PARAMETERS QUANTIZATION WITH A-CONNECT

ANDRÉS FELIPE CENTENO OCHOA
LUIS ALEJANDRO HERNÁNDEZ RÍOS

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTY OF PHYSICAL-MECHANICAL ENGINEERING
SCHOOL OF ELECTRICAL,
ELECTRONIC AND TELECOMMUNICATIONS ENGINEERING
BUCARAMANGA
2022

PARAMETERS QUANTIZATION WITH A-CONNECT

ANDRÉS FELIPE CENTENO OCHOA
LUIS ALEJANDRO HERNÁNDEZ RÍOS

Bachelor degree thesis to qualify for the title of
Electronic Engineer

Director
Elkim Felipe Roa Fuentes,
Philosophy Doctor.

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTY OF PHYSICAL-MECHANICAL ENGINEERING
SCHOOL OF ELECTRICAL,
ELECTRONIC AND TELECOMMUNICATIONS ENGINEERING
BUCARAMANGA
2022

Acknowledgment

We would like to thank all the people who supported and guided us throughout our career and the development of this project.

Special thanks to professors Luis Rueda and Elkim Roa who were supervising and supporting the work done.

CONTENTS

LIST OF FIGURES

pág.

LIST OF TABLES

pág.

RESUMEN

TÍTULO: CUANTIZACIÓN DE PARÁMETROS CON A-CONNECT [*]

AUTORES: ANDRES FELIPE CENTENO OCHOA, LUIS ALEJANDRO HERNANDEZ RIOS [**]

PALABRAS CLAVE: REDES NEURONALES CONVOLUCIONALES, REDES NEURONALES PROFUNDAS, REDES NEURONALES CUANTIZADAS.

DESCRIPCIÓN:

Los algoritmos de aprendizaje profundo logran una alta precisión de clasificación a expensas de un costo de cálculo significativo: los resultados se obtienen utilizando grandes conjuntos de entrenamiento y modelos grandes. Una red neuronal muy profunda normalmente involucra muchas capas con millones de parámetros, lo que hace que el almacenamiento del modelo de red sea extremadamente grande. Esto prohíbe el uso de redes neuronales profundas en hardware con recursos limitados, especialmente teléfonos móviles u otros dispositivos integrados. A-Connect es una metodología de entrenamiento estadístico de redes neuronales que mejora la resiliencia de las redes neuronales analógicas frente a la variabilidad estocástica. Esta metodología también demostró funcionar para la cuantificación de parámetros cuando se usa para pesos binarios. Ya se desarrolló una biblioteca A-Connect en un proyecto anterior de pregrado usando API existentes como Keras y TensorFlow. En este documento, presentamos una extensión de la biblioteca A-Connect para que pueda implementarse para diferentes niveles de cuantificación de parámetros, lo que permite la utilización de la metodología para la cuantificación en aplicaciones de mayor precisión en hardware con recursos limitados. Mostramos los resultados para la biblioteca desarrollada usando una red neuronal Fully-connected y algunas arquitecturas más comunes de Red Neural Convolucional (CNN). Trabajamos con los conjuntos de datos MNIST y CIFAR-10. Además, presentamos una implementación de FPGA con pesos y sesgos binarios utilizando A-Connect.

---

[*] Trabajo de grado

[**] Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingenierías Eléctrica, Electrónica y Telecomunicaciones. Director: Elkim Felipe Roa Fuentes, Philosophy Doctor.

ABSTRACT

TITLE: PARAMETERS QUANTIZATION WITH A-CONNECT [*]

AUTHORS: ANDRES FELIPE CENTENO OCHOA, LUIS ALEJANDRO HERNANDEZ RIOS [**]

KEYWORDS: CONVOLUTIONAL NEURAL NETWORKS, DEEP NEURAL NETWORKS, QUANTIZED NEURAL NETWORKS.

DESCRIPTION:

Deep learning algorithms achieve high classification accuracy at the expense of significant computation cost: results are obtained using large training sets and large models. A very deep neural network normally involves many layers with millions of parameters, making the storage of the network model to be extremely large. This prohibits the usage of deep neural networks on resource limited hardware, especially cell phones or other embedded devices. A-Connect is a neural network statistical training methodology that improves analog neural network resilience against stochastic variability. This methodology also proved to work for parameter quantization when used for binary weights. An A-Connect library has already been developed in a previous bachelor undergraduate project using existing APIs such as Keras and TensorFlow. In this paper, we present an extension of the A-Connect library so it can be deployed for different parameter quantization levels which allows the utilization of the methodology for quantization in higher precision applications on resource limited hardware. We show the results for the library developed using a fully-connected neural network and some more common architectures of Convolutional Neural Network (CNN). We worked with the MNIST and CIFAR-10 datasets. In addition, we present an FPGA implementation with binary weights and biases using A-Connect.

---

[*] Bachelor Thesis

[**] Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingenierías Eléctrica, Electrónica y Telecomunicaciones. Director: Elkim Felipe Roa Fuentes, Philosophy Doctor.

INTRODUCTION

DNN (Deep Neural Networks) have achieved high accuracy in a wide range of tasks, including object detection, speech recognition, and computer vision. One of the major factors that enabled faster progress in this field is implementing neural networks on hardware devices that have the capability of running efficiently the massive amount of multiply-accumulate operations required to compute the weighted sum of the neuron's inputs. Given the widespread adoption of power-hungry hardware such as GPUs (Graphics processing unit) for training neural networks, it has been necessary to run deep neural networks in low power devices. Not only it does affects computational resource but also memory capability, this being due to large model sizes (200MB for ResNet-101[1], 250MB for AlexNet[2], or 500MB for VGG-Net[3]).
Several works have treated this issue with different approaches. One of the most common approaches is to compress a trained network. Hashed-Nets[4] force the weights to share a single parameter value by using a hash function to randomly group connection weights. Other researchers focus on the design of novel network architectures that exploit computation and memory efficient operations, such as Mo-

---

[1] K. He, X. Zhang, S. Ren y J. Sun. "Deep Residual Learning for Image Recognition". En: *Conference on Computer Vision and Pattern Recognition(CVPR)* (2015), págs. 770-778. arXiv: 1512.03385 [cs.CV].

[2] Alex Krizhevsky, Ilya Sutskever y Geoffrey Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". En: *Neural Information Processing Systems* 25 (ene. de 2012). DOI: 10.1145/3065386.

[3] Karen Simonyan y Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV].

[4] W. Chen, J. Wilson, S. Tyree, K. Weinberger e Y. Chen. "Compressing neural networks with the hashing trick". En: *International Conference on Machine Learning(ICML)* (2015), págs. 2285-2294. arXiv: 1504.04788 [cs.LG].

bileNet[5], SqueezeNet[6], ShuffleNet[7] and DenseNet[8].

The quantization of weights and others network parameters is another of the most common approaches to solve the problem. This consists of changing the network parameters from 32 bits floating point into lower bit-depth representations. Some methodologies like BinaryConnect[9] take the extreme case of binarization, where the weights are constrained into two possible values (e.g, -1 and 1), thus, decreasing the accuracy of the trained network. TWN's[10] increase the number of quantization levels to 3 (-1, 0, and 1) improving the final accuracy. Other works[11] propose increasing the quantization levels using a defined function so that it is possible to increase the bitwidth until achieving accuracy comparable to the 32-bit floating-point counterpart. Other problem that comes with implementing a deep neural network in an analog processor is that it suffers from some stochastic variations that affect the accuracy

---

5 A. Howard, M. Zhu, B. Chen, D. Kalenichenko y W. Wang. *Mobilenets: Efficient convolutional neural networks for mobile vision applications*. 2017. arXiv: 1704.04861 [cs.CV].

6 F. Landola y col. "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and 1Mb model size". En: *International Conference on Learning Representations(ICLR)* (2017). arXiv: 1602.07360 [cs.CV].

7 X. Zhang, X. Zhou, M. Lin y J. Sun. *Shufflenet: An extremely efficient convolutional neural network for mobile devices*. 2017. arXiv: 1707.01083 [cs.CV].

8 G. Huang, Z. Liu, L. Van der Maaten y K. Weinberger. "Densely connected convolutional networks". En: *Conference on Computer Vision and Pattern Recognition(CVPR)* (2018). arXiv: 1608.06993 [cs.CV].

9 M. Courbariaux, Y. Bengio y J. David. "BinaryConnect: Training Deep Neural Networks with binary weights during propagations". En: *Conference on Neural Information Processing Systems(NIPS)* (2016). arXiv: 1511.00363 [cs.LG].

10 F. Li, B. Zhang y B. LiuF. "Ternary Weight Networks". En: *Conference on Neural Information Processing Systems(NIPS)* (2016). arXiv: 1605.04711 [cs.CV].

11 I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv e Y. Bengio. "Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations". En: *Journal of Machine Learning Research(JMLR)* (2018), págs. 1-30. arXiv: 1609.07061 [cs.NE].

of the operations. These stochastic variations could be temporal (noise) or spatial (mismatch). Therefore analog DNN accelerators are very imprecise even when the neural networks have shown some intrinsic immunity to noisy environments[12] so the digital accelerators still are the best option.

