PASNet: Polynomial Architecture Search Framework for Two-party Computation-based Secure Neural Network Deployment

*Hongwu Peng[1], *Shanglin Zhou[1], *Yukui Luo[2], Nuo Xu[3], Shijin Duan[2], Ran Ran[3], Jiahui Zhao[1], Chenghong Wang[4], Tong Geng[5], Wujie Wen [3], Xiaolin Xu[2], and Caiwen Ding[1]
*These authors contributed equally.
[1]University of Connecticut, USA. [2]Northeastern University, USA. [3]Lehigh University, USA.
[4]Duke University, USA. [5]University of Rochester, USA.
[1]{hongwu.peng, shanglin.zhou, jiahui.zhao, caiwen.ding}@uconn.edu, [2]{luo.yuk, duan.s, x.xu}@northeastern.edu,
[3]{nux219, rar418, wuw219}@lehigh.edu, [4]cw374@duke.edu, [5]tgeng@ur.rochester.edu

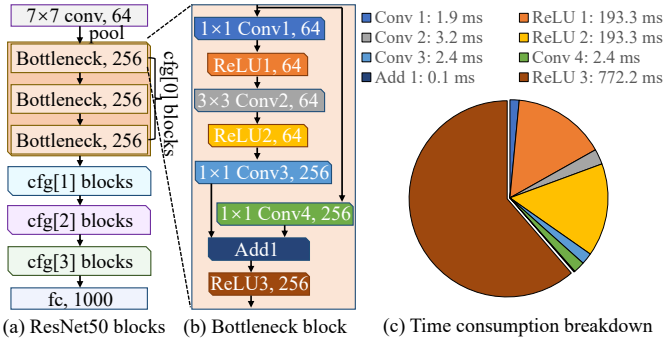
***Abstract*—Two-party computation (2PC) is promising to enable privacy-preserving deep learning (DL). However, the 2PC-based privacy-preserving DL implementation comes with high comparison protocol overhead from the non-linear operators. This work presents PASNet, a novel systematic framework that enables low latency, high energy efficiency & accuracy, and security-guaranteed 2PC-DL by integrating the hardware latency of the cryptographic building block into the neural architecture search loss function. We develop a cryptographic hardware scheduler and the corresponding performance model for Field Programmable Gate Arrays (FPGA) as a case study. The experimental results demonstrate that our light-weighted model PASNet-A and heavily-weighted model PASNet-B achieve 63 ms and 228 ms latency on private inference on ImageNet, which are 147 and 40 times faster than the SOTA CryptGPU system, and achieve 70.54% & 78.79% accuracy and more than 1000 times higher energy efficiency. The pretrained PASNet models and test code can be found on Github[1].**

***Index Terms*—Privacy-Preserving in Machine Learning, Multi Party Computation, Neural Architecture Search, Polynomial Activation Function, Software/Hardware Co-design, FPGA**

I. Introduction

Machine-Learning-As-A-Service (MLaaS) has been an emerging solution nowadays, to provide accelerated inference for diverse applications. However, most MLaaS require clients to reveal the raw input to the service provider [1] for evaluation, which may leak the privacy of users. Privacy-preserving deep learning (PPDL) and private inference (PI) have emerged to protect sensitive data in deep learning (DL). The current popular techniques include multi-party computation (MPC) [2] and homomorphic encryption (HE) [3]. HE is mainly used to protect small to medium-scale DNN models without involving costly bootstrapping and large communication overhead. MPC protocols such as secret-sharing [2] and Yao's Garbled Circuits (GC) [4] can support large-scale networks by evaluating operator blocks. This work mainly focuses on secure two-party computation (2PC), which represents the minimized system for multi-party computing (MPC) and is easy to extend [5].

[1]https://github.com/HarveyP123/PASNet-DAC2023



(a) ResNet50 blocks (b) Bottleneck block (c) Time consumption breakdown

Fig. 1: Lantecy of operators under 2PC PI setup. Network banwidth: 1 GB/s. Device: ZCU104. Dataset: ImageNet.

The primary challenge in 2PC-based PI is the comparison protocol overhead [6] for non-linear operators. As shown in Fig. 1, ReLU contributes over 99% of latency in a ciphertext setting for deep neural network (DNN), despite negligible overhead in plaintext. Replacing ReLU with second-order polynomial activation could yield 50× speedup.

To achieve high performance, good scalability, and high energy efficiency for *secure* deep learning systems, two orthogonal research directions have attracted enormous interest. The first one is the nonlinear operations overhead reduction algorithms. Existing works focus on *ReLU cost optimization*, e.g., minimizing ReLU counts (DeepReduce [7], CryptoNAS [8]) or replacing ReLUs with polynomials (CryptoNets [9], Delphi [10], SAFENet [11]), and *extremely low-bit* weights and activations (e.g., Binary Neural Network (BNN) [12]). However, these works neglect the accuracy impact. They often sacrifice the model comprehension capability, resulting in several accuracy losses on large networks and datasets such as ImageNet, hence are not scalable. The second trend is *hardware acceleration for PI* to speed up the MPC-based DNN through GPUs [2], [13]. Since no hardware characteristic is captured during DNN design, this top-down ("algorithm → hardware") approach can not effectively perform design space

exploration, resulting in sub-optimal solutions.

We focus on three observations: 1) preserving **prediction accuracy** for substantial benefits; 2) scalable **cryptographic overhead reduction** for various network sizes; 3) cohesive **algorithm/hardware optimizations** using closed loop "algorithm ↔ hardware" with design space exploration capturing hardware characteristics.

We introduce the **Polynomial Architecture Search (PASNet)** framework, which jointly optimizes DNN model structure and hardware architecture for high-performance MPC-based PI. Considering cryptographic DNN operators, data exchange, and factors like encoding format, network speed, hardware architecture, and DNN structure, PASNet effectively enhances the performance of MPC-based PI.

Our key design principle is to *enforce* exactly what is assumed in the DNN design—training a DNN that is both hardware efficient and secure while maintaining high accuracy.

To evaluate the effectiveness of our framework, we use FPGA accelerator design as a demonstration due to its predictable performance, low latency, and high energy efficiency for MLaaS applications (e.g., Microsoft Azure [14]). We summarize our contributions as follows:

1) We propose a trainable *straight through polynomial activation initialization* method for cryptographic hardware-friendly trainable polynomial activation function to replace the expensive ReLU operators.
2) Cryptographic hardware scheduler and the corresponding performance model are developed for the FPGA platform. The latency loop-up table is constructed.
3) We propose a differentiable cryptographic hardware-aware NAS framework to selectively choose the proper polynomial or non-polynomial activation based on given constraint and latency of cryptographic operators.

II. Basic of Cryptographic Operators

A. Secret Sharing

2PC setup. We consider a similar scheme involving two semi-honest in a MLaaS applications [5], where two servers receive the confidential inputs from each other and invoke a two party computing protocol for secure evaluation.

