Trainable Preprocessing for Reduced Precision Neural Networks

Gábor Csordás†, Kristof Denolf*, Nicholas Fraser*, Alessandro Pappalardo*, Kees Vissers*
*Xilinx Research Labs, USA and Ireland
{firstname.lastname}@xilinx.com
†EPFL, Switserland
{firstname.lastname}@epfl.ch

***Abstract*—Applications of neural networks are emerging in many fields and are frequently implemented in embedded environment, introducing power, throughput and latency constraints next to accuracy. Although practical computer vision solutions always involve some kind of preprocessing, most research focuses on the network itself. As a result, the preprocessing remains optimized for the human perception and is not tuned to neural networks. We propose the optimization of preprocesing along with the network using backpropagation and gradient descent. This open up the accuracy versus implementation cost design space towards more cost-efficient implementations by exploiting reduced precision input. In particular, we evaluate the effect of two preprocessing techniques: color conversion and dithering, using CIFAR10 and ImageNet datasets with different networks.**

***Index Terms*—Data Preprocessing, Quantized Neural Networks**

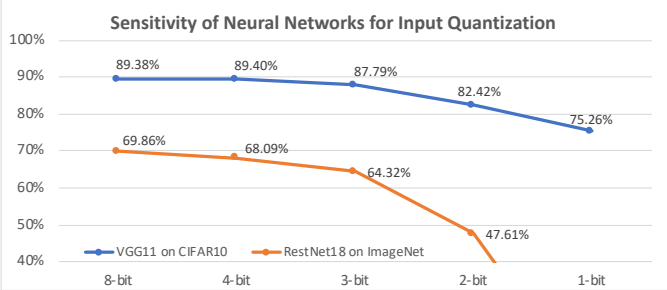

Fig. 1. The accuracy drops becomes significant when the precision of the input is less than 4 bit. The degradation is higher for ImageNet.

I. Introduction

Convolutional neural networks gained a lot of attention over the last decade thanks to their performance now exceeding humans for classification problems on the ImageNet challenge. Recent works, like EfficentNet [1] also carefully track the implementation cost of the network to enable their usage on embedded or edge devices imposing thermal and/or power constraints. This turns the development of the network into a two dimensional optimization problem of accuracy and energy efficiency. The goal is to find Pareto optimal implementations.

A well explored way of increasing energy efficiency is the reduction of the arithmetic precision. This not only simplifies the arithmetic units but it reduces the memory footprint and resulting bandwidth. In many cases 8-bit activations and weights are enough to achieve the accuracy of the equivalent 32-bit floating-point network [2]. More aggressive quantization often results in some accuracy degradation while further improving the accuracy versus hardware cost trade-offs. Binarized networks [3] push this to the extreme.

Quantization rarely extends to the input of the network because the input precision can strongly influence the accuracy of the neural network. Plotting the top-1 classification accuracy as a function of the input precision (Fig. 1) for a full precision VGG11 trained on CIFAR10 indicates a significant drop below 4-bits. For ImageNet the accuracy degradation is even larger and can already be seen using 4-bit input.

To avoid this accuracy drop, current reduced precision neural networks do not reduce the input precision, requiring a 'unique' first layer (compared to the remainder of the neural network layers). This results in a higher memory and compute cost of the first layer's processing core in case of a dataflow type implementation [4]. Larger networks are typically mapped to a homogeneous accelerator implementation with a single or multiple Deep learning Processing Unit(s) (DPUs), evaluating the network layer-by-layer [4]. The cost of the unique first layer is even higher in this case, since the DPU needs to support both 8-bit (first layer) and 4-bit input activations with 8-bit weights. This paper studies trainable preprocessing as technique to also reduce the precision of the input to the reduced precision neural network.

A. This Paper: Co-Optimization of Preprocessing

In practice the neural network is only a subpart of a complete computer vision pipeline. Raw data coming from an image sensors typically goes through image signal processing (ISP) [5], see Fig. 2, and these ISPs have traditionally been designed and configured for human perception. Neural networks might require different kind of processing or different parameters. In other words, the input data that can be the most easily classified with the human eye is not necessary the best for a (reduced precision) neural network.

