MOSAIC: A Prune-and-Assemble Approach for Efficient Model Pruning in Privacy-Preserving Deep Learning

Yifei Cai
ycai001@odu.edu
Old Dominion University

Qiao Zhang
qiaozhang@cqu.edu.cn
Chongqing University

Rui Ning
rning@cs.odu.edu
Old Dominion University

Chunsheng Xin
cxin@odu.edu
Old Dominion University

Hongyi Wu
mhwu@arizona.edu
University of Arizona

ABSTRACT

To enable common users to capitalize on the power of deep learning, Machine Learning as a Service (MLaaS) has been proposed in the literature, which opens powerful deep learning models of service providers to the public. To protect the data privacy of end users, as well as the model privacy of the server, several state-of-the-art privacy-preserving MLaaS frameworks have also been proposed. Nevertheless, despite the exquisite design of these frameworks to enhance computation efficiency, the computational cost remains expensive for practical applications. To improve the computation efficiency of deep learning (DL) models, *model pruning* has been adopted as a strategic approach to remarkably compress DL models. However, for practical deep neural networks, a problem called *pruning structure inflation* significantly limits the pruning efficiency, as it can seriously hurt the model accuracy. In this paper, we propose MOSAIC, a highly flexible pruning framework, to address this critical challenge. By first *pruning* the network with the carefully selected basic pruning units, then *assembling* the pruned units into suitable HE Pruning Structures through smart channel transformations, MOSAIC achieves a high pruning ratio while avoiding accuracy reduction, eliminating the problem plagued by the pruning structure inflation. We apply MOSAIC to popular DL models such as VGG and ResNet series on classic datasets such as CIFAR-10 and Tiny ImageNet. Experimental results demonstrate that MOSAIC effectively and flexibly conducts pruning on those models, significantly reducing the Perm, Mult, and Add operations to achieve the global cost reduction without any loss in accuracy. For instance, in VGG-16 on Tiny ImageNet, the total cost is reduced to 21.14% and 29.49% under the MLaaS frameworks GAZELLE and CrypTFlow2, respectively.

CCS CONCEPTS

• **Security and privacy** → *Privacy-preserving protocols.*

KEYWORDS

Model Pruning, Machine Learning as a Service, Privacy-Preserving Computation, Homomorphic Encryption

ACM Reference Format:
Yifei Cai, Qiao Zhang, Rui Ning, Chunsheng Xin, and Hongyi Wu. 2024. MOSAIC: A Prune-and-Assemble Approach for Efficient Model Pruning in Privacy-Preserving Deep Learning. In *ACM Asia Conference on Computer and Communications Security (ASIA CCS '24), July 1–5, 2024, Singapore, Singapore.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3634737.3637680

1 INTRODUCTION

Today, deep learning (DL) is widely used to improve productivity and solve challenging problems that have profound societal impacts. However, building a DL model requires access to extensive amounts of data, significant computing power, and professional expertise, which are a daunting challenge and often not possible for most users/organizations. To address this challenge, the Machine Learning as a Service (MLaaS) has been proposed as a practical solution [50]. MLaaS empowers technology giants with abundant resources to create well-trained advanced DL models and offer them as a service to a broad range of users.

However, privacy is a critical concern in MLaaS. On the one hand, the clients of MLaaS do not want the server to have access to their private data that can be precious business data or sensitive information, such as personal medical records. On the other hand, the server does not want to share its model parameters as they are considered valuable intellectual property that has been trained with tremendous computing resource and expertise. To effectively address the critical privacy issue, privacy-preserving MLaaS strategically integrates cryptographic primitives into the computation process of the DL model. Several ingeniously designed privacy-preserving frameworks have made inspiring efforts to bring MLaaS into practice [3, 10, 12, 21, 22, 25, 27, 29–31, 33, 36–41, 47–49, 52, 53]. The most commonly adopted cryptographic primitives in these frameworks include Homomorphic Encryption (HE) [11], Garbled Circuits (GC) [2], Oblivious Transfer (OT) [6], and Secret Sharing (SS) [44]. Of these cryptographic primitives, HE is more efficient for linear computation [4, 11]. On the other hand, GC and OT are more computationally-efficient for nonlinear operations [10]. As the DL model is the combination of linear and nonlinear functions, privacy-preserving DL frameworks typically adopt HE for linear operations and GC/OT for nonlinear operations. Examples of such frameworks include HE-GC-based frameworks like GAZELLE [22] and DELPHI [29], as well as the HE-OT-based framework CrypTFlow2 [38]. Despite the efforts to improve efficiency, the computation cost of these frameworks is still too high for practical applications. For instance, in our evaluations utilizing the CrypTFlow2 framework and the CKKS algorithm [9], the execution time for VGG-16 is approximately 248 seconds. This is observed even with just one CIFAR-10 image as input on an Intel i5 2.9GHz CPU. Such a duration is considered unacceptable for a majority of applications.

To improve the computation efficiency of DL models, *model pruning* [16, 28, 51] has been adopted as a strategic approach to remarkably compress DL models such as ResNet, which are usually very large and complex. The computational cost of a DL model arises from the calculations performed between its parameters and the input vectors. Model pruning eliminates a large number of model parameters, thereby eradicating the associated calculations. There are generally two types of pruning methods: unstructured pruning [13, 15] and structured pruning [28, 51]. Unstructured pruning removes individual model weights, while structured pruning removes model structures such as filters and even layers [28, 51].

Challenges for HE-based Model Pruning: Unfortunately, while existing model pruning methods have demonstrated a superior performance in reducing the computation time of DL models [13, 16, 28, 51], it was found that directly applying those plaintext model pruning methods onto privacy-preserving MLaaS frameworks provides little or no help in reducing their computation time [7]. The reason is summarized as follows. The privacy-preserving MLaaS frameworks are based on packed HE, which performs calculation by using three basic HE operations: Homomorphic Addition (Add), Homomorphic Multiplication (Mult), and Homomorphic Permutation (Perm). In the packed calculation between a ciphertext (the input features) and a plaintext vector (model parameters), the Perm operation over the ciphertext and the subsequent Mult or Add operation can be eliminated only if all the parameters in the plaintext vector are pruned. However, the plaintext model pruning schemes are oblivious to the packing structures and hence rarely obtain the desired pruning structures, resulting in marginal or no reduction in the corresponding HE-based computation, even though the model is significantly pruned. For example, experiments show that although the well-known pruning algorithm [16] can prune up to 65% of parameters in a convolution layer from AlexNet, it only results in a 3.6% reduction in HE operations. Moreover, pruning 90.8% of weights in a fully connected (FC) layer would not even reduce a single Perm operation out of the total 4096 Perm operations [7].

To address the above challenge, an HE-friendly structured pruning method, Hunter [7], was proposed to identify the novel HE Pruning Structures. It shows that an HE operation can be eliminated if all the elements in the corresponding HE Pruning Structure are pruned. Hunter performs pruning on these HE Pruning Structures in order to minimize the number of Perm operations and subsequent Mult and Add operations, which leads to a reduction in computing cost.

Limitation of Existing HE-friendly Pruning Schemes: However, our recent study has shown that Hunter cannot scale well to very deep networks. In privacy-preserving MLaaS frameworks, multiple input channels are often encrypted and packed into a single ciphertext for efficiency [1, 32, 39]. As we know, the size of input features generally decreases from the first layer to deeper layers in a DL model. For instance, in our experiments with the VGG-16 model on the CIFAR-10 dataset, the input size changes from 32×32 at the first layer to 8×8 at the 13th layer. For both safety and efficiency reasons, the length of a single ciphertext is typically set as 8192 slots (4096 slots for the first layer) in privacy-preserving MLaaS frameworks [1, 32, 39]. Thus, the number of input features that one ciphertext contains, denoted as c_n, increases significantly from 4 at the first layer to 128 at the 13th layer. Larger values of c_n correspond to larger HE Pruning Structures, which is called *pruning structure inflation.* An immediate consequence of such pruning structure inflation is that the pruning granularity becomes coarse, i.e., one HE Pruning Structure contains a large number of elements that must be removed altogether in order to eliminate the corresponding expensive HE operations [7]. *However, this often hurts the DL model's accuracy because this aggressive pruning can damage the essential model structures.*

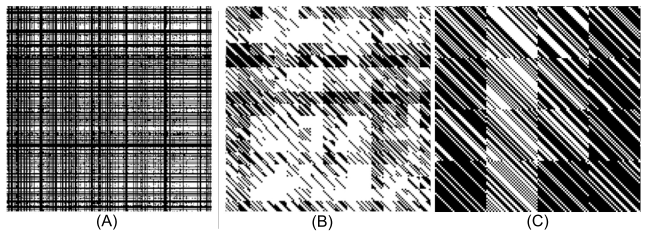


Figure 1: Pruned Convolutional Layers based on (a) traditional plaintext pruning, (b) Hunter with $C_n = 8$, and (c) Hunter with $C_n = 32$.

Figure 1 illustrates the binary representation of three pruned convolutional layers, where pruned parameters are shown in black, while unpruned parameters are shown in white. In Figure 1(A), we present the result of the so-called natural pruning, used in some plaintext model pruning schemes. For a well-trained model, unimportant parameters that fall below the defined threshold are pruned. Consequently, the remaining parameters represent essential model structures. Figure 1(B) and (C) depict the layers pruned by the HE Pruning Structures adopted by Hunter with different c_n values: 8 and 32, respectively. It is evident that the remaining parameters, shown in white, exhibit a distinct pattern from those in Figure 1(A). While many HE Pruning Structures are eliminated to result in cost reduction, a comparison with the essential model structures reveals that numerous important parameters have been pruned as well. As a consequence, the model's functionality is bound to be affected and may not be completely restored even with subsequent retraining.

In general, as c_n increases, the size of the HE Pruning Structure becomes larger, resulting in a coarser granularity for pruning, which, however, may lead to significant accuracy loss. Our experiments have shown that models become very sensitive to pruning when c_n is greater than 4 and pruning can easily cause significant accuracy loss. To avoid such accuracy loss, Hunter has to limit pruning efficiency, resulting in a sub-optimal pruning. Additionally, deep layers with large c_n usually contain dense computation, which also limits the potential reduction of the model's overall computational cost. For example, in our experiments with VGG-16 on the

CIFAR-10 dataset, to maintain the model accuracy, only 47.29% cost reduction can be achieved by Hunter's method. While it is possible to decrease c_n by reducing the size of the ciphertext, this also decreases the computation efficiency. More importantly, decreasing the number of slots of a ciphertext may result in safety issue of the encryption [1, 32, 39].

Our Contributions: To address the critical challenge of pruning structure inflation, in this paper, we propose a highly flexible pruning strategy, called MOSAIC, which is able to achieve fine-grained pruning in all layers, enabling a high pruning ratio while avoiding accuracy loss. MOSAIC employs a novel "*Prune-and-Assemble*" approach, which first *prunes* the convolutional layer with tessellated basic units, and then *assembles* the pruned units into HE Pruning Structures through a channel transformation scheme. (This process can be likened to creating a mosaic pattern by assembling small regular units.) As a result, the operations related to these HE Pruning Structures can be eliminated.

In our experiments[1], MOSAIC has been applied to popular DL models such as the VGG [45] and ResNet [18] series, utilizing classic datasets including CIFAR-10 [23] and Tiny ImageNet [46]. Experimental results demonstrate that MOSAIC can perform efficient and precise pruning, thus significantly reducing the computation costs of HE Perm and subsequently Mult and Add without sacrificing model accuracy. The cost reduction is remarkable, for instance, in the GAZELLE framework [22], MOSAIC can reduce the total cost by 78.86% in VGG-16 on Tiny ImageNet and save 91.27% of computational cost in ResNet-50 on Tiny ImageNet. Even under the deep optimized framework CrypTFlow2 [38], MOSAIC achieves a further cost reduction of up to 70.51%.

Note that in this paper, we often give examples for MOSAIC under the GAZELLE and CrypTFlow2 frameworks, which are two state-of-the-art privacy-preserving MLaaS frameworks. However, it is noteworthy that MOSAIC can also be applied on other frameworks as long as they use similar Multiple Input Multiple Output (MIMO) schemes for the HE computation, such as DELPHI [29] and GALA [53].

The remainder of this paper is organized as follows: Section 2 introduces the system model, threat model, packed HE, and state-of-the-art HE-friendly pruning schemes. Section 3 proposes MOSAIC, our efficient and flexible pruning schemes, including prune-and-assemble strategy, corresponding inter-layer channel transformation coordination, followed by the security analysis. In Section 4, we discuss the experimental results. Finally, Section 5 concludes the paper.

