TAS: Ternarized Neural Architecture Search for Resource-Constrained Edge Devices

Mohammad Loni, Hamid Mousavi, Mohammad Riazati, Masoud Daneshtalab, and Mikael Sjödin
School of Innovation, Design and Engineering, Mälardalen University, Västerås, Sweden
Email: {mohammad.loni, mohammad.riazati, masoud.daneshtalab, mikael.sjodin}@mdh.se

***Abstract*—Ternary Neural Networks (TNNs) compress network weights and activation functions into 2-bit representation resulting in remarkable network compression and energy efficiency. However, there remains a significant gap in accuracy between TNNs and full-precision counterparts. Recent advances in Neural Architectures Search (NAS) promise opportunities in automated optimization for various deep learning tasks. Unfortunately, this area is unexplored for optimizing TNNs. This paper proposes TAS, a framework that drastically reduces the accuracy gap between TNNs and their full-precision counterparts by integrating quantization into the network design. We experienced that directly applying NAS to the ternary domain provides accuracy degradation as the search settings are customized for full-precision networks. To address this problem, we propose (i) a new cell template for ternary networks with maximum gradient propagation; and (ii) a novel learnable quantizer that adaptively relaxes the ternarization mechanism from the distribution of the weights and activation functions. Experimental results reveal that TAS delivers 2.64% higher accuracy and ≈2.8× memory saving over competing methods with the same bit-width resolution on the CIFAR-10 dataset. These results suggest that TAS is an effective method that paves the way for the efficient design of the next generation of quantized neural networks.**

***Index Terms*—Quantization, Ternary Neural Network, Neural Architecture Search, Embedded Systems**

I. Introduction

Convolutional Neural Networks (CNNs) have successfully been adapted to various computer vision tasks. In general, there is an increasing demand to deploy CNNs onto resource-constrained edge devices due to energy efficiency, privacy, and stable connectivity concerns [1]. However, the enormous computational intensity of CNNs cannot be supported by resource-constrained edge devices leading to failure of existing processing paradigms in affording modern application requirements. Network quantization is a widely used technique that significantly amortizes the computational burden of CNNs. A Ternary Neural Network (TNN) [2]–[6], where both weights and activation functions are quantized to ternary tensors, is a variation of network quantization techniques that comes with the benefits of network compression and operation acceleration. TNNs are reported to have up to 16× memory compression ratio [2] and 20× speed-up on FPGA [7], in comparison with full-precision networks. Furthermore, TNNs yield a better trade-off between accuracy and network size with 38× higher expressive abilities [2] compared to other counterparts such as binary networks [8]. However, TNNs still suffer from a substantial accuracy drop issue, hampering them from being widely used in practice (up to ≈17% accuracy drop compared to full-precision networks on ImageNet [8]).

As an orthogonal direction, there have been advances in Neural Architecture Search (NAS) where we can automatically design high-performance networks [1], [9]–[11]. Because of this insight, we came up with the idea of integrating the ternarization mechanism into NAS with the hope of reducing the accuracy gap of TNNs. However, applying the same NAS methods that have been used for full-precision networks to the ternary domain is practically inefficient. We hypothesize three reasons for this failure: (i) because of the particularities of the ternarization mechanism, certain network characteristics used in full-precision networks, such as separable convolutions, may be undesired; (ii) it inherits from the DARTS [9] cell template, where inefficient gradients propagation can result in increasing quantization error, and (iii) prior works have a naïve assumption on the distribution of the weights and activation functions, which makes the ternarization mechanism less effective with real-world applications.

To tackle these challenges, we propose TAS, a gradient-based NAS method that efficiently reduces the accuracy gap between ternary and full-precision architectures, yet at the same time providing a significant compression ratio by up to ≈45×. Although several variations of NAS have been developed for designing low-precision networks [10], [12], to the best of our knowledge, TAS is the first attempt in the literature that successfully develops a robust to quantization design methodology for TNNs. Our main contributions are summarized as follows:

1) We perform extensive experiments to identify limitations of applying existing NAS methods to the ternary domain.
2) We develop a search method that is robust to quantization error by adding inter-cell skip connections to the DARTS cell template to effectively convey gradients propagation.
3) We propose a novel optimizer that ternarizes weights and activation functions by relaxing the ternarization mechanism from the data distribution assumption.