Additive Secret Sharing. In this work, we evaluate 2PC secret sharing. As a symbolic representation, for a secret value $x \in \mathbb{Z}_m$, $[\![x]\!] \leftarrow (x_{S_0}, x_{S_1})$ denotes the two shares, where $x_{S_i}, i \in \{0,1\}$ belong to server S_i. Other notations are as below:

- *Share Generation* shr(x): A random value r in $\mathbb{Z}_m$ is sampled, and shares are generated as $[\![x]\!] \leftarrow (r, x-r)$.
- *Share Recovering* rec$([\![x]\!])$: Given shares $[\![x]\!] \leftarrow (x_{S_0}, x_{S_1})$, it computes $x \leftarrow x_{S_0} + x_{S_1}$ to recover x.

An example of plaintext vs. secret shared based ciphertext evaluation is given in Fig. 2, where ring size is 4 and $\mathbb{Z}_m = \{-8, -7, ...7\}$. The integer overflow mechanism naturally ensures the correctness of ciphertext evaluation. Evaluation in the example involves secure multiplication, addition and comparison, and details are given in following sections.

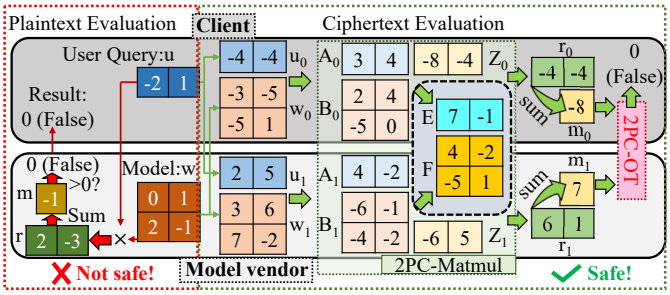


Fig. 2: A example of 4 bit plaintext vs. ciphertext evaluation.

B. Polynomial Operators Over Secret-Shared Data

Scaling and Addition. We denote secret shared matrices as $[\![X]\!]$ and $[\![Y]\!]$. The encrypted evaluation is given in Eq. 1.

$$[\![aX+Y]\!] \leftarrow (aX_{S_0} + Y_{S_0}, aX_{S_1} + Y_{S_1}) \tag{1}$$

Multiplication. We consider the matrix multiplicative operations $[\![R]\!] \leftarrow [\![X]\!] \otimes [\![Y]\!]$ in the secret-sharing pattern. where $\otimes$ is a general multiplication, such as Hadamard product, matrix multiplication, and convolution. We use oblivious transfer (OT) [15] based approach. To make the multiplicative computation secure, an extra Beaver triples [16] should be generated as $[\![Z]\!] = [\![A]\!] \otimes [\![B]\!]$, where A and B are randomly initialized. Specifically, their secret shares are denoted as $[\![Z]\!] = (Z_{S_0}, Z_{S_1})$, $[\![A]\!] = (A_{S_0}, A_{S_1})$, and $[\![B]\!] = (B_{S_0}, B_{S_1})$. Later, two matrices are derived from given shares: $E_{S_i} = X_{S_i} - A_{S_i}$ and $F_{S_i} = Y_{S_i} - B_{S_i}$, in each party end separately. The intermediate shares are jointly recovered as $E \leftarrow$ rec$([\![E]\!])$ and $F \leftarrow$ rec$([\![F]\!])$. Finally, each party, i.e, server S_i, will calculate the secret-shared R_{S_i} locally:

$$R_{S_i} = -i \cdot E \otimes F + X_{S_i} \otimes F + E \otimes Y_{S_i} + Z_{S_i} \tag{2}$$

Square. For the element-wise square operator shown $[\![R]\!] \leftarrow [\![X]\!] \otimes [\![X]\!]$, we need to generate a Beaver pair $[\![Z]\!]$ and $[\![A]\!]$ where $[\![Z]\!] = [\![A]\!] \otimes [\![A]\!]$, and $[\![A]\!]$ is randomly initialized. Then parties evaluate $[\![E]\!] = [\![X]\!] - [\![A]\!]$ and jointly recover $E \leftarrow$ rec$([\![E]\!])$. The result R can be obtained through Eq. 3.

$$R_{S_i} = Z_{S_i} + 2E \otimes A_{S_i} + E \otimes E \tag{3}$$

C. Non-Polynomial Operator Modules

Non-polynomial operators such as ReLU and MaxPool are evaluated using secure comparison protocol.

Secure 2PC Comparison. The 2PC comparison, a.k.a. millionaires protocol, is committed to determine whose value held by two parties is larger, without disclosing the exact value to each other. We adopt work [6] for 2PC comparison. Detailed modeling is given in Section III-C.

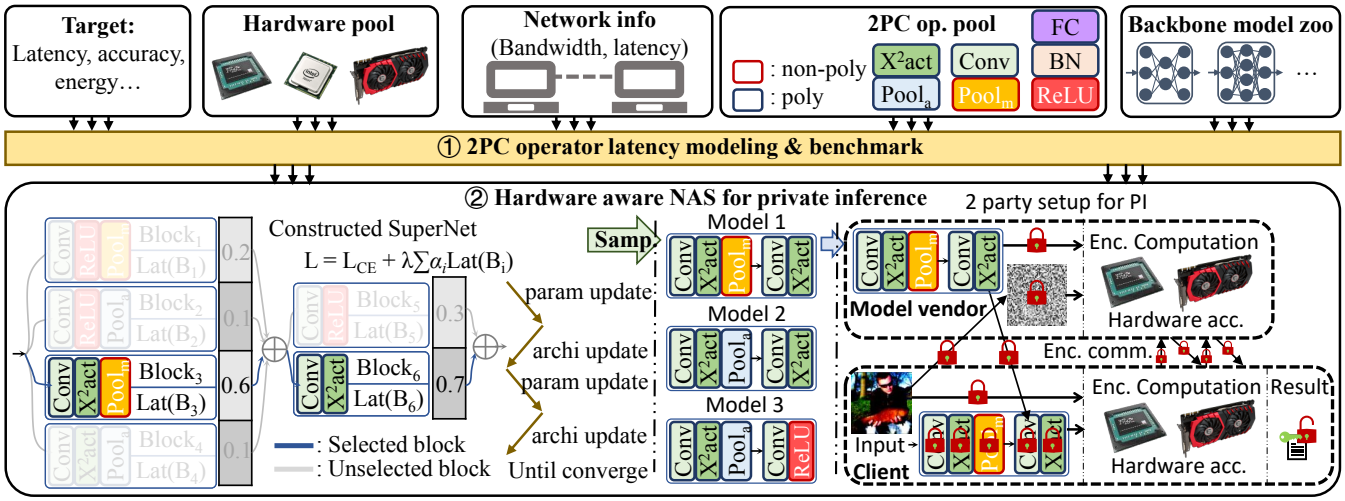


Fig. 3: Overview of PASNet framework for 2PC DNN based private inference setup.