2 PRELIMINARIES

2.1 System Model

In this paper, we consider MLaaS, as shown in Figure 2, which involves two parties: the client (C) and the server (S). The client owns sensitive data, such as medical records from a medical practitioner, while the server has a well-performed DL model that can provide prediction results for the client. However, privacy issues arise during the interaction between the two parties. The client does not want any other parties, including the server, to know its private data. Meanwhile, the server is unwilling to make its model parameters public, as training that model involves significant resources. To this end, privacy-preserving MLaaS aims to guarantee that the client's input is fully protected from the server, while the server's model parameters are entirely blind to the client.

[1]The pruned model and source code are available at *github.com/caiyifei2008/MOSAIC.*

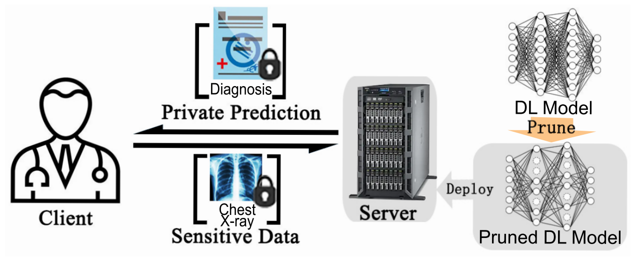


Figure 2: MLaaS with a Model Pruning Phase.

In line with the existing works on privacy-preserving MLaaS [3, 10, 12, 21, 22, 25, 27, 29–31, 33, 36–41, 47–49, 52, 53], we focus on deep Convolutional Neural Networks (CNNs), which are one of the most important and successful DL models widely used in various applications [24, 42, 45]. Convolution and dot product are two fundamental linear functions in CNNs. Convolution involves convolving the input feature map with the kernels of the convolutional layer, while the dot product is calculated between an input vector and the weight matrix of the FC layer. In MLaaS, the kernels and weight matrices are located in the server, whereas the input feature map and input vector are sourced from the client.

Please note that this paper primarily focuses on optimizing the computation efficiency of linear computations, which constitutes over 90% of the total cost of a DL model [53]. Meanwhile, we follow the efficient privacy-preserving nonlinear computation, such as max-pooling and ReLU, introduced in the state-of-the-art frameworks such as GAZELLE [22] and CrypTFlow2 [38]. Within the realm of linear computations, our emphasis is on convolution since it dominates the linear computations in the HE-based scenario. For instance, according to our experimental results, in VGG-16, convolution accounts for approximately 97.8% of the linear cost, and this ratio is even higher, reaching 99.05%, for ResNet-50.

2.2 Threat Model

MOSAIC follows the semi-honest adversary model, which is used in many state-of-the-art frameworks, such as GAZELLE [22], DELPHI [29], CrypTFlow2 [38] and MiniONN [27]. Under this model, both the client and the server follow the protocol while attempting to deduce extra information from the exchanged messages. For example, during the interaction in MLaaS, the server tries to discern the client's private input. In Section 3.5, we analyze and demonstrate that MOSAIC is secure under the semi-honest assumption.

2.3 Packed HE

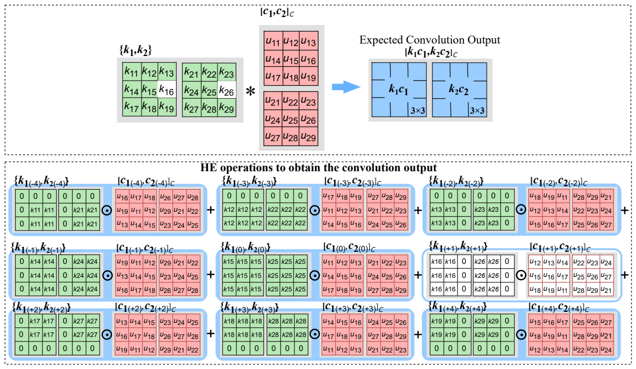
Homomorphic Encryption (HE) has been hailed as cryptography's holy grail due to its ability to achieve linear computation between ciphertexts without the need for decryption. This means that results can be directly exported in the encrypted form of the corresponding plaintext result. HE's crucial feature makes it a promising dominator in the field of privacy-preserving MLaaS. For instance, a client ($\mathcal{C}$) can encrypt its private data before sending it to the server ($\mathcal{S}$). The latter can then perform computations over the ciphertext using its DL model and obtain the result without gaining access to the client's secrets. Traditional HE algorithms encrypt each value of the plaintext individually, such as in the Paillier encryption scheme [34]. In contrast, packed HE encrypts the entire vector of plaintext, with multiple values, into one ciphertext. It then performs computations over that ciphertext in a Single Instruction Multiple Data (SIMD) manner [5]. As a result, packed HE achieves significantly higher efficiency and has been widely adopted in state-of-the-art MLaaS frameworks [50].

Different from the plaintext deep learning which applied by normal multiply-adds operations, in the packed HE-based scenario, all linear operations, convolution operations in convolutional layers and dot product operations in FC layers, are achieved through the combination of three basic HE linear operations: Homomorphic Addition (Add), Homomorphic Multiplication (Mult), and Homomorphic Permutation (Perm). Specifically, let's consider two plaintext vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$, which are respectively packed encrypted into $[\boldsymbol{v}_1]_C$ and $[\boldsymbol{v}_2]_C$. Here, we define $[\cdot]_C$ as the ciphertext encrypted by the C. The Add operation between ciphertext $[\boldsymbol{v}_1]_C$ and $[\boldsymbol{v}_2]_C$ can be regarded as the regular addition but in element-wise manner, which outputs the sum as ciphertext $[\boldsymbol{v}_1 + \boldsymbol{v}_2]_C$. The Mult operation between ciphertext $[\boldsymbol{v}_1]_C$ and plaintext $\boldsymbol{v}_2$ can be performed as element-wise multiplication which outputs the product as ciphertext $[\boldsymbol{v}_1 \odot \boldsymbol{v}_2]_C$. The Perm operation performs a cyclic rotation of the elements in one ciphertext. For example, given a ciphertext $[\boldsymbol{v}]_C$, rotating the ciphertext $[\boldsymbol{v}]_C$ in the i-th position moves all elements of $[\boldsymbol{v}]_C$ reversely by i-th position in a loop. This results in another ciphertext, denoted as $[\boldsymbol{v}_{(+i)}]_C$. Notice that Add and Mult operations are both performed in an element-wise manner. Therefore, it is not possible to directly sum up the values of a vector, since these elements are in the different slots. Thus, the Perm operation can solve this problem by rotating the ciphertext to align internal elements.

2.4 State-of-the-Art HE-based Convolution and HE-friendly Pruning Schemes

In this part, let's first introduce the HE-based convolution, which is achieved by the combination of three basic HE operations: Perm, Mult, and Add. We will then explore how the HE-friendly pruning scheme, named Hunter, efficiently prunes the model structures.

In Figure 3, we can see that the client C packs and encrypts its c_n (assuming $c_n = 2$ here) input channels $\boldsymbol{c_1}$ and $\boldsymbol{c_2}$ as $[\boldsymbol{c_1}, \boldsymbol{c_2}]_C$, where the size of each channel is $c_w \times c_h$. This ciphertext is then sent to the server $\mathcal{S}$ to convolve with the corresponding plaintext kernels $\boldsymbol{k_1}$ and $\boldsymbol{k_2}$, where the size of each kernel is $k_w \times k_h$. The encrypted output $[\boldsymbol{k_1 c_1}, \boldsymbol{k_2 c_2}]_C$ is obtained, where $\boldsymbol{k_1 c_1}$ and $\boldsymbol{k_2 c_2}$ can be seen as the outputs of two convolutions performed in parallel and in a SIMD fashion [5], and $*$ in the figure denotes the convolution operator. More specifically, the convolution operation is realized by first placing the kernel at each location of the input feature and then summing up all the element-wise products between the kernel values and the input values covered by the kernel window. However, as previously discussed, it is not possible for server $\mathcal{S}$ to directly sum up the element-wise products since the Add operation also works element-wise in the HE-base scenario. Add can only sum



$[\boldsymbol{c_1}, \boldsymbol{c_2}]_C$
$\{\boldsymbol{k_1}, \boldsymbol{k_2}\}$
Expected Convolution Output
$[\boldsymbol{k_1 c_1}, \boldsymbol{k_2 c_2}]_C$
$\boldsymbol{k_1 c_1}$
$\boldsymbol{k_2 c_2}$
3×3
HE operations to obtain the convolution output

Figure 3: HE-based Convolution Operation.

up the values at the same location (defined as a slot) of multiple ciphertexts but not the values at different slots within a given ciphertext. To solve this issue, the Perm operation is adopted to rotate the ciphertext.

To be specific, initially, $(k_w k_h - 1)$ Perm operations are performed over $[\boldsymbol{c_1}, \boldsymbol{c_2}]_C$ to obtain $k_w k_h$ rotated ciphertexts (including the original $[\boldsymbol{c_1}, \boldsymbol{c_2}]_C$). Among these rotated ciphertexts, $(k_w k_h - 1)/2$ are rotated in the forward (left) direction and $(k_w k_h - 1)/2$ are rotated in the backward (right) direction. This ensures that all the elements that need to be added up are relocated to the same slot of each ciphertext. For example, by 4 Perm operations in the forward direction[2], we obtain 4 rotated ciphertexts, as $[\boldsymbol{c_{1(+1)}}, \boldsymbol{c_{2(+1)}}]_C$, $[\boldsymbol{c_{1(+2)}}, \boldsymbol{c_{2(+2)}}]_C$, $[\boldsymbol{c_{1(+3)}}, \boldsymbol{c_{2(+3)}}]_C$ and $[\boldsymbol{c_{1(+4)}}, \boldsymbol{c_{2(+4)}}]_C$ shown in Figure 3. We obtain another 4 rotated ciphertexts: $[\boldsymbol{c_{1(-1)}}, \boldsymbol{c_{2(-1)}}]_C$, $[\boldsymbol{c_{1(-2)}}, \boldsymbol{c_{2(-2)}}]_C$, $[\boldsymbol{c_{1(-3)}}, \boldsymbol{c_{2(-3)}}]_C$ and $[\boldsymbol{c_{1(-4)}}, \boldsymbol{c_{2(-4)}}]_C$ by performing 4 Perm operations in the backward direction. Together with $[\boldsymbol{c_{1(0)}}, \boldsymbol{c_{2(0)}}]_C$ (i.e., the original $[\boldsymbol{c_1}, \boldsymbol{c_2}]_C$), we have 9 ciphertexts ready for the subsequent calculations.

Then Mult and Add operations are performed between the 9 ciphertexts and the corresponding transformed plaintext kernels (see $\{\boldsymbol{k_{1(-4)}}, \boldsymbol{k_{2(-4)}}\}$ to $\{\boldsymbol{k_{1(+4)}}, \boldsymbol{k_{2(+4)}}\}$, which are transformed from the original plaintext kernels) to finally calculate the convolution output. Notice that, the cyclic effect of Perm operation on elements in each rotated ciphertext makes the values in the corresponding transformed plaintext kernels associate with only "single" kernel value from the original kernel $\{\boldsymbol{k_1}, \boldsymbol{k_2}\}$ (since multiple kernel values at the same position behave like single value in SIMD manner). For example, the values in $\{\boldsymbol{k_{1(0)}}, \boldsymbol{k_{2(0)}}\}$ are only associate with k_{15}, k_{25} from $\{\boldsymbol{k_1}, \boldsymbol{k_2}\}$, and the values in $\{\boldsymbol{k_{1(+1)}}, \boldsymbol{k_{2(+1)}}\}$ are only associated with k_{16}, k_{26} from $\{\boldsymbol{k_1}, \boldsymbol{k_2}\}$, so on and so forth.

A major contribution of Hunter [7] is the interesting finding that each rotated ciphertext (that requires one Perm operation) is multiplied with a transformed plaintext kernel that includes only one kernel value. Thus, one Perm operation for obtaining one rotated ciphertext is no more needed and can be eliminated if the kernel value in that to-be-multiplied transformed kernel is zero. Based on this observation, one Perm operation is eliminated when one kernel value (excluding the central one) in $\{\boldsymbol{k_1}, \boldsymbol{k_2}\}$ is zero. Therefore, Hunter prunes the individual value (at the same position) in $\{\boldsymbol{k_1}, \boldsymbol{k_2}\}$ such that there is no need to rotate the corresponding

[2]In this paper, we use the positive and negative symbols "+" and "−" to denote the forward and backward directions, respectively. The same convention is applied to the subscript of the transformed kernel we will describe later.

ciphertext by one expensive Perm operation [7]. For example, as shown in Figure 3, one Perm operations for obtaining the rotated ciphertext $[c_{1(+1)}, c_{2(+1)}]_C$ is eliminated if the kernel values k_{16}, k_{26} are pruned. Then Hunter defines the *Internal Structure* as the kernel values at the same location of all the plaintext kernels that are convolved with the same input ciphertext, and pruning an Internal Structure correspondingly eliminates one Perm operation (and subsequent Mult and Add operations) over the input ciphertext.