We propose the optimization of preprocessing techniques inspired by traditional ISP along with the neural network using gradient descent and back-propagation and call them trainable preprocessing. The aim is to achieve implementations of neural networks that use the same reduced precision at the input as for the rest of the network and therefore consume less energy

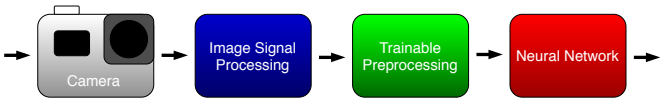

Fig. 2. A complete computer vision pipeline with camera, Image Signal Processor (ISP) and neural network. By adding trainable preprocessing, data can enter the neural network directly in a reduced precision format.

at the cost of an acceptable or no accuracy degradation. More specifically, this work focuses on color conversion and dithering. In the long term we plan to co-optimize the ISP or work directly on RAW data, but as the available labeled datasets contains already ISP-processed images, we add additional preprocessing and co-optimize it with the neural network to achieve more efficient implementations (Fig. 2).

Our contributions can be summarized as follows:

- We co-train preprocessing and neural networks.
- We train dither parameters with backpropagation.
- We exploit trainable preprocessing techniques to reduce input precision to reach more pareto optimal accuracy versus complexity (measured in Bit-Operations Performed (BOP) [6]) solutions.
- We conduct experiments on CIFAR10 and ImageNet.

All source code of our experiments is available (https://github.com/Xilinx/trainableProcessing) including the implementation of the preprocessing techniques, their training and the code of the networks with their training scripts and hyperparameters.

This paper is organized as follows. Section 2 reviews related work. Section 3 introduces the training of the two preprocessing techniques. Section 4 explains how we evaluate the proposed concept. In Section 5 we present our results. Sections 6 lists observations and presents future work. Finally, Section 7 draws the conclusions.

II. Related Work

EfficientNet [1] and others address the inference implementation cost by reducing the number of parameters without any accuracy degradation. An alternative method is reducing the precision of weights and activations. Ristretto [2] is an approximation framework for neural networks quantizing both weights and activations to 8-bit fixed point representation without losing significant accuracy. The usage of very reduced precision weights and activations has been studied extensively: binarized [3], QNN [7] and FINN [8]. These works only focus on improving the energy efficiency of the neural network and not on the optimization of the complete computer vision pipeline. A few studies analyze the importance of preprocessing in image classification. ColorNet [9] explores the effect of color spaces on classification accuracy. Kornia [10] provides a framework to insert image processing techniques into neural networks but does not analyze the optimization of preprocessing for neural networks. An adaptive method for dithering was proposed by Akarun et al. [11], but not for neural networks. The current paper trains parameters of conventional preprocessing techniques together with the network to improve the accuracy of reduced precision neural networks.

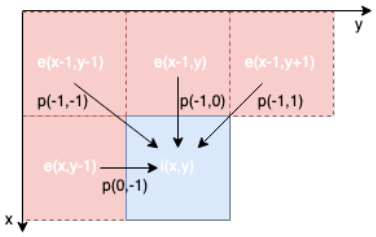

Fig. 3. Dithering diffusing the quantization error among the pixels. Pixels are processed in a sequential order: top left to bottom right.

III. Optimizing Preprocessing Just Like Other Layers of the Network

Preprocessing is an unconventional layer at the beginning of the network. Backpropagation of gradients can also be used to train parameters of preprocessing, if the preprocessing function is differentiable. Training calculates the derivative of the loss with respect to the preprocessing parameters:

$$\hat{y}_n = f(x_n, w, b, p) \tag{1}$$

$$\frac{\partial \ell(\hat{y}_n, y_n)}{\partial p} = \frac{\partial \ell(\hat{y_n}, y_n)}{\partial x_{preproc}} \frac{\partial x_{preproc}}{\partial p}, \tag{2}$$

