2023 IEEE Symposium Series on Computational Intelligence (SSCI)
Mexico City, Mexico. December 5-8, 2023

Results on the Empirical Design of a Residual Binary Multilayer Perceptron Architecture

Agustín Solís Winkler
UAP Tianguistenco
Universidad Autonoma del Estado de Mexico
Tianguistenco, México
asolisw@uaemex.mx
ORCID [0009-0009-6063-9977]

Asdrúbal López Chau
CU Zumpango
Universidad Autonoma del Estado de Mexico
Zumpango, México
alchau@uaemex.mx
ORCID [0000-0001-5254-0939]

Santiago Osnaya Baltierra
UAP Tianguistenco
Universidad Autonoma del Estado de Mexico
Tianguistenco, México
sosnayab@uaemex.mx
ORCID [0000-0003-0335-6662]

***Abstract*—Binary neural networks have emerged as an efficient solution for resource-constrained devices due to their reduced computational, memory, and storage requirements. However, binary neural networks often suffer from decreased accuracy compared to floating-point models. In this study, we propose a binary residual multilayer perceptron architecture that mitigates the degradation caused by binarization through the incorporation of normalization layers and residual connections. By leveraging design recommendations from state-of-the-art binary architectures, we aim to create a user-friendly model that can be easily implemented without requiring extensive neural network design expertise. This paper presents the empirical results of our proposed architecture, demonstrating its effectiveness in reducing degradation and improving performance for hardware-constrained devices.**

Keywords—binary neural network, multilayer perceptron, model compression, quantization, binarization, residual connection

I. Introduction

Despite the impressive performance of state-of-the-art deep learning models in laboratory settings, their size and resource requirements make them impractical for real-world applications [1]. The need to reduce the requirements of these models especially for resource-constrained devices has led to the development of the field of neural network compression [2]. Within this field, quantization [3] and binarization [4], [5] have gained significant interest due to the simplicity and potential for reducing storage, memory, and processing needs.

Binarization can be described as a quantizing neural network (NN) parameters to a precision of one bit [5], [6]. This technique offers savings by replacing floating-point (FP) operations with logical and bit-counting operations [7] [8], [9].

Regardless their appeal, binary neural networks (BNNs) suffer from accuracy degradation compared to their FP counterparts [9], [10]. To address this challenge various approaches and techniques have been developed, including reducing quantization errors, improving cost function and gradient approximation, employing specific training strategies, and designing architectures for binary operation [5].

Although a formal mathematical explanation of the functioning of BNNs is still lacking [11], it is known that NNs are universal approximators of continuous functions [12]. Moreover, it has been demonstrated the universal function approximation capability of three-layer BNNs [13], suggesting the potential for designing a binary multilayer perceptron capable of approximating problem solutions with convenient accuracy.

With the potential benefits mentioned above in mind, this study focuses on the empirical development of a residual binary multilayer perceptron architecture. Additional layers and residual connections are incorporated to assess whether these architecture modifications can effectively address the information loss caused by binarization. The objective is to determine if the resulting accuracy can compensate for the loss and if the precision achieved is comparable to that of a FP multilayer perceptron, all while reducing computational costs.

II. Related work

Several works have been developed to improve the precision and accuracy of BNNs since their introduction in 2015. The initial models, BinaryConnect [6] and BinaryNet [4], proposed different approaches to binarizing weights and activations. Subsequent models, such as XNOR-NET [7] and Binarized Neural Network [4] [14], incorporated scaling factors and probabilistic approximations to address information loss and improve efficiency. DoReFa-Net [15] introduced varying precision and width, but did not show significant improvement. In terms of architecture, Binary DenseNet [16] and MeliusNet [17] introduced shortcut connections and Dense Blocks, respectively, to enhance information flow and feature capacity. These advancements collectively aim to overcome limitations and improve the precision of binary neural networks.

III. Binary neural network basics

A. Binary neural network

A binary neural network is a specific type of NN model where the input and output layers are represented using FP values, while the hidden layers employ binary values for weights and activations [4]. The concept behind binarization is to constrain the weights and activations to *+1* and *–1* during training. This enables the substitution of FP multiplication and accumulation operations with faster 1-bit XNOR and POPCOUNT operations, which outperform the 32-bit counterparts in terms of computational speed [8].