Most state-of-the-art DL models, such as VGG and ResNet, utilize multiple filters and kernels (also known as multiple input and multiple output channels, MIMO) within each convolutional layer. Hunter also identifies another HE Pruning Structures, i.e., *External Structure* for pruning in MIMO case, following the same rule that the HE linear operation can be eliminated if all the elements in corresponding HE Pruning Structure are pruned. Thus, by pruning these HE Pruning Structures, Hunter aims to minimize the number of Perm and the subsequent Mult and Add operations, thereby reducing the overall computational cost.

Hunter sheds a light on HE-friendly structured pruning. However, in practical scenarios, the effectiveness of directly pruning the HE Pruning Structure is hindered by the problem of pruning structure inflation, which arises due to the shrinking size of the input feature map. Removing the inflating pruning structure causes a significant accuracy loss because it involves many important model parameters and damages essential model structures. To preserve the model accuracy, pruning ratio will be limited. This presents a key challenge that must be addressed.

3 PROPOSED PRUNE-AND-ASSEMBLE APPROACH, MOSAIC

We introduce MOSAIC, a flexible HE-friendly model pruning strategy to tackle the challenge discussed in the previous section. The MOSAIC approach involves two main steps. First, it prunes the convolutional layer using tessellated basic units. Next, it assembles the pruned units into HE Pruning Structures via a channel transformation. This way, the computationally expensive HE operations associated with these Pruning Structures can be eliminated. Moreover, we introduce the channel transformation coordination strategy to ensure that these channel transformations serve two important purposes. Firstly, they effectively assemble numerous pruned units into HE Pruning Structures, and secondly, they do not interfere with the overall model functions, preserving the model's accuracy and functionality. Our primary emphasis is on optimizing Convolutional layers since they contribute significantly to the overall cost, accounting for up to 99% of the linear cost.

The following subsections will elaborate on our adaptive pruning strategies tailored for three distinct MIMO schemes: (1) Ungrouped Output Rotation (Out-Rot) MIMO, (2) Input Rotation (In-Rot) MIMO, and (3) Grouped Out-Rot MIMO. Notably, the first two schemes find application within the frameworks like GAZELLE [22] and DELPHI [29], while the latter two are employed within the frameworks such as CrypTFlow2 [38] and GALA [53]. Subsequently, we will introduce the inter-layer channel transformation coordination, followed by a comprehensive security analysis.

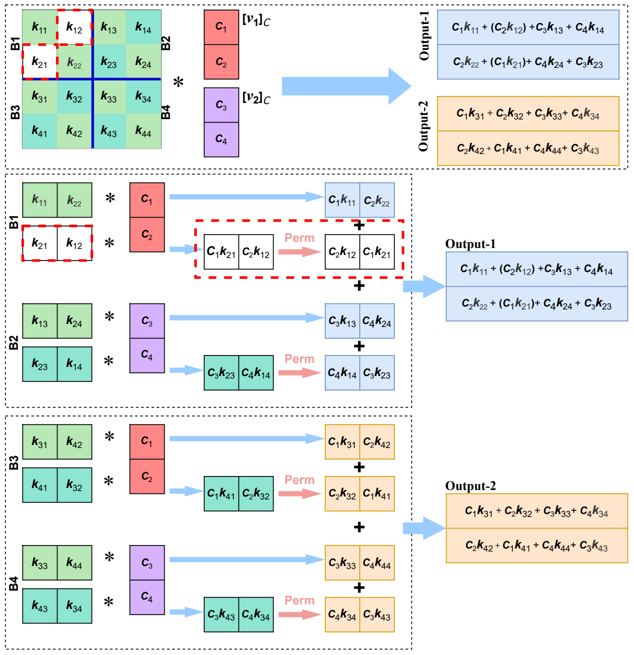
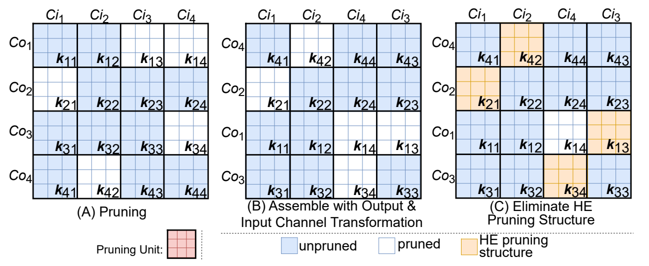

B1
B2
B3
B4
$[v_1]_C$
$[v_2]_C$
Output-1
Output-2
Perm
$c_1k_{11} + (c_2k_{12}) + c_3k_{13} + c_4k_{14}$
$c_2k_{22} + (c_1k_{21}) + c_4k_{24} + c_3k_{23}$
$c_1k_{31} + c_2k_{32} + c_3k_{33} + c_4k_{34}$
$c_2k_{42} + c_1k_{41} + c_4k_{44} + c_3k_{43}$

Figure 4: Ungrouped Out-Rot MIMO ($c_n = 2$).

3.1 MOSAIC's Pruning for Ungrouped Output Rotation MIMO Scheme

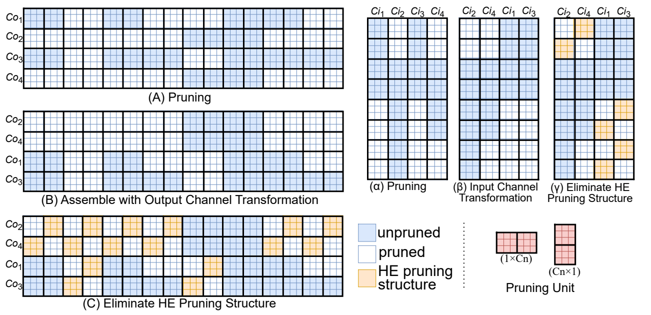
We begin with the Ungrouped Out-Rot MIMO scheme, which is the default scheme of GAZELLE frameworks [22] and current HE-friendly pruning method Hunter [7]. Figure 4 illustrates an example when $c_n = 2$. Four input channels to a convolution layer, c_1, c_2, c_3, c_4, are packed and encrypted into two ciphertexts $[v_1]_C$ and $[v_2]_C$, and then convolved with four filters, one at a row, each of which has four kernels (corresponding to the four input channels). Finally, the output is two encrypted ciphertexts, i.e., Output-1 and Output-2. Each output ciphertext includes two output channels. More specifically, to obtain the first output ciphertext Output-1 in the HE computation, we first convolve input ciphertext $[v_1]_C$ with the main-diagonal kernels group $\{k_{11}, k_{22}\}$ in the SIMD manner described earlier, which gives the ciphertext $[c_1k_{11}, c_2k_{22}]_C$ named as *intermediate ciphertext*. Then, input ciphertext $[v_1]_C$ is convolved with the kernels group $\{k_{21}, k_{12}\}$ to produce the ciphertext $[c_1k_{21}, c_2k_{12}]_C$. We cannot directly add these two convolved ciphertexts, since the first elements of $[c_1k_{11}, c_2k_{22}]_C$ and $[c_1k_{21}, c_2k_{12}]_C$ are actually partial results of output channel 1 and 2, respectively, and hence adding them does not make sense. To overcome this problem, a Perm operation is conducted to rotate ciphertext $[c_1k_{21}, c_2k_{12}]_C$ into $[c_2k_{12}, c_1k_{21}]_C$ which is regarded as an intermediate ciphertext and can then be added to $[c_1k_{11}, c_2k_{22}]_C$. Similarly, we get two more intermediate ciphertexts $[c_3k_{13}, c_4k_{24}]_C$ and $[c_4k_{14}, c_3k_{23}]_C$. Finally, the first output ciphertext, Output-1, is produced by summing up all four intermediate ciphertexts using the Add operation. The second output ciphertext, Output-2, can be obtained similarly.

As we can see, certain kernel groups involved in the convolution require the expensive Perm operation to rotate the resulting convolved ciphertext into the intermediate ciphertext. If these kernel



(A) Pruning
(B) Assemble with Output & Input Channel Transformation
(C) Eliminate HE Pruning Structure
Pruning Unit:
unpruned
pruned
HE pruning structure

Figure 5: MOSAIC's Pruning for Ungrouped Out-Rot MIMO ($c_n = 2$).



(A) Pruning
(B) Assemble with Output Channel Transformation
(C) Eliminate HE Pruning Structure
(α) Pruning
(β) Input Channel Transformation
(γ) Eliminate HE Pruning Structure
unpruned
pruned
HE pruning structure
(1×Cn)
(Cn×1)
Pruning Unit

Figure 6: MOSAIC's Pruning for Ungrouped Out-Rot MIMO, ResBlock ($c_n = 2$).

groups are all zeros, there is no need to rotate the convolved ciphertext to resequence associated filters. For example, the Perm operation over the convolved ciphertext $[\boldsymbol{c}_1\boldsymbol{k}_{21}, \boldsymbol{c}_2\boldsymbol{k}_{12}]_C$ can be eliminated if the involved kernels group $\{\boldsymbol{k}_{21}, \boldsymbol{k}_{12}\}$ are all zeros since the value of the corresponding intermediate ciphertext $[\boldsymbol{c}_2\boldsymbol{k}_{12}, \boldsymbol{c}_1\boldsymbol{k}_{21}]_C$ must be all zero. In fact, Hunter defines the *External Structure* based on these kernel groups, which *requires the expensive Perm operation to rotate the resulting convolved ciphertext*. In summary, pruning an External Structure can eliminate one Perm (and subsequent Mult and Add operations) over a convolved ciphertext [7]. But as discussed in Section 1, in practical scenarios, the opportunity of pruning the HE Pruning Structure, like the External Structure, is hindered by the problem of pruning structure inflation. Removing the inflated pruning structure can cause significant accuracy loss because it damages essential model structures.

The key to address the pruning structure inflation is a breakthrough scheme that can apply fine-granularity pruning even in the deep layers with the pruning structure inflation problem. To achieve this objective, MOSAIC smartly divides a convolutional layer into tessellated tiny pruning units. During pruning, the weights within the same unit are removed synchronously. Different units are pruned individually and separately. As a result, MOSAIC accomplishes pruning with a fine granularity that closely resembles unstructured pruning, thereby mitigating the impact on the model's accuracy. The selection of the basic pruning units plays a critical role, and generally two factors need to be taken into account. On the one hand, they should be small enough to minimize the damage to the model performance caused by pruning these units. On the other hand, the units should fit well with the subsequent assembly to form more HE Pruning Structures, in order to achieve maximal cost reduction.

For example, for Ungrouped Out-Rot MIMO scheme, the individual kernels are a good choice for the basic pruning unit given those two factors. As illustrated in Figure 5, the original layer is divided into 16 tessellated basic pruning units from $\boldsymbol{k}_{11}$ to $\boldsymbol{k}_{44}$. These units are independent of each other in pruning. The strategy of MOSAIC is to remove the unessential basic pruning units[3], resulting in the pruned layer (A).

After removing the unessential basic pruning units, MOSAIC applies the output and input channel transformations to reorder the output and input channels, which correspond to interchange of the rows and columns of the kernel matrix, respectively. The final channel transformation result is illustrated in Figure 5(B), where the row and column indices indicate the original row and column number, respectively. For example, the indices of the 3rd and 4th columns are Ci_4 and Ci_3, respectively, which indicate they were obtained by interchanging the 4th and 3rd columns of the kernel matrix in Figure 5(A). The objective of channel transformation, i.e., reordering rows and columns of the kernel matrix, is to assemble pruned units into as many HE Pruning Structures as possible, making it possible to prune those structures to eliminate the Perm and subsequent HE Mult/Add operations.

A local optimal search algorithm based on the greedy search method or a global optimal search algorithm based on the exhaustion method can be used to obtain the optimal kernel matrix row/column reordering. To conduct the greedy search, we could first select the row with the largest number of pruned units. We then search the remaining rows to pick up another $c_n - 1$ rows that can assemble the largest number of HE Pruning Structures with that selected row. We fix these c_n rows and then search for another c_n rows from the remaining rows of the kernel matrix. This process is repeated until all rows have been fixed, resulting in a sequence of rows that represent the local optimal solution. The same approach is used to obtain the sequence of the kernel matrix's columns. The global optimal search method involves exhaustively testing all possible sequences of rows or columns of the kernel matrix and selecting the one with the largest number of HE Pruning Structures. However, this approach requires massive computing power. In this work, we adopt the local optimal search algorithm for MOSAIC, to carry out channel transformation. Figure 5(C) illustrates the result of channel transformation on the kernel matrix that form two External Structures, which leads to the elimination of two Perm operations and subsequent HE operations. At last, to gain more insights into MOSAIC's pruning approach, the visualization of the prune-and-assemble process we discussed above is also shown in Figure 15 by applying MOSAIC onto a large kernel matrix.