TAS demonstrates its consistent effectiveness by achieving 2.64% accuracy improvement over counterparts with the same bit-width resolution on the CIFAR-10 [13] classification task. Meanwhile, TAS provides 2.84× memory saving on CIFAR-10 in comparison with other competing methods. Compared to full-precision ResNet-18 [14], TAS achieves up to 0.94% higher accuracy while also delivers up to 45.73× compression ratio. TAS provides the most accurate results on FPGA with a fair enough acceleration for most of the state-of-the-art CNNs. TAS generates similar results with STDEV=0.15% demonstrating our results are reproducible.

II. Related Work

In the following, we briefly discuss closely prior works that try to mitigate the ternarization error. Ternary neural networks (TNNs) provide a fair trade-off between accuracy and complexity [2], [8]. TWN [2], known as one of the earliest ternarization efforts, proposed $\{-\alpha, 0, +\alpha\}$ as quantization values to improve the accuracy of binary networks. TWN utilized two symmetric thresholds (Δ) alongside a scaling factor (α) for each layer to quantize weighs into $\{-\Delta, +\Delta\}$. However, there remains a significant performance gap between

TWN and the full-precision counterparts due to using symmetric thresholds. To solve this problem, TTQ uses two full-precision scaling factors (α_n, α_p) for positive and negative values [8]. TOT-Net [3] uses the same quantization method as TWN, but it learns scaling factors for each kernel. TRQ [4] claimed that the existing thresholding algorithms are not accurate enough to map the full-precision to ternary values. Therefore, TRQ introduced a recursive ternary quantization on full-precision weights for a refined reconstruction rather than directly thresholding. Although prior studies have improved the ternarization mechanism, they all have a naïve assumption on the distribution of the weights and activation functions. On the other hand, besides benefiting from TTQ and TOT-Net, TAS learns quantization thresholds jointly trained with the network parameters and is compatible with the distribution of each kernel's weights and activation functions.

III. Ternarized Neural Architecture Search

This section introduces our ternarized neural architecture search, named TAS, which designs ternary architectures that are robust to quantization error. In the following, we first motivate why we need specialized NAS for ternary networks. Afterward, we present our ternary quantization process and how we integrate it into the NAS procedure.

A. Specialized Ternary NAS: Motivation

Cell-based NAS method optimized by the gradient descent algorithm is highly popular in the community due to significantly reducing the computing cost of NAS methods [10]. Plus, the cell can be stacked any number of times to satisfy a given resource budget of an edge device. TAS utilizes a cell-based NAS method where the cell is ternarized in the first step (Section III-D). Then, the cell is optimized by an enhanced stochastic gradient descent (SGD) algorithm (Section III-E). Finally, we retrain the designed network from scratch to achieve maximum accuracy.

Using generic NAS methods for designing TNNs is not efficient. To demonstrate this claim, we add ternarized operations (using the TTQ ternarization mechanism [8]) to an existing NAS method. In this paper, we used DARTS [9] as the baseline NAS method. Then, we compare the performance of TAS with the ternarized DARTS (*DARTS+Ternarization*). Fig. 1 compares the learning curves of our proposed ternarized method (TAS) with ternarized DARTS on CIFAR-10 dataset. Disappointingly, the network designed by DARTS could not be trained properly, implying that full-precision NAS methods cannot trivially be extended to search ternary networks (TAS delivers 5.91% and 6.04% higher accuracy in 250 and 600 epochs, respectively). According to our investigations, we find three factors involved in the training failure of ternary DARTS: (i) the separable convolutional layer repetitively accumulates the ternarization error [10]; (ii) the cell template does not propagate the ternary gradients properly, leading to the vanishing gradient problem in the ternary domain; and (iii) existing ternarization mechanisms assume distribution of the weights is normal or uniform, which is a naïve assumption for real-world applications. Section III-B addresses the first issue, while the second and the third issues are addressed in Section III-C and Section III-D, respectively.