B. Binarization

The sign function (1) is used both to binarize the inputs and as an activation function, transforming FP values into binary values [4].

$$sign(x) = \begin{cases} +1 \; si \; x \geq 0, \\ -1 \; si \; x < 0 \end{cases} \tag{1}$$

C. Backpropagation

The problem of applying backpropagation with gradient descent to binarized neural networks poses a challenge, because of the non-differentiability of the binarization function. To overcome this, the Straight-Through Estimator (STE) technique (2) is utilized to train a network with binarized weights using the sign function [18].

$$\boldsymbol{b_w} = sign(\boldsymbol{w}) \tag{2}$$

Where $\boldsymbol{b_w}$ represents the binary weights tensor $\boldsymbol{w}$.

To binarize activations, the sign function is also applied to them, and the STE is used during the backward pass, similar to how weights are binarized. If the input to the activation function is very large, the gradient is canceled during the backward pass using (3) according with [8].

$$\frac{\partial L}{\partial \boldsymbol{a}} = \frac{\partial L}{\partial \boldsymbol{b_a}} \times 1_{|a| \leq 1} \tag{3}$$

Where L is the loss at the output, $\boldsymbol{\alpha}$ is the real valued input of the activation function and $\boldsymbol{b_\alpha}$ is the binary output of the activation function.

This allows for the effective use of binary values in the training process.

IV. Implementation and Testing

A. Multilayer perceptron

Fig. 1. Depicts a multilayer perceptron with three hidden layers. Each layer is fully connected meaning all processing units in a particular layer have a connection with all artificial neurons of the preceding one [19], [20]. This is the NN base model, whose binary equivalent used as starting point for the proposed architecture.

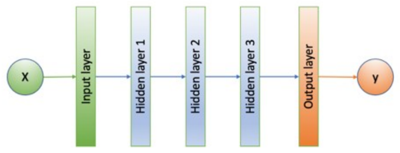


Fig. 1 Multilayer perceptron

B. Building block

For the development of the residual binary multilayer perceptron, the recommendations presented by [16], [21], [22] for binary neural networks were taken into account.

The input and output layers are kept in FP format, while a building block consisting of an ordered sequence of the following layers: dropout layer, binary dense layer, and batch normalization layer, is used as shown in Fig. 2. This block is repeated according to the number of hidden layers in the base model. If the perceptron has more than one hidden layer, a shortcut connection is added, but its width is adjusted to match that of the last hidden layer.

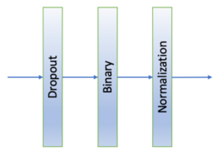


Fig. 2. Building block

C. Residual connection

The function of the shortcut is to restore the information flow at the end of the network [23], where it is combined with the output of the last hidden layer to contribute to the calculated value of the output layer, it takes the input layer as input and outputs the shape as the last hidden layer.

Typically, residual connections are implemented using convolutional layers with *1x1* filters, but for the proposed model, different methods for the implementation of shortcuts were tried including:

- Dense layer with no activation function.
- Quantized dense layer with no activation function. 8-bit quantization was used to reduce the memory required by the shortcut to one-fourth of the input layer.
- One-dimensional max pooling layer. This one requires an additional dense layer to adjust to the size of the last hidden layer when the width of that layer does not exactly match the output of the pooling layer.

D. Proposed model

Using the building block and the shortcut connection, which is added with the last hidden layer, a residual binary multilayer perceptron is obtained, as shown in Fig. 3.

As can be observed, the proposed architecture replaces each hidden layer of a standard multilayer perceptron with a block consisting of a dropout layer, a binary dense layer, and a batch normalization layer [24]. For the conducted tests, widths of the binary dense layers are kept the same as that of the base perceptron, to allow confirming that the accuracy improvements are due to the addition of layers rather than an increase in their width. However, in real-world applications, a scaling factor can be used to improve representativeness.

E. Model size anlysis

When implementing a model with quantized layers, k bits are used to represent each value. For binary networks $k = 1$, and the size of the input is equal to the number of parameters.