The stochastic variabilities are worse when a smaller and ultra-low power consumption CIM analog DNN accelerator is employed due to the random variations during the manufacturing process, all the currents and voltages deviate from their ideal values. For that reason, low-power smaller devices are more sensitive to the hardware stochastic variabilities.

The problems mentioned above converge in a Ph.D. thesis from the Integrated Systems Research Group - OnChip. The Ph.D. candidate proposes the A-Connect training methodology for neural networks. This methodology has shown interesting results by mitigating stochastic variability[13]. The results obtained with A-Connect were using a simple neural network of one hidden layer with the handwritten digits dataset MNIST[14]. A-Connect can also take into account another type of parameter corruption, such as weight quantization. At this moment, A-Connect only supports the extreme case with binary weights.

An A-Connect library has already been developed in a previous bachelor undergra-

---

[12] Hsin-yu Tsai, Stefano Ambrogio, Pritish Narayanan, Robert Shelby y Geoffrey Burr. “Recent progress in analog memory-based accelerators for Deep Learning”. En: *Journal of Physics D: Applied Physics* 51 (mayo de 2018). DOI: 10.1088/1361-6463/aac8a5.

[13] Luis Rueda y Elkim Roa. *A-Connect: Enabling Imprecise Analog Computation*. not published work.

[14] Y. LeCunn, C. Cortes y C. Burges. *The MNIST database of handwritten digits*. [Online] Available: http://yann.lecun.com/exdb/mnist/.

duate project[15] using existing APIs such as Keras[16] and TensorFlow[17], but this library only supports weights binarization.

To extend A-Connect for parameter quantization, this project makes the following contribution:

- A library update development using Keras[16] and TensorFlow[17] for the A-Connect training methodology. This library includes all the features of A-Connect including quantization of weights for different bit levels. This means the library has a fully connected layer and a convolutional layer with A-Connect for stochastic variability mitigation and weights quantization.
- Introduce Bias hyperparameter quantization to the A-Connect methodology library for different bit levels.

---

[15] R. Vergel, E. Silva, L. Rueda y E. Roa. *Library development for A-Connect*. Trabajo de investigación Ingeniero Electrónico. Buacaramanga. Universidad Industrial de Santander. Facultad de Ingenierías Físico-Mecánicas. 2021.

[16] François Chollet. *Deep learning API Keras*. [Online] Available: https://keras.io/. 2015.

[17] Google brian team. *Machine learning platform TensorFlow*. [Online] Available: https://www.tensorflow.org/. 2015.

1. OBJECTIVES

1.1. GENERAL OBJECTIVES

- To develop and validate an extension of the A-Connect methodology to diferent parameter quantization levels.

1.2. SPECIFIC OBJECTIVES

- To update the current weight binarization of the A-Connect library, with diferent quantization levels of diferent parameters.
- To test the A-Connect methodology using parameter quantization with deep neural networks in a well-studied dataset (e.g., CIFAR-10, ImageNet, or speech-recognition ones).
- To validate the trainned quantized neural network through FPGA acceleration.

2. A-CONNECT METHODOLOGY DESCRIPTION

The main idea with A-Connect is to mitigate the stochastic variability in a DNN analog accelerator. For this purpose, the A-Connect methodology uses a normal distribution to create a batch of error matrices for the weights and biases. Then multiply (element-wise) the weights and biases by their respective error matrices during the forward propagation. Figure 2.1 shows the operation of A-Connect. For the backward propagation, A-Connect applies the chain rule to get the proper gradient to update the weights and biases.

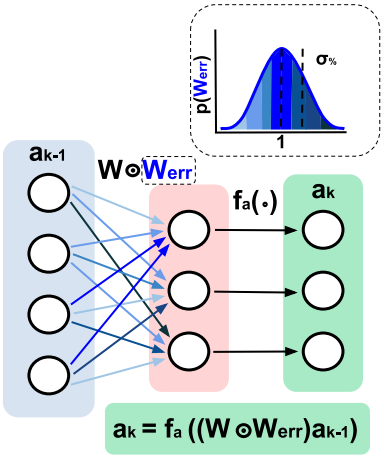
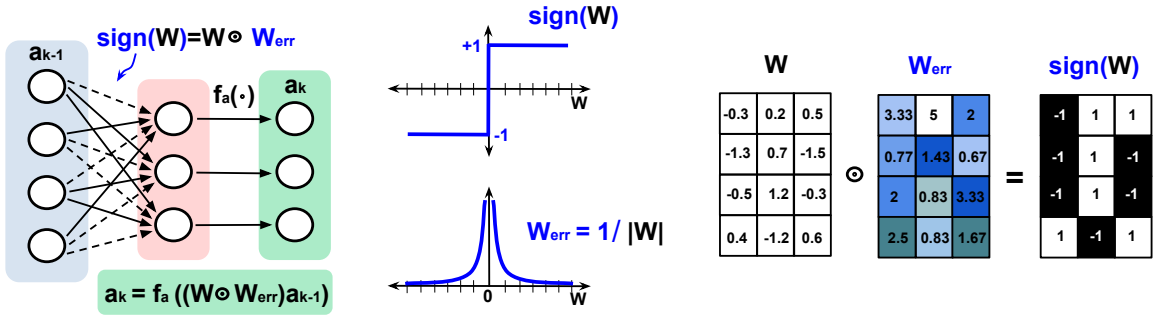


Figure 2.1. Neural network architecture for A-Connect.

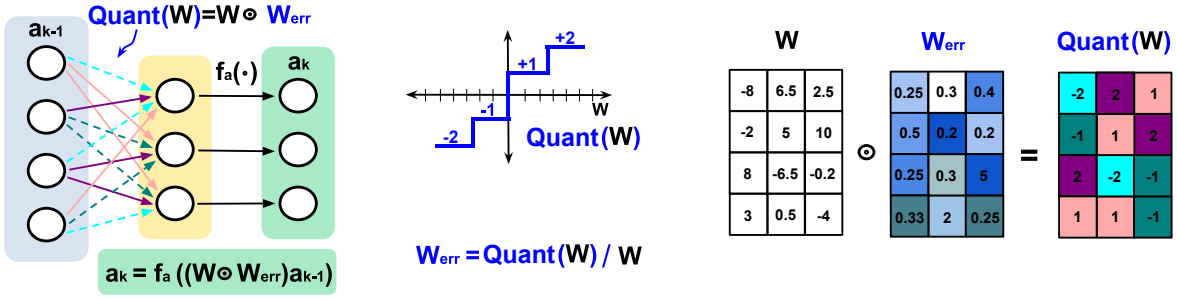
The other advantage of A-Connect is the weights quantization. Before this project, it was possible to binarize the layer weights by applying the operation shown in Figure 2.2.a. To binarize weights, A-Connect gives the value of -1 to each floating value less than 0 and the value of 1 to each floating value greater than 0. That is equivalent to applying the sign function as Figure 2.2.b. shows. After that, the layer makes the matrix multiplication between the weights and the inputs. An error matrix is created by dividing the binary weights between the floating weights. The error matrix is equivalent to 1/|W| as Figure 2.2.b. shows. This error matrix is later used during

the backward propagation process to get the proper gradient for the weights. Figure 2.2.c. illustrates the relationship between the error mask, the floating weights, and the binary weights. As it is shown the binary weights are equal to the multiplication between the floating weights and the error mask.



a. A-Connect for quantization: binary weights **b. W_{err} mapping** **c. Example error mask for binary weights**

Figure 2.2. a. A-Connect for weights binarization. b. Sign function for the weights and error mask. c. Relation between error mask, floating weights and binary weights.



a. A-Connect for quantization **b. W_{err} mapping** **c. Example error mask for 4 levels (2 bits) weights quantization**

Figure 2.3. a. A-Connect for weights quantization. b. Quantization function for the weights and error mask. c. Relation between error mask, floating weights and 2bits quantized weights.

Following this idea, A-Connect can be used for different levels of quantization, applying the methodology with a correct quantization function. Figure 2.3 shows how A-Connect methodology can be used for 2 bits weights quantization, where Quant can be any quantization function with any number of levels.

3. LIBRARY UPDATE DEVELOPMENT

The goal of the library update is to achieve a higher precision accuracy applying the A-Connect methodology. Currently, the A-Connect methodology focuses only on weight binarization as a quantization method, which is an extreme case of quantization. Higher precision applications include multiple quantization levels, not only in weights but in any network parameter. This update would be made possible for the A-Connect library to achieve similar accuracy to its continuous counterparts in different network architectures while still decreasing the storage usage.

3.1. Previous library.

In a previous thesis[15], a library was developed using TensorFlow/Keras, which allowed the use of the A-Connect methodology to mitigate stochastic variability and binarize weights, this library included a fully connected layer and a convolutional layer. Table 3.1 shows the fully connected layer attributes from that library and table 3.2 shows the convolutional layer attributes.