B. Search Space

Unlike full-precision NAS methods [1], [9], ternary NAS should leverage a search space that is robust to quantization error. Inspired by [10], we define a search space of different convolutional operations that are robust to quantization error. Note that although separable convolutions have been widely used in resource-efficient floating-point networks such as MobileNet, they are not suitable for ternary networks due to large quantization error [10]. Table I summarizes the operations of TAS search space.

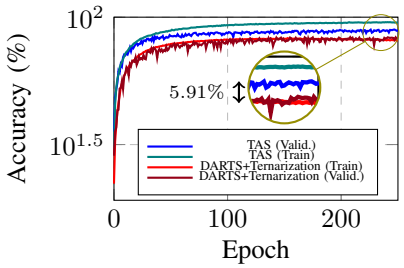

Fig. 1: Comparing train and test accuracies of the ternarized search using DARTS and TAS on CIFAR-10.

TABLE I: Operations of the TAS search space.

Operation Type	Ternary Convolution	Ternary Dilated Convolution	Max Pooling	Average pooling	Zeroise [9]
Kernel Size	3×3 5×5	3×3 5×5	3×3	3×3	N/A

C. Cell Template for TAS

The DARTS search space suffers from vanishing gradients for training ternarized networks since the skip-connections in the convolutional cell, i.e., intra-cell skip-connections, are limited to be inside a single cell. Thus, the intra-cell skip-connections do not properly propagate the gradients leading to aggregate outputs with quantization error inside the cell. To solve this problem, as inspired by ResNet [14], we add skip-connections between multiple cells (red lines in Fig. 2). This improves the gradients propagation between cells without accumulating quantization error, leading to improving training results (see the TAS curve in Fig. 1).

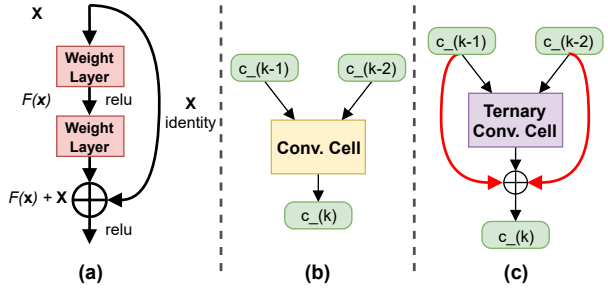

Fig. 2: Cell templates of (a) ResNet, (b) DARTS and (c) TAS. Conv. Cell indicates convolutional cell. `c_(k)` indicates the output of k^{th} cell.

D. New Ternary Quantization Method

In this section, we first propose a new ternarization mechanism. Then, we introduce new trainable parameters in the search objective and learn them jointly with the network parameters using SGD. For the sake of simplicity, we only present our ternarization mechanisms for the weights.

1) Ternary Weights: For ternary quantization, the full-precision weights $\mathbf{W}$ can be estimated with a non-negative scaling factor α and the ternary weights $\mathbf{W}^t$ as follows:

$$\mathbf{W} \approx \alpha \cdot \mathbf{W}^t \tag{1}$$

To build a high-performance TNN, prior works [2], [15] solve the optimization problem of minimizing the Euclidean distance between $\mathbf{W}$ and $\mathbf{W}^t$. Plus, [2] proposed a symmetric

threshold Δ for each layer to quantize weights, which sets the following constraint on ternary networks:

$$\mathbf{W}_i^t = f_t(\mathbf{W}_i|\Delta) = \begin{cases} +\alpha, & \text{if } \mathbf{W}_i > +\Delta \\ 0\ , & \text{if } |\mathbf{W}_i| \leq \Delta \\ -\alpha, & \text{if } \mathbf{W}_i < -\Delta \end{cases} \tag{2}$$