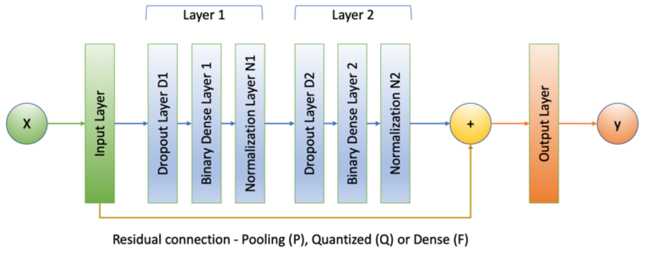


Fig. 3. Binary residual multilayer perceptron with 2 blocks

However, since the output is computed in floating point with m bits, the size S of the binary perceptron in bits is calculated as shown in (4). Fig. 4. Illustrates the basics for layer size calculation [25]:

$$S = \sum_{t=1}^{T} (k|V_{t-1}||V_t| + m|V_t|) + m(|V_T| + 1)|V_{t-1}| \tag{4}$$

Where $|V|$ represents the width of layer V [26], t is the current layer, $t–1$ is the previous layer and T is the number of hidden layers.

Normalization layers use floating point representation and the same number of inputs and outputs. Therefore, if m is the number of bits used to represent a FP value, the size required for a normalization layer S_N is determined by (5):

$$S_N = 2m|V_t|. \tag{5}$$

Dropout layers do not store any values; therefore, they don't require any storage.

Considering the three different implementations for shortcut connections described earlier, the residual connection size S_A will be calculated as (6):

$$S_A = k\frac{|V_0|}{p}|V_{T-1}| + m|V_T| \tag{6}$$

Where:

k – is the number of bits used for quantizing the input layer. If $k = 32$, a FP shortcut is used; if $k = 1$ we have a binary layer, and $k = 8$ represents an 8-bit quantized layer.

m – is the number of bits used to represent a FP number, which defaults to 32,

p – pool size for shortcuts based on pooling layers, if $p = 1$, no pooling layer is used.

Combining the 3 previous equations, the max size of the proposed model when T normalization layers are used is fully calculated using (7) combined with the shortcut size from (6):

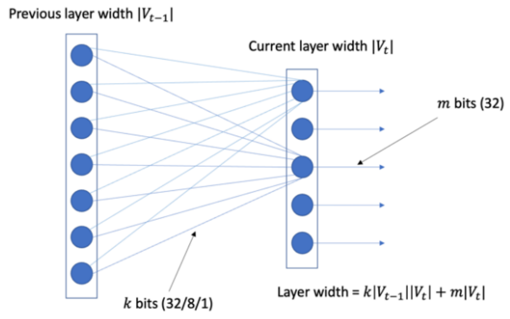
$$S = \sum_{t=1}^{T} (k|V_{t-1}||V_t| + m|V_t|) + m(|V_T| + 1)|V_{t-1}| + S_A + \sum_{t=1}^{T} 2m|V_t| \tag{7}$$

Selecting the values for k, m, and p, allows for model size tunning according with requirements and storage restrictions.

V. Testing

A. Dataset and hyperparameters

Three balanced datasets were used for the experiments: MNIST, IMDB, and CIFAR10, aiming to showcase the flexibility of the multilayer perceptron in image and text classification tasks, both binary and categorical. Table I summarizes the characteristics of the selected datasets.



Previous layer width $|V_{t-1}|$
Current layer width $|V_t|$
m bits (32)
k bits (32/8/1)
Layer width = $k|V_{t-1}||V_t| + m|V_t|$

Fig. 4. Base for layer width calculation

To keep the focus on the accuracy change when modifying the network architecture, a similar set of values for hyperparameters was used for all cases. In particular [11] suggests using small batch sizes, so in our case, we used 32. The values of the hyperparameters are shown in Table II.

TABLE I DATASETS USED FOR EPERIMENTS

Dataset	*Objects*	*Input*	*Classes*	*Training instances*	*Test Instances*
MNIST	28x28, 8 bits graysc. images	784	10	60000	10000
IMDB	Movie reviews	10000	2	25000	25000
CIFAR10	32x32, 24 bits color images	3072	10	50000	10000

B. Perceptron variants labeling

Each variant is identified using the following notation:

$$(M – DxN, DxN, DxN, S, N, D_s) \tag{8}$$

Where:

M – model type (F) floating-point, (B) binary

D – dropout layer before a dense layer

x – hidden dense layer width

N batch normalization after dense layer or shortcut

D_s – dropout layer before the output layer

S – shortcut (F) floating-point, (P) pooling, (Q) quantized

TABLE II. HYPERPARAMETER SETTINGS

Parameter	*Value*
Epochs	10
Batch	32
Learning rate	0.001
Dropout rate	0.05
Quantization	8 bits
Pool size	8

C. *Results description*

To evaluate a specific architecture of a residual binary multilayer perceptron model, comprehensive tests were conducted, generating various combinations of dropout layers, normalization layers, and shortcuts based on a predefined base design. The resulting models underwent training and evaluation to measure accuracy and size.

