1

Proposition B.1. *Let $\pi : \mathbb{R}^L \to \{0, 1\}$ be a deterministic policy on a two-exit MeNN $f_{\boldsymbol{\theta}}^{(2)}$ for a rational exit rate $\rho_0 = \frac{m}{M} \in (0, 1)$. Let D be the set of possible samples $(\boldsymbol{x}, y)$, $D_x = \{\boldsymbol{x} : \exists \ell \in [L], (\boldsymbol{x}, \ell) \in D\}$ be the inputs, and $D_\ell = \{\boldsymbol{x} : (\boldsymbol{x}, \ell) \in D\}$ be the inputs with label ℓ. Assume $\forall k \in \{0, 1\}$, $\pi^{-1}(k) \cap D_\ell \neq \emptyset$ where $\pi^{-1}(k) = \{\boldsymbol{x} \in D_x : \pi(f_{\boldsymbol{\theta}_0}^{(0)}(\boldsymbol{x})) = k\}$. Then, there exists an ordered dataset $\tilde{D} = [(\boldsymbol{x}^{(t)}, y^{(t)}) \in D]_{t=0}^{T-1}$ for $T = n \cdot LM$, $n > 1$ with the following.*

(1) The policy π exhibits an early exit rate of ρ_0 on $\tilde{D}$.

(2) There exists a function $g_{\boldsymbol{\phi}} : \{0, 1\}^{LM} \to [L]$ such that

$$g_{\boldsymbol{\phi}}(\pi(\hat{\boldsymbol{y}}^{(t)}), \ldots, \pi(\hat{\boldsymbol{y}}^{(t+LM-1)})) = MostFreq(y^{(t)}, \ldots, y^{(t+LM-1)})$$

where $t = j \cdot LM$ for any $j \in [n]$.

(3) Not all non-overlapping blocks of LM exit decisions (in property 2) have the same most frequent label.

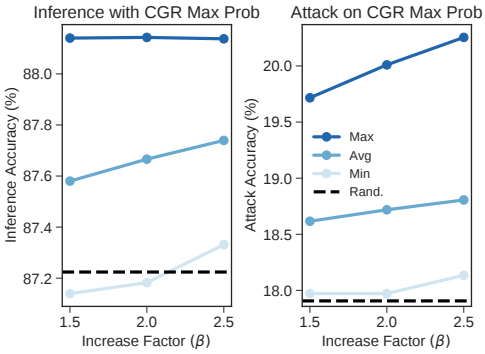

Figure 13: Mean inference accuracy (left) and worst-case attack accuracy (right) for different increase factors (β) aggregated across $\alpha \in \{0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$ and $\gamma = \{0.5, 0.7, 0.9, 0.99\}$.

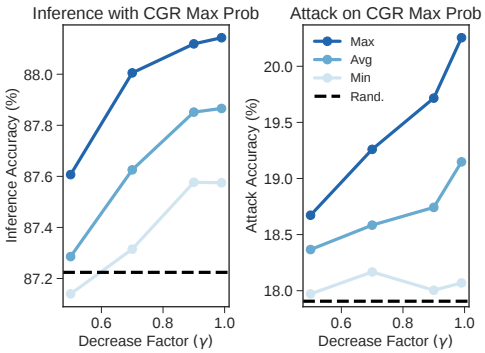

Figure 14: Mean inference accuracy (left) and worst-case attack accuracy (right) for different decrease factors (γ) aggregated across $\alpha \in \{0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$ and $\beta = \{1.5, 2.0, 2.5\}$.

