Prune, Permute and Expand: Efficient Machine Learning under Non-Client-Aided Homomorphic Encryption

[*] **Extended version available in [1]**

Ehud Aharoni[*] Moran Baruch[*] Pradip Bose[†] Alper Buyuktosunoglu[†] Nir Drucker[*]
Subhankar Pal[†] Tomer Pelleg[*] Kanthi Sarpatwar[†] Hayim Shaul[*] Omri Soceanu[*] Roman Vaculin[†]

[*] IBM Research - Israel, Haifa, Israel [†] IBM T.J. Watson Research Center, Yorktown Heights, NY, USA

***Abstract*—Privacy-preserving neural network (NN) inference solutions under homomorphic encryption (HE) have recently gained significant traction with several solutions that provide different latency-bandwidth trade-offs. Pruning the parameters of a NN model is a well-known approach to improving inference latency. However, pruning methods that are useful in the plaintext context may lend nearly negligible improvement in the HE case. In this work, we propose a novel set of pruning methods that reduce the latency and memory requirement, thus bringing the effectiveness of plaintext pruning methods to HE. We evaluate our methods on an autoencoder architecture on MNIST and show that our best method prunes $\sim 2\times$ more ciphertexts than our adaptation of a state-of-the-art scheme called Hunter, for negligible increase in mean-square reconstruction loss.**

I. Introduction

Data privacy and confidentiality are of utmost priority in today's world that is driven by information exchange. One promising solution that has been garnering significant traction is the use of homomorphic encryption (HE), which allows for the evaluation of certain functions on encrypted inputs. The major challenge that is constraining faster adoption is that HE applications involving a large number of operations, such as inference on large machine learning (ML) models under HE, is at best a few orders-of-magnitude slower than their plaintext counterparts and has significantly more memory requirement.

We focus on the use of HE in the context of privacy-preserving machine learning (PPML) inference frameworks. These frameworks run on systems that involve at least one client and one server, where the clients desire the privacy of input data (*e.g.* images), and the server performs inference over an encrypted ML model, which has been trained with proprietary data. Further, we focus on a non-client aided model, where the client (*e.g.* a hospital) dispatches encrypted data (*e.g.* images of the patient's X-Ray images) for classification on a server that runs inference under HE, without any client interaction. The encrypted result is decoded and decrypted on the client side to reveal the outcome.

Our Contributions. This work aims to improve the speed of ML inference under HE with minimum impact on the network accuracy. Our contributions are summarized here:

- We introduce a new set of methods to perform packing-aware pruning, which combines four main primitives: prune, permute, pack, and expand. Specifically, we provide a novel permutation algorithm for neural networks (NNs) with sequences of fully connected (FC) layers.
- We adapt a state-of-the-art pruning technique called Hunter [2] to HeLayers that supports non-interactive HE evaluations of deep NNs, which serves as our baseline.

Table I: Scope, criterion and target of pruning in our proposed methods listed in the first column.

P2T	Scope	Local (Lc), Global (Gl)
	Criterion	Average/Maximum/Minimum of tile (T-Avg/T-Max/T-Min)
	Target	Weight (-)
P2, P3, P3E	Scope	Local (Lc), Global (Gl)
	Criterion	L1 (L1), Rand (Rnd)
	Target	Weight (Wei), Neuron (Neu)
P4, P4E	Scope	Local (Lc), Global (Gl) [1st and 2nd prune]
	Criterion	L1 (L1), Rand (Rnd) [1st prune], threshold fraction of non-zeros in tile to prune [2nd prune]
	Target	Weight (Wei), Neuron (Neu) [1st prune]

II. Background

A. Ciphertext Packing in Homomorphic Encryption (HE)

Typical HE schemes perform computation in the form of single-instruction, multiple data (SIMD). One recent work, called HeLayers [3], is a framework that proposes efficient packing of multi-dimensional inputs by decomposing the input into tiles that are each packed into a separate ciphertext. For example, a simple 25×25 grayscale image may be packed into a set of 49 $(= \lceil \frac{25}{4} \rceil \cdot \lceil \frac{25}{4} \rceil)$ 4×4 tiles.