3.2. Quantization methodology.

Some works have implemented more than two levels of parameter quantization in their weight layers. In TWN[10] methodology, the weights are constrained to +1, 0 and -1. The Euclidian distance between full (float or double) precision weights and the ternary weights along with a scaling factor is minimized. In [11] a method of quantization is introduced using the clip function and defining a number of quantization levels with the variable bitwidth. However, the limits of the quantization methodology necessary to achieve the highest accuracy are not defined. Also, for different limit and bitwidth values, the quantization function had an odd number of levels and a different

Table 3.1. Fully connected A-Connect attributes in previous library.

Attribute	**Description**
output_size	The number of neurons or activation you want to use.
Wstd	Float between 0 and 1 for the weights standard deviation for training.
Bstd	Float between 0 and 1 for the bias standard deviation for training.
isBin	String yes or no for weights binarization.
pool	Integer for the pool size of the weights and bias error matrices.
Slice	Integer for batch slicing into 2, 4 or 8 parts.
d_type	Data type of the trainable parameters and error matrices.
weights_regularizer	Regularizer function for the weights.
biases_regularizer	Regularizer function for the biases.

quantization step value depending on the level.

Searching for the correct quantization equation to achieve the highest possible accuracy, the focus was into three items:

- Uniform distribution at weight initialization so that weights have the same probability of being approximated to any quantization level.
- Choosing the limits so that the highest accuracy can be achieved.
- Same step value for any quantization level.

We parted from the clip function for the design of the equation. This function has three variables as entry, the variable to be quantized (In our case, the weights), and the inferior and superior limit. With this in mind, the limits of the clip function were designed so that the quantization took an even number of levels in function of the bitwidth. Then, a shift in X axis and Y axis was applied with the objective of symmetric representation. At last, the function was normalized so that the limits

Table 3.2. Convolutional A-Connect layer attributes in previous library.

Attribute	**Description**
filters	Number of filters to be used during the convolution
kernel_size	Size of the filters
Wstd	Weights standard deviation
Bstd	Biases standard deviation
pool	Number of error matrices to be used during the forward propagation
isBin	String yes or no for weights binarization
strides	Number of strides to be used during the convolution
padding	VALID to reduce the image size or SAME for preserve
Op	1 or 2 to select map_fn or depth-wise convolution for the batch convolution
Slice	Integer 2,4 or 8 for batch slicing
d_type	Data type for the parameters
weights_regularizer	Regularizer function for the weights
bias_regularizer	Regularizer function for the biases

of the quantization were the same as in the A-Connect binary methodology, thus, obtaining the equation shown below:

$$QN(x) = \left(\text{clip}\left[\text{floor}\left(\frac{x}{x_{lsb}} + 1\right), -a, a+1\right] - 0.5\right) \cdot x'_{lsb} \tag{1}$$

Where $x_{lsb} = \text{lim}/2^{bw-1}$, $x'_{lsb} = 2 \cdot \text{lim}/(2^{bw} - 1)$, $a = 2^{bw-1} - 1$ and bw is the quantization number of bits (bitwidth).

In the initial implementation the limits used were the same as the given by the sign function, which were used for the development of the A-Connect binarization methodology. An example of the quantization can be seen in figure 3.1.

A simple network of one deep fully connected layer was trained with the Mnist da-

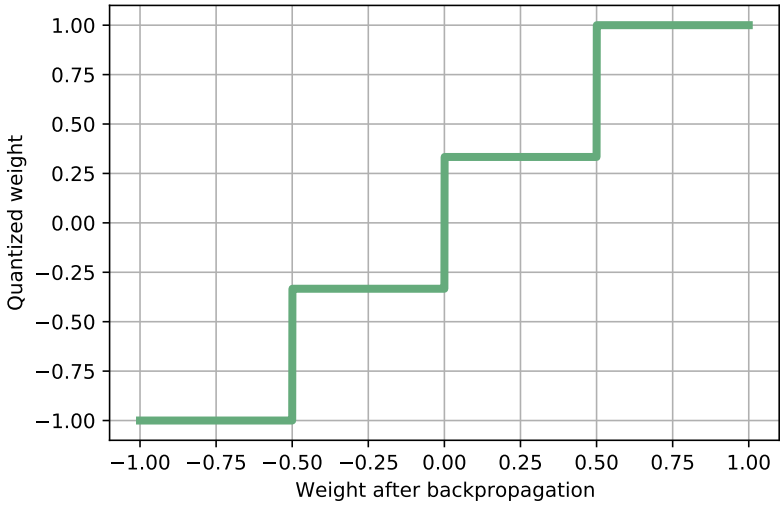


Figure 3.1. 2 bit quantization function with limits -1 and 1.

taset for the validation of the quantization. A set of trainings were made and the quantization was not getting high accuracy results compared to their 32 bit floating point counterparts. Visualizing the initial distribution of the weights showed that the levels closer to zero were gathering all of the weights, making clear that the limits implemented (-1 and 1) were not the best for achieving the highest accuracy possible, so a different number of limits were searched. Having smaller limits for the weight initialization than the used in the first implementation made not possible a uniform distribution at the beginning of the training. Considering Glorot Uniform as the standard form of weight initialization and the used in the A-Connect methodology, it was decided to use the limits given by this function. The initialization draws samples from a uniform distribution within $[-\mathrm{lim}, \mathrm{lim}]$, with:

$$\mathrm{lim} = \sqrt{\frac{6}{\mathrm{fanin} + \mathrm{fanout}}} \tag{2}$$

where *fanin* is the number of input units in the weight tensor and *fanout* is the number of output units. The limits of the quantization are in function of the size of the weight

matrix of each layer. For bias quantization, $\mathrm{lim} = \sqrt{6/\mathrm{fanout}}$.

3.3. Library modifications.

A-Connect uses the error matrices for the backward pass to generate the proper gradient for updating the weights and biases. The custom layer template from Keras does not provide a method to program the backpropagation because TensorFlow handles this by using an algorithm for automatic differentiation. The automatic differentiation from TensorFlow applies the chain rule to the operations in the forward pass to get the gradients of the trainable parameters. To do this, TensorFlow generates a computational graph of the layer, for A-Connect the graph is shown in Figure 3.2.

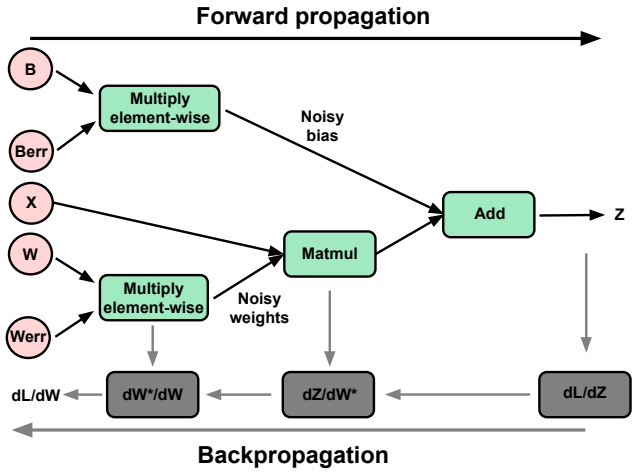


Figure 3.2. Computational graph for an A-Connect fully connected layer.

When the backward propagation begins, the algorithm travels from the last operation to the first one applying the chain rule to the differentiable operations to obtain the gradients. As Figure 3.2 shows, the first gradient to be obtained will be the gradient of the loss function. Then, using the chain rule, the algorithm will get the gradient to the noisy weights from the node Matmul. The node Add will be ignored because it

is the same gradient as Matmul. Finally, from the node multiplication element-wise TensorFlow will get the proper gradients of the weights which is $\frac{\partial L}{\partial W^*}W_{err}$. As shown in Figure 2.3, quantization with Quant function can also be viewed as multiplication by an error matrix of the form: $Werr = \frac{QN(X)}{X}$, where X is the matrix of the parameter to be quantized.

The automatic differentiation is a powerful tool that allows the user not to complicate with gradients. However, to apply the A-Connect methodology, it is necessary to program new gradients. To do this, TensorFlow provides the decorator *tf.custom_gradient*, Algorithms 1 and 2 shows how the weights quantization with A-Connect is done, the first shows the quantization for binary weights and the second for the rest of the levels.

Algoritmo 1: Custom gradient for A-Connect weights binarization. (/) is the element-wise division. To avoid the division by zero a small value ϵ ($< 1e-5$) must be added.

Input: Floating weights
Output: Binarized weights

function SIGN(W)
    $y \leftarrow tf.math.sign(W)$
    function GRAD(dy)
        $dW \leftarrow dy/(abs(W) + \epsilon)$
        return dW
    end function
    return y, GRAD
end function

Since A-Connect changes the gradient of the quantization function, the automatic differentiation will not get the correct value. To solve this, it is necessary to use the custom gradient decorator. Custom gradient replaces the gradient of the operations that are inside the function. For the weights quantization case, the operation with problems is the quantization function (SIGN or QN in Equation 1). The SIGN or QN function must receive as input the weights. Later, the function will do the forward

Algoritmo 2: Custom gradient for A-Connect weights 2 or more bits quantization. (/) is the element-wise division. (bw) bitwidth. To avoid the division by zero a small value ϵ ($< 1e-5$) must be added.