where the output of the network ($\hat{y}_n$) is a function ($f(\cdot)$) of the original sample from the dataset (x_n), the parameters of the network (w,b) and the preprocessing parameters (p). To get the derivative of the loss with respect to the preprocessing parameters ($\frac{\partial \ell(\hat{y}_n, y_n)}{\partial p}$), first the derivative of the loss with respect to pixels of the preprocessed image has to be backpropagated ($\frac{\partial \ell(\hat{y}_n, y_n)}{\partial x_{preproc}}$, where $x_{preproc}$ is the preprocessed image), and secondly the derivative of the preprocessed pixels with respect to the preprocessing parameters ($\frac{\partial x_{preproc}}{\partial p}$) has to be determined.

A. Dithering

Dithering is an image processing technique for error diffusion. The human eye processes regions of images by averaging color information in the neighborhood of the focus. If the spatial resolution is high enough, a diffusion of the quantization error can balance the average color sensed by the eye, and create an illusion of using a contiguous color palette. 2D convolution, the major operation of CNNs, extracts new features locally with a fixed window size. Therefore, it is natural to expect that dithering can improve the quality of input images for neural networks as well. The quantization error depends on the input image. By diffusing the quantization error over the image, dithering allows the quantization to act as if it is independent of the input [12]. Dithering updates pixels with the weighted average of the quantization error of the adjacent pixels (Fig. 3) according to Eq. 3 for input pixel $i(x, y)$ with quantization error $e(i, j)$ of an adjacent pixel, dithering parameters $p(i, j)$. $i_d(x, y)$ is the updated value of the pixel, $Q(\cdot)$ is the quantization function and $i_q(x, y)$ is the final quantized value.

$$
\begin{aligned}
i_d(x,y) = i(x,y) &+ e(x-1,y-1)p(-1,-1) \\
&+ e(x-1,y)p(-1,0) + e(x-1,y+1)p(-1,1) \\
&+ e(x,y-1)p(0,-1)
\end{aligned} \tag{3}
$$

$$
i_q(x,y) = Q(i_d(x,y)), \tag{4}
$$

The most known dithering filter is Floyd-Steinberg (FS) [12] and was designed to produce quantized images that look nice for the human eye. Dithering parameters leading to highest accuracy are not necessary identical to those. Training dithering parameters requires that the derivative of the dithered image with respect to dithering parameters exists. However, there are two challenges in expressing this derivative.

Firstly, the derivative of our quantization function (where it exists) is zero. If this zero derivative is used to train parameters, the gradient of the loss is always zero and the parameters would not change. The Straight Through Estimation (STE) [13] solves this by estimating the derivative of the quantization function with the derivative of the identity function within the quantization range:

$$
\frac{\partial i_q(x,y)}{\partial i_d(x,y)} = 1 \tag{5}
$$

where $i_q(x,y)$ is the final quantized value of the pixel, while $i_d(x,y)$ is the input to the quantization function. Thus,

$$
\frac{\partial i_q(x,y)}{\partial p(i,j)} = \frac{\partial i_q(x,y)}{\partial i_d(x,y)} \frac{\partial i_d(x,y)}{\partial p(i,j)} = \frac{\partial i_d(x,y)}{\partial p(i,j)} \tag{6}
$$

where $p(i,j)$ is a trainable dithering parameter.

The second problem is the data dependency among pixels. A pixel can only be processed if the adjacent pixels it depends on have all been processed. This means that in Eq. 3 all quantization errors are influenced by the dithering parameters p. Unfortunately, to determine the derivative of a corrected pixel with respect to p the derivative of the quantization error of the adjacent pixels with respect to p has to be expressed.

$$
\frac{\partial e(i,j)}{\partial p(m,n)} = \frac{\partial i_d(i,j)}{\partial p(m,n)} - \frac{\partial i_q(i,j)}{\partial p(m,n)} = 0, \tag{7}
$$