III. The PASNet Framework

The framework (Fig. 3) takes inputs like optimization target, hardware pool, network information, and 2PC operator candidates for cryptographic operator modeling, benchmarking, and automated design space optimization in PI using hardware-aware NAS. This section presents a new cryptographic-friendly activation function, its initialization method, DNN operator modeling under 2PC, and a hardware-aware NAS framework for optimizing DNN accuracy and latency. While evaluated on FPGA accelerators, the method can be easily adapted to other platforms like mobile and cloud.

A. *Trainable* X^2act *Non-linear Function.*

We use a hardware friendly trainable second order polynomial activation function as an non-linear function candidate, shown in Eq. 4, where w_1, w_2 and b are all trainable parameters. We propose ***straight through polynomial activation initialization*** **(STPAI)** method to set the w_1 and b to be small enough and w_2 to be near to 1 in Eq. 4 for initialization.

$$\delta(x) = \frac{c}{\sqrt{N_x}} w_1 x^2 + w_2 x + b \tag{4}$$

Convergence. Layer-wise second-order polynomial activation functions preserve the convexity of single-layer neural network [17]. Higher order polynomial activation function or channel-wise fine-grained polynomial replacement proposed in SAFENet [11] may destroy the neural network's convexity and lead to a deteriorated performance.

Learning rate. The gradient of w_1 must be balanced to match the update speed of other model weights. As such, we add a new scaling $\frac{c}{\sqrt{N_x}}$ prior to w_1 parameter. In the function, c is a constant, N_x is the number of elements in feature map.

B. *Search Space of Hardware-aware NAS.*

We focus on convolutional neural networks (CNNs) in our study. CNNs are mostly composed of Conv-Act-Pool and Conv-Act blocks. In work, we use the regular backbone model as a search baseline, such as the VGG family, mobilenetV3, and ResNet family. Each layer of supernet is composed of the layer structure obtained from baseline and its possible combination with X^2act and $Pool_a$ replacement. A toy example is shown in Fig. 3, where a two-layer supernet is constructed, and the first layer is Conv-Act-Pool, and the second layer is Conv-Act. The first layer has four combinations which are Conv-ReLU-Pool_m, Conv-ReLU-Pool_a, Conv-X^2act-Pool_m, and Conv-X^2act-Pool_a. The second layer has two combinations: Conv-ReLU and Conv-X^2act. The Conv block's parameters can be either shared among candidates or separately trained during the search.

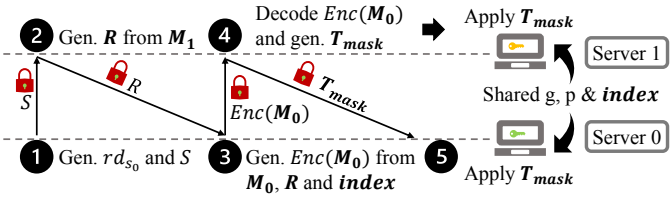


Fig. 4: Processing Steps of **2PC-OT** flow.

C. *Operator Modeling and Latency Analysis*

This section will analyze five different operators: 2PC-ReLU, 2PC-X^2act, 2PC-MaxPool, 2PC-AvgPool, and 2PC-Conv. Therefore, they require $(1, n)$-OT (noted as **OT flow** block to implement 2PC comparison flows. Batch normalization can be fused into the convolution layer and it's not listed.

1) 2PC-OT Processing Flow: While OT-based comparison protocol has been discussed in [15], we hereby provide other communication detail as shown in Fig. 4. Assume both servers have a shared prime number m, one generator (g) selected from the finite space $\mathbb{Z}_m$, and an **index** list with L length. As we adopt 2-bit part, the length of **index** list is $L = 4$.

① **Server 0** (S_0) generates a random integer rd_{s_0}, and compute mask number S with $S = g^{rd_{S_0}} \ mod \ m$, then shares

S with the Server 1 (S_1). We only need to consider communication ($COMM_1$) latency as $COMM_1 = T_{bc} + \frac{32}{Rt_{bw}}$, since computation ($CMP_1$) latency is trivial.

② **Server 1** (S_1) received S, and generates $\boldsymbol{R}$ list based on S_1's 32-bit dataset $\boldsymbol{M_1}$, and then send them to S_0. Each element of $\boldsymbol{M_1}$ is split into $U = 16$ parts, thus each part is with 2 bits. Assuming the input feature is square with size FI and IC denotes the input channel, and we denote the computational parallelism as PP. The CMP_2 is modeled as Eq. 5, and $COMM_2$ is modeled as Eq. 6.

$$CMP_2 = \frac{32 \times 17 \times FI^2 \times IC}{PP \times freq} \tag{5}$$

$$COMM_2 = T_{bc} + \frac{32 \times 16 \times FI^2 \times IC}{Rt_{bw}} \tag{6}$$

③ **Server 0** (S_0) received $\boldsymbol{R}$, it will first generate the encryption $\boldsymbol{key_0}(y,u) = \boldsymbol{R}(y,u) \oplus (S^{b2d(\boldsymbol{M_1}(y,u))+1} \ mod \ m)^{rd_{S_0}} \ mod \ m$. The S_0 also generates is comparison matrix for it's $\boldsymbol{M_0}$ with 32-bit datatype and $U = 16$ parts, thus the matrix size for each value (x) is 4×16. The encrypted $Enc(\boldsymbol{M_0}(x,u)) = \boldsymbol{M_0}(x,u) \oplus \boldsymbol{key_0}(y,u)$ will be sent to S_1. The $COMM_3$ of this step is shown in Eq. 8, and CMP_3 can be estimated as Eq. 7.

$$CMP_3 = \frac{32 \times (17 + (4 \times 16)) \times FI^2 \times IC}{PP \times freq} \tag{7}$$

$$COMM_3 = T_{bc} + \frac{32 \times 4 \times 16 \times FI^2 \times IC}{Rt_{bw}} \tag{8}$$

④ **Server 1** (S_1) decodes the interested encrypted massage by $\boldsymbol{key_1} = S^{rd_{S_0}} \ mod \ m$ in the final step. The CMP_4 and $COMM_4$ are calculated as following:

$$CMP_4 = \frac{((32 \times 4 \times 16) + 1) \times FI^2 \times IC}{PP \times freq} \tag{9}$$

$$COMM_4 = T_{bc} + \frac{FI^2 \times IC}{Rt_{bw}} \tag{10}$$

2) 2PC-ReLU Operator: 2PC-ReLU requires 2PC-OT flow. 2PC-ReLU latency ($Lat_{2PC-ReLu}$) model is given in Eq. 11.

$$Lat_{2PC-ReLu} = \sum_{i=2}^{4} CMP_i + \sum_{j=1}^{4} COMM_j \tag{11}$$

3) 2PC-MaxPool Operator: Original MaxPool function is shown in Eq. 12. The 2PC-MaxPool uses OT flow comparison, and the latency model is shown in Eq. 13.