Input: Floating weights
Output: Quantized weights

function QN(W)
    $y \leftarrow QN(W, bw)$
    function GRAD(dy)
        $dW \leftarrow dy * \frac{y}{W+\epsilon}$
        return dW
    end function
    return y, GRAD
end function

operation, for this case is the standard quantization. Inside GRAD, the new gradient should be defined (this gradient is equivalent to quantization error matrix) and then, returned to the main quantization function (SIGN or QN). At the end of the execution, the custom gradient function will return the quantized weights and the new gradient. This new gradient now will be used during the backward propagation with the automatic differentiation. The same process used for weights can be used for bias quantization.

Tables 3.3 and 3.4 show a detailed explanation for the Fully connected and convolutional layers parameters for the updated A-Connect library.

Table 3.3. Fully connected A-Connect attributes in updated library.

Attribute	**Description**
output_size	The number of neurons or activation you want to use.
Wstd	Float between 0 and 1 for the weights standard deviation for training.
Bstd	Float between 0 and 1 for the bias standard deviation for training.
isQuant	list of 2 strings yes or no, first element for weights quantization and second for bias.
bw	list of size 2, first element for weights bit-width and second for bias.
pool	Integer for the pool size of the weights and bias error matrices.
Slice	Integer for batch slicing into 2, 4 or 8 parts.
d_type	Data type of the trainable parameters and error matrices.
weights_regularizer	Regularizer function for the weights.
biases_regularizer	Regularizer function for the biases.

Table 3.4. Convolutional A-Connect layer attributes in updated library.

Attribute	**Description**
filters	Number of filters to be used during the convolution
kernel_size	Size of the filters
Wstd	Weights standard deviation
Bstd	Biases standard deviation
pool	Number of error matrices to be used during the forward propagation
isQuant	list of 2 strings yes or no, first element for weights quantization and second for bias
bw	list of size 2, first element for weights bit-width and second for bias
strides	Number of strides to be used during the convolution
padding	VALID to reduce the image size or SAME for preserve
Op	1 or 2 to select map_fn or depth-wise convolution for the batch convolution
Slice	Integer 2,4 or 8 for batch slicing
d_type	Data type for the parameters
weights_regularizer	Regularizer function for the weights
bias_regularizer	Regularizer function for the biases

4. RESULTS

In this section, we present the results obtained using the updated A-Connect library for parameter quantization. First, we test the quantization with a simple fully connected neural network architecture. Then, we present the results for the convolutional A-Connect layer with popular CNN architectures and compare them with state of art quantization methods. Finally, we present the results of an FPGA implementation of a simple quantized neural network with A-Connect.

4.1. Fully connected neural network.

The first step to test the quantization was to compare the results between a fully connected neural network training with full precision parameters and the same network but quantized for different amounts of bits. In this respect, we used a simple neural network of one hidden layer with 128 activations with ReLU as a non-linear function. We apply batch normalization[18] before the ReLU. This network is similar to the one used in [13].

The experiment consists of training the neural network using A-Connect with weights quantized for 1, 2, 4, and 8 bits, all with binary biases. These four neural networks were trained using two datasets, MNIST[14] 28x28 and MNIST 11x11 both with 8 or 4 bits pixel resolution. The networks were trained with stochastic variability of 0 %, stochastic gradient descent (SGD) optimizer, learning rate of 0.1, momentum of 0.9, batch size of 256, and 20 epochs. Also, we used weights and bias regularizer L2 from Keras with a regularization factor of 0.0001. Table 4.1 shows a comparison

[18] Sergey Ioffe y Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. arXiv: 1502.03167 [cs.LG].

between Full precision network results and quantized networks with A-Connect. As was expected, the results are very good reaching comparable results with the full precision network for the quantized network with 2-bits weights and binary bias.

Table 4.1. Fully connected architecture MNIST results.

Image Size	Pixel Res.	Binary	2Bits	4Bits	8Bits	Base
28x28	8bits	97.11 %	97.59 %	97.65 %	97.58 %	97.90 %
	4bits	96.93 %	97.36 %	97.46 %	97.61 %	97.78 %
11x11	8bits	96.38 %	96.90 %	97.46 %	97.49 %	97.81 %
	4bits	96.11 %	96.87 %	97.40 %	97.48 %	97.80 %

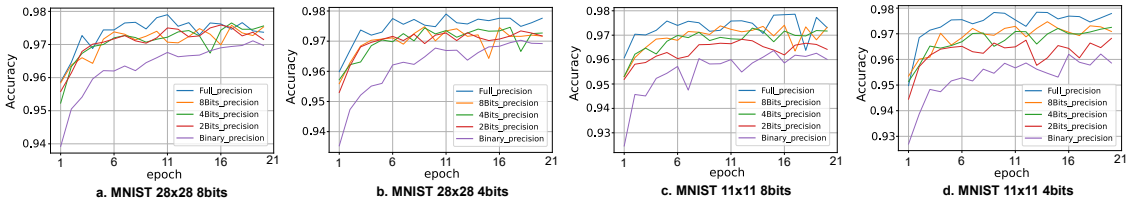


Figure 4.1. Validation accuracy results for: a. Standard 28x28 8 bits pixel resolution MNIST dataset. b. Standard 28x28 4 bits MNIST dataset. c. Downsampled MNIST 11x11 8 bits pixel resolution. d. Downsampled MNIST 11x11 4 bits pixel resolution.

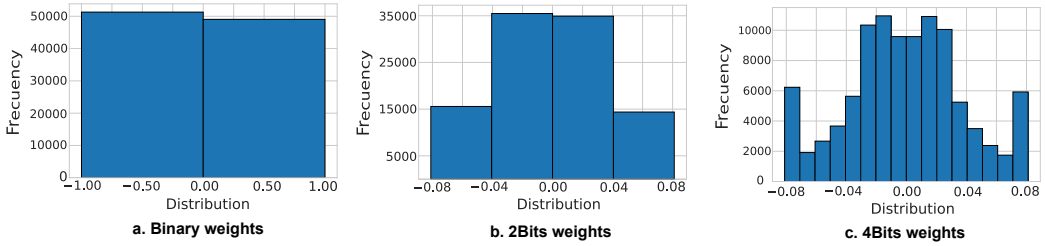


Figure 4.2. Weights histogram for hidden layer in fully connected network architecture: a. Binary weights quantization. b. 2Bits weights quantization. c. 4Bits weights quantization.

Figure 4.1 shows the validation accuracy curves for fully connected network architecture training with different size and pixel resolution MNIST datasets. As shown in the figures BWN (Binary weights network) converges slowly and achieves worse accuracy than 2 or more bits weights networks. However, 2 or more bits weights networks

converge almost as fast and stably as full precision weights networks.

In Figure 4.2 we show a weights histogram, in this figure, it is possible to see how the distribution of weights is in the hidden layer of fully connected network architecture for different bits of quantization. In 4.2.b the majority of the weights can be found in the levels closer to zero, showing that the weights tend to have lower values for achieving a higher accuracy. On the other hand, in figure 4.2.c we can see an accumulation of the weights in the superior an inferior levels, this is given by all the weights that exceed the limits of the quantization. These results were obtained with MNIST 28x28 dataset with 8 bits pixel resolution.

4.2. LeNet-5 with A-Connect quantization.

The convolutional layer presented in section 3.3 allows the users to create any CNN architecture with A-Connect. To test the performance of this layer, we used LeNet-5[19]. This architecture is widely used for handwritten digits recognition, and achieves high accuracy with the MNIST dataset.

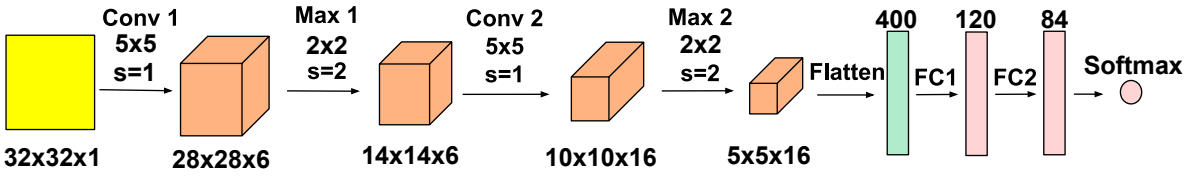


Figure 4.3. Common LeNet-5 architecture.

Figure 4.3 shows the LeNet-5 architecture which was used. Here, we perform the same test used for the fully connected layer. We trained four different neural networks with weights quantized for 1, 2, 4, and 8 bits. For each neural network, we used a TANH activation and batch normalization layers before each TANH. To train the model we used Stochastic Gradient Descent (SGD) with a learning rate of 0.01 (remains

[19] Y. Lecun, L. Bottou, Y. Bengio y P. Haffner. "Gradient-based learning applied to document recognition". En: *Proceedings of the IEEE* 86.11 (1998), págs. 2278-2324. DOI: 10.1109/5.726791.

constant during the training). Table 4.2 compares the results with A-Connect quantization (applied to fully connected and convolutional layers) against the base full precision neural network (using Keras layers). As was expected, A-Connect quantization for 2 bits or more, shows an improvement in the neural network accuracy approaching the base with full precision parameters, even surpassing the base for 4 and 8 bit quantization.