However, due to the straight through estimation (Eq. 5) these derivatives are identical (Eq. 6) and the derivative of the error with respect to the parameter is zero. To sum up, applying STE for dithering breaks the chain of dependencies to get a simple derivative, that is used to calculate the gradient of the loss with respect to p:

$$
\frac{\partial \mathcal{L}}{\partial p(m,n)} = \sum_i \sum_j \frac{\partial \mathcal{L}}{\partial i_q(i,j)} e(i+m, j+n) \tag{8}
$$

B. Color Conversion

The human vision perceives a wide spectrum of colors using the additive RGB color model meaning that the human eye has receptors to measure the intensity of red, green and blue light. In image processing Color Conversions (CC) are used frequently because some color models fit better for certain algorithms than others. When inserting neural networks into computer vision pipeline usually the RGB representation is used, even though it might not be the best representation of an image for the neural network [9].

$$
\begin{pmatrix} img_x \\ img_y \\ img_z \end{pmatrix} = \begin{pmatrix} w_{xr} & w_{xg} & w_{xb} \\ w_{yr} & w_{yg} & w_{yb} \\ w_{zr} & w_{zg} & w_{zb} \end{pmatrix} \begin{pmatrix} img_r \\ img_g \\ img_b \end{pmatrix} \tag{9}
$$

Instead of directly quantizing the RGB images, conversion of the original RGB channels ($img_{r,g,b}$) is proposed with a linear transformation into three new channels $img_{x,y,z}$ (Eq. 9). The goal is to train the coefficients of this transformation to get a new representation of the original image that is less sensitive for quantization.

IV. Experimental Setup and Results

Preprocessing a reduced input for the (quantized) neural network impacts both accuracy and complexity. The baseline for accuracy is set to an reduced precision implementation using the original 8-bit inputs. Trained dithering is also compared to traditional FS dithering, using the same kernel size and shape. To include the reduced precision aspect of the operations, the complexity is evaluated with the Bit-Operations Performed (BOP) [6] metric. We implemented and trained our preprocessing layers as Pytorch modules (see the open source training script).

The experiments use both low resolution CIFAR10 and the large scale ImageNet datasets, with an input precision adjusted to the reduced precision of the activations resulting in an IxWyAz network: x is the bit width of the input, y is the bit width of the weights and z is the bit width of the activations. The CIFAR10 experiments run on CNV I1W1A1 [8] whereas ImageNet uses MobileNetV1 I4W4A4 [14]. To isolate the effect of the trainable preprocessing techniques from the quantization of the weights and activations, they are also evaluated on a floating-point network where only the input precision is reduced (VGG11 [15] on CIFAR10 and ResNet18 [16] on ImageNet).

Next subsections first introduce the extension to the BOP metric for the complexity evaluation, then results and finally the ablation study.

A. Computational complexity estimation

BOP_C in Eq. 10 is conv2d cost, with b_w weight bits, b_a activation bits, n input and m output channels, k is the size of filters. CC uses the same formula with $k = 1$.

$$
\begin{aligned}
BOP_c &= mnk^2(b_a b_w + b_a + b_w + \log_2(nk^2)) \\
BOP_d &= ml((b_a+1)b_p + 2b_a + 2 + b_p + \log_2(l))
\end{aligned} \tag{10}
$$

BOP_d extends the BOP metric to dithering with b_a dither input bits (typically 8), b_p dither parameters bits, m channels and l dither parameters. The accumulation is on the result of a subtraction, so $b_a + 1$ bits and we need to add the cost of the subtraction, in addition $b_a + 1$. The bitwidth of the preprocessing parameters was 8-bit in our experiments.

