

Importance-Aware Encryption for Secure Distributed CNN Inference at the Edge

Fatemeh Mehrafrooz*, Roozbeh Siyadatzaheh*, Nele Mentens^{*,†}, Todor Stefanov*

*Leiden Institute of Advanced Computer Science (LIACS), Leiden University, the Netherlands

[†]Department of Electrical Engineering, KU Leuven, Belgium

Abstract—Distributed inference of Convolutional Neural Networks (CNNs) across multiple edge devices can overcome the resource limitations of a single edge device while avoiding the data transmission delays of cloud-based inference. However, the communicated intermediate data between edge devices may expose sensitive information to an attacker eavesdropping on the communication links. Authenticating and encrypting the entire communicated intermediate data can reduce this risk, but it introduces high encryption/decryption overhead for resource-constrained edge devices. This paper proposes a lightweight importance-aware authenticated encryption approach for secure distributed CNN inference at the edge. The proposed approach computes channel importance scores and uses them to guide an NSGA-II-based multi-objective optimization process that determines how much data should be encrypted and where the encrypted data should be located within the output feature maps. The optimization searches for Pareto-optimal data protection configurations that reduce CNN input reconstruction risk and information leakage while keeping the encryption overhead low. Experiments on ResNet50, VGG16, and MobileNet using CIFAR-10 and CIFAR-100 show that the proposed approach provides an effective trade-off between data privacy preservation and authenticated encryption overhead.

Index Terms—Distributed Edge Computing, Neural Networks, Security

I. INTRODUCTION

Today, deep neural networks (DNNs) play a crucial role in various aspects of our daily life, such as smart homes, healthcare, and agriculture. Among the different types of DNNs, convolutional neural networks (CNNs) are used for applications such as object detection and medical image classification due to their powerful image-processing capabilities. Modern large CNNs require significant resources for CNN inference and storage of the layers' weights. While cloud servers, due to their high computing and storage capabilities, are well-suited for inferring large CNN models, their distance from the source of data (sensors, edge devices, etc.) that needs to be processed by the CNNs often introduces delays in the data transmission and response time. Such delays may limit the suitability of a cloud server for CNN inference in real-time applications that require quick responses, such as face and voice detection.

To decrease the data transmission and response time delays, there is a trend to perform the CNN inference closer to the data source, e.g., by offloading it to edge devices. This eliminates the need to send raw data to a cloud server for processing. Although edge devices reduce the aforementioned delays, they

have limited resources in terms of compute power, memory, and energy budgets, and a single edge device may not be able to execute a large CNN model efficiently.

One solution approach to address the challenge of running a large CNN model on a resource-constrained edge device is to utilize a lightweight version of the CNN model [1]. Such lightweight version can be created by using CNN compression techniques such as pruning [2], quantization [3], and knowledge distillation [4]. Although a compressed CNN model requires fewer resources for deployment and inference, it may exhibit reduced accuracy. Another approach is to use cloud-edge hybrid CNN inference [5]. In this approach, part of the CNN model is executed on a cloud server, while the rest is executed on an edge device. Such cloud-edge inference approach maintains the CNN accuracy, but transmitting data to the cloud still may introduce unacceptable delays for some applications. Therefore, recently proposed approaches are to distribute the CNN model across multiple edge devices [6]–[9]. These approaches utilize the resources of several edge devices to deploy and infer the CNN model, thereby addressing the issue of the limited resources of a single edge device. Moreover, there is no need to send data to cloud servers, thus avoiding data transmission delays.

To distribute a CNN model across multiple edge devices, three common approaches are used: *vertical partitioning* [7], *horizontal partitioning* [8], and *data partitioning* [9]. In vertical partitioning, as illustrated in Fig. 1a, one or more layers of a CNN model are grouped and assigned to an edge device for execution. In horizontal partitioning (Fig. 1b), a part of a layer or multiple layers are divided into partitions and each partition is executed on an edge device. In data partitioning (Fig. 1c), the input data to a CNN model is split into smaller portions, and each portion is processed on an edge device by the CNN model.

While distributing a CNN model across multiple edge devices for inference solves the issues of the other solution approaches discussed above, it introduces security and privacy vulnerabilities. Since edge devices must exchange intermediate data coming from CNN layers to collaboratively execute the CNN model during inference, this communication can expose the intermediate data to potential attackers and cause information leakage. In particular, an attacker may eavesdrop on the communication and use the intermediate data to reconstruct the often private input data given to the CNN model or

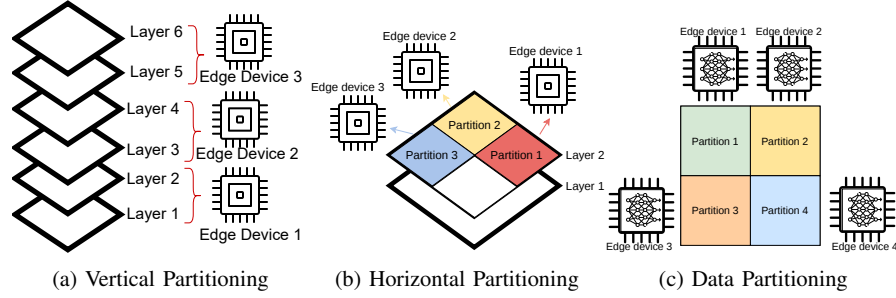


Fig. 1: Common CNN partitioning approaches for distributed inference across edge devices.

extract additional sensitive information. Therefore, protecting the communicated intermediate data is important for privacy-preserving distributed CNN inference on edge devices.

A common approach to reducing this privacy risk is to perform authenticated encryption of the communicated intermediate data. However, straightforward authenticated encryption and decryption introduce a significant overhead in terms of computation time and energy consumption. For resource-constrained edge devices, such overhead makes it impractical to encrypt large amounts of communicated data due to limited computation power and energy budgets of the edge devices. Therefore, an efficient strategy is needed to balance between the limited compute and energy resources of edge devices and the privacy protection strength of the communication among the devices during distributed CNN inference.