Based on Eq. 2, the ternary quantization problem reduces to calculating optimal α^* and Δ^*. In TWN [2], these optimal values are computed as follows:

$$\alpha_\Delta^* = \frac{1}{|\mathbf{I}_\Delta|}\sum_{i\in\mathbf{I}_\Delta}|\mathbf{W}_i|; \Delta^* = \underset{\Delta>0}{\text{argmax}}\frac{1}{|\mathbf{I}_\Delta|}(\sum_{i\in\mathbf{I}_\Delta}|\mathbf{W}_i|)^2 \tag{3}$$

where $\mathbf{I}_\Delta = \{i|\ |\mathbf{W}_i| > \Delta\}$. Δ^* in Eq. 3 has no straightforward solution. Prior works assume $\mathbf{W}_i$ is from uniform or normal distribution. If the $\mathbf{W}_i$ is distributed uniformly in $[-a, a]$, the approximated Δ^* is $\frac{2}{3}\cdot\mathbb{E}(|\mathbf{W}|)$. Thus, $\Delta^* \approx 0.7\cdot\mathbb{E}(|\mathbf{W}|)$ in the case of $\mathbf{W}_i$ is normally distributed.

To investigate the admissibility of the weights distribution assumption, Fig. 3.(a) shows the distribution of the weights for each layer of the AlexNet trained on CIFAR-10. According to our observations, the distribution of the weights in the last fully-connected layer is not symmetric and significantly deviates from the normal distribution (STDEV=6.9%), as depicted in Fig. 3.(b) (Q-Q plot). The same pattern for seventy networks trained on seven datasets is also observed in [16]. [17] also shows the weight distributions are different even for kernels in the same layer. Therefore, previous assumptions are not always valid and often leads to inefficient ternarization.

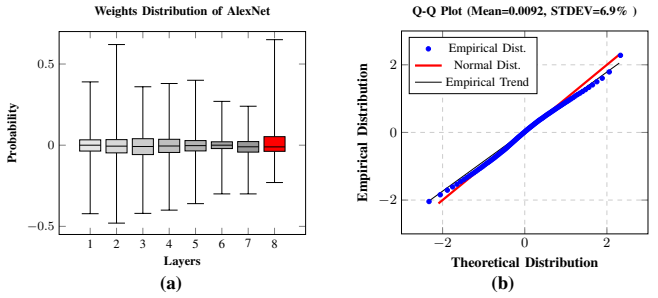

Fig. 3: (a) The distribution of the weights for each layer of AlexNet. The red box shows a non-uniform weights distribution. (b) Plotting the *Q-Q*-plot to pronounce the non-normality of the distribution of the weights of the last fully-connected layer of AlexNet (*Empirical data*).

To relax the distribution of the weights assumption, TAS uses a learning parameter δ_w and kernel-wise thresholding mechanism to compute Δ^* as follows:

$$\Delta^* = \delta_w \cdot \mathbb{E}(|\mathbf{W}|) = \frac{\delta_w}{n}\cdot\sum_{i=1}^{n}|\mathbf{W}_i| \tag{4}$$

where n is the total number of weights in a kernel. To show the expressive abilities of our method, we train a ternary AlexNet network based on the TTQ [8] in two scenarios with the proposed TAS thresholding and a fixed thresholding mechanism. Fig. 4 plots the validation loss values for these two scenarios. TAS mechanism improves the validation loss by 16.2%. The average δ_w for the kernels in the last fully-connected layer of AlexNet is 0.6 for our thresholding mechanism, which slightly differs from 0.7 that is fixed in other methods. However, using symmetric scaling factor α for both positive and negative quantization intervals causes a significant accuracy loss [5], [8]. In contrast, TAS use two trainable asymmetric scaling factors (α_n, α_p) in Eq. 2 for negative and positive values.

