Place for Landon

0.1 Case Study 1: Multilayer Perceptron

The purpose of this case study was to develop an understanding for how the multilayer perceptron (MLP) neural network architecture is affected by a *bit muting* attack function that seeks to manipulate the values of a floating-point number at a binary level. In the case of this experiment, we have chosen to model a function that forces the most-significant-bit (MSB) in the exponent of floating-point number to be muted to 0. In each MLP model, numerical values were stored as double precision floating-point numbers as per the IEEE 754 standard. Note that the exponent value was set to 0, rather than subtract 1023 to save further computation time and avoid invalid ranges when converting the mantissa and exponent back into it's floating-point counterpart.

When mute MSB attack occurs on a single floating-point number, it is then constrained to lie on the open interval $(-1, +1)$. A map of input values against MSB output values is shown in fig. (0.1). The inputs are a subset of the possible values of a double precision floating-point numbers for visual ease.

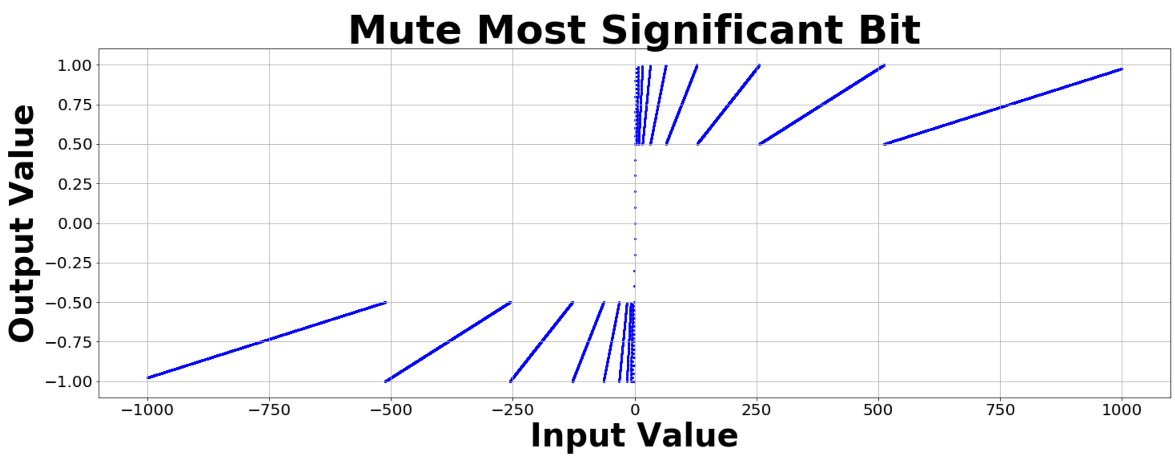


Figure 1: Input versus output values of float-ing point numbers subject to the *mute most-significant-bit* attack function

0.1.1 Experimental Setup

The MLP neural network model is composed of multiple layers of *neurons*, that each contain a real-valued floating-point number called an *activation*. Layers of neurons interact with the neurons in adjacent layers in sequence though a series of weighting coefficients, and bias elements, which is then subjected to an activation function. In general, given a layer of activations $\vec{x}^{(l)}$ (The superscript is used as a layer index), the activations in the next sequential layer, $\vec{x}^{(l+1)}$, are produced with weighting matrix $\hat{W}^{(l)}$, bias vector $\vec{b}^{(l)}$, and activation function f such that:

$$\vec{x}^{(l+1)} = f\Big(\hat{W}^{(l)}\vec{x}^{(l)} + \vec{b}^{(l)}\Big) \tag{1}$$

The recurrence of equation (1) is used to pass information forward through the network. In a network with L layers, raw information is given with $\vec{x}^{(0)}$ and the network's prediction is given by the final layer, $\vec{x}^{(L-1)}$.