To address this challenge, we propose a lightweight importance-aware authenticated encryption approach that selectively protects the communicated intermediate data between edge devices. The key idea is that not all parts of the intermediate data have the same effect on the CNN inference process. Therefore, instead of protecting the whole communicated intermediate data, our approach identifies the more important parts of this data and prioritizes them for protection. In our approach, for each output layer whose data is communicated to another edge device, we first compute an importance score for each output channel. This score represents the contribution of each channel to the CNN inference process. Then, based on these importance scores, we formulate an optimization problem to determine how much data should be protected inside each selected channel and where the protected data should be located within the corresponding channel feature map. In this way, the optimization process searches for protection configurations that reduce information leakage and reconstruction risk while keeping the amount of protected data low. As a result, the proposed approach avoids the high overhead of protecting the entire communicated intermediate data, while still prioritizing the parts that are more relevant to the CNN inference process. The main novel contributions of this paper can be summarized as follows:

- We propose a lightweight importance-aware authenticated encryption approach for secure distributed CNN inference on resource-constrained edge devices. Our approach

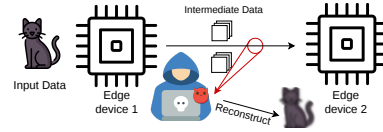


Fig. 2: An attack scenario in distributed CNN inference.

selectively encrypts the communicated data between devices in order to reduce information leakage and reconstruction risk while keeping the encryption overhead low.

- We formulate a multi-objective optimization problem and solve it using the well-known Non-dominated Sorting Genetic Algorithm II (NSGA-II) in order to minimize the information leakage, reconstruction risk and encryption overhead. The optimization process uses channel importance scores to guide the search toward effective configurations.
- We evaluate the proposed approach using different CNN models, datasets, and monitored layer positions. The results show that the approach reduces CNN input reconstruction risk and information leakage at low encryption/decryption overhead.

II. MOTIVATIONAL EXAMPLE

Consider the attack scenario illustrated in Fig. 2, where a distributed CNN model is deployed across multiple edge devices, and an attacker gains access to a communication link between two devices and to the intermediate data exchanged via the link during CNN inference. The attacker's objective is to use the exposed intermediate data to reconstruct the original CNN input data or extract sensitive information, thereby causing information leakage.

As discussed in Section I, a common approach to addressing this privacy risk is to apply authenticated encryption to the intermediate data before transmission. However, authenticating and encrypting the entire intermediate data that is communicated within a distributed CNN model introduces substantial time and energy costs, making it impractical for resource-constrained edge devices with strict real-time performance requirements. Even authenticating and encrypting the communicated data coming out of only a few selected output channels of a CNN layer still incurs considerable overhead.

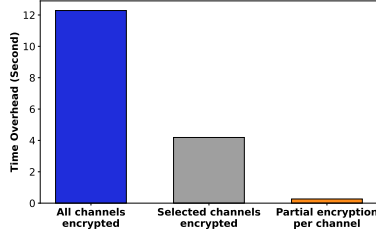


Fig. 3: Comparison of time overhead across different authenticated encryption cases.

For example, Fig. 3 shows the time overhead when authenticating and encrypting intermediate data coming from output channels of the second convolutional layer in the MobileNet CNN model in three cases: (1) authenticated encryption of the entire intermediate data, i.e., the data from all output channels (the blue bar), (2) authenticated encryption of the data from 25% selected output channels (the gray bar), and (3) authenticated encryption of only a portion of the data from each selected channel (the orange bar). All three cases were implemented in Python and measured on the same Linux machine, ensuring that the reported time overheads are compared under identical execution conditions. For all three cases, we use the *ASCON authenticated encryption* algorithm [10], which is selected as the new lightweight cryptography standard, offering state-of-the-art efficiency and security for resource-constrained platforms.

The results, shown in Fig. 3, indicate that authenticating and encrypting the entire intermediate data or the data from the selected channels introduces excessive overhead in the range of several seconds. In contrast, authenticating and encrypting only a portion of the data from each selected channel reduces the time overhead by up to $46.37\times$ on the selected platform compared to authenticated encryption of the entire intermediate data, making it a potentially practical privacy-preserving solution for resource-constrained edge devices.

The above observations motivate us to devise an approach that determines how much data in each output channel should be encrypted and where the encrypted data should be located in order to reduce information leakage and reconstruction risk, while keeping time and energy overheads low.

III. RELATED WORK

In this section, we discuss related studies that cover three relevant aspects, namely distributed CNN inference on multiple edge devices, selecting important channels in CNN models, and the security of intermediate data in distributed CNN inference.

A. Distributing CNN Inference Across Multiple Edge Devices

Numerous research studies have proposed approaches to distribute and optimize CNN model inference across multiple edge devices in order to improve performance [6]–[9], [11]–[14], reduce memory requirements [6], [7], [9], [11], [13], [15] and energy consumption [6], [11], [16], and

improve robustness [17], [18]. For example, MoDNN [14] and DeepThings [9] improve CNN inference on mobile and IoT devices using parallel execution and efficient task distribution. CoEdge [13] dynamically partitions workloads based on device capabilities and network conditions. AutoDiCE [6], [11] and RobustDiCE [17], [18] provide automated multi-objective exploration for distributed CNN inference at the edge.

The main focus of these studies is optimizing performance, memory, energy efficiency, and robustness. However, they do not address the security/privacy implications of distributed CNN inference. In contrast, our work introduces a lightweight importance-aware authenticated encryption approach that reduces information leakage and reconstruction risk from communicated intermediate data while keeping the encryption overhead low.

B. Identifying Important Channels in CNN Models

Identifying important channels in CNN models is commonly used in model compression to reduce computational complexity and memory usage. Various metrics have been proposed to measure channel importance, including filter weight norm, activation statistics, and gradient-based sensitivity.

The L1-norm of a CNN filter's weights is widely used to estimate the contribution of a channel to the CNN model accuracy [19]–[21]. Activation-based metrics, such as activation sum or standard deviation, measure the magnitude or variability of feature responses and are used to identify informative or redundant feature maps [22]–[24]. Gradient-based methods measure the sensitivity of the loss function with respect to filter outputs and capture the effect of each channel on the final prediction [25]–[27].

In summary, prior work demonstrates that the L1-norm, gradient-based methods, and activation standard deviation are effective for identifying important channels in CNNs. Building upon these findings, our approach combines these three metrics to compute an importance score for each channel and uses these scores to guide the NSGA-II optimization process towards finding effective data protection configurations.

C. Security of Data in Distributed CNN Inference

When distributing CNN inference across multiple edge devices, intermediate data must be transmitted between devices. This data may contain sensitive information and can be exploited by an attacker to reconstruct the original CNN input data or extract additional private information. Prior work has addressed this privacy risk using perturbation-based methods, privacy-preserving representation learning, and cryptographic protection.

Perturbation-based methods modify the communicated intermediate data to make reconstruction more difficult [28]. However, such methods intentionally alter the feature maps, thereby degrading the CNN inference accuracy. Representation-learning methods aim to learn privacy-preserving intermediate representations, for example through adversarial training [29]. However, they typically require retraining or modification of the CNN model. Cryptographic

methods, such as CryptoNets [30], Gazelle [31], and SecureML [32], provide strong privacy guarantees using homomorphic encryption, garbled circuits, or secure multi-party computation. However, their computational and memory overhead is typically very high for real-time CNN inference on resource-constrained edge devices.