Proof. We design an ordered dataset with $T = n \cdot LM$ elements in blocks of LM. Consider the j^{th} block for $j \in [n]$. We will fill this block with inputs of label $\ell \equiv j \pmod{L}$. Let $\boldsymbol{x}_\ell^{(0)} \in \pi^{-1}(0) \cap D_\ell$ and $\boldsymbol{x}_\ell^{(1)} \in \pi^{-1}(1) \cap D_\ell$ be arbitrary. In this block, for $0 \le t < L$, we set $\boldsymbol{x}_t$ to $\boldsymbol{x}_\ell^{(0)}$ when $t = \ell$ and $\boldsymbol{x}_\ell^{(1)}$ otherwise. On the remaining $LM - L$ inputs, we arbitrarily select $Lm - 1$ positions for early exiting, setting these elements to $\boldsymbol{x}_\ell^{(0)}$ and the remaining to $\boldsymbol{x}_\ell^{(1)}$. Note that there are $Lm - 1$ available slots because $M - 1 \ge m$ so $LM - L \ge Lm > Lm - 1$. This construction satisfies the desired properties. First, each block contains exactly Lm early exits, yielding an early exit rate of $(nLm)/(nLM) = m/M = \rho_0$. Second, let $g_{\boldsymbol{\phi}} : \{0,1\}^{LM} \to [K]$ output the position of the first early exit. By construction, the j^{th} block (with inputs of label ℓ) has only one early exit in the first L positions. This exit occurs in position ℓ, so the function $g_{\boldsymbol{\phi}}$ correctly maps to the most frequent label. Finally, there are $n > 1$ blocks. As the j^{th} block has inputs of label $\ell \equiv j \pmod{L}$, $\tilde{D}$ will have at least two blocks with different most frequent labels. □

B.2 Proposition 5.1

Let π_r be the CGR policy without exit quotas and $\delta, \epsilon : \mathbb{R} \to \mathbb{R}$ be functions such that $\delta(\alpha) = (1 - \alpha - \rho_0(1-\alpha))/\rho_0$ and $\epsilon(\alpha) = (1 - \rho_0(1-\alpha))/\rho_0$ where ρ_0 is the exit rate for $K = 2$.

Proposition B.2. *Consider a two-exit MeNN with a target early exit rate ρ_0. Suppose we observe a sequence of T exit decisions $\pi_r^{(t)} := \pi_r(\hat{\boldsymbol{y}}^{(0,t)}, 0) = v_t$ for $v_t \in \{0,1\}$ and $t \in [T]$. Let these T inputs belong to the same class and $n = \sum_{t=0}^{T-1} v_t$. Then, π_r displays the following bounds for any labels $\ell_0, \ell_1 \in [L]$.*

$$\left(\frac{\delta(\alpha)}{\epsilon(\alpha)}\right)^n \left(\frac{1-\rho_0\epsilon(\alpha)}{1-\rho_0\delta(\alpha)}\right)^{T-n} \le \frac{P(\pi_r^{(t)} = v_t \forall t | Y = \ell_0)}{P(\pi_r^{(t)} = v_t \forall t | Y = \ell_1)}$$
$$\left(\frac{\epsilon(\alpha)}{\delta(\alpha)}\right)^n \left(\frac{1-\rho_0\delta(\alpha)}{1-\rho_0\epsilon(\alpha)}\right)^{T-n} \ge \frac{P(\pi_r^{(t)} = v_t \forall t | Y = \ell_0)}{P(\pi_r^{(t)} = v_t \forall t | Y = \ell_1)}$$

Proof. The construction of the CGR policy yields the following bounds, resulting from the alignment (or lack thereof) of PCE with the decisions v_t under the largest possible bias α.

$$\rho_0\delta(\alpha) \le P(\pi_r^{(t)} = 1) \le \rho_0\epsilon(\alpha)$$
$$1 - \rho_0\epsilon(\alpha) \le P(\pi_r^{(t)} = 0) \le 1 - \rho_0\delta(\alpha)$$

The maximum probability of observing the sequence $\{v_t\}_{t=0}^{T-1}$ occurs when PCE aligns with each v_t. The minimal probability happens when PCE goes against all observed decisions. This logic holds any $\ell \in [L]$, creating the following.

$$(\rho_0\delta(\alpha))^n (1-\rho_0\epsilon(\alpha))^{T-n} \le P(\pi_r^{(t)} = v_t \forall t | Y = \ell)$$
$$(\rho_0\epsilon(\alpha))^n (1-\rho_0\delta(\alpha))^{T-n} \ge P(\pi_r^{(t)} = v_t \forall t | Y = \ell)$$

From these inequalities, we obtain the desired result. □