Memory requirements are influenced by layer selection, quantity, and the type of residual connection employed (refer to section *IV.E*).

In quantized and FP shortcuts, the parameter count equals the product of the input layer's width and the last hidden layer's width. However, if an 8-bit representation is utilized, the quantized shortcut requires only one-fourth of the memory compared to the FP shortcut. Pooling-based shortcuts reduce parameter count based on the chosen pool size.

These considerations allow for a priori insights into architecture size, even before construction. Furthermore, dropout layers enhance accuracy without increasing model size when used in conjunction with normalization layers.

The subsequent tables present average accuracy achieved by each layer for the tested datasets. Multivariable regression analysis supports these values. For reference, the first three rows of each table compare accuracy among FP base models, equivalent binary models, and the best-performing variant of the proposed model. Additionally, a second table compares size and accuracy across different model variants, including FP, pooling, quantized, and the best-performing variants, with the binary equivalent model listed last.

D. *Results with MNIST and CIFAR10*

Tests with the MNIST dataset show that normalization layers N1, N2, N3, lead to better accuracy of the models, and that is improved with the use of pooling shortcuts P. Table III shows the best performing variants outperform the Binary equivalent and get close the FP architecture.

Table IV displays that the binary equivalent is 0.04 size of the FP models, however we can observe best performing variants growing up to 0.16.

It is worth noting that the CIFAR10 dataset exceeds the classification capabilities of perceptrons, which can achieve a maximum accuracy of 0.59 for this dataset only when extensive preprocessing is applied to the input, including feature augmentation and image cropping [93]. However, this case is presented without using preprocessing to showcase the benefits of the proposed architecture.

TABLE III. AVERAGE ACCURACY FOR MNIST

Layer	*1 layer*	*2 layers*	*3 layers*
F original	0.9749	0.9771	0.9772
B equivalent	0.9024	0.9162	0.8510
Best variant	0.9401	0.9630	0.9652
D	0.9061	0.9429	0.9221
N	0.9080	0.9437	0.9241
P	0.9174	0.9464	0.9256
F	0.9022	0.9420	0.9214
Q	0.9069	0.9419	0.9218
D1	0.9110	0.9433	0.9223
N1	0.9238	0.9479	0.9309
D2		0.9428	0.9238
N2		0.9520	0.9448
D3			0.9225
N3			0.9514

For CIFAR10, shortcuts based on pooling layers, provide best results as shown in Table V. 3-layer model outperforms the original FP model. Again, best accuracy models stay under 0.15 from the FP model.

TABLE IV. ACCURACY VS RELATIVE SIZE FOR MNIST

Hidden layers	*Model*	*Accuracy*	*Model size (Kib)*	*Relative size*
1	F-128	0.9749	397.00	1.00
	B-D128N,P	0.9340	62.79	0.16
	B-D128N,Q	0.9206	117.29	0.30
	B-D128N,F	0.9243	411.29	1.04
	B-128N,PN,D	0.9401	63.79	0.16
	B-128	0.9024	17.79	0.04
2	F-128,88	0.9771	440.32	1.00
	B-D128N,D88N,P	0.9577	49.88	0.11
	B-D128N,D88N,Q	0.9562	87.35	0.20
	B-D128,D88N,F	0.9556	290.16	0.66
	B-D128N,D88N,PN	0.9630	50.57	0.11
	B-128,88	0.9162	17.95	0.04
3	F-128,88,88	0.9772	470.91	1.00
	B-D128N,D88N,D88N,P	0.9591	51.86	0.11
	B-D128N,D88N,D88N,Q	0.9520	89.33	0.19
	B-D128N,D88N,D88N,F	0.9576	292.14	0.62
	B-D128N,D88N,D88N,PN,D	0.9652	52.55	0.11
	B-128,88,88	0.8510	19.23	0.04

E. *Results with IMDB*

Table VI shows again that normalization layers improve the perceptron´s accuracy, and for this dataset quantized and FP residual connections work to improve the result. It can be observed that best variant outperforms the original FP perceptron.

