Spring 2022

California State University, Northridge

Department of Electrical & Computer Engineering

Final Project

Neural Network IPs

May 09, 2022

ECE 526

Written By: Christopher Y. Chen

Graphical user interface, text, application

Description automatically generated

**Definition:**

Artificial Neural Network is part of the machine learning that utilizes artificial neurons to mimic human braining decisions and processing. The signals will be process into the neuron, when through the activation function which usually are non-linear function of the sum of the inputs. The base on the weight of the neurons and edges, the computer will output the most accurate prediction of its own. Multiple neurons will form a layer of neuron networks. Different transformation will conduct in each layer all the way to the destination layer which is the output layer.

**Neural Network**:

Diagram, schematic

Description automatically generated

**Training:**

The difference between the processed output of the network and the target output is the error. The error will be elimated as the sample data grows bigger. The network will ajusted to the error, which is called supervised learning. Thhis learning algorithm analyzes the training data and prodcuted a relative function to mapping new example until the accuracy reach to a satisfactory level. Such action of learning are considered as examples, that will be used as data base for next prediction.

**Convolutional Neural Network:**

Converlutional Network is one of the many artificial neural network most used for visual image recongintion. The system will weight each edges of the images with certain weight and process it with acitivation function which is called feature mapping. It consist of many layers such as convolutional layers, pooling layers, and fully connected layers. The convolutiona layer offers the feature map with shape formula: number of inputs \* feature map height \* feature map width \* feature map channels. Pooling layber reduced the dimension of the data for less accuracy but much more efficient. The fully connected layers connect every neuron in a layer to another layer to classify the image at the end.

**Fully Connected Layer Neuron:**

Diagram

Description automatically generated

In this project I built a neuron as part of the fully connected layer. Firs the weighted weight will be added into the weighted memory, then it will be multiplied with actual inputs. Then we will it will pervious sum output. At the last stage, you will add the bias, and the activation function will sums up all the sum outputs and send it to the next neuron.

First, we will construct weighted memory. This weight memory will be used to store the weight value and later multiple with your weighted value to form a combined weight value for later operations. The module will take in clock value, a read enable, a read address, and a write output. the weighted memory is also parameterizes based on the needs of the input. You may customize the neuron number to be in certain layers or position. The data width and the address width can also be adjusted but has to match each others.

Graphical user interface, text, application, email

Description automatically generated

The memory will be an array, it will load 3 weighted value into the array with data length to be 16. It will step through each assignment in the weight File “weightfif.mif”. And as long as the read enable is positive, the memory will read the current memory output at the current address line and put it in write output.

Then we will construct the neuron.

Text

Description automatically generated

The Neuron is also sizeable. The parameter you established here will be the value be putted into all is sub-associated file, the weight Memory and the sig\_Rom or relu module. The neuron will first load weight values into the weighted memory we mentioned in pervious page. The testing condition will then put the current weight value associate it with respective neuron.

Graphical user interface, text, application, email

Description automatically generated

Text, letter

Description automatically generated

The order of process will go through sequentially. First you will process the input value. Then after the input, you will be dealing with weights. If the weights are loaded, then you will process into the multiplication. Then you will do a sigmoid activation function. After conduction the activation function, you will output the result.

Text

Description automatically generated

The above code shows load bias value into the register we declared earlier for final addition usage.

Text

Description automatically generated

The following code shown how the arithmetic function needs to take care of the underflow, overflow, and final value cases to conduct the operation. If the multiplier is available, then the circuit will under that this is not the last stage and process the test if the arithmetic operation will cause any type of underflow or overflow and taking care of it by either make it the largest negative or positive value. If none of the underflow or overflow has happened. Then the system will do the multiplication and addition regularly. If that this is the last operation before the send to the activation function, the system will add the bias on top of the arithmetic operation and still take care of the overflow and underflow cases. Once the output is sent or if rest button is triggered, the sum will be cleared.

Graphical user interface, text, application

Description automatically generated

The sigmoid activation function for sigmoid is precalculated because it requires extensive hardware to do it in firmware. But it’s much easier to just have the function calculated in software and then loaded into a ROM and ready for use.

Graphical user interface, text, application

Description automatically generated

Chart, line chart

Description automatically generated

The activation formula for sigmoid is just 1/(1+e^(-x)), just remember that the sigmoid function is a nonlinear activation function for most of the neural network to capture the features. The reason why we choose the sigmoid activation function is that a small change in the input will causes a relatively small change in the output, it is also bounded differentiable and real function with provide flexibility for further transportation.

**Ratified Linear Unit Activation Function:**

The ratified linear unit Activation Function is a linear function that rules out all the zero inputs and only outputs all the non-zero linear outputs. It provides advances such as sparse activation, which eliminates almost half of the outputs to increase efficiency of the neural network. It has better gradient propagation that only saturate the larger value and eliminate the small value so doesn’t cause error in training the neural network.

Graphical user interface, text, application

Description automatically generated

**Why we pack/no pack it as an IP:**

**Pros:**

The IP is highly efficient for implementation hardware, this makes it easier for the company to massive produce, efficient for the embedded FPGA to access the set data.

**Cons:**

Less flexible, you can not change the parameter later, but you must change many hardware specifications to meets the needs of the changing.

**In Conclusion:**

Neural Network is a interesting and complex topic to study. It mimics human behaviors might benefit future applications to decide what is the best outcome in a very short amount of time. In the FPGA, if the neural network is embedded onto the chip, the overall system will consume less power, data, and will have faster processing time. However, it gave us less flexibility in terms of modifying the neural network after it’s been produced or adopt to the newly evolved environment or requirements.