A special case of MOSAIC's pruning on the Ungrouped Out-Rot MIMO is for the DL model built by ResBlocks, which can be found in the ResNet series and many other networks. In the ResBlock, the first layer only undergoes output channel transformation while the last layer only undergoes input channel transformation, according to our inter-layer transformation coordination (to be discussed in Subsection 3.4). To this end, the first layer of ResBlocks is divided

[3]There are many approaches to determining the importance of model structure, ranging from the most straightforward approach that compares the absolute value of the weights with a given threshold [16] to advanced approach focused on the effect of certain structure on the model objective function [8]. In this work, we do not intend to reinvent the pruning criteria. The proposed MOSAIC can adopt any existing pruning criteria. Our experimental results are based on a pruning criteria: HSPG [8].

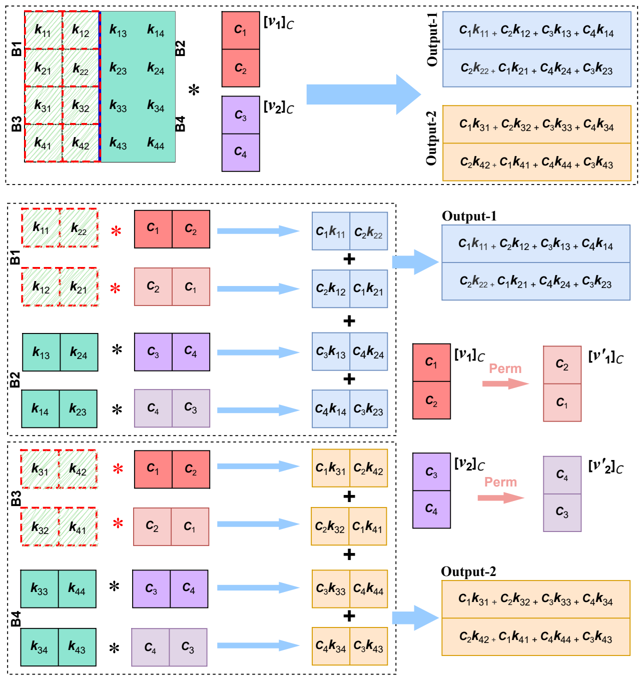

B1
B2
B3
B4
$[v_1]_C$
$[v_2]_C$
$[v'_1]_C$
$[v'_2]_C$
Perm
Output-1
Output-2
$c_1k_{11}+c_2k_{12}+c_3k_{13}+c_4k_{14}$
$c_2k_{22}+c_1k_{21}+c_4k_{24}+c_3k_{23}$
$c_1k_{31}+c_2k_{32}+c_3k_{33}+c_4k_{34}$
$c_2k_{42}+c_1k_{41}+c_4k_{44}+c_3k_{43}$

Figure 7: In-Rot MIMO ($c_n = 2$).

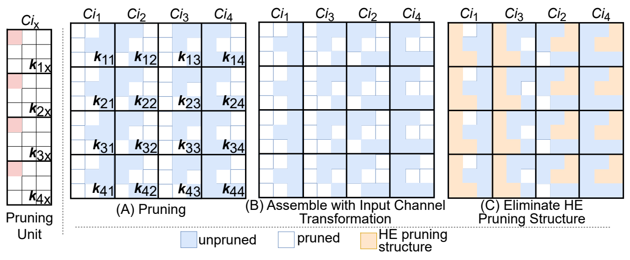

Ci_x
k_{1x}
k_{2x}
k_{3x}
k_{4x}
Pruning Unit
Ci_1 Ci_2 Ci_3 Ci_4
Ci_1 Ci_3 Ci_2 Ci_4
(A) Pruning
(B) Assemble with Input Channel Transformation
(C) Eliminate HE Pruning Structure
unpruned
pruned
HE pruning structure

Figure 8: MOSAIC's Pruning for In-Rot MIMO ($c_n = 2$).

using a $1 \times c_n$ kernel set as the pruning unit, and the last layer is divided using a $c_n \times 1$ kernel set. These pruning units are more efficient for assembly than individual kernels for ResBlocks. Figure 6(A) shows that the first layer of a ResBlock is divided into 32 tessellated $1 \times c_n$ basic pruning units and some unessential units are pruned. In Figure 6(B), the output channel transformation is conducted to assemble the pruned units, resulting in the elimination of eight External Structures and corresponding Perm shown in (C). The lack of input channel transformation in the first layer prevents adjustment of the sequence of the kernel matrix's columns. To ensure the formation of more External Structures, each of which involves a $c_n \times c_n$ area of the kernel matrix. This basic unit already covers the kernels from adjacent c_n columns, allowing for many External Structures to be formed even without column transformation. The same reasoning applies to the selection of the $c_n \times 1$ basic units for the last layer of ResBlock. The prune-and-assemble processes are illustrated in (α), (β) and (γ).

3.2 MOSAIC's Pruning for Input Rotation MIMO Scheme

The input rotation MIMO scheme, *In-Rot*, rotates the input of convolution. Figure 7 demonstrates the In-Rot MIMO scheme, where the input of the convolution, ciphertext $[\boldsymbol{v}_1]_C$, is rotated into $[\boldsymbol{v}'_1]_C$ and then convolved with the kernels group $\{\boldsymbol{k}_{12}, \boldsymbol{k}_{21}\}$ to obtain the intermediate ciphertext $[\boldsymbol{c}_2\boldsymbol{k}_{12}, \boldsymbol{c}_1\boldsymbol{k}_{21}]_C$ directly. Similarly, we can obtain three more intermediate ciphertexts: $[\boldsymbol{c}_1\boldsymbol{k}_{11}, \boldsymbol{c}_2\boldsymbol{k}_{22}]_C$, $[\boldsymbol{c}_3\boldsymbol{k}_{13}, \boldsymbol{c}_4\boldsymbol{k}_{24}]_C$, and $[\boldsymbol{c}_4\boldsymbol{k}_{14}, \boldsymbol{c}_3\boldsymbol{k}_{23}]_C$. Finally, the first output ciphertext, Output-1, is produced by summing up all four intermediate ciphertexts using the Add operation. The second output ciphertext, Output-2, can be obtained similarly.

Actually, in Figure 7, each $*$ symbol represents the convolution operator associated with one HE-based convolution, as discussed in Figure 3. Combining Figure 7 and Figure 3, we could obtain some observations. The In-Rot MIMO scheme converts Perm operation associated with External Structure (denoted as **ex-Perm**) to the Perm operation associated with Internal Structure (denoted as **in-Perm**). It eliminates the cost of ex-Perm at the expense of increasing in-Perm. Obviously, one Perm conversion is involved with converting input ciphertext $[\boldsymbol{v}_1]_C$ into $[\boldsymbol{v}'_1]_C$, and then additional $(k_w k_h - 1)$ in-Perm operations performed over $[\boldsymbol{v}'_1]_C$ to obtain the rotated ciphertext for the convolution between $[\boldsymbol{v}'_1]_C$ and the kernels group $\{\boldsymbol{k}_{12}, \boldsymbol{k}_{21}\}$ (and all other kernels group to be convolved with $[\boldsymbol{v}'_1]_C$).

For the In-Rot MIMO scheme, a basic pruning unit of MOSAIC consists of the weights located at the same position of all the kernels associated with the same input channel. For example, as illustrated in Figure 8, the four weights located at the top-left corner of each of the four kernels in the same column, which are associated with the same input channel, form one basic pruning unit.

In Figure 8, the original layer is divided into 36 tessellated basic pruning units, with 9 units for each column of the kernel matrix. The pruning process then removes 4, 4, 5 and 3 units for each column, resulting in the pruned layer in Figure 8(A). Next, the input channel transformation is performed to reorder the input channels, which corresponds to the column transformation of the kernel matrix, as shown in Figure 8(B). Thus, the pruned units are assembled into 7 Internal Structures, with 4 related to the first input ciphertext (corresponding to the 1st and 2nd columns of the kernel matrix) and 3 related to the second input ciphertext (corresponding to the 3rd and 4th columns of the kernel matrix). This assembly results in a reduction of 13 in-Perms[4], shown in Figure 8(C).

3.3 MOSAIC's Pruning for Grouped Output Rotation MIMO Scheme

Figure 9 illustrates the Grouped Out-Rot MIMO scheme when $c_n = 2$. Most of the operations are the same as the Ungrouped Out-Rot MIMO introduced before, except that the Perm operation is not applied to each of convolved ciphertext immediately. Instead, all convolved ciphertexts that require the same Perm operation are added together first and then one Perm is performed. For instance, in Figure 9, the input ciphertext $[\boldsymbol{v}_1]_C$ is convolved with the kernels group $\{\boldsymbol{k}_{21}, \boldsymbol{k}_{12}\}$ to produce the ciphertext $[\boldsymbol{c}_1\boldsymbol{k}_{21}, \boldsymbol{c}_2\boldsymbol{k}_{12}]_C$. Next, add this convolved ciphertext with another convolved ciphertext $[\boldsymbol{c}_3\boldsymbol{k}_{23}, \boldsymbol{c}_4\boldsymbol{k}_{14}]_C$ via Add operation. The resultant sum is rotated

[4]Internal Structure with weights at the center position of the kernel is involved with $(c_n - 1)$ in-Perm, while the other Internal Structures are involved with c_n in-Perm. So, when $c_n = 2$, the number of in-Perm eliminated is: $4 \times 2 + 1 \times (2 - 1) + 2 \times 2 = 13$.

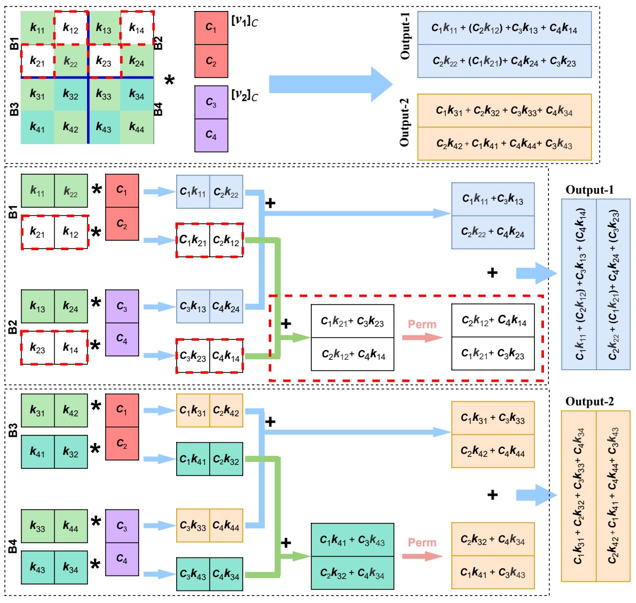


Figure 9: Grouped Out-Rot MIMO ($c_n = 2$).

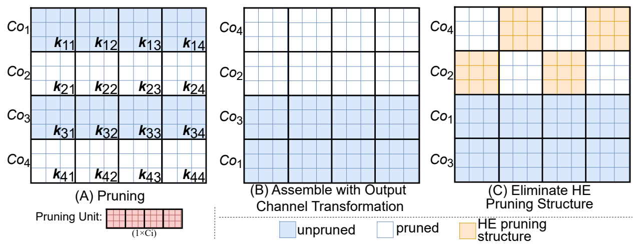

(A) Pruning
Pruning Unit:
(B) Assemble with Output Channel Transformation
(C) Eliminate HE Pruning Structure
unpruned
pruned
HE pruning structure

Figure 10: MOSAIC's Pruning for Grouped Out-Rot MIMO ($c_n = 2$).

into the intermediate ciphertexts $[c_2\boldsymbol{k}_{12}+c_4\boldsymbol{k}_{14}, c_1\boldsymbol{k}_{21}+c_3\boldsymbol{k}_{23}]_C$ by a Perm at last. Added to another intermediate ciphertexts $[c_1\boldsymbol{k}_{11} + c_3\boldsymbol{k}_{13}, c_2\boldsymbol{k}_{22} + c_4\boldsymbol{k}_{24}]_C$, the first output ciphertext Output-1 is obtained. The second output ciphertext, Output-2, can be calculated in a similar manner.