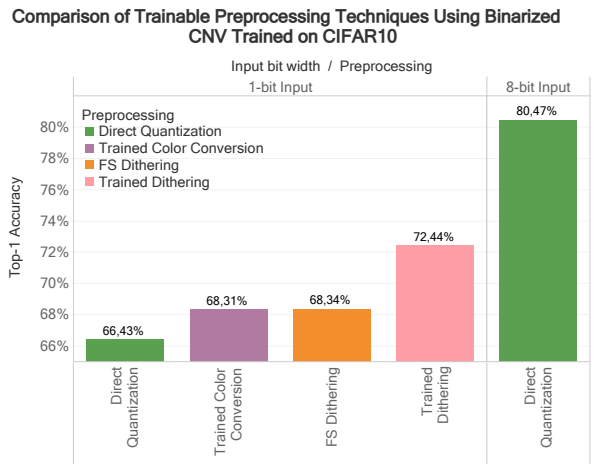

Fig. 4. Trained dithering outperforms FS dithering by far on a CNV W1A1 with CIFAR10.

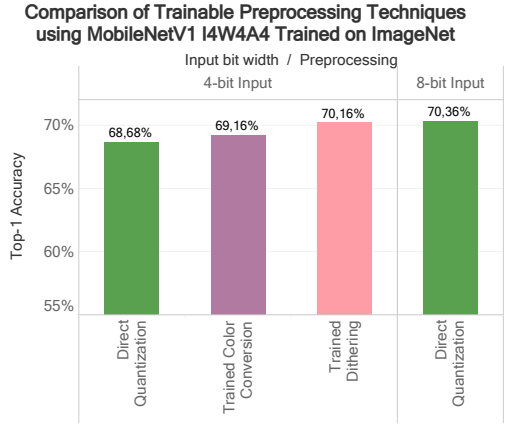

Fig. 5. 4-bit MobileNetV1 experiments on ImageNet recovering allmost all accuracy loss.

B. Results

On the fully binarized CNV W1A1 network (Fig. 4), direct quantization of the input to binary values leads to 14% accuracy degradation . FS dithering results regains 1.91% top-1 accuracy, CC 1.88%. Trained dithering outperforms all the other preprocessing techniques, 4.13% over FS, up to 6.01% improvement over direct quantization.

The W4A4 MobileNetV1 (Fig 5) experiments focus on this benefit of the trainable preprocessing and based on the results for CIFAR10 avoid FS dithering. While the trained CC already improves the accuracy over direct quantization, trained dithering is more effective: only 0.2% worse top-1 accuracy than using the 8-bit input for the W4A4 version.

Trained dithering achieves similar accuracy as FS but at a reduced complexity. Looking at the trained dither parameter values [1], the top-left values are zero (for all color channels),

[1]dither kernel parameters [p(-1,-1),p(-1,0),p(-1,1),p(0,-1)]: FS [0.0625, 0.3125,0.1875,0.4375]; trained R: [0,0.4192,0.0361,0.4453], G: [0,0.4161,0.0239,0.4463], B: [0,0.3899,0.0655,0.4463]

TABLE I
BOP COST OF 4-BIT MOBILENETV1'S FIRST LAYER.

	Preproc (BOPs)	Preproc + First layer (BOPs)
8-bit Input	-	50341
4-bit Direct Quant.	-	33061
4-bit Trained Color Conv.	763	33823
4-bit FS Dither.	1233	34294
4-bit Trained Dither.	925	33985

which is no the case for FS. This is exploited to reduce the complexity of the accelerator (Table I).

The complexity of first layer using reduced (4-bit) precision input is 34% lower (Table I) than the 8-bit variant. The cost of preprocessing is almost negligible compared to the cost of the first layer. Note that the complexity gain for a layer-by-layer solution would exceed this, since support the larger (8-bit) bitwidth of the input can completely be omitted from this type of accelerator.

C. Ablation Study

Fig. 6 shows the achieved top-1 accuracy of a floating-point VGG11 using direct quantization, trained CC, FS dithering and trained dithering when the input precision is reduced to 1,2 and 3-bit. The direct quantization of the input images to 1,2 and 3-bit increases the top-1 error by 14.12%, 6.96% and 1.59% respectively (similar to Fig. 1). FS dithering significantly improves this top-1 error. It reduces the accuracy gap between the binarized and the original 8 bit input to half (7.38% better than direct quantization). Training the dithering parameters outperforms FS dithering and leads to another 1.2% improvement. For 2 or 3-bit input images, FS dithering is similar to the trained one (3.17% improvement compared to direct quantization). Trained CC is less effective than dithering for binarzed inputs (6.14% worse), but it outperforms dithering when the input precision is reduced to 2 bits by 0.77%.

