Convolutional Neural Network (CNN) for Image Classification

Design Acronym:

CNN4IC

Team:

- Jacobo Morales Erazo (Student)
- Martín Calderón (Student)
- Hernando Diaz (Student)
- Daniel Pedraza (Student)
- Mentor: Fredy Enrique Segura Quijano
- Mento: Juan Sebastian Moya Baquero

Chapter/Section: CASS Universidad de los Andes Student Chapter / Colombia Section
Local Supervisor's name: Fredy Segura
Local Supervisor's email name: fsegura@uniandes@uniandes.edu.co

Description of the Design Idea:

A Convolutional Neural Network (CNN) where weights are changeable leverages the ability for offline and extra-low power classification, a commonly complex and memory intensive process. This IC will be built following the principles of convolutional neural networks, involving the convolutional and pooling layers that make these architectures strong image classifiers, while enabling the option of modified via an external connection to adjust the kernel's weights and, in this way, the CNN can be easily adapted to any context. For training the CNN for handwritten number recognition, the chosen dataset is MNIST, which is a subset of the bigger NIST dataset, which includes 60000 samples for training and 10000 samples for tests. During preprocessing, all the images were changed from gray scale into binary scale. The images will come preprocessed from an external device connected with the chip via an on-chip Serial Peripheral Protocol (SPI) that will manage input and output on information from the chip to the external devices. This serial protocol will keep the number of pins in a manageable quantity and simplifies communication while leaving room for the creation of the CNN. Also, the IC will use a small address protocol to dictate with type of information is entering the IC (image input values or weight parameter values).

Going into detail, the CNN architecture will receive images in binary format representing a black pixel as 0 and a white pixel as 1. Also, the image will be preprocessed to fit 28 x 28 pixels, this making in total 784 bits per image which is between 2^9 and 2^10 bits per image this size was decided for making possible the recognition of better-quality images from the normal creation of these CNN. Thus, image resolution makes a realistic objective to achieve proper behavior and isn’t too complex to be manageable by the SPI module which could cause delay in in-chip communication.

The chosen model was based on the LeNet-5 architecture created by Yann LeCun, which was conceived for the classification of the MNIST dataset.

Input -> Convolution layer 1 (5x5xn)-> Max pooling (2x2)-> Convolution layer 2 (5x5xm)-> Max pooling (2x2)-> Flatten -> Fully connected layer -> Output

To create a model as compact as possible, the third convolutional layer and the first fully connected layer were dropped due to their heavy influence on increased amounts of parameters.

Using this as our standard, the following methodology was used for the comparison of the models. Training was carried out using the whole 60,000 image dataset using out of sample tests. Using a small-scale Monte Carlo approach, 10 iterations for 10 models were done. Each one was trained on a basis of 10 epochs using categorical cross entropy as our loss function and stochastic gradient descent as our optimizer. ReLu was used as the activation function for all the layers except for the fully connected layer which uses SoftMax. From the LeNet-5 architecture, n = 6 and m = 16, however, these will be the independent variables to evaluate the relationship between the number of parameters and the achieved training and test accuracy. Due to the lack of space to represent the data obtained we will have to jump directly onto the analysis and conclusions; however, this is available in the Design Example run section. The accuracies tend to increase slightly in a linear pattern when bound to bigger amounts of n and m keeping the value over 93%, additionally, the number of parameters grows in the same matter as well. With this trend and trying to keep the parameters around the 10,000 parameters margin, the values for n and m were kept at n = 6 and m = 16 proving to be balanced. In this sense, the chosen architecture is constructed as follows:

Input -> Convolution layer 1 (5x5x6)-> Max pooling (2x2)-> Convolution layer 2 (5x5x16)-> Max pooling (2x2)-> Flatten -> Fully connected layer -> Output

This architecture achieved a training accuracy of 98.65%, a test accuracy of 98.05% and parameters count of 10,422.