In contrast, our work adopts a lightweight importance-aware authenticated encryption/decryption approach that selectively protects communicated intermediate data. The proposed approach does not modify the CNN architecture, does not require retraining of the CNN model, and does not change the CNN inference accuracy. It computes an importance score for each output channel and uses these scores to guide an NSGA-II-based optimization process that determines how much data in each channel should be encrypted and where the encrypted data should be located. In this way, our approach reduces the information leakage and reconstruction risk while keeping the encryption/decryption overhead low.

IV. PROPOSED APPROACH

In this section, we describe our proposed lightweight importance-aware authenticated encryption approach for secure distributed CNN inference on edge devices. The approach consists of two phases, one executed at design time before deployment of the CNN for inference, and another executed at runtime during the CNN inference.

In our approach, at runtime during the CNN inference, each edge device encrypts and authenticates a small portion of the data belonging to the output channels of a CNN layer which is connected to another layer executed on a different edge device referred to as destination device. Then, this protected small portion of data together with the unprotected data in all output channels of the CNN layer is sent to the destination device. Upon receiving the data, the destination device decrypts and verifies the protected portion and proceeds with the inference using both the protected and unprotected data. This approach reduces the exposure of sensitive information in the communicated intermediate data while keeping the time and energy overheads associated with the encryption/decryption low.

The aforementioned small portion of data belonging to the output channels of a CNN layer, that has to be encrypted and authenticated, is determined at design time before the deployment of the CNN for inference. To this end, we have devised an importance-aware data protection procedure which finds Pareto-optimal data portions to be protected such that information leakage is minimized while the encryption/decryption overhead during CNN inference remains low. The overall workflow of our importance-aware protection procedure is summarized in Algorithm 1. The algorithm takes as input a set of CNN layers $L = \{l_1, \dots, l_{|L|}\}$, where the output data of every layer $l_i \in L$ is transmitted between edge devices, and two sample datasets X_{imp} and X_{leak} . Dataset X_{imp} is used to compute importance scores of output channels belonging to every layer l_i , while dataset X_{leak} is used to evaluate candidate data protection configurations during the optimization process. The output of the algorithm is a set of Pareto-optimal data

Algorithm 1: Importance-aware Data Protection

Input: Set of layers L ; Sample datasets X_{imp}, X_{leak} ;

Output: Set of Pareto-optimal data protection configurations $D = \{\chi_1, \dots, \chi_{|L|}\}$

```

1  $D \leftarrow \emptyset$ ;
2 for  $l_i \in L$  do
3    $S_i \leftarrow \emptyset$ ;
4   // Step 1: channel importance analysis
5   for  $ch_j^{out} \in l_i$  do
6      $s_j^{L1} = \sum_{h=1}^{k_h} \sum_{w=1}^{k_w} \sum_{c=1}^{C_{in}} |W_j^{c,h,w}| + |b_j|$ ;
7      $s_j^{grad} = \frac{1}{|X_{imp}| \cdot H \cdot W} \sum_{n=1}^{|X_{imp}|} \sum_{h=1}^H \sum_{w=1}^W \left| \frac{\partial \mathcal{L}_n}{\partial a_{j,n}^{h,w}} \right|$ ;
8      $s_j^{std} = \frac{1}{\sqrt{\frac{1}{|X_{imp}| \cdot H \cdot W} \sum_{n=1}^{|X_{imp}|} \sum_{h=1}^H \sum_{w=1}^W (a_{j,n}^{h,w} - \mu_j)^2}}$ ;
9      $\hat{s}_j = \text{norm}(s_j^{L1}) + \text{norm}(s_j^{grad}) + \text{norm}(s_j^{std})$ ;
10     $S_i \leftarrow S_i \cup \{\hat{s}_j\}$ ;
11  // Step 2: multi-objective optimization
12   $\mathcal{P}_i \leftarrow \text{NSGA-II}(l_i, S_i, X_{leak})$ ;
13   $\chi_i \leftarrow \text{SELECTDATAProtectionConfiguration}(\mathcal{P}_i)$ ;
14   $D \leftarrow D \cup \{\chi_i\}$ ;
15 return  $D$ 

```

protection configurations $D = \{\chi_1, \dots, \chi_{|L|}\}$, where each configuration χ_i specifies the data portions of output channels belonging to layer l_i to which authenticated encryption should be applied.

Algorithm 1 starts with initializing an empty set D . Then, for each layer $l_i \in L$, the algorithm proceeds in two main steps (Lines 2-12). In Step 1, for each output channel ch_j^{out} belonging to layer l_i , a channel importance analysis is performed to estimate the contribution of the channel to the accuracy of the CNN inference process (Lines 5-9). Three complementary importance scores are computed, namely the L1-norm of the channel weights (Line 5), the gradient-based sensitivity of the loss function with respect to the channel activations computed over dataset X_{imp} (Line 6), and the activation standard deviation computed over the same dataset (Line 7). These three scores are normalized and aggregated to obtain the final importance score $\hat{s}_j$ of output channel ch_j^{out} (Line 8). The final importance scores of the channels belonging to layer l_i are collected in set S_i (Line 9). More details about Step 1 are given in Section IV-A.

In Step 2 of the algorithm (Lines 10-12), a multi-objective optimization is performed to obtain a set $\mathcal{P}_i = \{\chi_i^1, \dots, \chi_i^{|\mathcal{P}_i|}\}$ of Pareto-optimal data protection configurations for layer l_i . The NSGA-II algorithm takes as input the layer l_i , the channel importance scores S_i , and the dataset X_{leak} (Line 10). Layer l_i defines the output channels and feature map dimensions, S_i guides the initialization and mutation operators within NSGA-II, and X_{leak} is used to evaluate two of the three optimization objectives within NSGA-II, namely the reconstruction risk and the information leakage. NSGA-II explores different configurations by varying the size of protected data and its location in the output channels of l_i , while minimizing the reconstruction

risk, information leakage, and encryption/decryption overhead. Finally, one configuration $\chi_i \in \mathcal{P}_i$ is selected for layer l_i (Line 11) and added to set D (Line 12). More details about Step 2 are given in Section IV-B.

A. Channel Importance Analysis

To determine which portions of the communicated intermediate data should be protected, the proposed method first identifies the output channels that contribute most significantly to the CNN inference process. As summarized in Algorithm 1 (Lines 4-9), an importance score is computed for every output channel ch_j^{out} of layer l_i . The importance score is derived from three complementary metrics that capture different characteristics of the contribution of each channel to the final CNN prediction accuracy. More specifically, our proposed approach evaluates the magnitude of the convolutional filter parameters, the sensitivity of the loss function with respect to the channel activations, and the variability of the extracted feature responses. These metrics jointly capture structural, functional, and statistical aspects of the information represented by each channel.