Using the ImageNet dataset on a full-precision ResNet18 (Fig. 7) yields similar trends like on CIFAR10. Direct quantization of the input images results in 66.94%, 22.25%, 5.54%, and 1.77% accuracy degradation using 1, 2, 3 and 4-bit input respectively. Trained CC can recover a significant portion of this loss (44.24%, 13.92%, 2.17% and 0.3% improvement for 1, 2, 3 and 4-bit input respectively). Dithering appears to be the more effective technique to recover the most of the accuracy loss. FS dithering achieves only 3.5%, 2.41%, 1.27% and 0.62% (for 1, 2, 3 and 4-bit) worse top-1 accuracy than using the 8-bit original input. For the 1-bit input it recovers 63.44% accuracy loss.

V. OBSERVATIONS AND FUTURE WORK

Usually, the input of full-precision neural networks is normalized (to 0 mean and standard deviation of 1) to speed up the training process. On the other hand, the quantization of the input images requires a range in which we equally distribute the quantization levels. A too wide or or too tight range results in inefficient usage of bits. In our evaluation setup

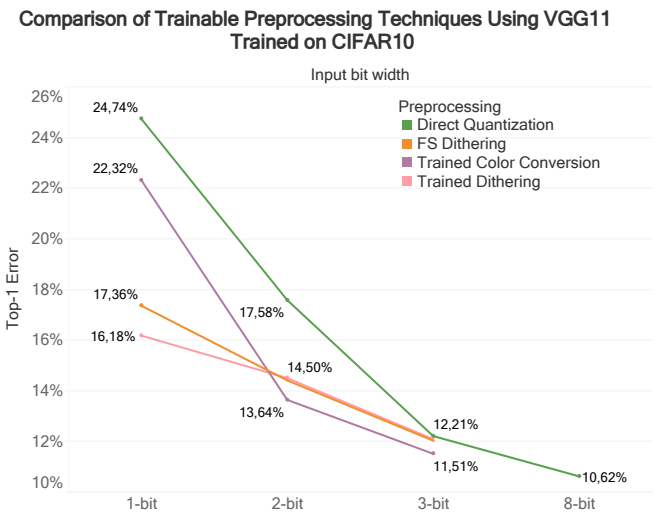

Fig. 6. Trained dithering leads to best accuracy with binarized input, CC slightly outperforms it using 2 bit input. Using 3-bit input the difference in accuracy becomes marginal.

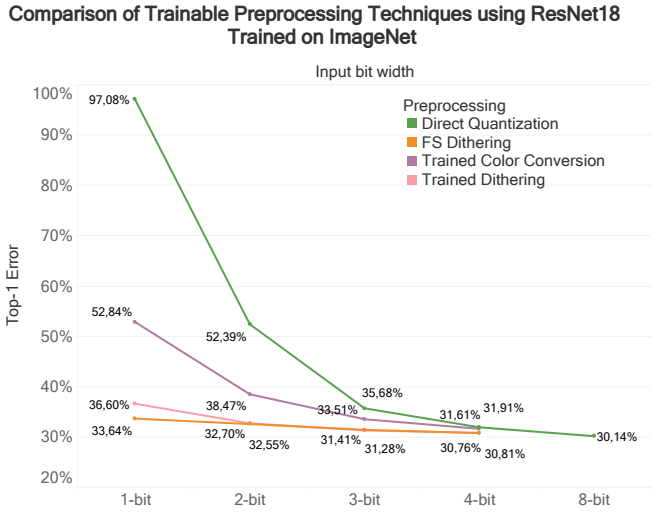

Fig. 7. Full-precision ResNet18 on ImageNet dataset. Dithering can more effectively recover accuracy loss than trained CC.

the quantization range was between -1 and 1. If normalization is enabled some values are pushed outside of this range and they are clamped. This loss of quality in the input images led to 3% accuracy degradation in the Imagenet experiments. Therefore, input normalization is disabled for quantized inputs, matching the maximum value of the image (255 in 8-bit) to 1 and the minimum value (0 in 8-bit) to -1.