Figure 4.4 shows the validation accuracy curves during training for LeNet5 with MNIST. As shown in the figure, 8bits weights have a good performance, even obtaining better accuracy than the full parameters base network, since quantization can operate as a regularization method.

Table 4.2. LeNet-5 MNIST validation accuracy results.

Quantization	**Validation Accuracy**
Full precision (Float 32)	98.94 %
8Bits weights	99.17 %
4Bits weights	98.95 %
2Bits weights	98.87 %
Binary weights	98.58 %

Image recognition is one of the most popular applications of neural networks. In particular, convolutional neural networks are the ideal architecture to do image recognition and classification. We present here some results using our convolutional A-Connect quantization with a state-of-the-art CNN architecture such as AlexNet using the CIFAR-10 dataset[20]. Also we present results with a residual neural network ResNet20[1] using the CIFAR-10 dataset.

[20] A. Krizhevsky. *The CIFAR-10 and CIFAR-100 datasets*. [Online] Available: https://www.cs.toronto.edu/ kriz/cifar.html/.

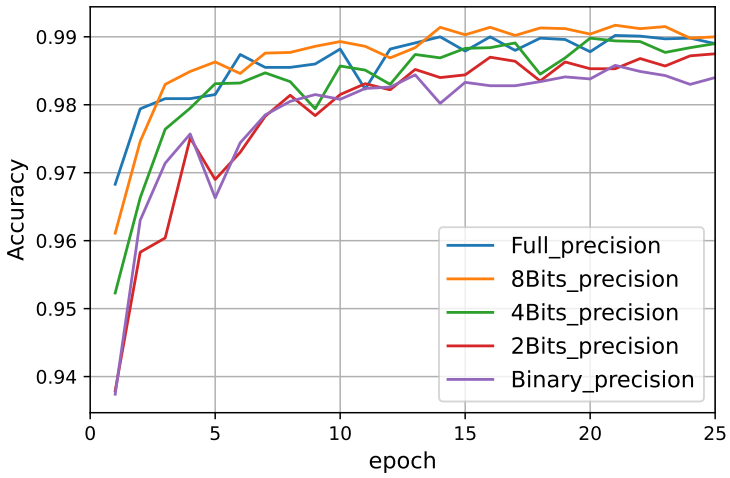


Figure 4.4. Validation accuracy for LeNet-5 MNIST network.

4.3. AlexNet with A-Connect quantization.

AlexNet[2] is maybe the most popular architecture for image recognition since it won the *ImageNet Large Scale Visual Recognition Challenge* in 2012. To test the A-Connect quantization performance in this kind of task we implemented a modified version of AlexNet. The standard architecture shown in Figure 4.5 uses two fully connected layers of 4096 activations before the output layer, but these two layers cause a high memory usage. We replaced these two layers for two of 1024 activations and one more of 512 activations. Figure 4.6 shows the modification to the fully connected layers. Also, we added two dropout regularization layers with a regularizing factor of 0.5, one after the flatten layer and the other after the first fully connected layer of 1024 activations.

To train this architecture, we used the CIFAR-10 dataset, batch normalization layers after each convolutional or fully connected layer (except the output layer), and ReLU as a non-linear activation function. Then, we used a resize layer with size 227x227 to resize the input images (which are 32x32). We trained the model using Stochastic

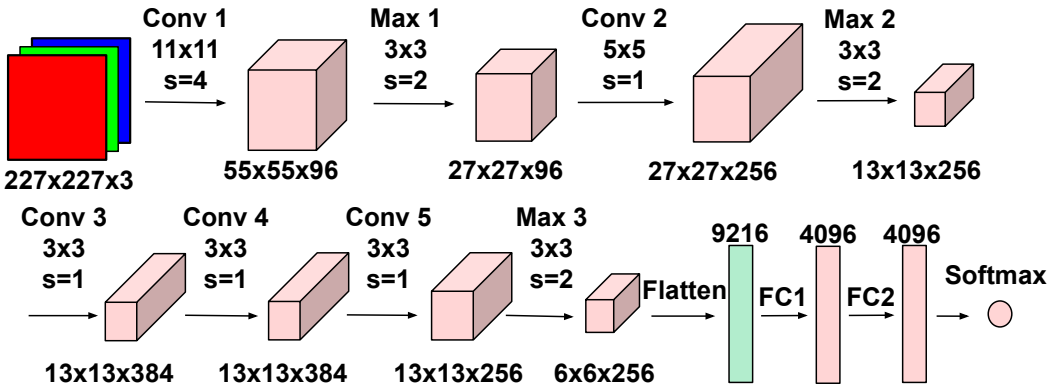


Figure 4.5. Common AlexNet architecture.

gradient descent (SGD) with an initial learning rate of 0.01 which decays by half every 20 epochs, momentum of 0.9, batch size of 256 and we trained the model during 100 epochs. The results for the base neural network and the A-Connect quantized neural networks are summarized in Table 4.3. The results shown in the table correspond to the highest validation accuracy obtained during training. As we expected, A-Connect quantization performance is very good, even obtaining better accuracy than the full precision parameters base network for 8 bits weights quantization. Figure 4.7 shows the validation accuracy curves during training for AlexNet with CIFAR-10 dataset.

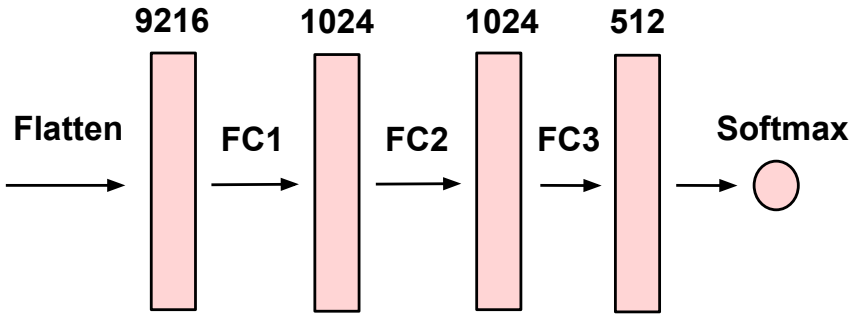


Figure 4.6. Modified fully connected layers in AlexNet.

Table 4.3. AlexNet CIFAR-10 validation accuracy results.

Quantization	Validation Accuracy
Full precision (Float 32)	85.17 %
8Bits weights	85.66 %
4Bits weights	83.67 %
2Bits weights	82.09 %
Binary weights	79.68 %

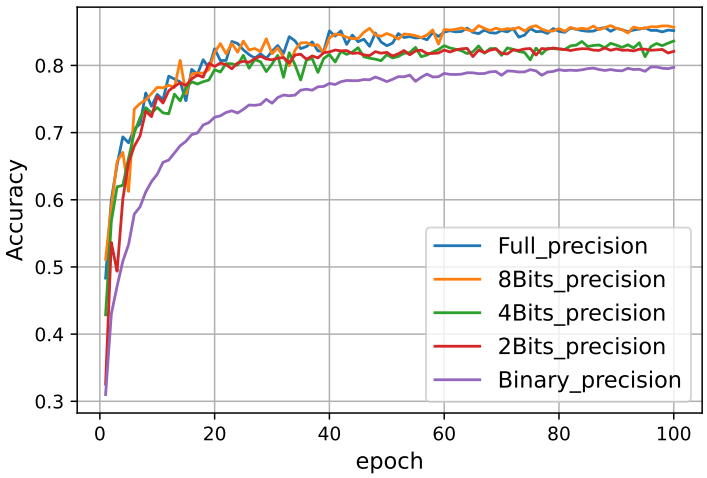


Figure 4.7. Validation accuracy for AlexNet CIFAR-10 network.

4.4. ResNet-20 with A-Connect quantization.

Another popular architecture for image recognition is Deep Residual Network ResNet[1]. Deeper neural networks are more difficult to train. ResNet architecture allows training deeper networks more simply through a residual learning framework, being easy to optimize and gaining accuracy from considerably increased depth. The ResNet-20 consists of a CONV layer followed by 3 ResNet blocks (18 CONV layers with 3x3

filter) and a final FC layer. We used the "BN after additionResNet structure[21], Figure 4.8 shows the ResNet block implemented. For training, we used SGD with a momentum of 0.9 and a learning rate starting from 0.1 and scaled by 0.1 at epoch 80, 120 and 160, and scaled by 0.5 at epoch 180. L2-regularizer with regularization factor of 1e-4 is applied to weight. The mini-batch size of 256 is used, and the maximum number of epochs is 200. We used the CIFAR-10 dataset, 50000 training images and used the 10000 test images for validation. We used data augmentation during training, using Keras layers: RandomFlip horizontal, vertical and horizontal RandomTranslation of 10 %, and vertical RandomZoom of 20 %. The results are summarized in Table 4.4. We achieve a validation accuracy of 91.35 % for the base full precision parameters network, and with 8 bits weights, we achieve 90.78 % pretty close to base network accuracy. Figure 4.9 shows the validation accuracy curves during training for ResNet-20 with CIFAR-10 dataset.

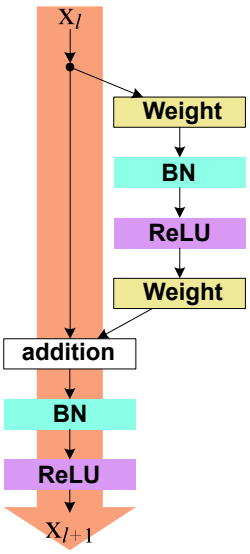


Figure 4.8. BN after addition ResNet Block.

[21] K. He, X. Zhang, S. Ren y J. Sun. “Identity Mappings in Deep Residual Networks”. En: *European Conference on Computer Vision(ECCV)* (2016). arXiv: 1603.05027 [cs.CV].

Table 4.4. ResNet20 CIFAR10 validation accuracy results.

Quantization	Validation Accuracy
Full precision (Float 32)	91.35 %
8Bits weights	90.78 %
4Bits weights	90.15 %
2Bits weights	88.69 %
Binary weights	83.67 %

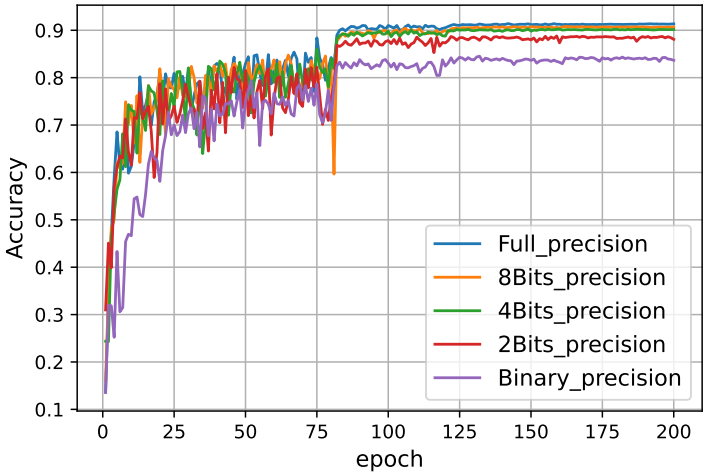


Figure 4.9. Validation accuracy for ResNet20 CIFAR-10 network.

4.5. State of the art comparison.

4.5.1. LeNet5 architecture For comparison with the state-of-the-art, we include most of the recent binary quantized neural networks that used the same experimental settings (including ternary quantizations that effectively use 2-bits) and compare the accuracy numbers from their papers: TWNs, BPWNs[10], BinaryConnect[9] and Binarized Neural Networks[22]. For this, we train a LeNet5 architecture network with

[22] I. Hubara, M. Courbariaux, D. Soudry e Y. Bengio. *Binarized Neural Networks*. 2016. arXiv: 1602.02830 [cs.LG].

A-Connect quantization for 1 and 2 bits weights with MNIST standard dataset, all with binary bias. We used a TANH activation and batch normalization layers before each TANH. To train the model we used Stochastic Gradient Descent (SGD) with a momentum of 0.9 and starting learning rate of 0.01 that scaled by 0.1 at epoch 15. The mini-batch size of 256 is used, and the maximum number of epochs is 25. Table 4.5 shows the results obtained and compared the degradation of the accuracy of quantized neural networks with base full precision networks. With binary weights and bias A-Connect quantization, we achieve comparable degradation with BPWNs and overcome BinaryConnect and Binarized Neural Networks quantization. Also with 2 bits weights and binary bias we won in degradation.

Table 4.5. State of art comparison results with LeNet5 architecture.

	Bit_width (W/B)	Baseline	Quantized	Degradation
A-Connect	**2/1**	**98.95 %**	**98.91 %**	**0.04 %**
A-Connect	**1/1**	**98.95 %**	**98.59 %**	**0.36 %**
TWNs	2/32	99.41 %	99.35 %	0.06 %
BPWNs	1/32	99.41 %	99.05 %	0.36 %
BinaryConnect	1/32	99.41 %	98.82 %	0.59 %
Binarized NN	1/32	99.41 %	88.6 %	10.81 %

4.5.2. ResNet20 architecture It was also fundamental to compare the A-Connect quantization results with architectures that are widely used in the state-of-the-art such as ResNet20. For this comparison, we focused on the accuracy results obtained by two types of ternary weights quantization such as TWNs and TTQ[23], and

[23] C. Zu, S. Han, H. Mao y W. Dally. "Trained Ternary Quantization". En: *International Conference on Learning Representations(ICLR)* (2017). arXiv: 1612.01064 [cs.LG].

Table 4.6. State of art comparison results with ResNet20 architecture.

	Bit_width (weights)	Baseline	Quantized	Degradation
A-Connect	**2**	**91.3 %**	**88.7 %**	**2.6 %**
A-Connect	**4**	**91.3 %**	**90.2 %**	**1.1 %**
A-Connect	**8**	**91.3 %**	**90.8 %**	**0.5 %**
SAWB-fpsc	2	91.8 %	91.6 %	0.2 %
DoReFa	2	91.8 %	90.8 %	1.0 %
LQ-Nets	2	92.1 %	91.8 %	0.3 %
TWN	2	91.8 %	90.9 %	0.9 %
TTQ	2	91.8 %	91.1 %	0.7 %

other methodologies that employ 2-bit weights such as DoReFa-Net[24], LQ-Nets[25] and SAWB[26]. Our network was trained with the same parameters mentioned in section 4.4. Table 4.6 shows the results obtained and compared the degradation of the accuracy of quantized neural networks with base full precision networks. In the table, we see that AConnect 2bW (2-bit weights) has a lower performance than the other state-of-the-art quantizations, this may be because the A-Connect methodology implemented is working with quantization limits fixed per layer. In future works, the performance of the quantization could be improved, making the limit a trainable parameter. On the other hand, AConnect 4bW (4-bit weights) and AConnect 8bW (8-bit weights) have performance comparable to other state-of-the-art quantizations. During the development of the previous library, they found that batch normalization helps A-Connect to regularize the models. For this reason, every model that we used

---

[24] S. Zhou y col. *DoReFa-Net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients*. 2018. arXiv: 1606.06160 [cs.NE].

[25] D. Zhang, J. Yang, D. Ye y G. Hua. "LQ-Nets: Learned Quantization for Highly Accurate and Compact Deep Neural Networks". En: *European Conference on Computer Vision(ECCV)* (2018), págs. 365-382. arXiv: 1807.10029 [cs.CV].

[26] J. Choi y col. "Accurate and Efficient 2-bit Quantized Nueral Networks". En: *Machine Learning and Systems(MLSys)* (2019).

has batch normalization layers inside them. A-Connect is a statistical training methodology that introduces errors to the trainable parameters. This operation modifies a lot of the statistics of the data. Therefore, batch normalization mitigates this effect and then, in some cases, helps the A-Connect models to train properly and even get higher accuracies. The influence of bias on the performance of neural networks is very low compared to weights, for this reason, many of the quantization studies focus on weights, however in this work bias quantization is shown mainly in the most extreme case (binary bias).

4.6. FPGA Implementation.

The main objective of this project is to extend the A-Connect methodology to parameter quantization. The A-Connect quantization was designed to decrease memory footprint without sacrificing accuracy and to be able to implement well performed neural networks in resource limited hardware. For this reason, it is necessary to validate the quantization methodology in hardware. With this purpose, a neural network trained with the A-Connect quantization methodology was implemented on an FPGA. The implemented network consists of a hidden layer with 60 activations and ReLu as a nonlinear function. It was trained on Keras/TensorFlow with binary weights and biases. Additionally, with stochastic gradient descent (SGD), a learning rate of 0.1, momentum of 0.9, 80 epoch, and 32 batch size. We used the MNIST dataset for training and testing, keeping the 8 bit resolution for each pixel but resizing the images from 28x28 to 11x11 pixels. The trained model achieved an accuracy of 85.18 % when validating with the 10,000 images of dataset MNIST for testing.

For implementing the neural network prediction process a Verilog hardware description was made, emulating each one of the processes for obtaining the final result. Figure 4.10 top shows the blocks diagram of the implementation. The module Top-Network contains the description of the neural network, it consists of 121 8-bits inputs

(each one representing a pixel). The binary weights and bias were initialized in the Top-Network. Now, as we are working with binary weights and binary bias, we work with +1 and 0 instead of +1 and -1 as the sign function. The values greater than one are represented by +1 and those less than zero with 0. For the Hidden sub-module, its weights were organized inside of a one dimension array with 60 values. Each value indicates the weights corresponding to the 121 synapses of each neuron in the hidden layer. On the other hand, the biases are defined inside a one dimension array of 60 bits. The 121 8-bits values of the image were arranged into a bit string, where each byte represented a pixel value of the image. For the output layer, a one dimension array with 10 values was organized, the weights matrix having 60 bits, one bit for each value and the bias 10 bits, each one representing a neuron output. A neuron and a ReLu sub-module were described for the hidden layer operations, those can be seen in Figure 4.11, Out_HL is the output of a Hidden layer neuron, W_HL is the weight vector entering the hidden layer neuron and B_HL is the bias associated to the neuron. Inside the neuron sub-module, we perform the multiplication between the image and the weights using mux 2 to 1 for each neuron. Our implementation takes the weights as a control signal for the mux 2 to 1. Thus, this multiplication required fewer resources than a standard multiplication. After this, an accumulative sum of all multiplications is done and another mux 2 to 1 was implemented to add or subtract the bias value then we obtain the output of the layer applying the ReLu activation function on the ReLu sub-module. In the output layer, a different neuron sub-module is applied where the ReLu function is not used. For the prediction, an equivalent to the softmax function is applied, where the maximum value of the ten possible predictions of the network is found and it is taken as the correct prediction.
To test the neural network implemented in the FPGA it was necessary to establish communication between the computer and the neural network. The protocol used for this was SPI (Serial Peripheral Interface). The SPI protocol is responsible for