As we can observe, certain kernel groups involved in the calculation require the expensive Perm operation. For example, during the process to calculate output ciphertext Output-1, the kernels group $\{\boldsymbol{k}_{21}, \boldsymbol{k}_{12}\}$ and $\{\boldsymbol{k}_{23}, \boldsymbol{k}_{14}\}$, marked by the red dashed boxes, are involved with Perm operation. Thus, if these kernel groups are all zeros, there is no need to perform the Perm since the final result of the following calculations must also be zero. In this way, we could eliminate one Perm operation by pruning these involved kernel groups, treating them as an HE Pruning Structure. However, as previously discussed, the challenge of pruning structure inflation impedes the removal of the inflating HE Pruning Structure, as such removal can lead to substantial accuracy loss due to the disruption of essential model structures. Next, we refine our adaptable prune-and-assemble framework, MOSAIC, to effectively work on the Grouped Out-Rot MIMO scheme, while avoiding the pruning structure inflation problem.

For the Grouped Out-Rot MIMO scheme, the basic pruning unit of MOSAIC is the entire filter, which corresponds to a row in the kernel matrix. As shown in Figure 10, the original layer is divided into 4 tessellated basic pruning units. Then pruning is conducted to remove unessential units, e.g., the two rows of the kernel matrix in Figure 10(A). Then, the output channel transformation is performed to reorder the output channels, which correspond to the rows of the kernel matrix, as shown in Figure 10(B). Lastly, in Figure 10(C), the assembled pruned units constitute an HE Pruning Structure. This eliminates one Perm operation and subsequent HE Mult/Add operations.

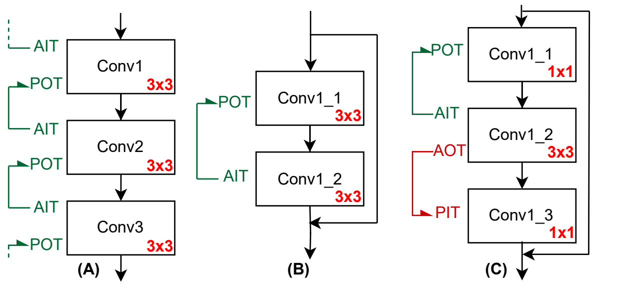


Figure 11: MOSAIC's Channel Transformation Coordination Strategies: (A) VGG-like model. (B) 2-layer ResBlock. (C) 3-layer bottleneck block.

3.4 MOSAIC's Channel Transformation Coordination

So far, we have discussed MOSAIC's prune-and-assemble approach for Ungrouped Out-Rot MIMO scheme, In-Rot MIMO scheme and Grouped Out-Rot MIMO scheme. As we know, a DL model is a stack of layers. The output channels of a layer are the input channels of the following layer [26, 45]. The channel transformation is used to reorder the input or the output channel for assembling the basic pruning units into HE Pruning Structures. It is crucial to ensure that the channel transformation conforms to the channel logic between the layers to maintain the function of the model. When an output channel transformation is made on a layer, the order of the layer's output channels changes. Therefore, a corresponding input channel transformation must be performed on the next layer to align its input channels. Similarly, when an input channel transformation is made on a layer, it must lead to another output channel transformation on the previous layer. The active and passive channel transformations must be applied in pair.

Next, we discuss how MOSAIC coordinates the channel transformations between the neighboring layers of a DL model. We propose three inter-layer channel transformation coordination schemes specifically designed for (A) VGG-like model, (B) 2-layer ResBlock, and (C) 3-layer ResBlock. These three coordination schemes cover the most popular CNN models and offer comprehensive and efficient pruning, while maintaining the functionality of the model.

Channel transformation coordination for VGG-like model: As shown in Figure 11 (A), for VGG-like model with sequentially stacked convolutional layers, we perform the active input channel transformation (AIT) on each layer, followed by the corresponding passive output channel transformation (POT) on the previous layer. The AIT is performed with the objective to maximize the number of HE pruning structures to be obtained. On the other hand, the POT is performed with a simple objective that is to align the order of the (output) channels to match the desired order of the immediately following AIT. Note that the output channels of a POT become the input channels of the following AIT.

Channel transformation coordination for 2-layer ResBlock: Some DL models are built with ResBlocks, which have "shortcuts" to connect the head and tail of each ResBlock. While the "shortcut" has many benefits for the performance of deep networks, it presents a great challenge in channel order alignment, or coordination of channel transformation. Assume that there is one node between every two adjacent ResBlocks, and the shortcuts connect all nodes together. Thus, all nodes share the same channel orders. Moreover, any output channel transformation on the last layer of a ResBlock or input channel transformation on the first layer of a ResBlock will affect the channel order of all these nodes. Consequently, we are compelled to align the first and last layers of each ResBlock to match the same channel order, even though this is actually what we aim to evade.

The MOSAIC channel transformation coordination scheme for ResBlock is devised with those constraints. Specifically, in DL models with 2-layer ResBlocks, all channel transformations are performed inside the ResBlock. There is no cross-ResBlocks channel transformation. As shown in Figure 11(B), only AIT is conducted on the last layer, while the corresponding POT is performed on the first layer of the ResBlock.

Channel transformation coordination for 3-layer ResBlocks: The 3-layer ResBlock is typically found in some large-scale networks such as ResNet-50, ResNet-101 and ResNet-152. This ResBlock consists of a stack of three layers: 1×1, 3×3, and 1×1 convolutional layers. The 3×3 layer serves as the feature extractor, while the 1×1 layers are responsible for expanding and compressing dimensions [18]. We make slight adjustments to the coordination scheme for 2-layer ResBlocks to make it suitable for 3-layer ResBlocks. Given the dominant contribution of the 3×3 layer to the model functionality and calculation density, we perform both the AIT and active output channel transformation (AOT) on the 3×3 layer, as shown in Figure 11(C). We then conduct the corresponding POT and the passive input channel transformation (PIT) on the first and last 1×1 layers, respectively.

By integrating these transformation coordination schemes with the pruning scheme for the three MIMO schemes: (1) Ungrouped Out-Rot MIMO, (2) In-Rot MIMO, and (3) Grouped Out-Rot MIMO, as detailed earlier, we can achieve remarkable cost reduction for DL models in privacy-preserving MLaaS. For instance, the cost is reduced by 91.27% for ResNet-50 on the Tiny ImageNet dataset under the GAZELLE framework on which the first two MIMO schemes are applied. We also achieve a 58.59% cost reduction under CrypTFlow2 which employs the latter two MIMO schemes. While this pruning ratio might be lower compared to the one for GAZELLE, it is important to recognize that CrypTFlow2 is already an extensively optimized framework, resulting in a significantly lower baseline cost. Hence, this achieved pruning ratio remains noteworthy, given its ability to further compress an already highly optimized computation cost.

Note that, the selection of MIMO schemes usually varies from layer to layer in a DL model, to achieve optimal performance. In Section A of the Appendix, we conduct a comprehensive analysis of the intricacies and complexities associated with Ungrouped Out-Rot, In-Rot MIMO and Grouped Out-Rot schemes. This analysis can help to determine for a particular layer, which of these schemes can benefit most, considering factors like complexity and actual computational cost.

3.5 Security Analysis

Similar to GAZELLE [22] and CrypTFlow2 [38], the security of MOSAIC relies on the semantic security of packed homomorphic encryption algorithms, such as BFV [11] and CKKS [9], for linear computations, including convolution. Specifically, MOSAIC prunes the original model by setting some weights to zero and performs channel transformation by reordering the input and output channels of the convolutional layer. The pruned models are then used by the server for the MLaaS service. For linear computations, MOSAIC uses a more efficient homomorphic encryption-based computation method with lower complexity during the MLaaS process, compared to other frameworks like GAZELLE. Therefore, MOSAIC does not introduce any extra computational modules. Moreover, for nonlinear computations, MOSAIC follows the same paradigm as GAZELLE or CrypTFlow2, which is based on Garbled Circuits (GC) [2] or Oblivious Transfer (OT) [6] for ReLU computation. Overall, under the semi-honest assumption, MOSAIC is considered secure.

4 PERFORMANCE EVALUATION

In this section, We utilize two state-of-the-art MLaaS frameworks, GAZELLE [22] and CrypTFlow2 [38] as example, which encompass all three MIMO schemes discussed earlier, to evaluate the performance of MOSAIC, and compare it with Hunter [7]. Another notable privacy-preserving MLaaS framework is Cheetah [19]. However, Cheetah is highly sensitive and dependent on delicate adjustment of encryption parameters [19]. Furthermore, concerns have been raised by certain research studies about its communication overhead [14, 17]. As of now, GAZELLE and CrypTFlow2 remain the widely accepted frameworks for MLaaS. We train and prune the DL model using MOSAIC's flexible HE-friendly pruning strategy with *Pytorch* [35] on a workstation with NVIDIA A6000 GPU and AMD Ryzen3995 CPU. The experiments are conducted on five popular DL models: VGG-11, VGG-13, VGG-16, ResNet-34, and ResNet-50, with two mainstream datasets: CIFAR-10 [23] and Tiny ImageNet [46]. We then deploy MOSAIC's pruned model with the Homomorphic Encryption (HE) library *SEAL-Python* [20, 43] on a machine with Intel Core i5-10400 CPU to evaluate the actual computational cost under CKKS [9]. The performance evaluation focuses on the linear computation cost of the convolutional layer. The nonlinear cost and communication cost are the same as the baseline privacy-preserving framework.

We train and prune all models from scratch without any other improvement methods, such as Transfer Learning or Knowledge Distillation. For all evaluations, we follow the Homomorphic Encryption Standard [1]. To ensure security and efficiency, the slots of ciphertext are set to 4096 for the first layer and 8192 for the subsequent layers of all models [1, 32, 39]. This results in the layer-wise c_n breakdown shown in Table 1. As we can observe, c_n inflates dramatically as the model goes deeper.

Table 1: Layer-wise c_n breakdown of eight models

Method		Layer-wise c_n breakdown
CIFAR	VGG16	$\mathbf{4}_{\times 1} \rightarrow \mathbf{8}_{\times 3} \rightarrow \mathbf{32}_{\times 6} \rightarrow \mathbf{128}_{\times 3}$
	VGG13	$\mathbf{4}_{\times 1} \rightarrow \mathbf{8}_{\times 3} \rightarrow \mathbf{32}_{\times 4} \rightarrow \mathbf{128}_{\times 2}$
	VGG11	$\mathbf{4}_{\times 1} \rightarrow \mathbf{8}_{\times 1} \rightarrow \mathbf{32}_{\times 4} \rightarrow \mathbf{128}_{\times 2}$
Tiny-IN	VGG16	$\mathbf{1}_{\times 1} \rightarrow \mathbf{2}_{\times 3} \rightarrow \mathbf{8}_{\times 6} \rightarrow \mathbf{32}_{\times 3}$
	VGG13	$\mathbf{1}_{\times 1} \rightarrow \mathbf{2}_{\times 3} \rightarrow \mathbf{8}_{\times 4} \rightarrow \mathbf{32}_{\times 2}$
	VGG11	$\mathbf{1}_{\times 1} \rightarrow \mathbf{2}_{\times 1} \rightarrow \mathbf{8}_{\times 4} \rightarrow \mathbf{32}_{\times 2}$
	ResNet34	$\mathbf{1}_{\times 1} \rightarrow \mathbf{2}_{\times 6} \rightarrow \mathbf{8}_{\times 8} \rightarrow \mathbf{32}_{\times 12} \rightarrow \mathbf{128}_{\times 6}$
	ResNet50	$\mathbf{1}_{\times 1} \rightarrow \mathbf{2}_{\times 9} \rightarrow \mathbf{8}_{\times 12} \rightarrow \mathbf{32}_{\times 18} \rightarrow \mathbf{128}_{\times 9}$

Table 2: MOSAIC v.s. Hunter.

Method	Baseline Cost	MOSAIC/Hunter Cost	Accuracy Baseline/MOSAIC/Hunter
VGG16-cifar	906.87s	**168.93s**/478.05s**(35.3%)**	94.50%/94.16%/94.19%
VGG13-cifar	583.40s	**153.56s**/347.11s**(44.2%)**	94.09%/94.01%/94.46%
VGG11-cifar	536.57s	**130.02s**/178.80s**(72.7%)**	93.80%/93.56%/93.35%
VGG16-tiny	2512.62s	**531.09s**/920.46s**(57.7%)**	65.00%/64.87%/64.68%
VGG13-tiny	1615.05s	**409.01s**/746.40s**(54.8%)**	64.36%/64.33%/64.70%
VGG11-tiny	1496.16s	**400.12s**/663.12s**(60.3%)**	63.00%/63.88%/63.75%

*The comparison in this table is conducted within the GAZELLE framework, which is the Hunter was initially designed for.