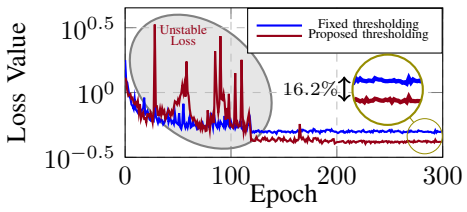

Fig. 4: Effect of optimizing the thresholding mechanism on AlexNet.

Following the chain rule for gradient descent, the derivatives of loss w.r.t α_n, α_p, and δ_w are computed as follows:

$$\begin{aligned} \frac{\partial L}{\partial \alpha_p} &= \sum_{j\in I_i^p}\frac{\partial L}{\partial \mathbf{W}_i^t(j)}, \frac{\partial L}{\partial \alpha_n} = \sum_{j\in I_i^n}\frac{\partial L}{\partial \mathbf{W}_i^t(j)} \\ \frac{\partial L}{\partial \delta_w} &= (\sum_{j\in I_i^p, k\in I_i^n}\frac{\partial L}{\partial \mathbf{W}_i^t(j)} + \frac{\partial L}{\partial \mathbf{W}_i^t(k)})\cdot(\sum_{i=1}^{n}|\mathbf{W}_i|/\delta_w) \end{aligned} \tag{5}$$

where $I_i^p = \{j|\mathbf{W}_i(j) > +\Delta^*\}$ and $I_i^n = \{j|\mathbf{W}_i(j) < -\Delta^*\}$. The gradients of full-precision weights is calculated by using a scaled gradient for 32-bit weights by α_n and α_p.

E. Search Objective

The problem of ternary architecture search is defined as:

$$\gamma_t^* = \underset{\gamma_t\in A_t(S_t,T_t)}{\text{argmin}}\ \mathcal{L}_S(D;\boldsymbol{\theta}_{\gamma_t}) \tag{6}$$

where $\mathcal{L}_S(\cdot)$ is the search objective that includes cross-entropy loss and entropy-based regularizer to force selecting diverse operations [10]. A_t is the feasible set of ternary network architectures, T_t and S_t show TAS cell template and search space, respectively. $\boldsymbol{\theta}_{\gamma_t}$ indicates all trainable parameters of ternary network including $\{\mathbf{W}, \gamma_t, \delta_w, \delta_v, \alpha_p, \alpha_n\}$. γ_t^* is the best searched architecture parameters. The final network architecture is retrained to find the optimal weights. Note that the validation loss is unstable in early training epochs (is shown by a gray ellipse in Fig. 4) since increasing the number of learning parameters (δ_w, δ_v are proposed by TAS) can result in overfitting on training data in some epochs.

IV. Experiments

A. Experimental Setup

Details on Searching Networks. We split CIFAR-10 classification dataset [13] into 30k data points for training and 30k for validation. We train a network using SGD with 16 initial channels and eight cells for 50 epochs with a batch size of 32. Each cell consists of seven nodes equipped with a depth-wise concatenation operation as the output node. The convolutional operations follow the *ReLU+Convolution+Batch Normalization* order. The rest of the setup follows [9]. We use weight decay=3×10^{-4} and momentum=0.9 with initial learning rate of 0.025 using cosine annealing [18]. The search step takes ≈14 GPU hours on a single NVIDIA® RTX A4000.

Details on Training the Searched Networks. We split CIFAR-10 into 50k data points for training and 10k for validation. We train the best network for 600 epochs with a batch size of 128. We use the SGD algorithm with weight decay=3×10^{-6}, momentum=0.9, and the learning rate= 5×10^{-2} to 4×10^{-4} with one-cycle policy [19].

Details on FPGA Implementation. We utilize the Xilinx High-Level-Synthesis (HLS) tool to automatically deploy a TNN represented in high-level APIs, such as Keras, to FPGA. Unfortunately, HLS tools usually support OpenCL, C, or C++, while neural network designers typically use high-level APIs

to describe the network. To solve this issue, we leverage DeepHLS [20] conversion tool to convert Keras to ANSI C. DeepHLS also verifies conversion results. For the implementations using HLS, we use Xilinx ZCU104 (XCZU7EV) development board with 2-bit signed format.