$$out = \max_{\substack{k_h \in [0, K_h - 1] \\ k_w \in [0, K_w - 1]}} in(n, c, hS_h + k_h, wS_w + k_w) \tag{12}$$

$$Lat_{2PC-MaxPool} = \sum_{i=2}^{4} CMP_i + \sum_{j=1}^{4} COMM_j + 3T_{bc} \tag{13}$$

4) 2PC-X^2act Operator: The original X^2act has been shown in Eq. 4. The X^2act needs a ciphertext square operation and 2 ciphertext-plaintext multiplication operations. The basic protocol is demonstrated in Sec. II-B. The latency of computation and communication can be modeled as: $CMP_{x^2} = \frac{2 \times FI^2 \times IC}{PP \times freq}$ and $COMM_{x^2} = T_{bc} + \frac{32 \times FI^2 \times IC}{Rt_{bw}}$. The latency model of 2PC-X^2act (Lat_{2PC-X^2act}) is shown in Eq. 14.

$$Lat_{2PC-X^2act} = CMP_{x^2} + 2 \times COMM_{x^2} \tag{14}$$

5) 2PC-AvgPool Operator: The 2PC-AvgPool operator only involves addition and scaling, the latency is

$$Lat_{2PC-AvgPool} = \frac{2 \times FI^2 \times IC}{PP \times freq} \tag{15}$$

6) 2PC-Conv Operator: The 2PC-Conv operator involves multiplication between ciphertext, and the basic computation and communication pattern are given Sec. II-B. The computation part follows tiled architecture implementation [18]. Assuming we can meet the computation roof by adjusting tiling parameters, the latency of the 2PC-Conv computation part can be estimated as $CMP_{Conv} = \frac{3 \times K \times K \times FO^2 \times IC \times OC}{PP \times freq}$, where K is the convolution kernel size. The communication latency is modeled as $COMM_{Conv} = T_{bc} + \frac{32 \times FI^2 \times IC}{Rt_{bw}}$. Thus, the latency of 2PC-Conv is given in Eq. 16.

$$Lat_{2PC-Conv} = CMP_{Conv} + 2 \times COMM_{Conv} \tag{16}$$

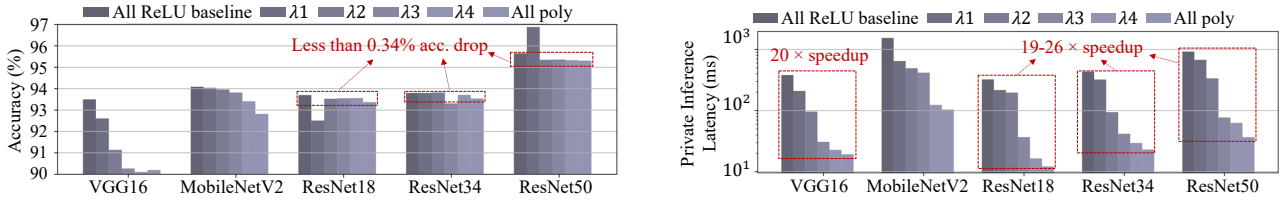
D. *Differentiable Harware Aware NAS Algorithm*

Algorithm 1 Differentiable Polynomial Architecture Search.

Input: M_b: backbone model; D: a specific dataset
$Lat(OP)$: latency loop up table; H: hardware resource
Output: Searched polynomial model M_p

while not converged **do**
  Sample minibatch x_{trn} and x_{val} from trn. and val. dataset
  // Update architecture parameter α:
  Forward path to compute $\zeta_{trn}(\omega, \alpha)$ based on x_{trn}
  Backward path to compute $\delta\omega = \frac{\partial \zeta_{trn}(\omega,\alpha)}{\partial \omega}$
  Virtual step to compute $\omega' = \omega - \xi\delta\omega$
  Forward path to compute $\zeta_{val}(\omega', \alpha)$ based on x_{val}
  Backward path to compute $\delta\alpha' = \frac{\partial \zeta_{val}(\omega',\alpha)}{\partial \alpha}$
  Backward path to compute $\delta\omega' = \frac{\partial \zeta_{val}(\omega',\alpha)}{\partial \omega'}$
  Virtual steps to compute $\omega^{\pm} = \omega \pm \varepsilon\delta\omega'$
  Two forward path to compute $\zeta_{trn}(\omega^{\pm}, \alpha)$
  Two backward path to compute $\delta\alpha^{\pm} = \frac{\partial \zeta_{trn}(\omega^{\pm},\alpha)}{\partial \alpha}$
  Compute hessian $\delta\alpha'' = \frac{\delta\alpha^{+} - \delta\alpha^{-}}{2\varepsilon}$
  Compute final architecture parameter gradient $\delta\alpha = \delta\alpha' - \xi\delta\alpha''$
  Update architecture parameter using $\delta\alpha$ with Adam optimizer

  // Update weight parameter ω:
  Forward path to compute $\zeta_{trn}(\omega, \alpha)$ based on x_{trn}
  Backward path to compute $\delta\omega = \frac{\partial \zeta_{trn}(\omega,\alpha)}{\partial \omega}$
  Update architecture parameter using $\delta\omega$ with SGD optimizer
end while
Obtain architecture by $OP_l(x) = OP_{l,k^*}(x)$, $s.t.\ k^* = \text{argmax}_k\ \theta_{l,k}$



(a) Searched model accuracy comparison

(b) Searched model private inference latency comparison

Fig. 5: PASNet framework evaluation on CIFAR-10 dataset under 2PC PI setup. Network banwidth: 1 GB/s. Device: ZCU104.

Early work [19] focus on using RL for NAS. The RL based method effectively explores the search space but still requires a significant amount of search overhead such as GPU hours and energy. Hardware-aware NAS have also been investigated [20]. In this work, we incorporate latency constraint into the target loss function of the DARTS framework [21], and develop a differentiable cryptographic hardware-aware micro-architecture search framework. We firstly determine a supernet model for NAS, and introduces gated operators $OP_l(x)$ which parametrizes the candidate operators $OP_{l,j}(x)$ selection with a trainable weight $\alpha_{l,k}$ (Eq. 17). For example, a gated pooling operator consists of MaxPool and AvgPool operators and 2 trainable parameters for pooling selection. The latency of the operators could be determined based on Sec. III-C. A parameterized latency constraint is given as $Lat(\alpha) = \sum_{l=1}^{n}\sum_{j=1}^{m}\theta_{l,j}Lat(OP_{l,j})$, where the latency of gated operators are weighted by $\theta_{l,j}$. We incorporate the latency constraint into the loss function as $\zeta(\omega,\alpha) = \zeta_{CE}(\omega,\alpha) + \lambda Lat(\alpha)$, and penalize the latency $Lat(\alpha)$ by λ.