The Multilayer Perceptron network must then find a series of parameters, which are the elements in all weighting matrices and all bias vectors, that allows for a given sample to produce a low value for a given *Loss function.* The loss function measures the difference between a networks given output and expected output; with better trained networks producing generally low loss values over a set of samples. The process of finding the values is called *optimization.* There are many different algorithms that perform optimization, and for this experiment, we have chosen to use *Stochastic Gradient Descent* (SGD) due to its widespread use and versatility.

To model a bit-muting attack, we introduce and *attack function* that acts in the matrix-vector product $\hat{W}^{(l)}\vec{x}^{(l)}$ in eqn. (1). When a Boolean *trigger condition* parameter is met, the MSB attack as described above, eqn. (1) is instead replaced with it's attack variant:

$$\vec{x}^{(l+1)} = f\Big(A\big[\hat{W}^{(l)}\vec{x}^{(l)}\big] + \vec{b}^{(l)}\Big) \tag{2}$$

Where A is the attack function, applied element-wise to each object in it's argument. Equation (2) is then producing a perturbation from expected values that is not explicitly accounted for in the optimization procedure.

For a practical demonstration of this attack function, we have tested it on variants of an image classification neural network. Each network was given of subset of training images from the MNIST data set, each containing a pixelized handwritten digit, labeled 0 through 9 according to the digit in that image. Each model was then trained on 6400 similar, but non-identical images for evaluation. A subset of samples is visualized with a binary color map in fig. (0.1.1)

Each sample is 28×28 pixels, each containing by a single integer 0 through 255. Images were shaped shaped into a 784×1 array object and used as input into the network. This means for each network model, there are always 784 input neurons and 10 output neurons (one for each class). To test different model architecture complexities, we tested networks containing 1, 2, 3, and 4 hidden layers (referred to as *network depth*) and used 20, 40, 60, 80, 100, and 120 neurons per layer (referred to as *neuron density* or *network width*). Permutations of these parameters allowed for us to record behavior over 24 unique MLP model variants.

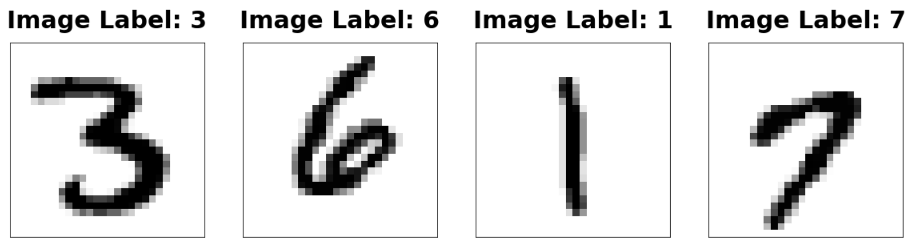


Figure 2: Four MNIST figures shaped as 28 pixel by 28 pixel images, colored with a binary color map along with their corresponding labels.

0.1.2 Impact of Bit-Muting Attack Classification Metrics

To measure the impact of this type of attack on the image classifier model, we test a baseline classifier against a classifier subject to a random 50/50 trigger condition. From each variant, we record the following metrics

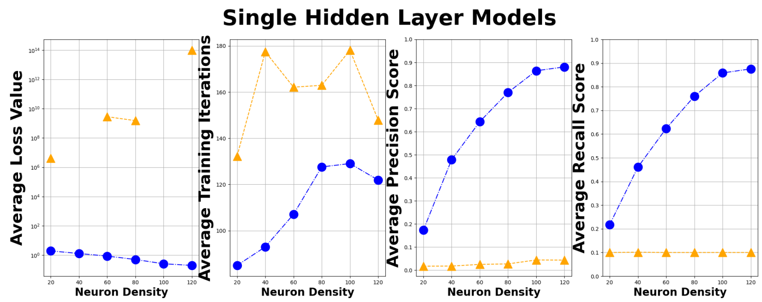
1. Accumulated loss function for the final iteration.
2. Iterations it took to train the model (Out of a Maximum possible 400)
3. Average precision score all 10 classes (Bounded $[0, 1]$, higher is favorable)
4. Average recall score for all 10 classes (Bounded $[0, 1]$, higher is favorable)

In each of the 24 model variants (4 layer depths, 6 densities), 50 models were trained, with the average, minimum and maximum values for each of the four metrics noted. The average for each metric, across each model depth and neuron density is plotted in fig. (0.1.2). Note that the lack of a data point means that the value of that metric diverged to infinity or some other undefined value.