Details on Architecture Configuration. We stack the best cell searched by TAS to build the final network. We have 18 normal cells with two reduction cells, where every six normal cells are followed by one reduction cell.

B. Results on CIFAR-10

Table II compares TAS against state-of-the-art ternarization approaches, binary NAS, and DARTS on CIFAR-10 dataset. Note that the compression ratio is determined by measuring memory utilization. Consider q is quantization resolution (q-bit) of layer l, L is the maximum number of layers in each network, $\#W_l$ and $\#W_l^t$ are the number of weights in layer l for full-precision (32-bit) and ternary networks, respectively. Hence, the compression ratio is expressed as $\frac{\sum_{l=1}^{L} \#W_l \times 32}{\sum_{l=1}^{L} \#W_l^t \times q}$. ResNet-18 [14] is selected as the compression ratio baseline in our experiments.

TAS significantly outperforms all existing methods in terms of accuracy and compression ratio. TAS obtains a 0.94% accuracy improvement with a 45.73× higher compression ratio compared to full-precision ResNet-18. In comparison with TRQ [4] with the same quantization resolution, TAS achieves 2.64% accuracy improvement with 2.84× higher compression ratio. We can see that trivially extending DARTS to design TNNs (*DARTS+Ternarization*) results in a 6.04% accuracy degradation. Compared to binary NAS [10], TAS provides 1.28% higher accuracy without compromising memory saving.

TABLE II: Comparing the TAS results with state-of-the-art methods on CIFAR-10 dataset.

Method (Backbone Arch.)	# bits (W/A)‡	Compression Ratio (×)†	Top-1 Accuracy (%)
Full-precision (ResNet-18) [14]	32/32	1	91.0
TBN (VGG-7) [6]	2/32	1.33	90.85
TWN (ResNet-18) [2]	2/32	16.06	92.56
TRQ (ResNet-18) [4]	2/2	16.06	89.3
Binary NAS (A) [10]	1/1	45.73	90.66∗
DARTS+Ternarization [9]	2/32	-	85.9
TAS (Ours)	2/2	**45.73**	**91.94**

† The baseline for comparing the compressing ratio is ResNet-18.
‡ (Weights/Activation Function).
∗ Experiments obtained by re-running the official implementation.

C. Results on FPGA Implementation

To evaluate the TAS performance on real hardware, we deploy the best TAS architecture for CIFAR-10 on FPGA (Table III). These results suggest that TAS is effective for accurate implementation of TNNs on FPGA by providing up to ≈2.8% higher accuracy. Note that we did not perform any optimization in the FPGA compilation level, and the utilized HLS tool is far slower than super optimized accelerators [21]. Nevertheless, TAS provides a fair enough acceleration for most existing CNNs (up to 90 frames-per-second throughput).

TABLE III: Comparing the FPGA implementation results of TAS with state-of-the-art methods on CIFAR-10.

Work (FPGA Device)	# bits (I/W)‡	Freq. (MHz)	Latency (ms)	Resource Utilization (%)	Accuracy (%) (Network)
T-DLA [7] (Zedboard)	8/2	125/250 (Logic/Adder)	2.188	LUT:78.71, FF:37.76 BRAM:75.0, DSP:49.55	89.08 (VGG-like)
[22] (VC709)	8/2	250	**0.036**	LUT:35.8 BRAM:23.1	86.71 (VGG-like)
TAS (XCZU7EV)	32/2	300	10.88	LUT:52, FF:8 BRAM:81, DSP:2	**91.94**

‡ (Input Image/Weights).

V. Conclusion

This paper proposed TAS, a variation of NAS for TNNs. TAS significantly improves the performance of TNNs by: (i) proposing a new cell template for ternary networks; and (ii) adaptively relaxing the network quantization process from the distribution assumption of weights and activation functions. According to experimental results, TAS yields 45× higher compression ratio while significantly reduces the accuracy gap between TNNs and full-precision counterparts on CIFAR-10.