$$\theta_{l,j} = \frac{\exp(\alpha_{l,j})}{\sum_{k=1}^{m}\exp(\alpha_{l,k})},\ OP_l(x) = \sum_{k=1}^{m}\theta_{l,k}OP_{l,k}(x) \quad (17)$$

The optimization objective of our design is shown in Eq. 18, we aim to minimize the validation loss $\zeta_{val}(\omega^*,\alpha)$ with regard to architecture parameter α. The optimal weight ω^* is obtained through minimize the training loss. The second order approximation of the optimal weight is given as $\omega^* \approx \omega' = \omega - \xi\,\delta\zeta_{trn}(\omega,\alpha)/\delta\omega$, the approximation is based on current weight parameter and its' gradient. The virtual learning rate ξ can be set equal to that of weight optimizer.

$$\text{argmin}_{\alpha}\ \zeta_{val}(\omega^*,\alpha),\ s.t.\ \omega^* = \text{argmin}_{\omega}\ \zeta_{trn}(\omega,\alpha) \quad (18)$$

Eq. 19 gives the approximate α gradient using chain rule, the second term of α gradient can be further approximated using small turbulence ε, where weights are $\omega^{\pm} = \omega \pm \varepsilon\,\delta\zeta_{val}(\omega',\alpha)/\delta\omega'$ and Eq. 20 is used for final α gradient.

$$\delta\zeta_{val}(\omega',\alpha)/\delta\alpha - \xi\delta\zeta_{val}(\omega',\alpha)/\delta\omega'\delta\delta\zeta_{trn}(\omega,\alpha)/\delta\omega\delta\alpha \quad (19)$$

$$\frac{\delta\delta\zeta_{trn}(\omega,\alpha)}{\delta\omega\delta\alpha} = \delta(\zeta_{trn}(\omega^+,\alpha) - \zeta_{trn}(\omega^-,\alpha))/(2\varepsilon\delta\alpha) \quad (20)$$

With the help of analytical modeling of optimization objective, we are able to derive the differentiable polynomial architecture search framework in Algo. 1. The input of search framework includes backbone model M_b, dataset D, latency loop up table $Lat(OP)$, and hardware resource H. The algorithm returns a searched polynomial model M_p. The algorithm iteratively trains the architecture parameter α and weight ω parameter till the convergence. Each α update requires 4 forward paths and 5 backward paths according to Eq. 18 to Eq. 20, and each ω update needs 1 forward paths and 1 backward paths. After the convergence of training loop, the algorithm returns a deterministic model architecture by applying $OP_l(x) = OP_{l,k^*}(x),\ s.t.\ k^* = \text{argmax}_k\ \alpha_{l,k}$. The returned architecture is then used for 2PC based PI evaluation.

IV. Evaluation

Hardware setup. Our platform uses two ZCU104 MPSoCs connected via a 1 GB/s LAN router. With a 128-bit load/store bus and 32-bit data, we process four data simultaneously at 200MHz. The fixed point ring size is set to 32 bits for PI.

Datasets and Backbone Models. PASNet is evaluated on CIFAR-10 and ImageNet for image classification tasks. CIFAR-10 [22] has colored 32×32 images, with 10 classes, $50,000$ training, and $10,000$ validation images. ImageNet [22] has RGB 224×224 images, with 1000 categories, 1.2 million training, and $50,000$ validation images.

Systems Setup. Polynomial architecture search experiments are conducted using Ubuntu 18.04, Nvidia Quadro RTX 6000 GPU, PyTorch v1.8.1, and Python 3.9.7. Pretrained weights for CIFAR-10 and ImageNet are from [23] and Pytorch Hub [24], respectively. Cryptographic DNN inference is performed on FPGA-based accelerators using two ZCU104 boards, connected via Ethernet LAN. The FPGA accelerators are optimized with coarse-grained and fine-grained pipeline structures, as discussed in Sec. III-C.

A. *Hardware-aware NAS Evaluation*

Our hardware-aware PASNet evaluation experiment (algorithm descripted in Sec. III-D) was conducted on CIFAR-10 training dataset. A new training & validation dataset is randomly sampled from the CIFAR-10 training dataset with 50%-50% split ratio. The new training dataset is used to update the weight parameter of PASNet models, and the new validation dataset is used to update the architecture parameter.

TABLE I: PASNet evaluation & cross-work comparison with CryptGPU [13] and CryptFLOW [1]. Batch size = 1

	CIFAR-10 dataset				ImageNet dataset				
Model	Top 1 (%)	Lat. (ms)	Comm. (MB)	Effi. (1/(ms*kW))	Top 1 (%)	Top 5 (%)	Lat. (s)	Comm. (GB)	Effi. (1/(s*kW))
PASNet-A	93.37	12.2	2.86	5.12	70.54	89.59	0.063	0.035	999
PASNet-B	95.31	36.74	13.18	1.70	78.79	93.99	0.228	0.162	274
PASNet-C	95.33	62.91	30.03	0.99	79.25	94.38	0.539	0.368	115
PASNet-D	92.82	104.09	25.01	0.60	71.36	90.15	0.184	0.103	339
CryptGPU ResNet50	\	\	\	\	78	92	9.31	3.08	0.15
CryptFLOW ResNet50	\	\	\	\	76.45	93.23	25.9	6.9	0.096

The hardware latency is modeled through section. III-C, and the λ for latency constraint in loss function is tuned to generate architectures with different latency-accuracy trade-off. Prior search starts, the major model parameters are randomly initialized and the polynomial activation function is initialized through **STPAI** method. We use VGG-16 [25], ResNet-18, ResNet-34, ResNet-50 [26], and MobileNetV2 [27] as backbone model structure to evaluate our PASNet framework.

With the increase of latency penalty, the searched structure's accuracy decreases since the DNN structure has more polynomial operators. After the proper model structure is found during architecture search process, the transfer learning with **STPAI** is conducted to evaluate the finetuned model accuracy.

The finetuned model accuracy under 2PC setting with regard to λ setting can be found in Fig. 5(a). The baseline model with all ReLU setting and all-polynomial operation based model are also included in the figure for comparison. Generally, a higher polynomial replacement ratio leads to a lower accuracy. The VGG-16 model is the most vulnerable model in the study, while the complete polynomial replacement leads to a 3.2% accuracy degradation (baseline 93.5%). On the other side, ResNet family are very robust to full polynomial replacement and there are only 0.26% to 0.34% accuracy drop for ResNet-18 (baseline 93.7%), ResNet-34 (baseline 93.8%) and ResNet-50 (baseline 95.6%). MobileNetV2's is in between the performance of VGG and ResNet, in which a full polynomial replacement leads to 1.27% degradation (baseline 94.09%).

On the other hand, Fig. 5(b) presents the latency profiling result of searched models performance on CIFAR-10 dataset under 2PC setting. All polynomial replacement leads to 20 times speedup on VGG-16 (baseline 382 ms), 15 times speedup on MobileNetV2 (baseline 1543 ms), 26 times speedup, ResNet-18 (baseline 324 ms), 19 times speedup on ResNet-34 (baseline 435 ms), and 25 times on speedup ResNet-50 (baseline 922 ms). With most strict constraint λ, the searched model latency is lower.