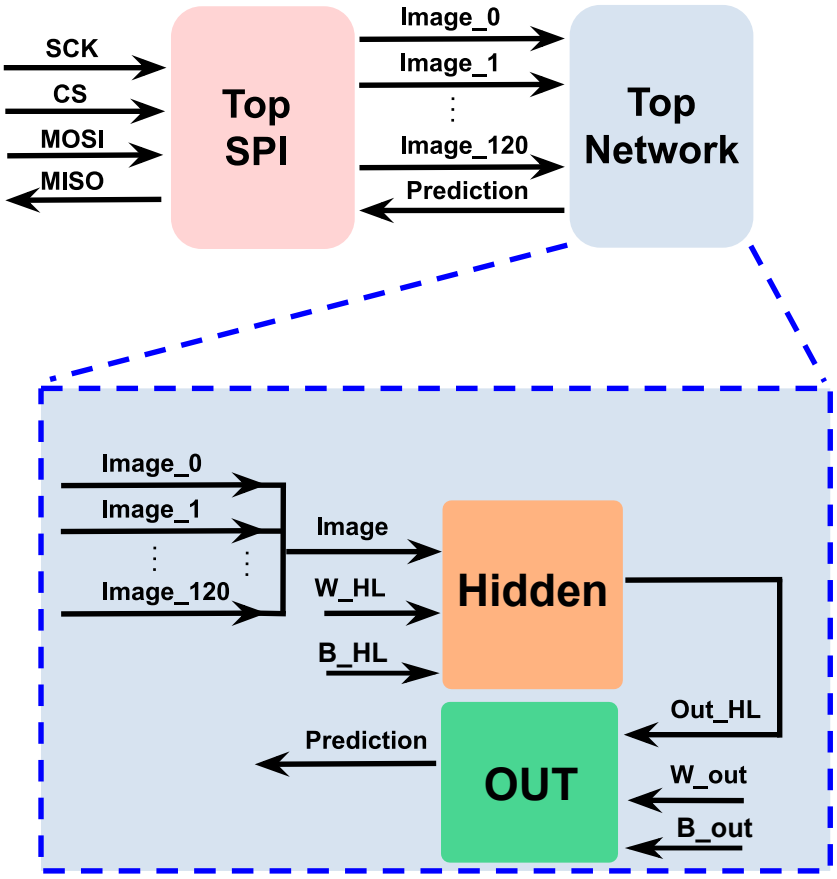


Figure 4.10. Implemented Structure.

sending the input data (image) from the computer to the network and extracting the prediction from the network to the computer. Figure 4.10 shows the complete implementation made in the FPGA including the neural network and the SPI protocol. Top SPI module receives the data coming from the computer and then, through a state machine (which controls the actions of writing the input image to the network or reading the prediction performed) stores the data in a register bank or reads the network prediction of a read-only register. The image stored in the registers will be the inputs of the Top-Network module, and the read-only register will be the input coming from the Top-Network module.

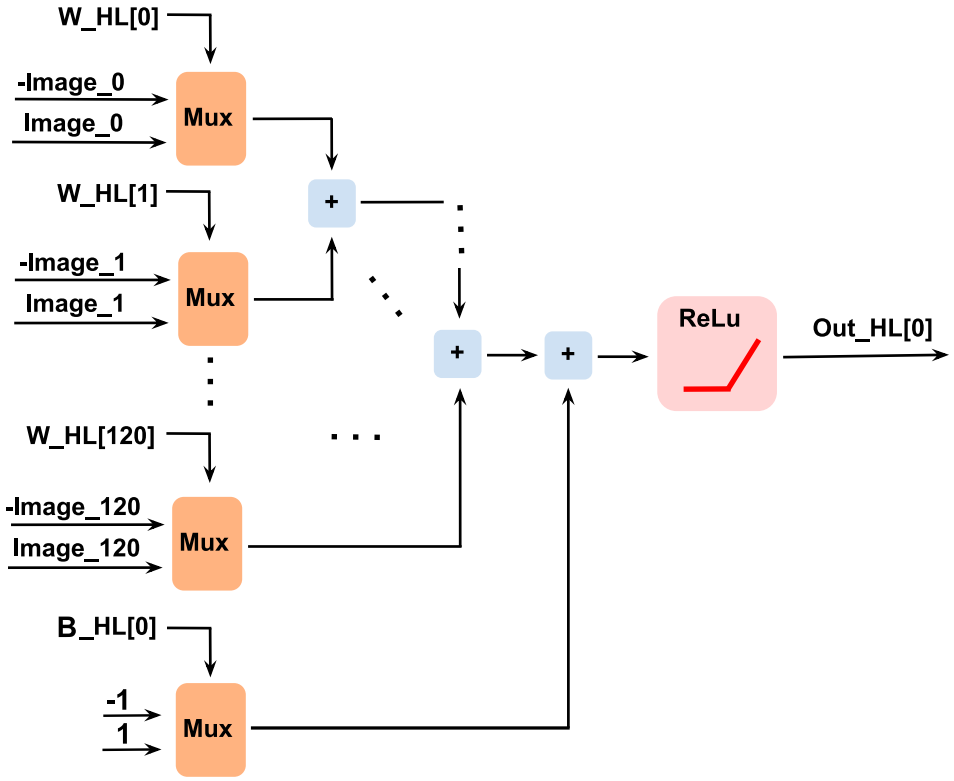


Figure 4.11. Description of a neuron in the FPGA implementation.

The communication between the computer and the SPI protocol implemented in the FPGA is done by USB-Serial conversion cable C232HM[27], which has the four wires used for the protocol (SSK, CS, MOSI, and MISO) and the ground wire. Using a Python script, the input image is transferred through the C232HM wire and then sent through the SPI protocol to the Top-Network module and thus, evaluate the input and determine its respective prediction. Through the Python script, it is possible to extract the network prediction as well.

FPGA network implementation achieves achieves an accuracy up to 85.15 % when we validated the 10,000 images of dataset MNIST for testing. The above shows that

[27] Future Technology Devices International Ltd. *C232HM USB 2.0 HI-SPEED TO MPSSE CABLE Datasheet*. [Online] Available: http://www.ftdichip.com/Support/Documents/DataSheets/ Cables/DS_C232HM_MPSSE_CABLE.pdf.

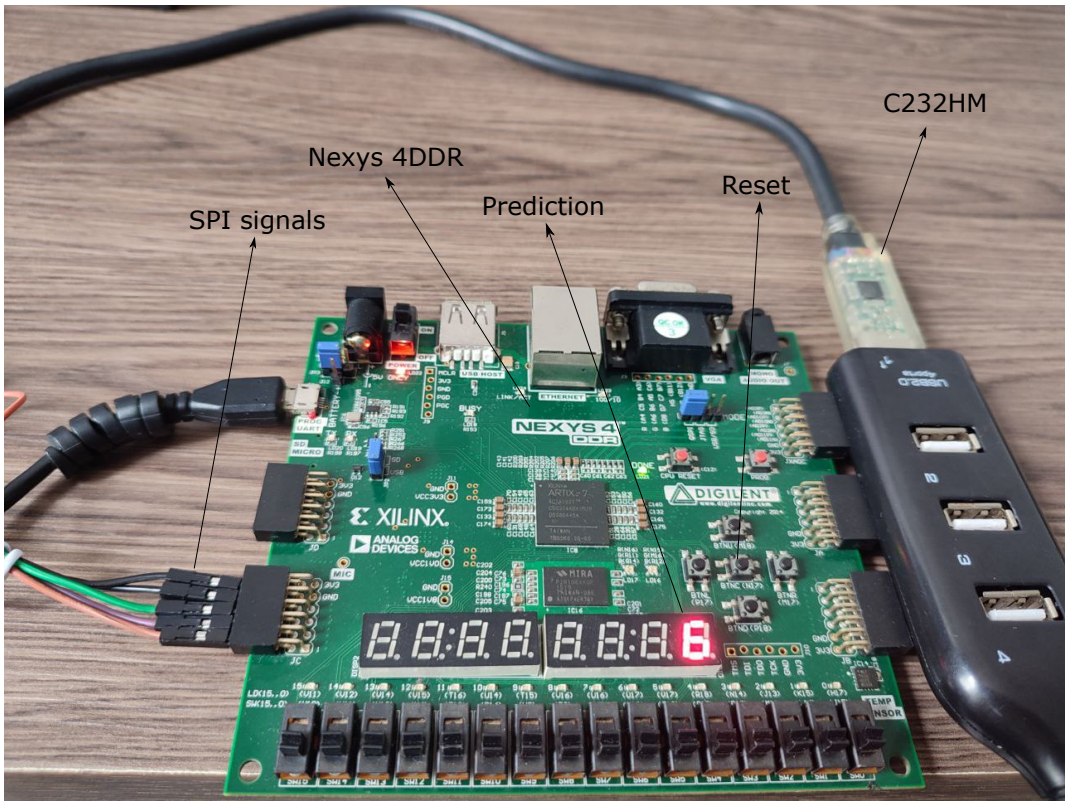


Figure 4.12. Implementation Setup.