The first metric (Algorithm 1, Line 5) measures the magnitude of the convolutional filter parameters using the L1 norm [20]. Filters with larger weight magnitudes typically contribute more strongly to the feature extraction process and therefore have a greater influence on the CNN output. In this formulation (Line 5), $W_j^{c,h,w}$ denotes the weight connecting the c -th input channel to output channel ch_j^{out} at spatial position (h, w) , b_j represents the bias associated with channel j , and k_h and k_w denote the height and width of the convolutional kernel.

The second metric (Algorithm 1, Line 6) evaluates the sensitivity of the CNN loss function with respect to the channel activations [25]. Channels whose activations produce larger gradients with respect to the loss function have a stronger influence on the final CNN prediction outcome. The gradients are computed over a sample dataset X_{imp} that is representative of the same task for which the CNN model has been trained. In this formulation (Line 6), $a_{j,n}^{h,w}$ denotes the activation of channel ch_j^{out} at spatial location (h, w) for the n -th input sample, while $\mathcal{L}_n$ represents the corresponding loss function.

The third metric (Algorithm 1, Line 7) captures the variability of the feature responses [22] produced by channel ch_j^{out} . Channels that exhibit higher activation variability across different input samples typically encode richer and more informative features, indicating a stronger contribution to the CNN inference process. The quantity μ_j in Line 7 denotes the mean activation of channel ch_j^{out} computed over the sample dataset X_{imp} .

Since the three metrics may operate on different numerical scales, they are normalized using min-max normalization (Algorithm 1, Line 8) such that their values lie within the range $[0, 1]$. In addition, because the three metrics capture complementary aspects of the channel importance, the normalized metrics are aggregated by summation to obtain the final importance score $\hat{s}_j$ for channel ch_j^{out} .

B. Multi-Objective Optimization

After computing the importance scores of the output channels of convolutional layer l_i (Algorithm 1, Lines 4-9), the next step is to determine how the communicated intermediate data should be protected (Algorithm 1, Lines 10-12). More specifically, for each output channel $ch_j^{out} \in l_i$, an optimization process should determine how much data should be encrypted and where the spatial location of this data is within the channel feature map. Encrypting a large portion of the intermediate data reduces the CNN input reconstruction risk because less information remains unprotected. However, this increases the encryption/decryption overhead. On the other hand, encrypting a small portion of the data reduces the overhead but may leave sufficient unprotected information for an attacker to reconstruct the original CNN input. Therefore, there is a trade-off between privacy protection and encryption/decryption overhead.

To explore this trade-off, we formulate our data protection problem as a multi-objective optimization problem and solve it using the NSGA-II algorithm. The goal is to find Pareto-optimal protection configurations for each layer l_i . Each configuration specifies the amount and location of the data that should be encrypted, within the output channels of the layer. All output channels of layer l_i are considered during the optimization process. The computed channel importance scores S_i are used to guide the search toward Pareto-optimal protection configurations. In this way, protection configurations of channels with higher importance scores are more likely to have a larger amount of encrypted data, while channels with lower importance scores are more likely to have a smaller amount of encrypted data.

Within NSGA-II, for our optimization problem, each individual/chromosome χ_i represents a candidate data protection configuration for layer l_i . We encode the chromosome as follows

$$\chi_i = \{z_1, z_2, \dots, z_{|l_i|}\}, \quad z_j = (p_j, y_j, x_j), \quad (1)$$

where each gene z_j specifies the amount and location of the data to be encrypted within output channel $ch_j^{out} \in l_i$. Parameter $p_j \in [0, 1]$ determines the amount of encrypted data as a fraction of the whole data within the channel. The spatial location of the encrypted data region within the channel is specified by the parameters y_j and x_j , which are the coordinates of the top-left corner of this region. Starting from the top-left corner (y_j, x_j) , the region expands horizontally and vertically to cover a square of size $r_j \times r_j$ within the feature map of channel ch_j^{out} . The side length r_j is determined as follows

$$r_j = \left\lceil \sqrt{p_j \cdot H \cdot W} \right\rceil, \quad (2)$$

where H and W denote the height and width of the feature map. To ensure that the encrypted region remains fully contained within the feature map boundaries, the following constraints are imposed on parameters y_j and x_j during the optimization process

$$0 \leq y_j \leq H - r_j, \quad 0 \leq x_j \leq W - r_j, \quad (3)$$

where y_j and x_j are integer values. We consider only square regions of data to encrypt in order to keep the protection configuration compact and to reduce the optimization search space. For a given $p_j \in [0, 1]$, the amount of encrypted data is $p_j \times H \times W$ data elements out of all $H \times W$ data elements of the channel. When $p_j = 0$, no elements of the channel are encrypted and parameters y_j and x_j are ignored.

For our optimization problem, a population in NSGA-II is a set of individuals, i.e., candidate data protection configurations for layer l_i . During the optimization process, these configurations are evolved through genetic operations such as selection, crossover, and mutation. To facilitate a more effective optimization process, we incorporate the computed channel importance scores S_i associated with l_i into the initialization stage of NSGA-II. In this stage, some individuals are initialized with very small values of parameter p_j to represent low-overhead configurations, while the remaining individuals are initialized using importance-guided sampling based on S_i . In the sampling strategy, channels with higher importance scores are more likely to be assigned larger p_j values, whereas channels with lower importance scores are more likely to remain unprotected or assigned with smaller p_j values.

Moreover, to improve the search efficiency of NSGA-II during the optimization process, we propose a directional importance mutation (DIM) operator for mutations of individuals in NSGA-II. For every individual/candidate configuration undergoing mutation, we first determine a stochastic volume k , which defines the number of channels to be modified. We then perform a weighted random selection to identify k channels with high importance and k channels with low importance based on their importance scores $s_j \in S_i$. For the selected channels with high importance, our mutation operator increases the value of parameter p_j by a random step size. Conversely, for the selected channels with low importance, it decreases the value of p_j by a random step size. Additionally, the spatial coordinates (y_j, x_j) of these modified channels are stochastically shifted to explore different data locations within the feature maps of the channels. For crossover/recombination of individuals, we apply the standard simulated binary crossover (SBX) operator, which blends the parameters of parent configurations to produce offspring configurations.

During the optimization process, NSGA-II aims at minimizing three objective functions, namely CNN Input Reconstruction Risk, Information Leakage, and Encryption Overhead. Thus, every candidate data protection configuration is evaluated using these functions described below.