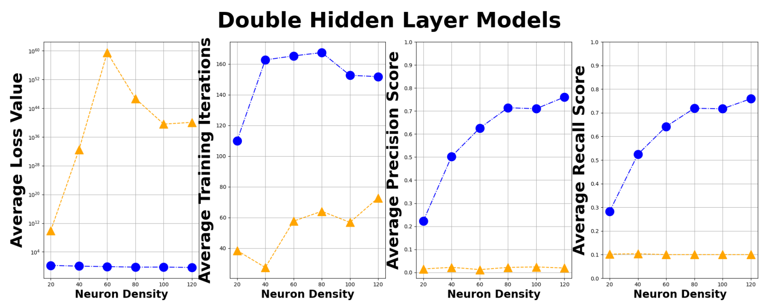
0.1.3 Conclusions from Bit-Muting Attack

The accumulated loss function value measures the difference between expected

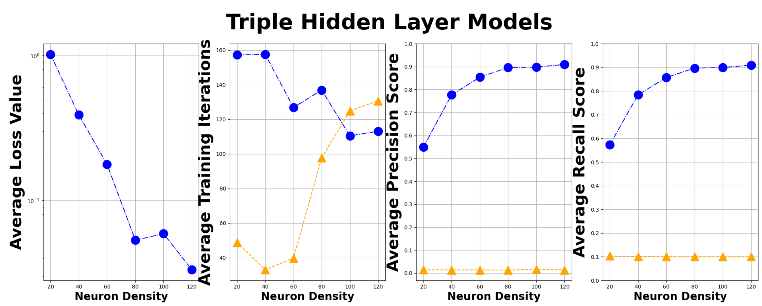
Examining all model depths and all neuron densities, the baseline model shows a substantially higher precision and recall score than the corresponding attacked model. These differences are shown the two right-most subplots in each row. These classification metrics are commonly used together to quantify a network's ability to match samples with it's known target label in a validation set. Both are bound between 0 and 1, with a higher value indicating a better performing network.


Single Hidden Layer Models
Average Loss Value
Average Training Iterations
Average Precision Score
Average Recall Score
Neuron Density

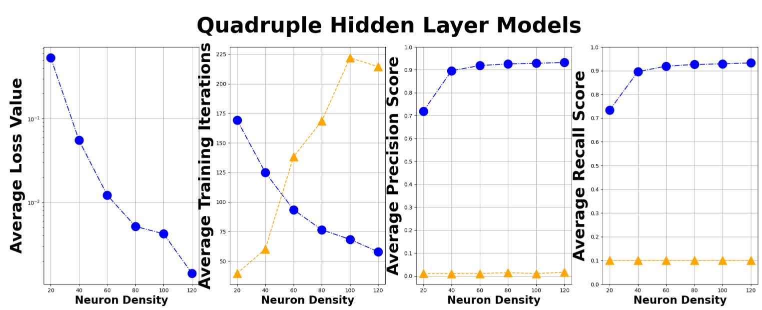
(a)


Double Hidden Layer Models
Average Loss Value
Average Training Iterations
Average Precision Score
Average Recall Score
Neuron Density

(b)


Triple Hidden Layer Models
Average Loss Value
Average Training Iterations
Average Precision Score
Average Recall Score
Neuron Density

(c)


Quadruple Hidden Layer Models
Average Loss Value
Average Training Iterations
Average Precision Score
Average Recall Score
Neuron Density

(d)