B. *Cross-work ReLU Reduction Performance Comparison*

A futher accuracy-ReLU count analysis is conducted and compared with SOTA works with ReLU reduction: DeepReDuce [7], DELPHI [10], CryptoNAS [8], and SNI [28]. As shown in Fig. 6, we generate the pareto frontier with best accuracy-ReLU count trade-off from our architecture search result. We name the selected models as **PASNet**, and compare it with other works. The accuracy-ReLU count comparison is show in Fig. 7. Our work achieves a much better accuracy vs. ReLU comparison than existing works, especially at the situation with extremely few ReLU counts.

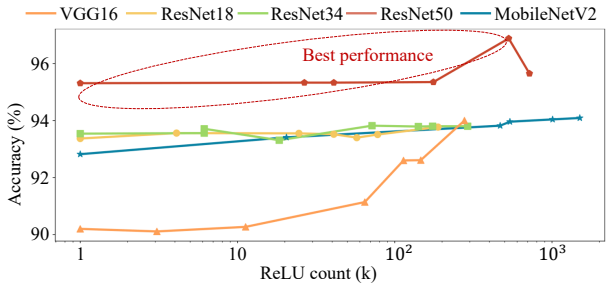


Fig. 6: Accuracy-ReLU count trade-off on CIFAR-10.

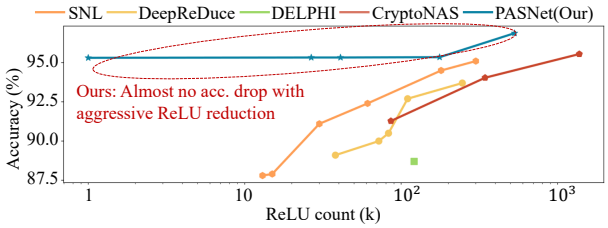


Fig. 7: ReLU reduction comparison on CIFAR-10.

C. *Cross-work PI System Performance Comparison*

We pick up 4 searched PASNet model variants for CIFAR-10 & ImageNet dataset accuracy & latency evaluation and name them as **PASNet-A**, **PASNet-B**, **PASNet-C**, **PASNet-D**. PASNet-A is a light-weighted model and shares the same backbone model as ResNet-18 but has only polynomial operators. PASNet-B and PASNet-C are heavily-weighted models that share the same backbone model as ResNet-50. PASNet-B has only polynomial operators and PASNet-C has 4 2PC-ReLU operators. PASNet-D is a medium-weighted model derived from MobileNetV2 with all polynomial layers. Note that the baseline top-1 accuracy of ResNet-18 on CIFAR-10 and ImageNet are 93.7% and 69.76%, baseline top-1 accuracy of ResNet-50 on CIFAR-10 and ImageNet are 95.65% and 78.8%, and the baseline top-1 accuracy of MobileNetV2 on CIFAR-10 and ImageNet are 94.09% and 71.88%.

The PASNet variants evaluation results and ImageNet cross-work comparison with SOTA CryptGPU [13] and CryptFLOW [1] implementation can be found in Tab. I. We observe a 0.78% top-1 accuracy increase for our light-weighted

PASNet-A compared to baseline ResNet-18 performance on ImageNet. Heavily-weighted models PASNet-B and PASNet-C achieve comparable (-0.01%) or even higher accuracy (+0.45%) than the ResNet-50 baseline. we achieve only a 0.13% accuracy drop for our medium-weighted PASNet-D compared to baseline MobileNetV2 performance on ImageNet. Even with the ZCU 104 edge devices setting, we can achieve a much faster secure inference latency than the SOTA works implemented on the large-scale server system. Our light-weighted PASNet-A achieves 147 times latency reduction and 88 times communication volume reduction compared to CryptGPU [13]. Our heavily-weighted model PASNet-B achieved 40 times latency reduction and 19 times communication volume reduction than CryptGPU [13] while maintaining an even higher accuracy. Our highest accuracy model PASNet-C achieved 79.25% top-1 accuracy on the ImageNet dataset with 17 times latency reduction and 8.3 times communication volume reduction than CryptGPU [13]. Note that our system is built upon the ZCU104 edge platform, so our energy efficiency is much higher (more than 1000 times) than SOTA CryptGPU [13] and CryptFLOW [1] systems.

V. Discussion

Existing MLaaS accelerations focused on plaintext inference acceleration [29]–[52]. Others target on plaintext training acceleration [53]–[63], federated learning [64]–[66] to protect the privacy of training data, and privacy protection of model vendor [67], [68].

In this work, we propose PASNet to reduce high comparison protocol overhead in 2PC-based privacy-preserving DL, enabling low latency, high energy efficiency, and accurate 2PC-DL. We employ hardware-aware NAS with latency modeling. Experiments demonstrate PASNet-A and PASNet-B achieve 147x and 40x speedup over SOTA CryptGPU on ImageNet PI test, with 70.54% and 78.79% accuracy.

Acknowledgement

This work was in part supported by the NSF CNS-2247891, 2247892, 2247893, CNS-2153690, DGE-2043183, and the Heterogeneous Accelerated Compute Clusters (HACC) program at UIUC. Any opinions, findings and conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

References

[1] Nishant Kumar, Mayank Rathee, Nishanth Chandran, Divya Gupta, Aseem Rastogi, and Rahul Sharma. Cryptflow: Secure tensorflow inference. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 336–353. IEEE, 2020.

[2] Brian Knott, Shobha Venkataraman, Awni Hannun, Shubho Sengupta, Mark Ibrahim, and Laurens van der Maaten. Crypten: Secure multi-party computation meets machine learning. *Advances in Neural Information Processing Systems*, 34:4961–4973, 2021.

[3] Miran Kim, Xiaoqian Jiang, Kristin Lauter, Elkhan Ismayilzada, and Shayan Shams. Secure human action recognition by encrypted neural network inference. *Nature Communications*, 13(1):4799, 2022.

[4] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Adaptively secure garbling with applications to one-time programs and secure outsourcing. In *Advances in Cryptology–ASIACRYPT 2012: 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings 18*, pages 134–153. Springer, 2012.

[5] Daniel Demmler, Thomas Schneider, and Michael Zohner. Aby-a framework for efficient mixed-protocol secure two-party computation. In *NDSS*, 2015.

[6] Juan Garay, Berry Schoenmakers, and José Villegas. Practical and secure solutions for integer comparison. In *Public Key Cryptography–PKC 2007: 10th International Conference on Practice and Theory in Public-Key Cryptography Beijing, China, April 16-20, 2007. Proceedings 10*, pages 330–342. Springer, 2007.

[7] Nandan Kumar Jha, Zahra Ghodsi, Siddharth Garg, and Brandon Reagen. Deepreduce: Relu reduction for fast private inference. In *International Conference on Machine Learning*, pages 4839–4849. PMLR, 2021.