1) *CNN Input Reconstruction Risk (f_1)*: The reconstruction risk f_1 quantifies the classification-based reconstruction accuracy. Similar to the reconstruction-based attack model in [33], a trained convolutional decoder $D_i(\cdot)$ is used to reconstruct the CNN input data from an attacker-accessible intermediate representation $\hat{\mathbf{A}}_{i,n}$. The reconstructed input data is obtained as $\hat{x}_{i,n} = D_i(\hat{\mathbf{A}}_{i,n})$ and is then fed into the original CNN model $cnn(\cdot)$ to obtain the predicted label

$\hat{y}_{i,n} = \arg \max cnn(\hat{x}_{i,n})$. Then, the reconstruction risk is computed as follows

$$f_1 = Acc_{recon} = \frac{1}{|X_{leak}|} \sum_{n=1}^{|X_{leak}|} \mathbb{I}[\hat{y}_{i,n} = y_n]. \quad (4)$$

In this equation, y_n is the true label of the original CNN input data x_n , and $\mathbb{I}[\cdot]$ is an indicator function that returns 1 if $\hat{y}_{i,n} = y_n$ and 0 otherwise.

Minimizing f_1 reduces the ability of an attacker to reconstruct input data that can still be classified correctly by the CNN model. Since the protected part of the intermediate representation is determined by configuration parameter p_j , increasing the value of p_j for important channels reduces the amount of unprotected intermediate data available to the attacker, which can decrease the reconstruction accuracy.

2) *Information Leakage (f_2)*: The information leakage f_2 quantifies the similarity between the reconstructed input data $\hat{x}_{i,n}$ and the original CNN input data x_n using the structural similarity index measure (SSIM) [34] as follows

$$f_2 = \frac{1}{|X_{leak}|} \sum_{n=1}^{|X_{leak}|} SSIM(\hat{x}_{i,n}, x_n). \quad (5)$$

Minimizing f_2 implies that the reconstructed data is less similar to the original CNN input data, leading to lower information leakage. Function f_2 is mostly affected by the spatial coordinates (y_j, x_j) in a candidate configuration, as they allow the optimization to target specific sensitive regions within the feature maps that contribute most to the leakage.

3) *Encryption Overhead (f_3)*: The encryption overhead is computed as follows

$$f_3 = \frac{E}{|l_i| \times H \times W}, \quad (6)$$

where $E = \sum_{j=1}^{|l_i|} r_j \times r_j$ is the total number of encrypted data elements in layer l_i and $|l_i| \times H \times W$ is the total number of data elements in the layer. Minimizing f_3 reduces the amount of intermediate data that is authenticated and encrypted. Recall that the size $r_j \times r_j$ of an encrypted data region depends on configuration parameter p_j (Eq. 2). Thus, minimizing f_3 by reducing the value of p_j also implies low encryption overhead in terms of time and energy consumption, which is very important for resource-constrained edge devices.

V. EXPERIMENTAL EVALUATION

In this section, we evaluate the effectiveness of the proposed importance-aware authenticated encryption approach for secure distributed CNN inference on edge devices. The evaluation aims to analyze the trade-off between the CNN input reconstruction risk, information leakage, and the overhead introduced by authenticated encryption. We first describe the experimental setup, including the CNN models, datasets, NSGA-II configuration parameters, and the different evaluation experiments. Then, we present and discuss the experimental results for every experiment.

A. Experimental Setup

We evaluate the proposed approach using three representative CNN models, namely VGG16, ResNet50, and MobileNet, that have different architectural characteristics and feature map structures. This allows us to analyze the approach effectiveness across diverse CNN model architectures. VGG16 follows a pipeline-like model architecture of convolutional layers, ResNet50 uses residual connections in the architecture, and MobileNet uses depthwise separable convolutions. Using these three models, we conduct experiments on two image classification datasets, namely CIFAR-10 and CIFAR-100. CIFAR-10 contains 60,000 color images belonging to 10 classes, while CIFAR-100 contains 60,000 color images belonging to 100 classes. In all experiments, the input images are resized to 224×224 pixels before being fed to the CNN models.

For each model–dataset pair, we first train the CNN model for the target classification task. Then, for each model, we monitor two layers whose output feature maps are assumed to be communicated between edge devices and exposed to an attacker. Monitoring these layers, allows us to evaluate how the amount of exposed information changes across different depths of the CNN model. The monitored layers are selected to represent early and late feature map representations for each CNN model. For VGG16, we monitor the 2nd and 13th convolutional layer. For ResNet50, we monitor layers from the 2nd and 5th residual stage. For MobileNet, we monitor the 1st and 13th convolutional block. For every monitored layer, we train a decoder model that simulates a reconstruction attacker as in [33]. This decoder-based attack is used during the NSGA-II optimization process to evaluate the CNN input reconstruction risk (f_1) and information leakage (f_2) for a given data protection configuration. The NSGA-II configuration parameters for all experiments are the same, namely the population size is set to 60, the initial population is evolved for 80 generations, crossover is applied with a probability of 0.9, and mutation is applied with a probability of 0.5.

To evaluate our proposed approach, we conduct two experiments. In the first experiment, we evaluate the effectiveness of our approach in reducing the CNN input reconstruction risk and information leakage while partially encrypting the output feature maps of monitored layers. To this end, we first apply our approach (Algorithm 1) on the aforementioned CNN models, datasets, and monitored layers in order to obtain Pareto-optimal data protection configurations (Line 10 in Algorithm 1). Then, we compare the CNN input reconstruction risk, information leakage, and encryption overhead metrics for these Pareto-optimal configurations with the same metrics for a baseline configuration where the output feature maps are fully encrypted. In addition, we analyze how the dataset complexity affects the amount of data encryption needed to reduce the CNN input reconstruction risk and information leakage. The CIFAR-10 and CIFAR-100 datasets contain the same number of images with the same input resolution, but CIFAR-100 is more complex in the sense that the images belong to 100 classes instead of 10 classes. Moreover, we analyze how the

CNN input reconstruction risk and information leakage change across the different monitored layers by comparing the Pareto-optimal protection configurations obtained for the early and late monitored layers of each CNN model. Output feature maps of early and late layers contain different amounts and types of information about the original CNN input data.

In the second experiment, we evaluate the effect of using channel importance scores to guide NSGA-II in our approach (Lines 4-9 in Algorithm 1) on the quality of the obtained Pareto-optimal data protection configurations (Line 10). To this end, we compare our proposed approach (Algorithm 1) with two baseline approaches, namely *Random Importance* and *No Importance*. The three approaches use the same NSGA-II configuration parameters and objective functions but differ in how channel importance is used during the NSGA-II initialization phase and in the mutation operation. The proposed approach uses our importance scores in initialization and mutation (our DIM operator) as explained in Section IV-B, *Random Importance* replaces our importance scores with randomly assigned scores, and *No Importance* does not use any importance scores in initialization and mutation. This experiment is conducted using the ResNet50 model trained on CIFAR-100, and the monitored layer is the third residual stage of ResNet50.