Hardware centered simulations

After a software version of the CNN was developed to test the general behavior of the architecture, a compact model was developed to mimic the IC in terms of register usage the bit-width constraints associated. The first approach encapsulates the quantization of the model to use 4-bit weight values and 8-bit bias values effectively reducing the maximum register size to 9 bits. Using Pytorch and Brevitas libraries a quantization aware model was developed to guarantee the fulfillment of the bit constraints established and to train the model accordingly, at this point the quantization process has been established for the convolution and activation layers while pending on the fully connected one which represent a further challenge. Diagrams and simulations can both be found with an explanation on the attached files, nevertheless. This approach allows us to gather reliable weights and biases to hard code on the CNN as well as the necessary scaling coefficients to reduce the cost of quantization operations, additionally, we will be able to effectively scale up or down the size of our model with these simulations to be able to fit the architecture within the area constraints.

Max activation function

Considering the output of the Fully connected layer is a vector of N elements V = [V_1, V_2, ..., V_N], where N corresponds to the number of labels in the dataset (which in this project equals to nine), it was possible to formulate an activation function f(V_i) for predictions based on the Softmax activation function, defined as:

Softmax(V_i) = exp(V_i) / Σ_{j=1..N} exp(V_j)

where exp(x) represents e^(x).

According to this definition, Softmax maps V into a vector of probabilities that sum to 1. Here the predicted class is obtained by taking the index i that maximizes the softmax(V_i). In other words: Predicted class = argmax_i softmax(V_i).

Moreover, Σ_{j=1..N} exp(V_j) will be the same constant for every Softmax(V_i). So argmax_i softmax(V_i) = argmax_i{ exp(V_i) / Σ_{j=1..N} exp(V_j)} = argmax_i{ exp(V_i)}. So, it is possible to firstly set f(V_i) as f(V_i)= exp(V_i).

Finally, since the exponential function is strictly increasing, the predicted class can be equivalently obtained by directly taking the index of the maximum logit, without explicitly computing the softmax. So, argmax_i (exp(V_i))= argmax_i (V_i) . Finally, f(V_i) can be set as the activation function f(V_i)=V_i, where the predicted class is obtained by taking the index i that maximizes V_i.

Convolution and Pooling Layer

In the convolution and pooling section of architecture, there are challenges related to memory management and the dynamic range of the mapping tensors. Although the weights are stored in 8-bit integer precision, the results of the first convolution may require up to 14 bits to be represented. If this wider precision were to propagate through the rest of the architecture, it would force all subsequent registers and datapaths to also use 14-bit (and later even 22-bit) widths, significantly increasing resource usage.

To avoid this, the convolution output is temporarily stored in a dedicated register bank sized to hold the required wider precision. The results are then rescaled and quantized back to 8-bit integers before continuing to the next stage. While this reduces precision, it was considered a better trade-off compared to the additional area and resources that would be consumed by maintaining wider registers across the entire datapath.

After rescaling, the matrix passes through the first max-pooling stage. The second convolution produces tensors that can reach values requiring up to 22 bits, so the same strategy is applied: results are held in a wider temporary register bank and then rescaled to 8 bits. Finally, the matrices go through the second max-pooling stage, after which the final mapping tensors are ready for the fully connected layer.

Estimated Number of Pins:

- **Input: 5 (SCLK, SS_n, MISO, Vcc, GND)**
- **Output: 1 (MOSI)**
- **Bidirectional: 0**

Design Type: Digital

Expected Outcome:

It is well known that many students are eager to learn mathematics. However, students with upper-body disabilities often face significant challenges when it comes to paying attention while simultaneously taking notes, an essential step in the learning process. In Colombia alone, nearly three million people live with this condition. To help create a more inclusive and supportive learning environment, this project proposes the development of an integrated circuit (IC) capable of recognizing handwritten numbers. This represents an important milestone toward building more advanced systems capable of identifying full mathematical expressions and storing them, thereby reducing the burden of rapid notetaking.

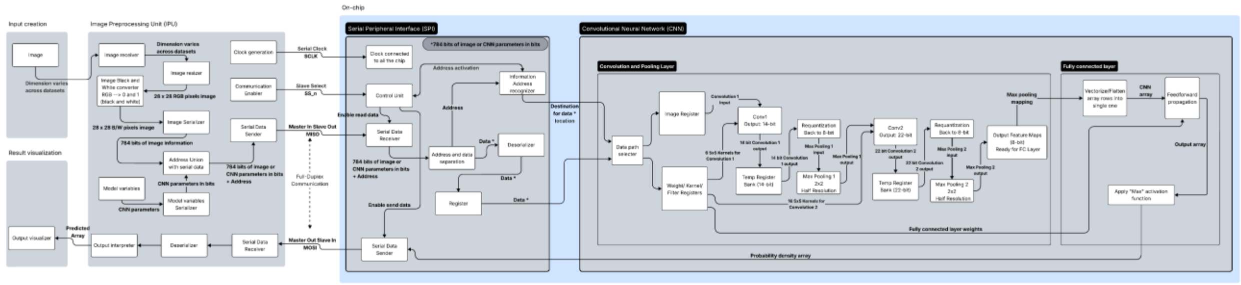
The proposed system is composed of two main components: an Image Preprocessing Unit (IPU) and a Convolutional Neural Network (CNN) chip, which will communicate through the Serial Peripheral Interface (SPI) protocol. The IC will be able to receive an image representing a number, process it through the IPU, and return a predicted number as output. This modular design ensures both efficiency and scalability, making it possible to extend the system toward more complex recognition tasks in the future.

Beyond its technical scope, this project represents an outstanding opportunity for our team. From a technological perspective, it allows us to take a significant step forward in innovation and design, particularly in the implementation of convolutional neural networks. At the same time, it challenges us to strengthen our teamwork abilities, which are essential for ensuring high performance in collaborative projects. Together, these aspects make the initiative not only impactful for accessibility in education but also transformative for the professional growth of our group.

References:

[1] DANE, El diamante del cuidado frente a la experiencia de la discapacidad en Colombia: Una aproximación a los requerimientos diferenciales de las personas con discapacidad y de sus propios cuidadores en 2021, Nota Estadística No. 1 de 2023, Apr. 2023. [Online]. Available: https://www.dane.gov.co/files/investigaciones/notas-estadisticas-casen/abril-2023-DiscapCuidadores.pdf

Block Diagram:


Input creation
Image
Dimension varies across datasets
Result visualization
Output visualizer
Predicted Array
Image Preprocessing Unit (IPU)
Image receiver
Dimension varies across datasets
Image resizer
28 x 28 RGB pixels image
Image Black and White converter RGB --> 0 and 1 (black and white)
28 x 28 B/W pixels image
Image Serializer
784 bits of image information
Address Union with serial data
CNN parameters in bits
Model variables
CNN parameters
Model variables Serializer
784 bits of image or CNN parameters in bits + Address
Clock generation
Communication Enabler
Serial Data Sender
Output interpreter
Deserializer
Serial Data Receiver
Serial Clock SCLK
Slave Select SS_n
Master In Slave Out MISO
Full-Duplex Communication
Master Out Slave In MOSI
On-chip
Serial Peripheral Interface (SPI)
*784 bits of image or CNN parameters in bits
Clock connected to all the chip
Address activation
Control Unit
Information Address recognizer
Address
Enable read data
Serial Data Receiver
Address and data separation
Data *
Deserializer
784 bits of image or CNN parameters in bits + Address
Data *
Register
Data *
Enable send data
Serial Data Sender
Destination for data * location
Convolutional Neural Network (CNN)
Convolution and Pooling Layer
Data path selecter
Image Register
Convolution 1 input
Conv1 Output: 14-bit
14 bit Convolution 1 output
6 5x5 Kernels for Convolution 1
Weight/ Kernel/ Filter Registers
Temp Register Bank (14-bit)
14 bit Convolution 1 output
Requantization Back to 8-bit
Max Pooling 1 input
Max Pooling 1 2x2 Half Resolution
Max Pooling 1 output
16 5x5 Kernels for Convolution 2
Conv2 Output: 22-bit
22 bit Convolution 2 output
Temp Register Bank (22-bit)
22 bit Convolution 2 output
Requantization Back to 8-bit
Max Pooling 2 input
Max Pooling 2 2x2 Half Resolution
Max Pooling 2 output
Output Feature Maps (8-bit) Ready for FC Layer
Max pooling mapping
Fully connected layer weights
Probability density array
Fully connected layer
Vectorize/Flatten array rows into single one
CNN array
Feedforward propagation
Output array
Apply "Max" activation function