MOSAIC's flexible pruning strategy v.s. Hunter: Table 2 provides a comprehensive comparison between MOSAIC and Hunter. This comparison illustrates, while aiming to maintain accuracy, directly removing the HE Pruning Structure doesn't yield a desirable pruning ratio for models with inflating c_n. This is due to the coarse granularity of the pruning. In contrast, MOSAIC's highly flexible "Prune-and-Assemble" scheme proficiently reduces computational cost. In the third column of the table, the value "168.93s/478.05s (35.3%)" represents the practical time cost of MOSAIC's pruned model, which is 168.93s, in comparison to Hunter's time cost of 478.05s. Therefore, MOSAIC's cost stands at only 35.3% of Hunter's.

MOSAIC's performance compared with baseline: Table 3 and Table 4 show that MOSAIC can efficiently prune the original model without sacrificing accuracy, reducing the actual cost of Perm, Mult, and Add operations within GAZELLE framework and CrypTFlow2 framework. For example, in the state-of-the-art ResNet-50 on Tiny ImageNet within GAZELLE (Table 3), MOSAIC reduces the actual cost of Perm by 92.42% and the overall cost by 91.27% while maintaining accuracy. Within CrypTFlow2, which is a highly customized HE framework, the attainable pruning ratio might be somewhat lower compared to the GAZELLE framework due to lack of redundancy and smaller pruning space. Nonetheless, benefiting from its lower baseline cost, the final cost of the pruned model through MOSAIC remains competitive with that of the GAZELLE framework. The effectiveness of MOSAIC lies in its flexible and accurate pruning approach that can overcome the pruning ratio limitation caused by pruning structure inflation. Overall, MOSAIC performs well in all eleven experiments under two most widely accepted state-of-the-art HE frameworks, efficiently reducing the computational cost of the model without any accuracy drop.

Layer-wise performance breakdown: We further analyze the pruning performance by breaking down the entire model into a series of layers. In Figure 12 and 13, we provide the layer-wise breakdown of VGG-16 on the CIFAR-10 and Tiny ImageNet dataset,

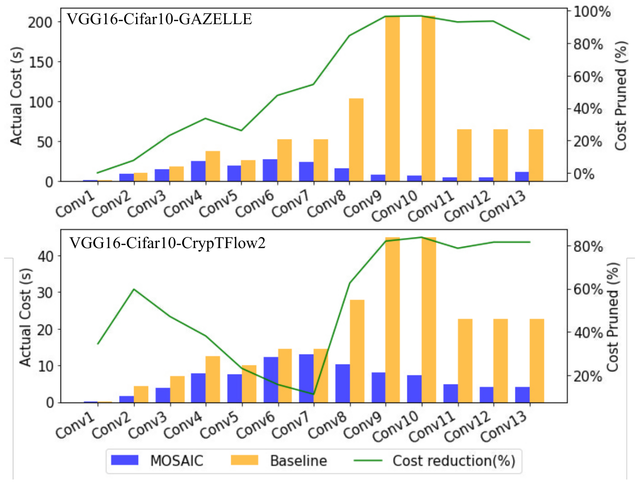


Figure 12: VGG16 CIFAR-10 Layer-Wise Breakdown.

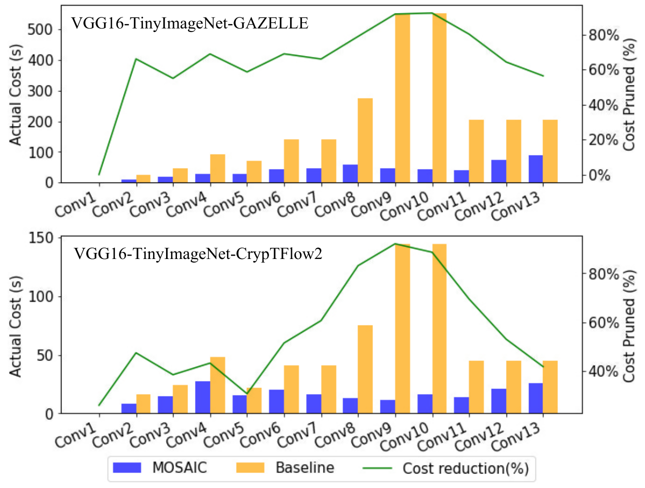


Figure 13: VGG16 TinyImageNet Layer-Wise Breakdown.

respectively. Additionally, Figure 14 illustrates the layer-wise breakdown of ResNet-50 on the Tiny ImageNet dataset. The breakdown yields several noteworthy observations: **Firstly**, unlike the scenario in plaintext, where the computational cost is relatively evenly distributed across layers, the privacy-preserving scenario (even with the same model and dataset) exhibits a tendency for the model's cost to concentrate around specific layers. For instance, in Figure 12 and 13, this concentration is evident around Conv9 and Conv10. The figures also demonstrate that MOSAIC strategically targets these layers to achieve efficient model cost compression. **Secondly**, the pruning ratio (indicated by the green line) exhibits distinct trends between different HE frameworks due to the variations in the adopted MIMO scheme. The pruning ratio line for GAZELLE is more consistent and smooth, while the CrypTflow2 line displays more fluctuations, and at times, even multiple spikes, shown in Figure 14. Generally speaking, this indicates that the process of pruning within the former has a larger space. These observations hold true in our other experiments as well. The interested reader is referred to Section B in the Appendix for more details.

Table 3: MOSAIC's computation performance compared with baseline on five popular DL models with two mainstream datasets within the GAZELLE framework.

Dataset	CIFAR10			Tiny ImageNet				
Models	VGG11	VGG13	VGG16	VGG11	VGG13	VGG16	ResNet34	ResNet50
Parameters	13M	13M	19M	161M	161M	166M	24M	38M
Computation Cost in Convolution								
Perm(s)	110.56/458.49 75.89%↓	131.91/495.59 73.38%↓	142.02/772.29 81.61%↓	219.04/1184 81.5%↓	218.17/1263.91 82.74%↓	299.89/1974.43 84.81%↓	572.72/1157.08 50.5%↓	367.5/4850.45 92.42%↓
Mult(s)	17.3/69.4 75.07%↓	19.24/78.06 75.35%↓	23.92/119.62 80%↓	160.96/277.47 41.99%↓	169.64/312.10 45.65%↓	205.49/478.36 57.04%↓	105.12/211.65 50.34%↓	68.21/202.8 66.37%↓
Add(s)	2.16/8.68 75.1%↓	2.4/9.76 75.38%↓	2.99/14.96 80.03%↓	20.12/34.7 42.01%↓	21.2/39.03 45.68%↓	25.71/59.83 57.03%↓	13.13/26.46 50.38%↓	7.79/25.31 69.20%↓
Overall Cost(s)	130.02/536.57 **75.77%↓**	153.56/583.4 **73.68%↓**	168.93/906.87 **81.37%↓**	400.12/1496.16 **73.26%↓**	409.01/1615.05 **74.68%↓**	531.09/2512.62 **78.86%↓**	690.96/1395.19 **50.48%↓**	443.51/5078.88 **91.27%↓**
Model Accuricy								
Baseline	93.80%	94.09%	94.50%	63%	64.36%	65%	65.26%	65.50%
MOSAIC's	93.56%	94.01%	94.16%	63.88%	64.33%	64.87%	65.55%	65.30%

*The pruned model maintains similar accuracy to the baseline model. Computation performance data entries show the actual time cost achieved by the pruned model compared to the baseline. For example, a reduction from 485.49s to 110.56s is represented as 110.56/485.49 (75.89% ↓), indicating a 75.89% cost reduction for the Perm operation.

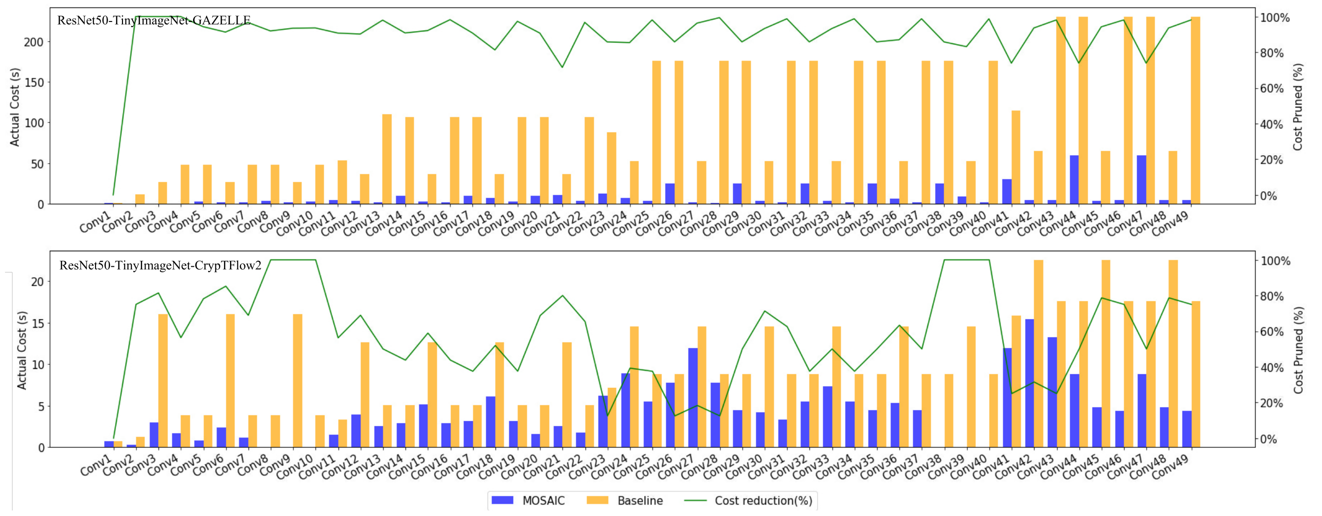


Figure 14: ResNet50 TinyImageNet Layer-Wise Performance Breakdown.

Table 4: MOSAIC's computation performance compared with baseline on VGG-16 and ResNet-50 with CIFAR-10 and Tiny ImageNet dataset within the CrypTFlow2 frameworks.

Dataset	CIFAR10	Tiny ImageNet	
Models	VGG16	VGG16	ResNet50
Computation Cost in Convolution			
Perm(s)	49.31/113.83 56.68%↓	80.65/151.88 46.90%↓	137.07/293.85 53.35%↓
Mult(s)	31.98/119.62 73.27%↓	109.22/478.36 77.17%↓	70.46/202.80 65.26%↓
Add(s)	4/14.96 73.28%↓	13.65/59.83 77.18%↓	8.63/25.31 65.90%↓
Overall Cost(s)	85.28/248.41 **65.67%↓**	203.51/690.07 **70.51%↓**	216.16/521.96 **58.59%↓**
Model Accuricy			
Baseline	94.50%	65%	65.50%
MOSAIC's	94.41%	64.94%	65.54%

Visualization of MOSAIC: To gain a deeper understanding of the Prune-and-Assemble approach of MOSAIC, we have provided visualizations of MOSAIC applied to various MIMO schemes. These include: (1) Ungrouped Out-Rot MIMO, (2) Ungrouped Out-Rot MIMO ResBlock, (3) In-Rot MIMO, and (4) Grouped Out-Rot MIMO. These visualizations are presented in Figures 15, 16, 17, and 18. In these binary representations, pruned values are depicted in black, while unpruned values are represented in white. In each of the figures, (A) corresponds to the kernel matrix pruned by the basic units, while (B) shows the kernel matrix where the pruned units are assembled together into HE Pruning Structures using corresponding channel transformations. As we can observe, the basic pruning units have been cleverly designed to fulfill two primary objectives. Firstly, these units are tessellated and compact, ensuring minimal model accuracy disruption during the pruning process. Secondly, they seamlessly align with the subsequent channel transformations.

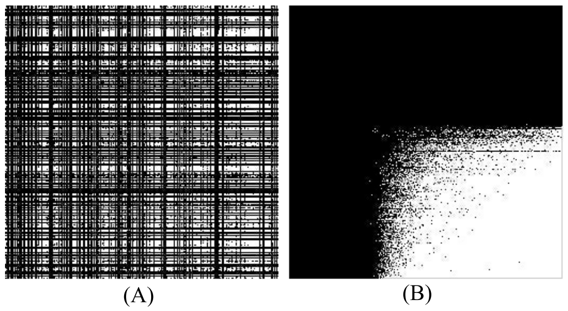


Figure 15: Visualization of MOSAIC's Pruning for Ungrouped Out-Rot MIMO.