B. Experiment 1: Results and Discussion

The results from the first experiment are shown in Fig. 4. The results for ResNet50 are shown in Fig. 4a and Fig. 4b, the results for VGG16 are shown in Fig. 4c and Fig. 4d, and the results for MobileNet are shown in Fig. 4e and Fig. 4f. In each plot, the axes represent the CNN input reconstruction risk, information leakage, and encryption overhead. The blue circles and orange squares correspond to Pareto-optimal protection configurations obtained when the CIFAR-10 and CIFAR-100 datasets are used, respectively. The stars and pentagons correspond to the baseline configuration, where the output feature maps of the monitored layers are fully encrypted.

1) *Effectiveness of the Proposed Approach:* The results in Fig. 4 show that the proposed approach is effective across different CNN models in comparison to the aforementioned baseline configuration in every plot. This configuration provides the strongest protection, i.e., very low CNN input reconstruction risk and low information leakage, but it exhibits the full/highest encryption overhead. As can be seen in Fig. 4, the proposed approach finds Pareto-optimal protection configurations that approach the strongest protection level while encrypting only a portion of the output feature maps of the monitored layers, thereby reducing significantly the encryption overhead.

The results also show that the proposed approach is not limited to a specific CNN model architecture. It is effective for ResNet50 with residual connections, VGG16 with a sequential convolutional structure, and MobileNet with depthwise separable convolutions. This is important because different CNN architectures distribute the information differently across their intermediate feature maps. Therefore, using an optimization-

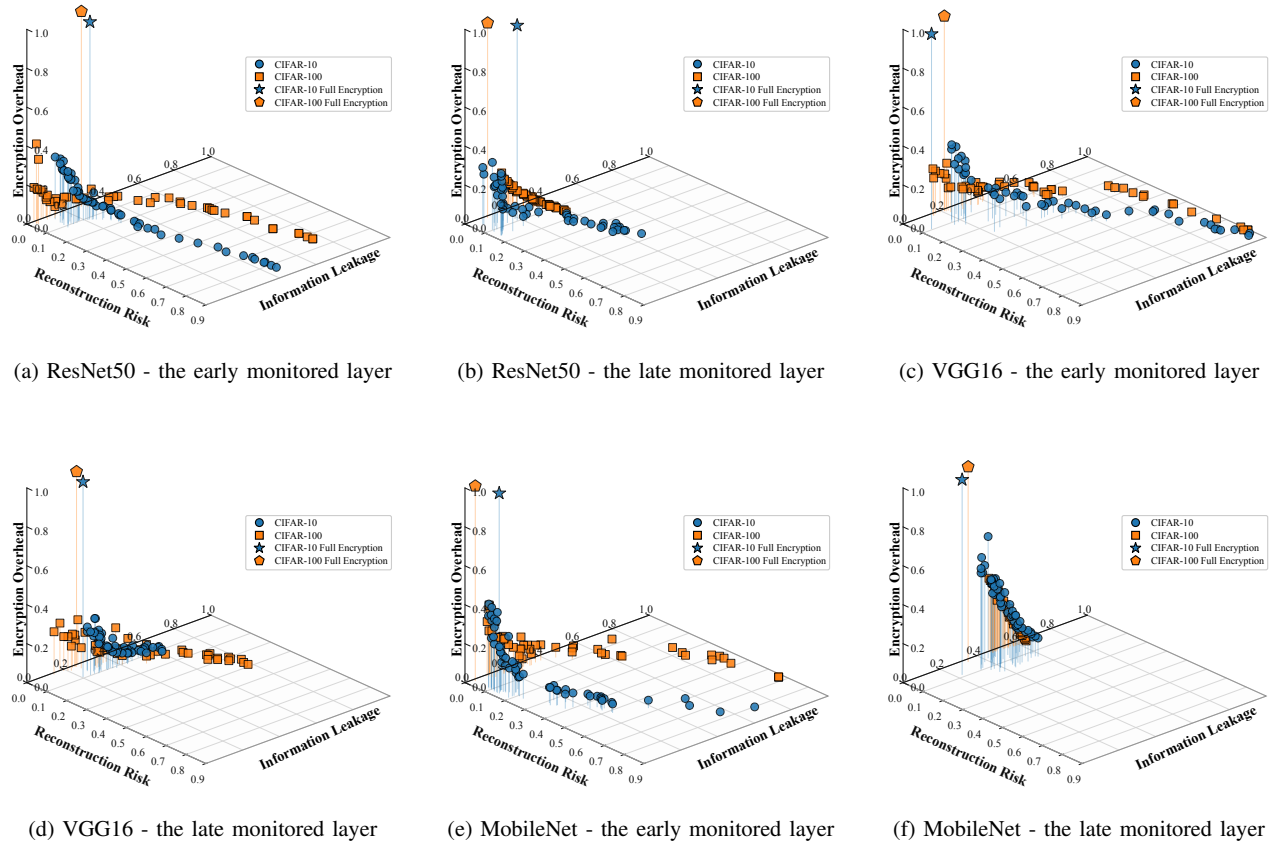


Fig. 4: Pareto-optimal protection configurations for ResNet50, VGG16, and MobileNet on CIFAR-10 and CIFAR-100.

based protection approach like ours for every CNN model is more suitable than applying a fixed data protection ratio to all models and monitored layers.

To make the trade-offs shown in Fig. 4 easier to interpret, Table I reports in more detail one representative low-overhead Pareto-optimal configuration for each CNN model, monitored layer, and dataset, taken from Fig. 4. Each representative configuration has the lowest CNN input reconstruction risk among the configurations which encryption overhead is at most 10%. This corresponds to the configuration selection step in Line 11 of Algorithm 1, where one data protection configuration is selected from the Pareto front produced by NSGA-II (Line 10). The *Model* column in Table I indicates the CNN architecture and monitored layer depth, while the *Data* column indicates the dataset. The *Sel. f_3* column shows the encryption overhead of the selected representative configurations. *Base f_1* and *Base f_2* show the CNN input reconstruction risk and information leakage, respectively, of the baseline. *Sel. f_1* and *Sel. f_2* show the CNN input reconstruction risk and information leakage of the selected configuration.

The numbers in Table I more clearly show the aforemen-

TABLE I: Low-overhead Pareto-optimal configurations.