As future research, we will map our trainable preprocessing techniques to real hardware implementations to get better insight into to cost of neural networks with trainable preprocessing. Moreover, we plan to explore options to directly use raw (image sensor) data. This will enable training the parameters of the ISP itself.

VI. Conclusion

Cost efficient implementations of neural networks often only focus on reducing the precision in the network, while keeping full precision at its input. This paper explored how trainable prepossessing (dithering and color conversion) can be applied to also reduce the input precision as technique to further improve the energy efficiency. We extend the training process of the neural network to the parameters of this preprocessing.

Experiments with the CIFAR10 and ImageNet on different neural network topologies showed a significant accuracy improvement compared to direct quantization of the input data. On the binary CNV network trained on CIFAR10, the trained dithering outperforms direct quantization by 6.01%, recovering 45% of the accuracy degradation related to input quantization. The quantized MobileNetV1 experiments showed the same trends using the more complex ImageNet dataset. In case of MobileNetV1 I4W4A4 trained dithering achieves only 0.2% worse top-1 accuracy than using the full-precision input, but at 32% lower computational complexity.

References

[1] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," *CoRR*, vol. abs/1905.11946, 2019.

[2] P. Gysel, J. Pimentel, M. Motamedi, and S. Ghiasi, "Ristretto: A framework for empirical study of resource-efficient inference in convolutional neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2018.

[3] H. Qin, R. Gong, X. Liu, X. Bai, J. Song, and N. Sebe, "Binary neural networks: A survey," *Pattern Recognition*, vol. 105, p. 107281, 2020.

[4] K. Goetschalckx and M. Verhelst, "Breaking high-resolution cnn bandwidth barriers with enhanced depth-first execution," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 2, pp. 323–331, 2019.

[5] M. Petrou and C. Petrou, *Image Processing: The Fundamentals*. Wiley, 2010.

[6] C. Baskin, E. Schwartz, E. Zheltonozhskii, N. Liss, R. Giryes, A. M. Bronstein, and A. Mendelson, "UNIQ: uniform noise injection for the quantization of neural networks," *CoRR*, vol. abs/1804.10969, 2018.

[7] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *J. Mach. Learn. Res.*, vol. 18, no. 1, p. 6869–6898, Jan. 2017.

[8] Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre, and K. Vissers, "FINN: A framework for fast, scalable binarized neural network inference," in *Proc. ACM/SIGDA Int. Symp. on Field-Programmable Gate Arrays (FPGA)*, 2017, p. 65–74.

[9] S. N. Gowda and C. Yuan, "Colornet: Investigating the importance of color spaces for image classification," in *ACCV 2018*, vol. 11364. Springer, 2018, pp. 581–596.

[10] E. Riba, D. Mishkin, D. Ponsa, E. Rublee, and G. Bradski, "Kornia: an open source differentiable computer vision library for pytorch," 2019.

[11] L. Akarun, Y. Yardimci, and A. Enis Cetin, "Adaptive methods for dithering color images," in *Proceedings., International Conference on Image Processing*, vol. 3, Oct 1995, pp. 125–128 vol.3.

[12] R. W. Floyd, "Adaptive algorithm for spatial gray scale," *SID Int. Sym. Digest of Tech. Papers, 1975*, pp. 36–37, 1975.

[13] Y. Bengio, N. Léonard, and A. C. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *CoRR*, vol. abs/1308.3432, 2013.

[14] A. Pappalardo, "Xilinx/brevitas." [Online]. Available: https://doi.org/10.5281/zenodo.3333552

[15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. 3rd Int. Conf. on Learning Representations (ICLR)*, Y. Bengio and Y. LeCun, Eds., 2015.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 770–778.