B. Neural Network (NN) Pruning

One common technique to address the problem of complex and large ML models with high memory and latency requirements is pruning, *e.g.* pruning NN convolutional filters, FC weights or even entire nodes from a trained model. We adapt various NN pruning strategies from plaintext ML to our scenario of non-interactive HE under SIMD packing and show that they fail for any reasonably-sized packing shape. This is due to the fundamental reason that while pruning may introduce zeros, if the zeros reside in a ciphertext where even one other element is a non-zero, then the ciphertext cannot be eliminated. For instance, when we aggressively pruned 85% of weights in our network with our best pruning scheme, only 17% of the ciphertexts were eliminated.

III. Proposal

In this work, we explore several pruning methods that we summarize in Table I, however, we deep-dive on the P4E scheme only, for brevity. Universally, our methods accept a pruning fraction as input which indicates the fraction of the **target** to prune, where the target can be weights or neurons. Weight pruning is illustrated in Figure 1 (i)-(ii). The pruning **criteria** we consider include *(i)* pruning uniformly at random, *i.e.* randomly pruning neurons or randomly setting weights to zero, and *(ii)* pruning based on some threshold, *e.g.* the L1-norm of a weight. The **scope** of pruning indicates whether the pruning is done layer-by-layer, or all at once. When using the random criterion, the scope does not have an

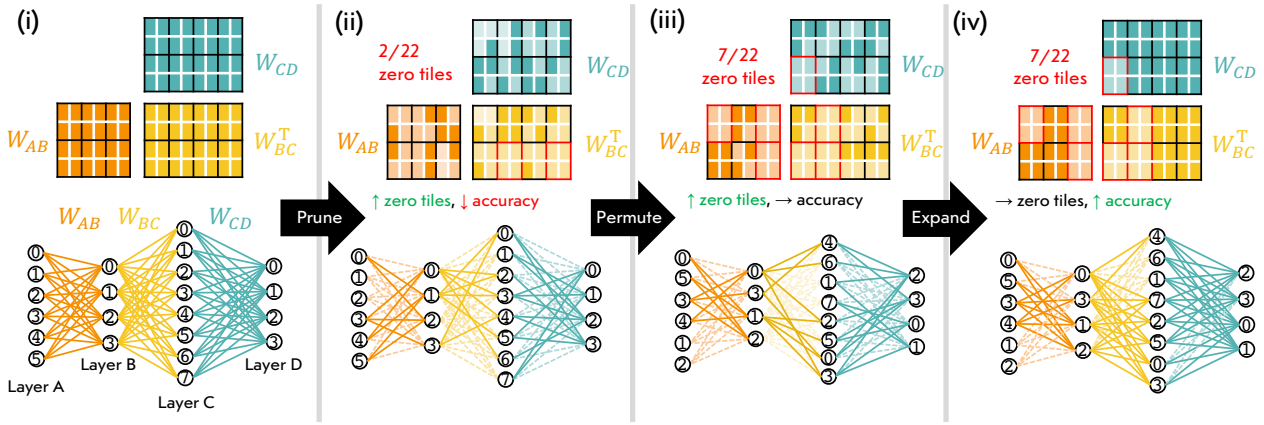

Figure 1: Illustration of our proposed P3E pruning method when considering a 4-layer network with 6,4,8,4 neurons in layers A-D, respectively. We divide the weight matrices into 2×2 tiles and perform $54/88 = 61\%$ weight-based pruning. After pruning we can only reduce $2/22 \approx 10\%$ tiles. After permutation, however, it is possible to reduce $7/22 = 35\%$ of the tiles. Expansion (with an implicit post-expansion re-training step) is useful to restore most of the accuracy loss.

effect. However, it has a major effect when considering, *e.g.* L1-based pruning, particularly if there is a large variance in the weights across the different layers.