Model	Data	Sel. f_3 (%)	Base f_1	Sel. f_1	Base f_2	Sel. f_2
ResNet-Early	CIFAR-10	7.32	0.096	0.102	0.239	0.249
ResNet-Early	CIFAR-100	8.79	0.012	0.000	0.283	0.107
ResNet-Late	CIFAR-10	8.59	0.093	0.063	0.183	0.139
ResNet-Late	CIFAR-100	9.91	0.011	0.000	0.111	0.257
VGG-Early	CIFAR-10	8.40	0.092	0.164	0.055	0.441
VGG-Early	CIFAR-100	8.30	0.011	0.008	0.212	0.387
VGG-Late	CIFAR-10	1.84	0.089	0.063	0.209	0.586
VGG-Late	CIFAR-100	7.90	0.011	0.023	0.258	0.374
MobileNet-Early	CIFAR-10	9.86	0.109	0.172	0.068	0.092
MobileNet-Early	CIFAR-100	9.47	0.011	0.023	0.044	0.315
MobileNet-Late	CIFAR-10	8.12	0.082	0.109	0.232	0.587
MobileNet-Late	CIFAR-100	4.07	0.016	0.016	0.336	0.621

tioned trend observed in Fig. 4. For the selected configurations, they require about 7.72% encryption overhead on average. At such low overhead, the CNN input reconstruction risk (*Sel. f_1*) remains very low, ranging from 0.000 to 0.172. For example, the selected configuration for ResNet50 on CIFAR-100 with the early monitored layer reaches CNN input reconstruction risk of 0.000 with only 8.79% encryption overhead.

Similarly, for MobileNet on CIFAR-100 with the early monitored layer, the selected configuration reaches reconstruction risk of 0.023 with 9.47% overhead. The information leakage values for the selected low-overhead configurations range from 0.092 to 0.621. In summary, both Fig. 4 and Table I show that the proposed approach delivers a diverse set of Pareto-optimal configurations, enabling an effective configuration selection with trade-off among reconstruction risk, information leakage, and encryption overhead.

2) *Effect of the Dataset Complexity*: Comparing Pareto-optimal configurations for CIFAR-10 and CIFAR-100 in Fig. 4 reveals that the dataset complexity affects the trade-off between the CNN input reconstruction risk, information leakage, and encryption overhead. The CIFAR-100 images have the same resolution as the CIFAR-10 images but they belong to 100 classes instead of 10 classes. Therefore, after partial encryption of the data in the output feature maps of the monitored layers, the remaining unencrypted data may still preserve some structural information while no longer containing enough classification-relevant information for the reconstructed data to be classified correctly among a larger number of classes.

This effect is clearly visible in the selected configurations reported in Table I. For the early monitored layers, the selected CIFAR-100 configurations reach lower reconstruction risk than the corresponding CIFAR-10 configurations for all three CNN models. ResNet50 reaches 0.000 on CIFAR-100 compared to 0.102 on CIFAR-10, VGG16 reaches 0.008 compared to 0.164, and MobileNet reaches 0.023 compared to 0.172. The same trend is visible for the late monitored layers, where the selected CIFAR-100 configurations also have lower reconstruction risk compared to the corresponding CIFAR-10 configurations.

The information leakage values do not always follow the same trend as the reconstruction risk. Some selected CIFAR-100 configurations have higher information leakage even though their reconstruction risk is lower. This shows that the information leakage and reconstruction risk metrics capture different aspects of data protection. The reconstruction risk quantifies whether the reconstructed CNN input data remains useful for the CNN classification task, whereas the information leakage quantifies the structural similarity of the reconstructed data to the original CNN input data. Therefore, minimizing both metrics is important while also keeping the encryption overhead as low as possible.

3) *Effect of the Monitored Layer Position*: The position of the monitored layer in the CNN model architecture affects both the CNN input reconstruction risk and information leakage. The early monitored layers are closer to the CNN input and therefore usually preserve more spatial structure and low-level feature information. This makes the output feature maps of these layers more useful for an attacker to analyze. In Fig. 4a, Fig. 4c, and Fig. 4e, this effect is reflected by the Pareto-front trade-off among the CNN input reconstruction risk, information leakage, and encryption overhead. Reducing the encryption overhead relative to the fully encrypted baseline can leave more useful information exposed to the attacker.

The late monitored layers show different behavior. Since the

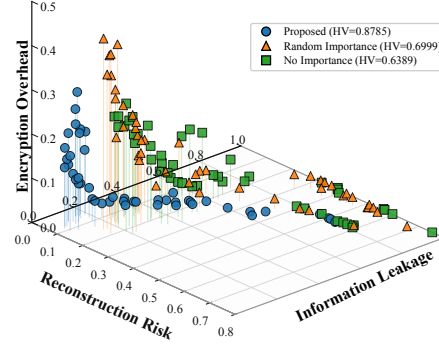


Fig. 5: Comparison of proposed *Importance-aware Approach* with the *Random Importance* and *No Importance* baselines.

data in their output feature maps is more task-specific and less directly related to the input data, the CNN input reconstruction risk is often lower for these layers. However, the information leakage values in Table I show that low reconstruction risk does not always imply low information leakage. For example, the selected configurations for MobileNet with the late monitored layer provide low reconstruction risk, but the information leakage values remain relatively high compared with some early layer configurations. This means that the reconstructed CNN input data may lose the details needed for correct classification by the CNN model while still preserving some structural similarity information that may be exploited by an attacker. Therefore, both the CNN input reconstruction risk and information leakage metrics should be considered when deciding how much data to encrypt at each monitored layer. In the proposed approach, the layer-wise optimization is important because it allows the Pareto-optimal data protection configurations to reflect the feature map information content of each monitored layer instead of applying uniform protection across monitored layers irrespective of their positions and information content.

C. Experiment 2: Results and Discussion

The results from the second experiment are shown on the 3-dimensional plot in Fig. 5 where the dimensions represent the CNN input reconstruction risk, information leakage, and encryption overhead. The blue circles show the Pareto front obtained by our proposed approach. The orange triangles and the green squares show the Pareto fronts obtained by the *Random Importance* and *No Importance* baseline approaches, respectively. The hypervolume (HV) for each Pareto front is computed and shown as well.

The proposed approach delivers the highest hypervolume value of 0.8785 compared to *Random Importance* and *No Importance* with hypervolume values of 0.6999 and 0.6389, respectively. A higher hypervolume indicates a better Pareto front meaning that the obtained data protection configurations cover a larger region of desirable trade-offs among the three objectives. In other words, the configurations obtained by the proposed approach have better overall quality compared to

the configurations obtained by the baselines, i.e., they provide lower CNN input reconstruction risk, lower information leakage, and lower encryption overhead.