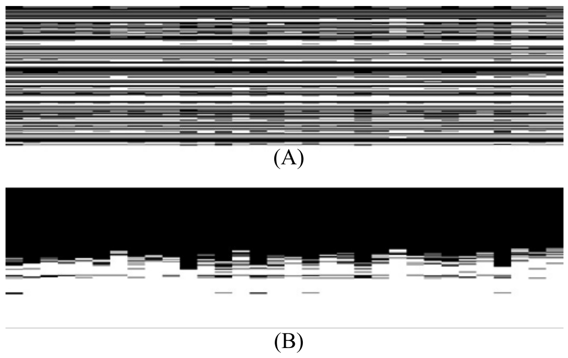


Figure 16: Visualization of MOSAIC's Pruning for Ungrouped Out-Rot MIMO, ResBlock.

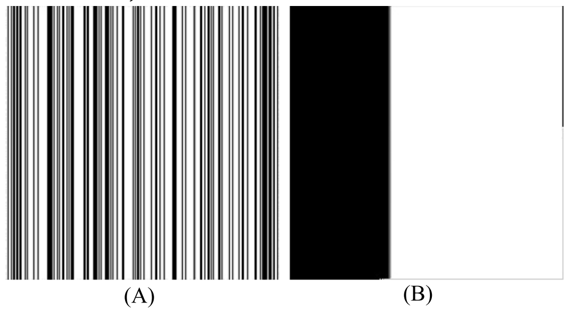


Figure 17: Visualization of MOSAIC's Pruning for In-Rot MIMO.

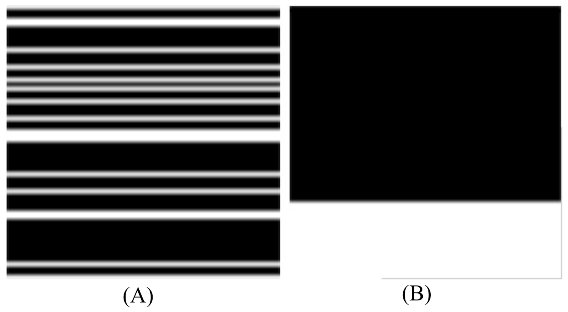


Figure 18: Visualization of MOSAIC's Pruning for Grouped Out-Rot MIMO.

This compatibility enables them to be effectively assembled into more HE Pruning Structures, leading to enhanced cost reduction.

5 CONCLUSION

In this paper, we introduce "MOSAIC," a highly flexible pruning framework characterized by its fine pruning granularity. The core of MOSAIC is based on a **"Prune-and-Assemble"** strategy. This entails initially pruning the convolutional layer with numerous tessellated basic units, followed by the assembly of these pruned units into HE Pruning Structures through our proposed channel transformations. By adopting this approach, MOSAIC attains a high pruning ratio while circumventing accuracy drops that typically accompany direct pruning of inflated structures. Utilizing the state-of-the-art HE-based frameworks GAZELLE and CrypTFlow2 as our testbed, we successfully apply MOSAIC to popular DL models like VGG and ResNet series on classic datasets including CIFAR-10 and Tiny ImageNet. The experimental results show that MOSAIC can perform efficient and precise pruning, reducing the Perm, Mult, and Add operations to achieve significant cost reduction without any loss in accuracy. For instance, when applied to VGG-16 on Tiny ImageNet, MOSAIC achieves total cost reductions of 78.86% and 70.51% within the GAZELLE and CrypTFlow2 frameworks, respectively. This underscores the high efficiency of MOSAIC's HE-friendly pruning approach.

ACKNOWLEDGMENTS

The authors would like to express thanks of gratitude to the lab members for helping and collaborating on this work and anonymous reviewers for the constructive comments. The work of H. Wu and R. Ning was supported in part by the National Science Foundation under Grants OAC-2320999, CNS-2120279, CNS-2153358 and DUE-2153358, DoD Center of Excellence in AI and Machine Learning (CoE-AIML) under Contract Number W911NF-20-2-0277, and the Commonwealth Cyber Initiative.

REFERENCES

[1] Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin Lauter, et al. 2021. Homomorphic encryption standard. *Protecting privacy through homomorphic encryption* (2021), 31–62.
[2] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. 2012. Foundations of garbled circuits. In *Proceedings of the ACM CCS.* 784–796.
[3] Fabian Boemer, Rosario Cammarota, Daniel Demmler, Thomas Schneider, and Hossein Yalame. 2020. MP2ML: a mixed-protocol machine learning framework for private inference. In *Proceedings of the ARES.* 1–10.
[4] Zvika Brakerski. 2012. Fully homomorphic encryption without modulus switching from classical GapSVP. In *Annual Cryptology Conference*. Springer, 868–886.
[5] Zvika Brakerski, Craig Gentry, and Shai Halevi. 2013. Packed ciphertexts in LWE-based homomorphic encryption. In *Proceedings of the PKC.* 1–13.
[6] Gilles Brassard, Claude Crépeau, and Jean-Marc Robert. 1986. All-or-nothing disclosure of secrets. In *Proceedings of the EUROCRYPT.* 234–238.
[7] Yifei Cai, Qiao Zhang, Rui Ning, Chunsheng Xin, and Hongyi Wu. 2022. Hunter: He-friendly structured pruning for efficient privacy-preserving deep learning. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security.* 931–945.
[8] Tianyi Chen, Bo Ji, Tianyu Ding, Biyi Fang, Guanyi Wang, Zhihui Zhu, Luming Liang, Yixin Shi, Sheng Yi, and Xiao Tu. 2021. Only train once: A one-shot neural network training and pruning framework. *Advances in Neural Information Processing Systems* 34 (2021), 19637–19651.
[9] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. 2017. Homomorphic encryption for arithmetic of approximate numbers. In *Proceedings of the ASIACRYPT.* 409–437.
[10] Daniel Demmler, Thomas Schneider, and Michael Zohner. 2015. ABY-A framework for efficient mixed-protocol secure two-party computation.. In *Proceedings of the NDSS.*
[11] Junfeng Fan and Frederik Vercauteren. 2012. Somewhat practical fully homomorphic encryption. *IACR Cryptol. ePrint Arch.* (2012), 144.

[12] 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 *Proceedings of the ICML*. 201–210.

[13] Yiwen Guo, Anbang Yao, and Yurong Chen. 2016. Dynamic network surgery for efficient dnns. *arXiv preprint arXiv:1608.04493* (2016).

[14] Kanav Gupta, Deepak Kumaraswamy, Nishanth Chandran, and Divya Gupta. 2022. Llama: A low latency math library for secure inference. *Cryptology ePrint Archive* (2022).

[15] Song Han. 2017. *Efficient methods and hardware for deep learning*. Ph. D. Dissertation. Stanford University.

[16] Song Han, Jeff Pool, John Tran, and William J Dally. 2015. Learning both weights and connections for efficient neural networks. *arXiv preprint arXiv:1506.02626* (2015).

[17] Meng Hao, Hongwei Li, Hanxiao Chen, Pengzhi Xing, Guowen Xu, and Tianwei Zhang. 2022. Iron: Private inference on transformers. *Advances in Neural Information Processing Systems* 35 (2022), 15718–15731.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE CVPR*. 770–778.

[19] Zhicong Huang, Wen-jie Lu, Cheng Hong, and Jiansheng Ding. 2022. Cheetah: Lean and fast secure {two-party} deep neural network inference. In *31st USENIX Security Symposium (USENIX Security 22)*. 809–826.

[20] Huelse. 2023. *Microsoft SEAL 4.X For Python*. https://github.com/Huelse/SEAL-Python

[21] Xiaoqian Jiang, Miran Kim, Kristin Lauter, and Yongsoo Song. 2018. Secure outsourced matrix computation and application to neural networks. In *Proceedings of the ACM SIGSAC*. 1209–1222.

[22] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. 2018. {GAZELLE}: A low latency framework for secure neural network inference. In *Proceedings of the USENIX Security*. 1651–1669.

[23] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *Proceedings of the NeurIPS* 25 (2012), 1097–1105.

[25] Nishant Kumar, Mayank Rathee, Nishanth Chandran, Divya Gupta, Aseem Rastogi, and Rahul Sharma. 2020. Cryptflow: Secure tensorflow inference. In *Proceedings of the IEEE S&P*. 336–353.

[26] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[27] Jian Liu, Mika Juuti, Yao Lu, and Nadarajah Asokan. 2017. Oblivious neural network predictions via minionn transformations. In *Proceedings of the ACM SIGSAC*. 619–631.

[28] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. 2017. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE ICCV*. 2736–2744.

[29] Pratyush Mishra, Ryan Lehmkuhl, Akshayaram Srinivasan, Wenting Zheng, and Raluca Ada Popa. 2020. Delphi: A Cryptographic Inference Service for Neural Networks. In *Proceedings of the USENIX Security*. 2505–2522.

[30] Payman Mohassel and Peter Rindal. 2018. ABY3: A mixed protocol framework for machine learning. In *Proceedings of the ACM SIGSAC*. 35–52.

[31] Payman Mohassel and Yupeng Zhang. 2017. Secureml: A system for scalable privacy-preserving machine learning. In *Proceedings of the IEEE S&P*. 19–38.

[32] Deepika Natarajan and Wei Dai. 2021. Seal-embedded: A homomorphic encryption library for the internet of things. *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2021), 756–779.

[33] Lucien KL Ng and Sherman SM Chow. 2021. GForce: GPU-Friendly Oblivious and Rapid Neural Network Inference. In *Proceedings of the USENIX Security*.

[34] Pascal Paillier. 1999. Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of the EUROCRYPT*. 223–238.

[35] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, and et al Chanan. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[36] Arpita Patra, Thomas Schneider, Ajith Suresh, and Hossein Yalame. 2021. ABY2.0: Improved mixed-protocol secure two-party computation. In *Proceedings of the USENIX Security*.

[37] Arpita Patra and Ajith Suresh. 2020. BLAZE: blazing fast privacy-preserving machine learning. *arXiv preprint arXiv:2005.09042* (2020).

[38] Deevashwer Rathee, Mayank Rathee, Nishant Kumar, Nishanth Chandran, Divya Gupta, Aseem Rastogi, and Rahul Sharma. 2020. CrypTFlow2: Practical 2-party secure inference. In *Proceedings of the ACM SIGSAC*. 325–342.

[39] M Sadegh Riazi, Mohammad Samragh, Hao Chen, Kim Laine, Kristin E Lauter, and Farinaz Koushanfar. 2019. XONN: XNOR-based Oblivious Deep Neural Network Inference.. In *USENIX Security Symposium*. 1501–1518.

[40] M Sadegh Riazi, Christian Weinert, Oleksandr Tkachenko, Ebrahim M Songhori, Thomas Schneider, and Farinaz Koushanfar. 2018. Chameleon: A hybrid secure computation framework for machine learning applications. In *Proceedings of the ASIACCS*. 707–721.

[41] Bita Darvish Rouhani, M Sadegh Riazi, and Farinaz Koushanfar. 2018. Deepsecure: Scalable provably-secure deep learning. In *Proceedings of the DAC*. 1–6.

[42] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE CVPR*. 815–823.

[43] SEAL 2019. Microsoft SEAL (release 3.3). https://github.com/Microsoft/SEAL. Microsoft Research, Redmond, WA..

[44] Adi Shamir. 1979. How to share a secret. *Commun. ACM* 22, 11 (1979), 612–613.

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

[46] Stanford.edu. 2015. *Tiny ImageNet Dataset*. http://cs231n.stanford.edu

[47] Sijun Tan, Brian Knott, Yuan Tian, and David J Wu. 2021. CRYPTGPU: Fast Privacy-Preserving Machine Learning on the GPU. *arXiv preprint arXiv:2104.10949* (2021).

[48] Sameer Wagh, Divya Gupta, and Nishanth Chandran. 2019. SecureNN: 3-Party Secure Computation for Neural Network Training. *Proc. Priv. Enhancing Technol.* 2019, 3 (2019), 26–49.

[49] Sameer Wagh, Shruti Tople, Fabrice Benhamouda, Eyal Kushilevitz, Prateek Mittal, and Tal Rabin. 2020. Falcon: Honest-majority maliciously secure framework for private deep learning. *arXiv preprint arXiv:2004.02229* (2020).

[50] Wei Wang, Sheng Wang, Jinyang Gao, Meihui Zhang, Gang Chen, Teck Khim Ng, and Beng Chin Ooi. 2018. Rafiki: Machine learning as an analytics service system. *arXiv preprint arXiv:1804.06087* (2018).

[51] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. 2016. Learning structured sparsity in deep neural networks. *Proceedings of the NeurIPS* 29 (2016), 2074–2082.