[8] Zahra Ghodsi, Akshaj Kumar Veldanda, Brandon Reagen, and Siddharth Garg. Cryptonas: Private inference on a relu budget. *Advances in Neural Information Processing Systems*, 33:16961–16971, 2020.

[9] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International conference on machine learning*, pages 201–210. PMLR, 2016.

[10] Pratyush Mishra, Ryan Lehmkuhl, Akshayaram Srinivasan, Wenting Zheng, and Raluca Ada Popa. Delphi: a cryptographic inference system for neural networks. In *Proceedings of the 2020 Workshop on Privacy-Preserving Machine Learning in Practice*, pages 27–30, 2020.

[11] Qian Lou, Yilin Shen, Hongxia Jin, and Lei Jiang. Safenet: A secure, accurate and fast neural network inference. In *International Conference on Learning Representations*, 2021.

[12] Anshul Aggarwal, Trevor E Carlson, Reza Shokri, and Shruti Tople. Soteria: In search of efficient neural networks for private inference. *arXiv preprint arXiv:2007.12934*, 2020.

[13] Sijun Tan, Brian Knott, Yuan Tian, and David J Wu. Cryptgpu: Fast privacy-preserving machine learning on the gpu. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1021–1038. IEEE, 2021.

[14] Jeff Barnes. Azure machine learning. *Microsoft Azure Essentials. 1st ed, Microsoft*, 2015.

[15] Joe Kilian. Founding crytpography on oblivious transfer. In *STOC*, pages 20–31, 1988.

[16] Donald Beaver. Efficient multiparty protocols using circuit randomization. In *Crypto*, 1991.

[17] Sarath Sivaprasad, Ankur Singh, Naresh Manwani, and Vineet Gandhi. The curious case of convex neural networks. In *Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part I 21*, pages 738–754. Springer, 2021.

[18] Chen Zhang, Peng Li, Guangyu Sun, Yijin Guan, Bingjun Xiao, and Jason Cong. Optimizing fpga-based accelerator design for deep convolutional neural networks. In *Proceedings of the 2015 ACM/SIGDA international symposium on field-programmable gate arrays*, pages 161–170, 2015.

[19] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[20] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10734–10742, 2019.

[21] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[23] Huy Phan. huyvnphan/pytorch_cifar10, January 2021.

[24] Pytorch. Pytorch hub. https://pytorch.org/hub/research-models.

[25] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint*, 2014.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[27] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[28] Minsu Cho, Ameya Joshi, Brandon Reagen, Siddharth Garg, and Chinmay Hegde. Selective network linearization for efficient private inference. In *International Conference on Machine Learning*, pages 3947–3961. PMLR, 2022.

[29] Hongwu Peng, Shaoyi Huang, Shiyang Chen, Bingbing Li, Tong Geng, Ang Li, Weiwen Jiang, Wujie Wen, Jinbo Bi, Hang Liu, et al. A length adaptive algorithm-hardware co-design of transformer on fpga through sparse attention and dynamic pipelining. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pages 1135–1140, 2022.

[30] Shaoyi Huang, Dongkuan Xu, Ian EH Yen, Yijue Wang, Sung-En Chang, Bingbing Li, Shiyang Chen, Mimi Xie, Sanguthevar Rajasekaran, Hang Liu, et al. Sparse progressive distillation: Resolving overfitting under pretrain-and-finetune paradigm. *arXiv preprint arXiv:2110.08190*, 2021.

[31] Lei Zhang, Jie Zhang, Bowen Lei, Subhabrata Mukherjee, Xiang Pan, Bo Zhao, Caiwen Ding, Yao Li, and Dongkuan Xu. Accelerating dataset distillation via model augmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11950–11959, 2023.

[32] Yanfu Zhang, Runxue Bao, Jian Pei, and Heng Huang. Toward unified data and algorithm fairness via adversarial data augmentation and adaptive model fine-tuning. In *2022 IEEE International Conference on Data Mining (ICDM)*, pages 1317–1322. IEEE, 2022.

[33] Yawen Wu, Zhepeng Wang, Zhenge Jia, Yiyu Shi, and Jingtong Hu. Intermittent inference with nonuniformly compressed multi-exit neural network for energy harvesting powered devices. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020.

[34] Runxue Bao, Bin Gu, and Heng Huang. Efficient approximate solution path algorithm for order weight l_1-norm with accuracy guarantee. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 958–963. IEEE, 2019.

[35] Hongwu Peng, Deniz Gurevin, Shaoyi Huang, Tong Geng, Weiwen Jiang, Orner Khan, and Caiwen Ding. Towards sparsification of graph neural networks. In *2022 IEEE 40th International Conference on Computer Design (ICCD)*, pages 272–279. IEEE, 2022.

[36] Xuan Kan, Hejie Cui, and Carl Yang. Zero-shot scene graph relation prediction through commonsense knowledge integration. In *Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part II 21*, pages 466–482. Springer, 2021.

[37] Yixuan Luo, Payman Behnam, Kiran Thorat, Zhuo Liu, Hongwu Peng, Shaoyi Huang, Shu Zhou, Omer Khan, Alexey Tumanov, Caiwen Ding, et al. Codg-reram: An algorithm-hardware co-design to accelerate semi-structured gnns on reram. In *2022 IEEE 40th International Conference on Computer Design (ICCD)*, pages 280–289. IEEE, 2022.

[38] Runxue Bao, Bin Gu, and Heng Huang. Fast oscar and owl regression via safe screening rules. In *International Conference on Machine Learning*, pages 653–663. PMLR, 2020.

[39] Yuhong Li, Tianle Cai, Yi Zhang, Deming Chen, and Debadeepta Dey. What makes convolutional models great on long sequence modeling? *arXiv preprint arXiv:2210.09298*, 2022.

[40] Hongwu Peng, Shaoyi Huang, Tong Geng, Ang Li, Weiwen Jiang, Hang Liu, Shusen Wang, and Caiwen Ding. Accelerating transformer-based deep learning models on fpgas using column balanced block pruning. In *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, pages 142–148. IEEE, 2021.

[41] Xuan Kan, Wei Dai, Hejie Cui, Zilong Zhang, Ying Guo, and Carl Yang. Brain network transformer. *arXiv preprint arXiv:2210.06681*, 2022.

[42] Panjie Qi, Yuhong Song, Hongwu Peng, Shaoyi Huang, Qingfeng Zhuge, and Edwin Hsing-Mean Sha. Accommodating transformer onto fpga: Coupling the balanced model compression and fpga-implementation optimization. In *Proceedings of the 2021 on Great Lakes Symposium on VLSI*, pages 163–168, 2021.

[43] Xia Xiao, Zigeng Wang, and Sanguthevar Rajasekaran. Autoprune: Automatic network pruning by regularizing auxiliary parameters. *Advances in neural information processing systems*, 32, 2019.

[44] Hongwu Peng, Shanglin Zhou, Scott Weitze, Jiaxin Li, Sahidul Islam, Tong Geng, Ang Li, Wei Zhang, Minghu Song, Mimi Xie, et al. Binary complex neural network acceleration on fpga. In *2021 IEEE 32nd International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pages 85–92. IEEE, 2021.

[45] Zhepeng Wang, Yawen Wu, Zhenge Jia, Yiyu Shi, and Jingtong Hu. Lightweight run-time working memory compression for deployment of deep neural networks on resource-constrained mcus. In *Proceedings of the 26th Asia and South Pacific Design Automation Conference*, pages 607–614, 2021.

[46] Shaoyi Huang, Shiyang Chen, Hongwu Peng, Daniel Manu, Zhenglun Kong, Geng Yuan, Lei Yang, Shusen Wang, Hang Liu, and Caiwen Ding. Hmc-tran: A tensor-core inspired hierarchical model compression for transformer-based dnns on gpu. In *Proceedings of the 2021 on Great Lakes Symposium on VLSI*, pages 169–174, 2021.

[47] Xuan Kan, Hejie Cui, Joshua Lukemire, Ying Guo, and Carl Yang. Fbnetgen: Task-aware gnn-based fmri analysis via functional brain network generation. In *International Conference on Medical Imaging with Deep Learning*, pages 618–637. PMLR, 2022.

[48] Xiaofan Zhang, Yuhong Li, Junhao Pan, and Deming Chen. Algorithm/accelerator co-design and co-search for edge ai. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 69(7):3064–3070, 2022.

[49] Panjie Qi, Edwin Hsing-Mean Sha, Qingfeng Zhuge, Hongwu Peng, Shaoyi Huang, Zhenglun Kong, Yuhong Song, and Bingbing Li. Accelerating framework of transformer by hardware design and model compression co-optimization. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pages 1–9. IEEE, 2021.

[50] Yi Sheng, Junhuan Yang, Yawen Wu, Kevin Mao, Yiyu Shi, Jingtong Hu, Weiwen Jiang, and Lei Yang. The larger the fairer? small neural networks can achieve fairness for edge devices. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pages 163–168, 2022.

[51] Yuhong Li, Cong Hao, Pan Li, Jinjun Xiong, and Deming Chen. Generic neural architecture search via regression. *Advances in Neural Information Processing Systems*, 34:20476–20490, 2021.

[52] Shaoyi Huang, Ning Liu, Yueying Liang, Hongwu Peng, Hongjia Li, Dongkuan Xu, Mimi Xie, and Caiwen Ding. An automatic and efficient bert pruning for edge ai systems. In *2022 23rd International Symposium on Quality Electronic Design (ISQED)*, pages 1–6. IEEE, 2022.

[53] Yawen Wu, Zhepeng Wang, Dewen Zeng, Meng Li, Yiyu Shi, and Jingtong Hu. Decentralized unsupervised learning of visual representations. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI*, pages 2326–2333, 2022.

[54] Shaoyi Huang, Haowen Fang, Kaleel Mahmood, Bowen Lei, Nuo Xu, Bin Lei, Yue Sun, Dongkuan Xu, Wujie Wen, and Caiwen Ding. Neurogenesis dynamics-inspired spiking neural network training acceleration. *arXiv preprint arXiv:2304.12214*, 2023.

[55] Yawen Wu, Zhepeng Wang, Yiyu Shi, and Jingtong Hu. Enabling on-device cnn training by self-supervised instance filtering and error map pruning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(11):3445–3457, 2020.

[56] Ran Xu, Yue Yu, Hejie Cui, Xuan Kan, Yanqiao Zhu, Joyce Ho, Chao Zhang, and Carl Yang. Neighborhood-regularized self-training for learning with few labels. *arXiv preprint arXiv:2301.03726*, 2023.

[57] Yawen Wu, Zhepeng Wang, Dewen Zeng, Yiyu Shi, and Jingtong Hu. Synthetic data can also teach: Synthesizing effective data for unsupervised visual representation learning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2023.

[58] Runxue Bao, Xidong Wu, Wenhan Xian, and Heng Huang. Doubly sparse asynchronous learning for stochastic composite optimization. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI*, pages 1916–1922, 2022.

[59] Yawen Wu, Zhepeng Wang, Dewen Zeng, Yiyu Shi, and Jingtong Hu. Enabling on-device self-supervised contrastive learning with selective data contrast. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 655–660. IEEE, 2021.

[60] Shaoyi Huang, Bowen Lei, Dongkuan Xu, Hongwu Peng, Yue Sun, Mimi Xie, and Caiwen Ding. Dynamic sparse training via balancing the exploration-exploitation trade-off. *arXiv preprint arXiv:2211.16667*, 2022.

[61] Yawen Wu, Dewen Zeng, Zhepeng Wang, Yiyu Shi, and Jingtong Hu. Distributed contrastive learning for medical image segmentation. *Medical Image Analysis*, 81:102564, 2022.

[62] Runxue Bao, Bin Gu, and Heng Huang. An accelerated doubly stochastic gradient method with faster explicit model identification. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 57–66, 2022.

[63] Bowen Lei, Dongkuan Xu, Ruqi Zhang, Shuren He, and Bani K Mallick. Balance is essence: Accelerating sparse training via adaptive gradient correction. *arXiv preprint arXiv:2301.03573*, 2023.

[64] Yawen Wu, Dewen Zeng, Zhepeng Wang, Yi Sheng, Lei Yang, Alaina J James, Yiyu Shi, and Jingtong Hu. Federated contrastive learning for dermatological disease diagnosis via on-device learning. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pages 1–7. IEEE, 2021.

[65] Yijue Wang, Jieren Deng, Dan Guo, Chenghong Wang, Xianrui Meng, Hang Liu, Chao Shang, Binghui Wang, Qin Cao, Caiwen Ding, et al. Variance of the gradient also matters: Privacy leakage from gradients. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2022.

[66] Yawen Wu, Dewen Zeng, Zhepeng Wang, Yiyu Shi, and Jingtong Hu. Federated contrastive learning for volumetric medical image segmentation. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2021: 24th International Conference, Strasbourg, France, September 27–October 1, 2021, Proceedings, Part III 24*, pages 367–377. Springer, 2021.

[67] Yijue Wang, Chenghong Wang, Zigeng Wang, Shanglin Zhou, Hang Liu, Jinbo Bi, Caiwen Ding, and Sanguthevar Rajasekaran. Against membership inference attack: Pruning is all you need. *arXiv preprint arXiv:2008.13578*, 2020.

[68] Yijue Wang, Nuo Xu, Shaoyi Huang, Kaleel Mahmood, Dan Guo, Caiwen Ding, Wujie Wen, and Sanguthevar Rajasekaran. Analyzing and defending against membership inference attacks in natural language processing classification. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 5823–5832. IEEE, 2022.