References

[1] M. Loni, A. Zoljodi, A. Majd, B. H. Ahn, M. Daneshtalab, M. Sjödin, and H. Esmaeilzadeh, "Faststereonet: A fast neural architecture search for improving the inference of disparity estimation on resource-limited platforms," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2021.

[2] F. Li, B. Zhang, and B. Liu, "Ternary weight networks," *arXiv preprint arXiv:1605.04711*, 2016.

[3] N. Nazari, M. Loni, M. E. Salehi, M. Daneshtalab, and M. Sjodin, "Totnet: An endeavor toward optimizing ternary neural networks," in *2019 22nd Euromicro Conference on Digital System Design (DSD)*. IEEE, 2019, pp. 305–312.

[4] Y. Li, W. Ding, C. Liu, B. Zhang, and G. Guo, "Trq: Ternary neural networks with residual quantization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 10, 2021, pp. 8538–8546.

[5] D. Zhang, J. Yang, D. Ye, and G. Hua, "Lq-nets: Learned quantization for highly accurate and compact deep neural networks," in *Proceedings of the European conference on computer vision*, 2018, pp. 365–382.

[6] D. Wan, F. Shen, L. Liu, F. Zhu, J. Qin, L. Shao, and H. T. Shen, "Tbn: Convolutional neural network with ternary inputs and binary weights," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 315–332.

[7] Y. Chen, K. Zhang, C. Gong, C. Hao, X. Zhang, T. Li, and D. Chen, "T-dla: An open-source deep learning accelerator for ternarized dnn models on embedded fpga," in *2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2019, pp. 13–18.

[8] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," *arXiv preprint arXiv:1612.01064*, 2016.

[9] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," *arXiv preprint arXiv:1806.09055*, 2018.

[10] D. Kim, K. P. Singh, and J. Choi, "Learning architectures for binary networks," in *European Conference on Computer Vision*. Springer, 2020, pp. 575–591.

[11] A. Bulat, B. Martinez, and G. Tzimiropoulos, "Bats: Binary architecture search." Springer, 2020, pp. 309–325.

[12] H. Yu, Q. Han, J. Li, J. Shi, G. Cheng, and B. Fan, "Search what you want: Barrier panelty nas for mixed precision quantization," in *European Conference on Computer Vision*. Springer, 2020, pp. 1–16.

[13] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10 and cifar-100 datasets," *URl: https://www. cs. toronto. edu/kriz/cifar. html*, 2009.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[15] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European conference on computer vision*. Springer, 2016, pp. 525–542.

[16] M. Thoma, "Analysis and optimization of convolutional neural network architectures," *arXiv preprint arXiv:1707.09725*, 2017.

[17] W. Xu, X. He, T. Zhao, Q. Hu, P. Wang, and J. Cheng, "Soft threshold ternary networks," in *IJCAI*, 2020, pp. 2298–2304.

[18] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.

[19] L. N. Smith, "A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay," *arXiv preprint arXiv:1803.09820*, 2018.

[20] M. Riazati, M. Daneshtalab, M. Sjödin, and B. Lisper, "Deephls: A complete toolchain for automatic synthesis of deep neural networks to fpga," in *2020 27th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. IEEE, 2020, pp. 1–4.

[21] K. Guo, S. Zeng, J. Yu, Y. Wang, and H. Yang, "A survey of fpga-based neural network accelerator," *arXiv preprint arXiv:1712.08934*, 2017.

[22] A. Prost-Boucle, A. Bourge, F. Pétrot, H. Alemdar, N. Caldwell, and V. Leroy, "Scalable high-performance architecture for convolutional ternary neural networks on fpga," in *2017 27th International Conference on Field Programmable Logic and Applications*. IEEE, 2017, pp. 1–7.