

Simon Chaisouang

[shchaiso@uark.edu](mailto:shchaiso@uark.edu)

010878218

CSCE 5613 Artificial Intelligence

Spring 2025

Late Days: 0

## 1 – Binary Network

**Introduction**

For this section we are tasked with creating network architecture for a binary network. The data is generated using four different types of binary operators: and, or, xor, and nor. The truth table is given below for each binary operator.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x | y | x and y | x or y | x xor y | x nor y |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 |

**Table 1. Truth Table of Binary Operators**

**Network Architecture**

The binary network is defined as 1 input layer, 2 hidden layers, and 1 output layer. The first layer takes an input size of 2 for two binary inputs and maps the input to an output size of 4. The ReLu (Rectified Linear Unit) activation function is applied on the first hidden layer. This layer introduces non-linearity to the network to learn more complex mappings such as the XOR and NOR binary operations.

The next hidden layer takes an input size of 4 and maps it to an output size of 1 which represents the network’s confidence that the binary inputs belong to a class of 0 or 1. The output layer applies a sigmoid activation function which converts the network’s output into a probability. The network is optimized with a learning rate of 0.01 and 1000 training steps.

**Results**

The following graphs and information below outline the results of the binary network for each of the four operators: AND, OR, XOR, and NOR.

A white board with blue lines

AI-generated content may be incorrect.

**Figure 1.1 Graph of Training Loss for AND Operator**

For the AND operator, the final accuracy was calculated as 0.75 (75%).

A white board with blue lines

AI-generated content may be incorrect.

**Figure 1.2 Graph of Training Loss for OR Operator**

For the OR operator, the final accuracy was calculated as 0.75 (75%).

A white rectangular object with blue lines

AI-generated content may be incorrect.

**Figure 1.3 Graph of Training Loss for XOR Operator**

For the XOR operator, the final accuracy was calculated as 0.50 (50%).

A white background with blue outline

AI-generated content may be incorrect.

**Figure 1.4 Graph of Training Loss for NOR Operator**

For the NOR operator, the final accuracy was calculated as 0.75 (75%).

## 2 – Digit Classification

**Introduction**

For this section, we are tasked with designing and implementing a neural network for digit classification using the MNIST dataset. Hyper-parameters such as batch size, number of epochs, and learning rate are configured to optimize the model. After training and evaluation of the data, statistical metrics such as accuracy score, inference time, and the confusion matrix are derived to evaluate the performance of our defined model.

**Network Architecture**

We implement a simple convolutional neural network defined as the following:

Neural Network Layers:

* Convolutional Layer 1: 320 parameters
* ReLu Layer (Activation Function)
* Convolutional Layer 2: 18,496 parameters
* Pooling Layer
* Fully Connected Layer 1: 1,605,760 parameters
* Fully Connected Layer 2: 1,290 parameters

Total Parameters: 1,625,866 parameters

**Results**

**A screenshot of a graph

AI-generated content may be incorrect.**

**Figure 2.1 Training Loss and Accuracy Curve of Digit Classification Model**

|  |  |
| --- | --- |
| Performance Metrics of Digit Classification Model | |
| Total Training Time of Model | 142.68 seconds |
| Average Time of Each Iteration | 70 97 seconds |
| Final Accuracy | 94% |
| Inference Time of Model | 0.000999 seconds |

**Table 2. Performance Metrics of Digit Classification Model**

**A screenshot of a graph

AI-generated content may be incorrect.**

**Figure 2.2 Confusion Matrix of Digit Classification Model**

|  |
| --- |
|  |
|  |
|  |

**Figure 2.3 Sample Visualizations of Correct and Incorrect Predictions with Confidence Score**

## 3 - Backpropagation

**Introduction**

For this section, the task is given to implement a function that consists of forward and backwards passes, that calculates the output of the forward pass and the functions’ derivatives with respect to input and weights.

**Network Architecture**

For this model, the following modifications were made to tune the hyper-parameters to enhance the performance of the neural network model:

* Batch Size: 64
* Number of Epochs: 300
* Learning Rate: 0.01

**Results**

A final accuracy of 96% was obtained as results of backpropagation method implementation.