For the IMDB dataset, the best performing models grow up to 0.28 of the original architecture with a remarkable 3-layer model that outperforms the original perceptron with 0.09 of its size. As can be seen on Table VII.

VI. DISCUSSION

The experimental investigation conducted in this study centered on the development of the proposed residual binary multilayer perceptron architecture. The results obtained convincingly demonstrate that models adhering to the proposed building block consistently outperform their binary counterparts. Remarkably, certain model variants even surpass the performance of floating-point (FP) models. This noteworthy outcome can be ascribed to the salutary impact of normalization layers in augmenting the binary model's accuracy.

TABLE V. ACCURACY VS RELATIVE SIZE FOR CIFAR10

Hidden layers	Model	Accuracy	Model size (KiB)	Relative size
1	F-256	0.4224	3083.04	1.00
	B-D256N,P	0.3922	451.04	0.15
	B-D256N,Q	0.3617	878.04	0.28
	B-D256N,F	0.3670	3184.04	1.03
	B-D256N,P,D	0.4061	451.04	0.15
	B-256	0.2612	107.04	0.03
2	F-256,256	0.4567	3340.04	1.00
	B-D256N,D256N,P	0.4679	462.04	0.14
	B-D256N,D256N,Q	0.4091	889.04	0.27
	B-D256N,D256N,F	0.4477	3195.04	0.96
	B-D256N,D256N,P	0.4679	462.04	0.14
	B-256,256	0.1	116.04	0.03
3	F-256,256,256	0.4350	3597.04	1.00
	B-D256N,D256N,D256N,P	0.4423	473.04	0.13
	B-D256N,D256N,D256N,Q	0.4049	900.04	0.25
	B-D256N,D256N,D256N,F	0.4314	3204.04	0.89
	B-D256N,D256,D256N,PN,D	0.4661	475.04	0.13
	B-256,256,256	0.1000	125.04	0.03

Moreover, the influence of residual connections is distinctly beneficial, as visually depicted in Figure 5 (depicting individual contributions of normalization layers) and Figure 6 (clearly illustrating the positive influence of residual connections on overall results).

TABLE VI AVERAGE ACCURACY FOR IMDB

Model /Layer	1 layer	2 layers	3 layers
F original	0.8577	0.8549	0.8637
B equivalent	0.8303	0.8565	0.8533
Best variant	0.8515	0.8713	0.8748
D	0.8359	0.8417	0.8411
N	0.8348	0.8355	0.8330
P	0.8336	0.8378	0.8383
F	0.8377	0.8421	0.8511
Q	0.8396	0.8437	0.8425
D1	0.8386	0.8421	0.8431
N1	0.8361	0.8345	0.8363
D2		0.8404	0.8421
N2		0.8390	0.8425
D3			0.8410
N3			0.8380

The empirical evidence garnered from the conducted experiments strongly suggests that the incorporation of normalization layers and shortcuts within a binary multilayer perceptron constitutes an effective strategy for enhancing its performance and accuracy.

It is pertinent to note that, for models handling real-valued inputs, such as images, th implementation of a residual connection achieved by combining a maximum pooling layer followed by a dense FP layer proves to be the best choice. In contrast, for text datasets employing one-hot encoding, quantized shortcuts exhibit superior performance.

Fig. 7. shows how the performance of the best variant of the developed perceptron outperforms the original FP perceptron in two and three-layer models, and in all cases, it is better than the equivalent binary perceptron.