These results clearly indicate that our channel importance scores guide the NSGA-II search towards higher-quality data protection configurations. Compared with *Random Importance*, the proposed approach does not rely on randomly generated scores to decide which output channels should receive higher priority for authenticated encryption. Compared with *No Importance*, the proposed approach uses the channel importance scores during the initialization phase and mutation operators of NSGA-II. In this way, candidate configurations are generated and evolved by giving higher priority to output channels that contribute more to the CNN inference process. This helps the optimization process find Pareto-optimal data protection configurations with lower CNN input reconstruction risk, lower information leakage, and lower encryption/decryption overhead.

VI. CONCLUSIONS

In this paper, we propose a lightweight importance-aware authenticated encryption approach for secure distributed CNN inference on edge devices. The proposed approach selectively encrypts only a portion of the communicated intermediate data instead of encrypting the entire data. To determine which portion of data should be encrypted, it computes channel importance scores and uses them to guide an NSGA-II-based multi-objective optimization process that tries to jointly minimize the CNN input reconstruction risk, information leakage, and encryption overhead metrics. The experimental results show that our approach can obtain Pareto-optimal data protection configurations that reduce both the reconstruction risk and information leakage. The results also indicate that the dataset complexity and the position of the monitored layers affect the trade-off between the metrics, strongly suggesting that the amount and location of encrypted data should be determined separately for each monitored layer. Overall, our approach enables an effective trade-off between data privacy preservation and authenticated encryption overhead for distributed CNN inference on resource-constrained edge devices.

REFERENCES

- [1] K. Sun, X. Wang, X. Miao, and Q. Zhao, "A review of ai edge devices and lightweight cnn and llm deployment," *Neurocomputing*, 2025.
- [2] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in Neural Information Processing Systems*, 2015.
- [3] M. Nagel *et al.*, "A white paper on neural network quantization," *arXiv preprint arXiv:2106.08295*, 2021.
- [4] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.
- [5] M. Han *et al.*, "Transformer-based distributed task offloading and resource management in cloud-edge computing networks," *IEEE Journal on Selected Areas in Communications*, 2025.
- [6] X. Guo, A. D. Pimentel, and T. Stefanov, "Automated exploration and implementation of distributed CNN inference at the edge," *IEEE Internet of Things Journal*, pp. 5843–5858, 2023.
- [7] E. Tang and T. Stefanov, "Low-memory and high-performance CNN inference on distributed systems at the edge," in *Proceedings of the 14th IEEE/ACM International Conference on Utility and Cloud Computing Companion*, 2022.
- [8] R. Stahl *et al.*, "Fully distributed deep learning inference on resource-constrained edge devices," in *SAMOS*, 2019.
- [9] Z. Zhao *et al.*, "Deepthings: Distributed adaptive deep learning inference on resource-constrained IoT edge clusters," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2018.
- [10] C. Dobraunig, M. Eichlseder, F. Mendel, and M. Schl  ffer, "Ascon v1.2: Lightweight authenticated encryption and hashing," in *International Conference on Selected Areas in Cryptography*, 2016.
- [11] X. Guo, A. D. Pimentel, and T. Stefanov, "Hierarchical design space exploration for distributed CNN inference at the edge," in *Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2023, pp. 545–556.
- [12] X. Hou, Y. Guan, T. Han, and N. Zhang, "Distredge: Speeding up convolutional neural network inference on distributed edge devices," in *IPDPS*, 2022, pp. 1097–1107.
- [13] L. Zeng *et al.*, "Coedge: Cooperative DNN inference with adaptive workload partitioning over heterogeneous edge devices," *IEEE/ACM Transactions on Networking*, 2021.
- [14] J. Mao, X. Chen, K. W. Nixon, C. Krieger, and Y. Chen, "MoDNN: Local distributed mobile computing system for deep neural network," in *DATE*, 2017.
- [15] S. Zhang *et al.*, "Deepslicing: Collaborative and adaptive CNN inference with low latency," *IEEE Transactions on Parallel and Distributed Systems*, pp. 2175–2187, 2021.
- [16] E. Tang, X. Guo, and T. Stefanov, "The effects of partitioning strategies on energy consumption in distributed CNN inference at the edge," *arXiv preprint arXiv:2210.08392*, 2022.
- [17] X. Guo, Q. Jiang, A. D. Pimentel, and T. Stefanov, "RobustDiCE: Robust and distributed CNN inference at the edge," in *ASPDAC*, 2024, p. 26–31.
- [18] —, "Model and system robustness in distributed cnn inference at the edge," *Integration*, 2025.
- [19] A. Kumar *et al.*, "Pruning filters with l1-norm and capped l1-norm for cnn compression," *Applied Intelligence*, pp. 1152 – 1160, 2020.
- [20] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," *ArXiv*, vol. abs/1608.08710, 2016.
- [21] Y. He, P. Liu, L. Zhu, and Y. Yang, "Filter pruning by switching to neighboring cnns with good attributes," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 8044–8056, 2023.
- [22] X. Sun *et al.*, "Pruning filters with L1-norm and standard deviation for CNN compression," in *ICMV 2018*, 2019, p. 110412J.
- [23] Z. Wu *et al.*, "A filter pruning method of cnn models based on feature maps clustering," *Applied Sciences*, 2022.
- [24] H. Li, C. Ma, W. Xu, and X. Liu, "Feature statistics guided efficient filter pruning," ser. IJCAI'20, 2021.
- [25] P. Molchanov *et al.*, "Pruning convolutional neural networks for resource efficient inference," *arXiv: Learning*, 2016.
- [26] H. Yu *et al.*, "Gradient aggregation based fine-grained image retrieval: A unified viewpoint for cnn and transformer," *Pattern Recogn.*, 2024.
- [27] H. Yan *et al.*, "Cifs: Improving adversarial robustness of cnns via channel-wise importance-based feature selection," in *International Conference on Machine Learning*, 2021, pp. 11 693–11 703.
- [28] Y. Mao *et al.*, "Secure split learning against property inference, data reconstruction, and feature space hijacking attacks," in *ESORICS*, 2024, pp. 23–43.
- [29] X. Ding *et al.*, "Privacy-preserving feature extraction via adversarial training," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1967–1979, 2022.
- [30] R. Gilad-Bachrach *et al.*, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *Proceedings of The 33rd International Conference on Machine Learning*, 2016, pp. 201–210.
- [31] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, "Gazelle: a low latency framework for secure neural network inference," in *USENIX*, 2018, p. 1651–1668.
- [32] P. Mohassel and Y. Zhang, "Securml: A system for scalable privacy-preserving machine learning," in *2017 SP*, 2017, pp. 19–38.
- [33] F. Mehrafrooz, R. Siyadatzaheh, N. Mentens, and T. Stefanov, "A security analysis of cnn partitioning strategies for distributed inference at the edge," in *Security, Privacy, and Applied Cryptography Engineering*, 2026, pp. 262–281.
- [34] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, pp. 600–612, 2004.