[52] Qiao Zhang, Cong Wang, Hongyi Wu, Chunsheng Xin, and Tran V Phuong. 2018. GELU-Net: A Globally Encrypted, Locally Unencrypted Deep Neural Network for Privacy-Preserved Learning.. In *Proceedings of the IJCAI*. 3933–3939.

[53] Qiao Zhang, Chunsheng Xin, and Hongyi Wu. 2021. GALA: Greedy ComputAtion for Linear Algebra in Privacy-Preserved Neural Networks. *arXiv preprint arXiv:2105.01827* (2021).

A ANALYSIS OF COMPLEXITY, ACTUAL COST AND SELECTION GUIDELINES FOR THREE MIMO SCHEMES

As previously mentioned, there are three schemes for calculating convolution in the MIMO manner: (1) Ungrouped Out-Rot MIMO, (2) In-Rot MIMO, and (3) Grouped Out-Rot MIMO. The first two schemes find application within the frameworks like GAZELLE and DELPHI, while the latter two are employed within some framework such as CrypTFlow2 and GALA. However, existing privacy-preserving pruning mechanisms [7], often tend to employ Ungrouped Out-Rot MIMO without a comprehensive analysis. Our research delves deeper into this issue by analyzing the complexity and actual time cost of three schemes. Furthermore, we provide selection guidelines to facilitate the determination of the most suitable MIMO scheme for a given layer. This aids in maximizing the benefits derived from MOSAIC's pruning method, ultimately leading to the attainment of the most optimal cost reduction.

Table 5: Complexity of Three MIMO schemes.

MIMO Scheme	# Perm
Ungrouped Out-Rot	$\frac{c_i(k_w k_h-1)}{c_n}+\frac{c_i c_o(c_n-1)}{c_n^2}$
In-Rot	$\frac{c_i(k_w k_h-1)}{c_n}+\frac{c_i(c_n-1)k_w k_h}{c_n}$
Grouped Out-Rot	$\frac{c_i(k_w k_h-1)}{c_n}+\frac{c_o(c_n-1)}{c_n}$

*c_i: number of input channels, c_o: number of output channels, c_n: number of channels packed in one ciphertext, $k_w k_h$: kernel size in the horizontal and vertical directions.

Let's get straight to the point, the complexity of the three schemes is outlined in Table 5. The complexity here is quantified by the number of Perm operations for each scheme. Perm operation carries a higher cost compared to Mult and Add operation, and since these three schemes exhibit a relatively similar number of Mult and Add operations, the number of Perm operations becomes the determining factor in assessing complexity. Foremost, it's indisputable that the selected MIMO scheme should possess a lower complexity. Let's assume that our preference is to employ the In-Rot MIMO, rather than Ungrouped Out-Rot MIMO or Grouped Out-Rot MIMO, for GAZELLE and CrypTFlow2 frameworks respectively. This preference is based on the following preconditions:

$$GAZELLE: \frac{c_i c_o\,(c_n-1)}{c_n^2} > \frac{c_i}{c_n}(c_n-1)k_w k_h \qquad \frac{c_o}{k_w k_h} > c_n, \tag{1}$$

$$CrypTFlow2: \frac{c_o\,(c_n-1)}{c_n} > \frac{c_i}{c_n}(c_n-1)k_w k_h \qquad \frac{c_o}{k_w k_h} > c_i, \tag{2}$$

Thus far, Equation (1) and (2) can help to determine whether a layer can benefit from Out-Rot or In-Rot in terms of complexity. Furthermore, we know complexity serves as an indicator that influences cost. Ultimately, the real-world cost is rooted in the actual time taken for operations to be executed.

In certain scenarios, despite having the same number of operations (equivalent complexity), the actual time can vary. For instance, the actual time cost of a Perm operation hinges on two parameters: the length of the ciphertext to be rotated and the number of positions by which it must be rotated. Let's delve into an illustrative example to better comprehend this concept.

In the input rotation scheme, as shown Figure 7, one conversion Perm must be done to convert input ciphertext $[v_1]_C$ into $[v'_1]_C$. Additionally, $(k_w k_h - 1)$ Perm operations are performed to obtain the rotated ciphertext for the convolution between $[v'_1]_C$ and the kernels groups. As illustrated in Figure 19, assuming that $c_n = 2$, the stride is 1 and the 3×3 kernels are used ($k_w = k_h = 3$), the input ciphertext $[v]_C$ will be firstly rotated ($k_w k_h - 1 = 8$) times by Perm (as indicated by the black arrows). As a result, we obtain nine rotated ciphertexts from $[v_{(-4)}]_C$ to $[v_{(4)}]_C$, including the original $[v_{(0)}]_C$. Furthermore, there are two methods for converting the ciphertext: Method A and Method B. In Method A, $[v_{(0)}]_C$ is first converted to $[v'_{(0)}]_C$ (indicated by the red arrow), then $[v'_{(0)}]_C$ is rotated eight times to generate $[v'_{(-4)}]_C$ to $[v'_{(4)}]_C$ (indicated by the green arrows). In Method B, the nine rotated ciphertexts, ranging from $[v_{(-4)}]_C$ to $[v_{(4)}]_C$, are directly obtained by performing nine convert Perm (indicated by the red arrows). Both methods require nine additional Perms, but Method B is found out more cheap in most cases.

As we mentioned, the cost of a Perm is determined by two parameters: the length of the ciphertext to be rotated and the number of positions by which it needs to be rotated. Longer ciphertexts require more time to be rotated. Regarding the latter parameter, we found that rotating the ciphertext by a power of two positions requires less computation time, based on our experiments. When the size of the feature map of one input channel is $N \times N$, the convert Perm (red) rotates the ciphertext by $N \times N$ positions, and the size of N is typically a power of two for efficiency reasons

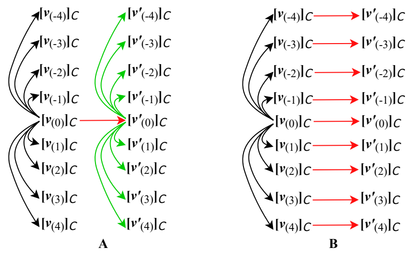


Figure 19: Input Ciphertext Conversion.

in HE-based scenarios [1]. Consequently, the cost of the convert Perm (red) is low. However, the eight Perms (green) that rotate the ciphertext by positions $-(N+1)$, $-(N)$, $-(N-1)$, -1, $+1$, $+(N-1)$, $+(N)$ and $+(N+1)$, which are not the power of two. These Perms (green) require more computation time than convert Perm (red). This conclusion holds true for different kernel sizes, strides, and c_n. Thus, Method B is selected to conduct the conversion.

Next, we analyse the selection guidelines in terms of actual time cost. For Perm operation, first, we record the actual cost of each type of Perm with different lengths and rotation positions in the experiments. Then, we count the number of each type of Perm in the DL model, and the sum of their products is the model's actual Perm cost. The actual time cost of Mult and Add can be obtained similarly. Thus, the actual time cost of a DL model can be expressed at a high level as follows:

$$T = \Theta \times \tau \times (1 - \beta), \tag{3}$$

Here, T represents the overall actual time cost, Θ denotes complexity (i.e., the number of operations), τ signifies the set of actual time costs for each operation, and β stands for the pruning ratio. Through this approach, we can acquire the actual time cost of both the original model and the pruned model employing each MIMO scheme. This help us to identify the most optimal scheme. Certainly, the determination of τ and β demands significant experimental effort. However, what we truly require are selection guidelines that can assist in identifying the optimal MIMO scheme prior to the commencement of training and pruning. This proactive approach is essential for efficient decision-making during the deployment phase. As a result, we have distilled our comprehensive experimentation along with Equations (1), (2), and (3) into the following heuristic formulas for practical application:

$$\rho_1 = \frac{c_o}{k_w k_h c_n}, \tag{4}$$

$$\rho_2 = \frac{c_o}{k_w k_h c_i}, \tag{5}$$

where ρ_1 and ρ_2 function as indicators guiding the selection of the appropriate MIMO scheme for specific layers within the GAZELLE and CrypTFlow2 frameworks, respectively. If $\rho_1 < 1$ or $\rho_2 < 1$, it's advisable to opt for Ungrouped or Grouped Out-Rot MIMO within their respective frameworks. Conversely, if $\rho_1 > 2$ or $\rho_2 > 1.3$, it suggests that the model greatly benefits from In-Rot MIMO, with a higher value indicating greater benefits. When $1 \le \rho_1 \le 2$ or $1 \le \rho_2 \le 1.3$, a relatively rare scenario, the efficiency of Out-Rot and In-Rot is nearly equivalent. In such cases, the optimal choice should be verified through actual validation. It's worth noting that

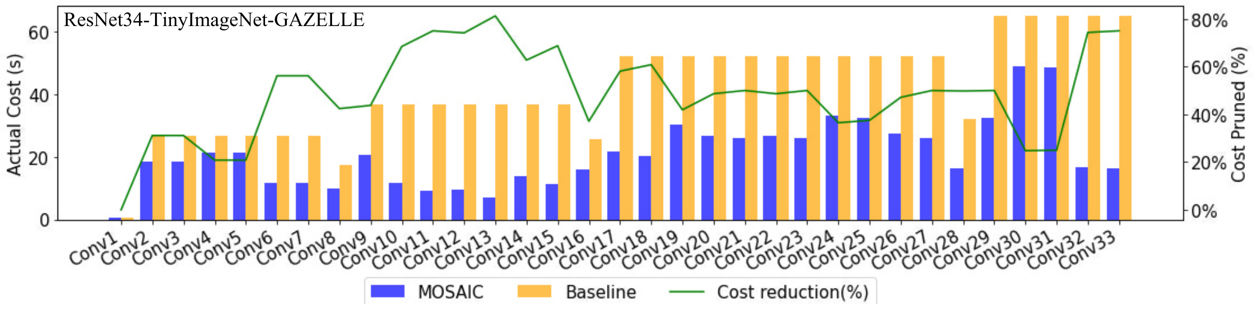


Figure 20: ResNet34 Layer-Wise Performance Breakdown.

the efficacy and efficiency of this formula have been thoroughly validated across all our extensive experimentation.

B LAYER-WISE PERFORMANCE BREAKDOWN OF MODEL PRUNED BY MOSAIC:

We conduct the analysis of pruning performance by dissecting the complete model into individual layers. In the main text, we have provided certain layer-wise breakdowns. Additionally, Figure 20, 21, 22 visually demonstrate the layer-wise breakdown for the remaining experiments conducted within the GAZELLE framework. These encompass VGG-11 and VGG-13 on CIFAR-10, VGG-11 and VGG-13 on Tiny ImageNet, as well as ResNet-34 on Tiny ImageNet.

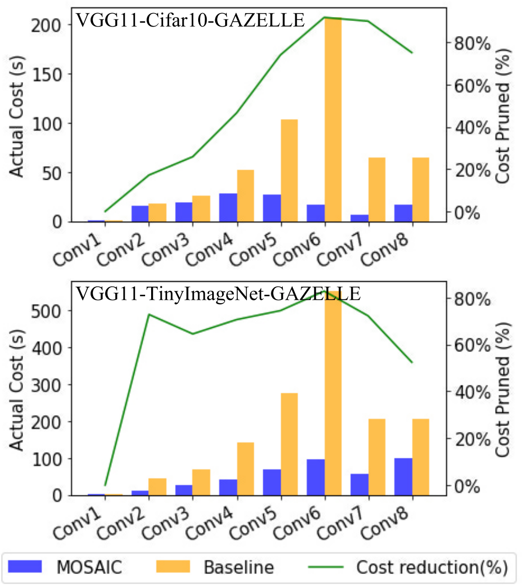


Figure 21: VGG11 Layer-Wise Performance Breakdown.

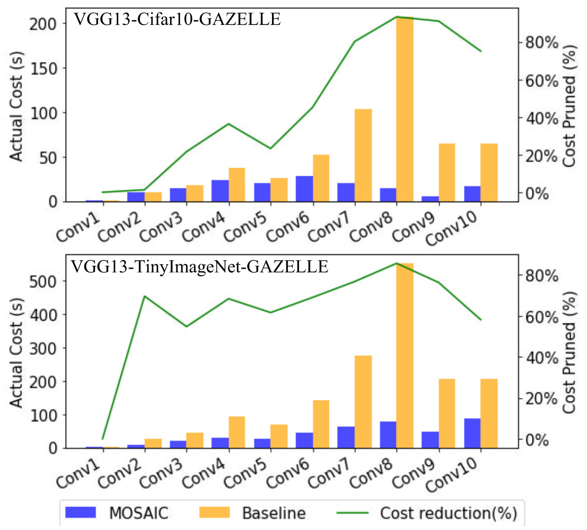


Figure 22: VGG13 Layer-Wise Performance Breakdown.