TABLE VII ACCURACY VS RELATIVE SIZE FOR IMDB

Hidden layers	Model	Accuracy	Model size (KiB)	Relative size
1	F-250	0.8577	9767.00	1.00
	B-D250N,P	0.8320	1395.03	0.14
	B-D250N,Q	0.8500	2751.47	0.28
	B-D250N,F	0.8463	10075.69	1.03
	B-D250N,Q,D	0.8514	2751.27	0.28
	B-250	0.8303	307.13	0.03
2	F-32,32	0.8549	1250.38	1.00
	B-D32N,D32N,P	0.8266	179.07	0.14
	B-D32N,D32N,Q	0.8432	352.69	0.28
	B-D32N,D32N,F	0.8474	1290.19	1.03
	B-32,32,Q	0.8698	352.19	0.28
	B-32,32	0.8565	39.57	0.03
3	F-32,16,8	0.8637	1252.75	1.00
	B-D32N,D16N,D8N,P	0.8338	74.58	0.06
	B-D32N,D16N,D8N,Q	0.8468	117.99	0.09
	B-D32N,D16N,D8N,F	0.8362	352.36	0.28
	B-32N,D16N,D8,Q,D	0.8652	117.93	0.09
	B-32,16,8	0.8533	39.39	0.03

Table VIII provides an overview of the changes in model size compared to the FP perceptron. The binary equivalent model has a size of 0.04, while the FP shortcuts result in a size of 0.78. It is worth noting that the Quantized and Pooling-based shortcuts offer more favorable options in terms of size.

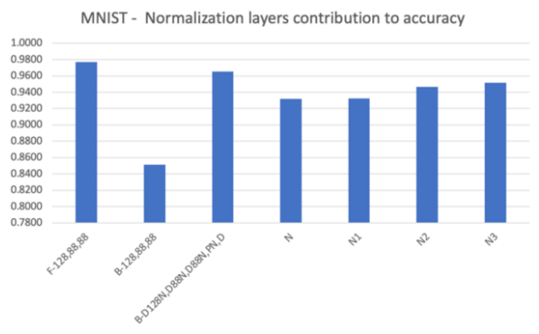


Fig. 5. Normalization layer contribution

VII. CONCLUSIONS

In summary, this study demonstrates that the modifying the architecture of a binary multilayer perceptron can significantly improve its performance in terms of accuracy, while preserving a compact model size. The the inclusion of normalization layers and residual connections is an effective strategy to achieve this objective, therefore, it is recommended that this technique be judiciously integrated into forthcoming developments of machine learning systems focused on resource-constrained devices, particularly those founded upon multilayer perceptrons.

VIII. Future work

The current study does not conduct parameter tuning to solely showcase the benefits of normalization layers and residual connections. However, it's crucial to explore hyperparameters like learning rates, batch sizes, and optimization for extensive model fine-tuning, identifying optimal configurations across diverse datasets and scenarios.

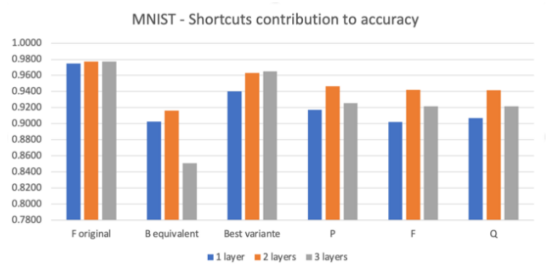


Fig. 6. Shortcut contribution to accuracy

A second direction involves further exploration in the binaryization of other deep learning models. This exploration will delve into techniques that enable the replacement of normalization layers with alternative methods that require fewer computations, as well as the investigation of the utilization of binary optimizers aiming to further reduce the floating-point operations required for both model training and inference.

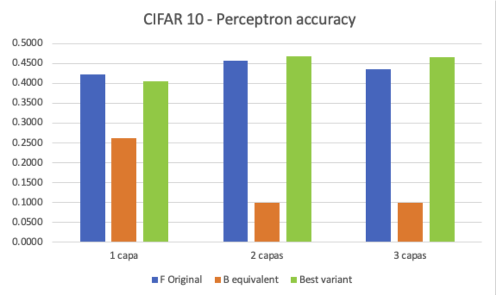


Fig. 7. Accuracy comparing floating-point, binary and proposed

TABLE VIII. Relative size according with shortcut used

	Average relative size			
Layers	**B**	**P**	**Q**	**F**
1	0.04	0.15	0.29	1.03
2	0.03	0.12	0.23	0.79
3	0.03	0.09	0.16	0.52
Average	0.04	0.12	0.22	0.78

References

[1] S. Pokhrel, "4 Popular Model Compression Techniques Explained," *Xailient*, Jan. 19, 2022. https://xailient.com/blog/4-popular-model-compression-techniques-explained/ (accessed Mar. 01, 2023).