Our most important contributions are the permute and expand operations (Figure 1 (ii)-(iv)). Permutation is performed after the pruning step for P2, P3, and P3E. Expansion is used in the P4 and P4E schemes. The *permute* operation permutes the rows and columns of the weight matrices after the pruning operation to essentially congregate the zero elements together. For our particular implementation, we first concatenate adjacent matrices, *e.g.* W_{AB} and W_{BC}^{T} in round 1, and W_{BC} and W_{CD}^{T} in round 2. We then feed each row of the concatenated matrix as a vector (after thresholding) into a k-means clustering pass, that groups together rows with zeros in similar indices. The Hamming distance is used as the distance function for clustering. Each set of rounds is repeated for several iterations, until the number of zero tiles cannot be improved further. Figure 1 (iii) shows how this approach increases the percentage of zero tiles from 10% to 35%.

The *expand* operation is a reversal of the pruning operation, where we search for tiles that are *not completely zero*, *i.e.* they hold zeros and non-zeros, and we un-prune the zero elements inside them. The motivation behind this action lies in the motivation to perform pruning, which is to reduce the number of active tiles. If a tile is not reduced, *i.e.* it has non-zero elements, then we cannot ignore it, and because we do not gain any performance benefits, it is best to fully utilize its elements to improve the accuracy of the model.

IV. Evaluation

We compare our best method (P4E) with an adaption of a scheme called Hunter [2] to a non-interactive solution, which we refer to as P2T. We construct four variants of P2T, *viz.* *Gl/T-Avg/-*, *Gl/T-Max/-*, *Lc/T-Avg/-* and *Lc/T-Max/-*. All of these use a one-shot pruning (no permutation or expansion) that is packing-aware, with the scope and reduction criterion specified by the first two parameters in the scheme name.

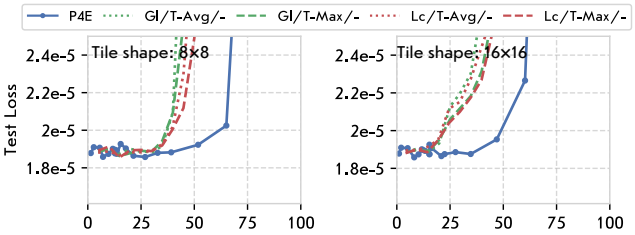

Figure 2: A comparison of the best overall pruning method (P4E, Gl/L1/Wei) against P2T variants over an autoencoder with 1 FC layer of size 32 and different tile shapes (8×8, 16×16). The y-axis is the value of the mean square error loss and the x-axis is the percentage of pruned zero tiles.

The results for MNIST inference on an autoencoder with 1 FC layer of size 32 are shown in Figure 2, for 2 tile shapes. Note that we obtain similar results for CIFAR-10, but we omit the analysis for this work. Our scheme outperforms P2T because the variance of weights held in each tile grows with the size of the tile, *i.e.* the larger the tile size, the more the chances that it holds a set of weights that are different in magnitude. When all of these weights are pruned at once, as is done in P2T, it leads to degradation in model accuracy, since important weights are pruned away along with unimportant ones. The general trend is that the degradation for P2T is much worse as the tile size increases, compared to P4E. These results effectively showcase the benefits of our prune, permute and expand approaches, in lieu of blind packing-aware pruning schemes, such as P2T.

V. Conclusion

We present a framework of pruning for HE-enabled NN inference that combines four critical primitives, *viz.* pruning, permutation, expansion and packing. We adapted a state-of-the-art pruning technique called Hunter [2] to the non-interactive scenario and compared our best scheme against theirs, in terms of reconstruction loss and the fractions of ciphertexts eliminated, achieving ∼2× reduction in number of ciphertexts for roughly the same loss, compared to Hunter.

REFERENCES

[1] E. Aharoni, M. Baruch, P. Bose, A. Buyuktosunoglu, N. Drucker, S. Pal, T. Pelleg, K. Sarpatwar, H. Shaul, O. Soceanu *et al.*, "He-pex: Efficient machine learning under homomorphic encryption using pruning, permutation and expansion," *arXiv preprint arXiv:2207.03384*, 2022.

[2] Y. Cai, Q. Zhang, R. Ning, C. Xin, and H. Wu, "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*, ser. ASIA CCS '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 931–945.

[3] E. Aharoni *et al.*, "HeLayers: A Tile Tensors Framework for Large Neural Networks on Encrypted Data," *CoRR*, vol. abs/2011.0, 2020.