our implementation has a closer accuracy to the trained model on Keras/TensorFlow. Therefore, we performed an FPGA implementation that allows recognizing a handwritten digit with high accuracy using A-Connect methodology for parameters quantization with binary weights and bias (quantization more critical case). We used Nexys A7-100T which features the XC7A100T-1CSG324C chip, which contains 450MHZ max clock, 5 push-button, up to 15,850 logic slices, 4 Pmod connectors, and eight 7-segment displays[28]. On the implementation, we only needed one Pmod connector (five GPIO) for the SPI protocol, one 7-segment display that will show the prediction

[28] Digilent. *Nexys A7 FPGA Board Reference Manual*. [Online] Available: https://reference.digilentinc.com/_media/reference/programmable-logic/nexys-a7/nexys-a7_rm.pdf.

obtained, and one push-button for the reset. Figure 4.12 shows the setup implementation on FPGA.

5. FURTHER CONTRIBUTIONS

A-Connect is still in development since the base work (Ph.D. Thesis) is not finished yet. A-Connect needs more work to test the methodology with more complex models and other types of neural networks. Also, A-Connect needs more work on memory usage optimization. Currently, one undergraduate project is working on the methodology.

- **A-Connect for speech recognition**: There are many applications of neural networks, one of that is speech recognition. Speech recognition commonly uses Recurrent Neural Networks (RNN). An A-Connect layer is been created on this work to test A-Connect in this type of application.

Finally, the works mentioned above will be included in a new version of the library.

6. SUMMARY

This paper presented the extension of the A-Connect methodology for parameters (weights and bias) quantization for different bit levels, including an update of the Keras/TensorFlow A-Connect library. We presented satisfactory results for weights and bias quantization in simple neural networks using the MNIST dataset. In addition, we exposed how we achieve excellent performance for weight quantization through the A-Connect methodology for more robust architecture such as AlexNet and more innovative architecture such as ResNet20. We compared A-Connect quantization against some state of the art methodologies using LeNet-5. Better or equivalent performance is achieved with the A-Connect quantization methodology. Finally, a neural network FPGA implementation using binary weights and bias achieves a high accuracy up to 85 %. The library is available in our repository [29].

[29] A. Centeno y col. *Library for A-Connect*. [Online] Available: https://github.com/onchipuis/A-Connect. 2022.

BIBLIOGRAPHY

Centeno, A. y col. *Library for A-Connect*. [Online] Available: https://github.com/onchipuis/A-Connect. 2022 (vid. pág. 45).

Chen, W., J. Wilson, S. Tyree, K. Weinberger e Y. Chen. "Compressing neural networks with the hashing trick". En: *International Conference on Machine Learning(ICML)* (2015), págs. 2285-2294. arXiv: `1504.04788 [cs.LG]` (vid. pág. 11).

Choi, J. y col. "Accurate and Efficient 2-bit Quantized Nueral Networks". En: *Machine Learning and Systems(MLSys)* (2019) (vid. pág. 37).

Courbariaux, M., Y. Bengio y J. David. "BinaryConnect: Training Deep Neural Networks with binary weights during propagations". En: *Conference on Neural Information Processing Systems(NIPS)* (2016). arXiv: `1511.00363 [cs.LG]` (vid. págs. 12, 35).

Digilent. *Nexys A7 FPGA Board Reference Manual*. [Online] Available: https://reference.digilenti
logic/nexys-a7/nexys-a7_rm.pdf (vid. pág. 42).

François Chollet. *Deep learning API Keras*. [Online] Available: https://keras.io/. 2015 (vid. pág. 14).

Future Technology Devices International Ltd. *C232HM USB 2.0 HI-SPEED TO MPSSE CABLE Datasheet*. [Online] Available: http://www.ftdichip.com/Support/Documents/DataShe
Cables/DS_C232HM_MPSSE_CABLE.pdf (vid. pág. 41).

Google brian team. *Machine learning platform TensorFlow*. [Online] Available: https://www.tenso
2015 (vid. pág. 14).

He, K., X. Zhang, S. Ren y J. Sun. “Deep Residual Learning for Image Recognition”. En: *Conference on Computer Vision and Pattern Recognition(CVPR)* (2015), págs. 770-778. arXiv: 1512.03385 [cs.CV] (vid. págs. 11, 30, 33).

— “Identity Mappings in Deep Residual Networks”. En: *European Conference on Computer Vision(ECCV)* (2016). arXiv: 1603.05027 [cs.CV] (vid. pág. 34).

Howard, A., M. Zhu, B. Chen, D. Kalenichenko y W. Wang. *Mobilenets: Efficient convolutional neural networks for mobile vision applications*. 2017. arXiv: 1704.04861 [cs.CV] (vid. pág. 12).

Huang, G., Z. Liu, L. Van der Maaten y K. Weinberger. “Densely connected convolutional networks”. En: *Conference on Computer Vision and Pattern Recognition(CVPR)* (2018). arXiv: 1608.06993 [cs.CV] (vid. pág. 12).

Hubara, I., M. Courbariaux, D. Soudry e Y. Bengio. *Binarized Neural Networks*. 2016. arXiv: 1602.02830 [cs.LG] (vid. pág. 35).

Hubara, I., M. Courbariaux, D. Soudry, R. El-Yaniv e Y. Bengio. “Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations”. En: *Journal of Machine Learning Research(JMLR)* (2018), págs. 1-30. arXiv: 1609.07061 [cs.NE] (vid. págs. 12, 18).

Ioffe, Sergey y Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. arXiv: 1502.03167 [cs.LG] (vid. pág. 27).

Krizhevsky, A. *The CIFAR-10 and CIFAR-100 datasets*. [Online] Available: https://www.cs.toronto (vid. pág. 30).

Krizhevsky, Alex, Ilya Sutskever y Geoffrey Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. En: *Neural Information Processing Systems* 25 (ene. de 2012). DOI: 10.1145/3065386 (vid. págs. 11, 31).

Landola, F. y col. “Squeezenet: Alexnet-level accuracy with 50x fewer parameters and 1Mb model size”. En: *International Conference on Learning Representations(ICLR)* (2017). arXiv: 1602.07360 [cs.CV] (vid. pág. 12).

Lecun, Y., L. Bottou, Y. Bengio y P. Haffner. “Gradient-based learning applied to document recognition”. En: *Proceedings of the IEEE* 86.11 (1998), págs. 2278-2324. DOI: 10.1109/5.726791 (vid. pág. 29).

LeCunn, Y., C. Cortes y C. Burges. *The MNIST database of handwritten digits*. [Online] Available: http://yann.lecun.com/exdb/mnist/ (vid. págs. 13, 27).

Li, F., B. Zhang y B. LiuF. “Ternary Weight Networks”. En: *Conference on Neural Information Processing Systems(NIPS)* (2016). arXiv: 1605.04711 [cs.CV] (vid. págs. 12, 18, 35).

Rueda, Luis y Elkim Roa. *A-Connect: Enabling Imprecise Analog Computation*. not published work (vid. págs. 13, 27).

Simonyan, Karen y Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV] (vid. pág. 11).

Tsai, Hsin-yu, Stefano Ambrogio, Pritish Narayanan, Robert Shelby y Geoffrey Burr. “Recent progress in analog memory-based accelerators for Deep Learning”. En: *Journal of Physics D: Applied Physics* 51 (mayo de 2018). DOI: 10.1088/1361-6463/aac8a5 (vid. pág. 13).

Vergel, R., E. Silva, L. Rueda y E. Roa. *Library development for A-Connect*. Trabajo de investigación Ingeniero Electrónico. Buacaramanga. Universidad Industrial de Santander. Facultad de Ingenierías Físico-Mecánicas. 2021 (vid. págs. 14, 18).

Zhang, D., J. Yang, D. Ye y G. Hua. "LQ-Nets: Learned Quantization for Highly Accurate and Compact Deep Neural Networks". En: *European Conference on Computer Vision(ECCV)* (2018), págs. 365-382. arXiv: 1807.10029 [cs.CV] (vid. pág. 37).

Zhang, X., X. Zhou, M. Lin y J. Sun. *Shufflenet: An extremely efficient convolutional neural network for mobile devices*. 2017. arXiv: 1707.01083 [cs.CV] (vid. pág. 12).

Zhou, S. y col. *DoReFa-Net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients*. 2018. arXiv: 1606.06160 [cs.NE] (vid. pág. 37).

Zu, C., S. Han, H. Mao y W. Dally. "Trained Ternary Quantization". En: *International Conference on Learning Representations(ICLR)* (2017). arXiv: 1612.01064 [cs.LG] (vid. pág. 36).