[2] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "A Survey of Model Compression and Acceleration for Deep Neural Networks," Oct. 2017, [Online]. Available: http://arxiv.org/abs/1710.09282

[3] P. E. Novac, G. B. Hacene, A. Pegatoquet, B. Miramond, and V. Gripon, "Quantization and deployment of deep neural networks on microcontrollers," *Sensors*, vol. 21, no. 9, May 2021, doi: 10.3390/s21092984.

[4] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1," Feb. 2016, [Online]. Available: http://arxiv.org/abs/1602.02830

[5] C. Yuan and S. S. Agaian, "A comprehensive review of Binary Neural Network," Artificial Intelligence Review 2023, pp. 1–65, Oct. 2021, doi: 10.1007/S10462-023-10464-W/METRICS.

[6] M. Courbariaux, Y. Bengio, and J.-P. David, "BinaryConnect: Training Deep Neural Networks with binary weights during propagations," Nov. 2015.

[7] M. Rastegari, V. Ordonez, J. Redmon, and Farhadi Ali, "XNOR-Net: ImageNet Classification Using Binary Convolutional Networks," 2016.

[8] T. Simons and D. J. Lee, "A review of binarized neural networks," *Electronics (Switzerland)*, vol. 8, no. 6. MDPI AG, Jun. 01, 2019. doi: 10.3390/electronics8060661.

[9] H. Qin, R. Gong, X. Liu, X. Bai, J. Song, and N. Sebe, "Binary Neural Networks: A Survey," Mar. 2020, doi: 10.1016/j.patcog.2020.107281.

[10] J. Bethge, H. Yang, M. Bornstein, and C. Meinel, "Back to Simplicity: How to Train Accurate BNNs from Scratch?," Jun. 2019, [Online]. Available: http://arxiv.org/abs/1906.08637

[11] M. Alizadeh, J. Fernández-Marqués, N. D. Lane, and Y. Gal, "An empirical study of binary neural networks' optimization.," in *International Conference on Learning Representations*, 2018.

[12] M. T. Hagan, H. B. Demuth, M. H. Beale, and O. De Jesus, *Neural Network Design*, 2nd ed. 1996.

[13] A. J. Redfern, L. Zhu, and M. K. Newquist, "BCNN: A Binary CNN With All Matrix Ops Quantized To 1 Bit Precision," 2021.

[14] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations," 2018.

[15] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "DoReFa-Net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients," Jun. 2016, [Online]. Available: http://arxiv.org/abs/1606.06160

[16] J. Bethge, H. Yang, M. Bornstein, and C. Meinel, "BinaryDenseNet: Developing an Architecture for Binary Neural Networks," 2019. [Online]. Available: https://github.com/hpi-xnor/BMXNet-v2

[17] J. Bethge, C. Bartz, H. Yang, Y. Chen, and C. Meinel, "MeliusNet: Can Binary Neural Networks Achieve MobileNet-level Accuracy?," Jan. 2020.

[18] C. Yuan and S. S. Agaian, "A comprehensive review of Binary Neural Network," Feb. 2022, [Online]. Available: http://arxiv.org/abs/2110.06804

[19] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[20] C. Aggarwal, *Artificial Intelligence. A textbook*, 1st ed. Swtizerland: Springer Nature Switzerland AG, 2021.

[21] H. Li, S. De, Z. Xu, C. Studer, H. Samet, and T. Goldstein, "Training quantized nets: A deeper understanding.," *Adv Neural Inf Process Syst*, pp. 5811–5821, 2017.

[22] H. Kim, K. Kim, J. Kim, and J.-J. Kim, "Reducing gradient mismatch in binary activation network by coupling binary activations.," 2020.

[23] Z. Liu, B. Wu, W. Luo, X. Yang, W. Liu, and K.-T. Cheng, "Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm.," in *Proceedings of the European conference on computer vision (ECCV)*, 2018.

[24] A. Solis Winkler, J. L. Tapia Fabela, and S. Osnaya Baltierra, "Diseño empírico de una arquitectura de perceptrón multicapa binario residual," *Research i*, no. In press, 2023.

[25] Google, "Keras: the Python deep learning API," *keras.io*, 2023. https://keras.io/ (accessed Feb. 19, 2023).

[26] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*, First. New York: Cambridge University Press, 2014. [Online]. Available: http://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning