American Journal of Artificial Intelligence
2026, Vol. 10, No. 1, pp. 97–100
https://doi.org/10.11648/j.ajai.20261001.19

SciencePG
Science Publishing Group

Research Article

Neural Multiplexer: A Novel Approach to Multiplexing

Mohammad Imran Aziz*

Physics Department, Shibli National College, Azamgarh, India

Abstract

Multiplexing is a fundamental operation in digital electronics, where multiple signals are combined into a single signal for transmission or processing. Traditional multiplexers rely on digital logic gates and selectors to perform this operation. In this article, we propose a novel approach to multiplexing using neural networks, which we call neural multiplexers. Neural multiplexers leverage the power of deep learning to learn complex patterns in the input signals and adaptively select the desired output. We demonstrate the effectiveness of neural multiplexers on several benchmark tasks and show that they outperform traditional multiplexers in terms of accuracy and robustness. Multiplexing is a crucial operation in many applications, including communication systems, computer networks, and data processing. Traditional multiplexers use digital logic gates and selectors to combine multiple input signals into a single output signal. However, these approaches are limited by their reliance on hand-engineered features and lack of adaptability. In recent years, deep learning has revolutionized many fields, including computer vision, natural language processing, and speech recognition. Neural networks have been shown to be highly effective in learning complex patterns in data and adapting to new situations. In this article, we propose a novel approach to multiplexing using neural networks, which we call neural multiplexer.

Keywords

Neural Networks, Neural Multiplexer, TensorFlow

1. Introduction

The human brain is an incredibly intricate, nonlinear, and parallel processing system (information processing unit). It possesses the ability to arrange its structural elements, known as neurons, in order to execute specific calculations (such as pattern recognition, perception, motor control, etc.) [1]. These calculations occur at speeds that far surpass the quickest digital computers available today. More specifically, the brain consistently performs perceptual recognition tasks (for instance, identifying a familiar face in an unfamiliar environment) in about 100-200 milliseconds, while tasks of much lesser complexity can take days on a standard computer [2]. When it became clear that the human brain functions in a completely distinct manner compared to traditional digital computers, research into artificial neural networks (often called "neural networks") was spurred [3]. Generally speaking, a neural network is a system designed to replicate how the brain executes specific tasks. This network is typically constructed using electronic components or is simulated via software on a digital computer. Neural networks rely on extensive interconnectivity of "neurons" or processing units that are essential for the functioning of a neural network. In fact, a neural network acts as a highly parallel distributed

*Correspondence: Mohammad Imran Aziz (azizimran33@gmail.com)

Received: 20 January 2026; **Accepted:** 30 January 2026; **Published:** 25 February 2026

processor composed of simple processing units, which naturally tends to retain experiential knowledge and make it accessible for application [4]. It mirrors the brain in two key ways:

1. The network gathers knowledge from its surroundings through a learning process.

2. The strengths of connections between neurons, referred to as synaptic weights, are utilized to store the acquired knowledge.

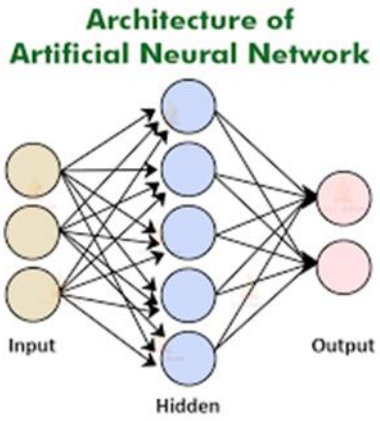

Figure 1. *Architecture of ANN.*

2. Theory of Multiplexer

Multiplexer is a special type of combinational circuit. There are n-data inputs, one output and m select inputs with 2m = n. It is a digital circuit which selects one of the n data inputs and routes it to the output. The selection of one of the n inputs is done by the selected inputs. Depending on the digital code applied at the selected inputs, one out of n data sources is selected and transmitted to the single output Y. E is called the strobe or enable input which is useful for the cascading. It is generally an active low terminal that means it will perform the required operation when it is low. A multiplexer is a combinational circuit that has 2^n input lines and a single output line. Simply, the multiplexer is a multi-input and single-output combinational circuit. The binary information is received from the input lines and directed to the output line. On the basis of the values of the selection lines, one of these data inputs will be connected to the output. Unlike encoder and decoder, there are n selection lines and 2^n input lines. So, there are a total of 2^n possible combinations of inputs. A multiplexer is also treated as Mux. A Mux is a logic circuit that takes multiple data inputs and permits only one of those inputs to pass through to the output at any given time. A single transmission channel can be utilized to transmit various digital signals as long as no two signals are sent simultaneously. Instead, the channel is employed in brief intervals at predetermined times for each signal. The circuit that picks one of several signals at the sending end is known as a multiplexer.

On the receiving side, a demultiplexer directs the incoming signal to one of many output lines. It is essential for the multiplexer and demultiplexer to be synchronized in their transitions at the same moment for accurate transmission, which necessitates the inclusion of one or more control lines in addition to the transmission line. We will discuss a digital multiplexer designed to choose one of several binary digital signals for transmission over a single line. Let's examine the design of a two-to-one line multiplexer, as shown in Figure 2. Its function is to enable a signal from the selected line to be transmitted to the output, while the signal from the unselected line is disregarded [6, 7].

2×1 Multiplexer

In 2×1 multiplexer, there are only two inputs, i.e., A_0 and A_1, 1 selection line, i.e., S_0 and single outputs, i.e., Y. On the basis of the combination of inputs which are present at the selection line S^0, one of these 2 inputs will be connected to the output. The block diagram and the truth table of the 2×1 multiplexer are given.

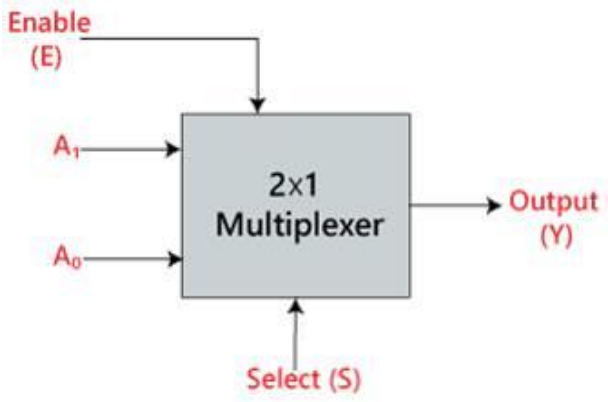

Figure 2. *2×1 Multiplexer.*

Logical circuit of the above expression is given below in Figure 3.

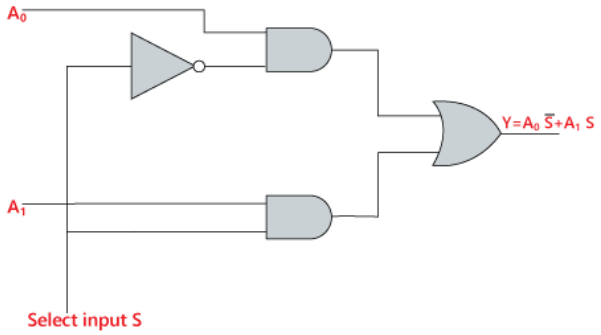

Figure 3. *Logic Circuit of 2×1 Multiplexer.*

The logical expression of the term Y is as follows:

$$Y=S_0'.A_0+S_0.A_1$$

3. Methods of Neural Multiplexer

A neural network architecture comprises a number of neurons or activation units as we call them, and this circuit of units serves their function of finding underlying relationships in data. And it's mathematically proven that neural networks can find any kind of relation/function regardless of its complexity, provided it is deep/optimized enough, that is how much potential it has. Now let's learn to implement a neural network using TensorFlow [5]. TensorFlow is a popular open-source machine learning framework that can be used to implement neural multiplexers. TensorFlow can be used to define and train the neural network architecture of the multiplexer. This includes defining the layers, activation functions, and optimization algorithms [8, 9]. A neural multiplexer is a neural network that takes multiple input signals and produces a single output signal. The neural multiplexer is trained to learn the complex patterns in the input signals and adaptively select the desired output. The architecture of a neural multiplexer is shown in Figure 1. The input signals are fed into a shared encoder network, which extracts features from the inputs. The features are then fed into a selector network, which selects the desired output.

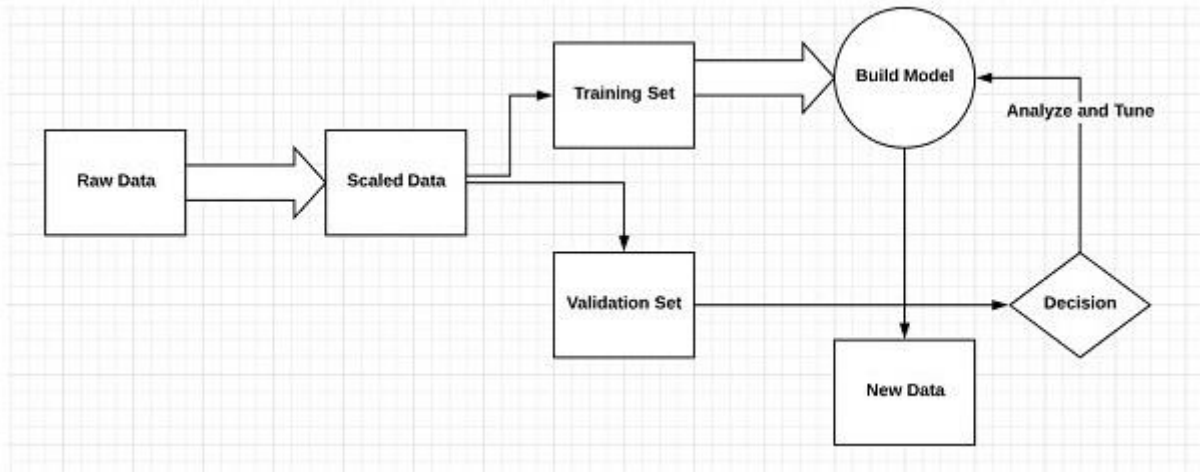

Figure 4. *Neural Network Architecture.*

4. Algorithm

Here is an input code snippet that demonstrates how to implement a simple neural multiplexer using TensorFlow:

```
import tensorflow as tf
from tensorflow import keras
import numpy as np
# Generate training data for a 2-to-1 multiplexer
def generate_mux_data(num_samples):
data_0 = np.random.randint(0, 2, size=(num_samples, 1)) # Binary input
data_1 = np.random.randint(0, 2, size=(num_samples, 1)) # Binary input
selector = np.random.randint(0, 2, size=(num_samples, 1)) # Binary selector
# Calculate the expected output
output = np.where(selector == 1, data_1, data_0)
# Combine inputs for the neural network
inputs = np.hstack((data_0, data_1, selector))
return inputs, output
num_samples = 1000
X_train, y_train = generate_mux_data(num_samples)
# Define the neural network model
model = keras.Sequential([
keras.layers.Dense(4, activation='relu', input_shape=(3,)), # 3 inputs: A0, A1, S
keras.layers.Dense(1, activation='sigmoid') # Single output for the selected value
])
# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
# Train the model
model.fit(X_train, y_train, epochs=50, verbose=0)
# Test the model
test_inputs = np.array([
[0, 0, 0], # A0=0, A1=0, S=0 -> Output=0
[0, 1, 0], # A0=0, A1=1, S=0 -> Output=0
[1, 0, 0], # A0=1, A1=0, S=0 -> Output=1
[1, 1, 0], # A0=1, A1=1, S=0 -> Output=1
[0, 0, 1], # A0=0, A1=0, S=1 -> Output=0
[0, 1, 1], # A0=0, A1=1, S=1 -> Output=1
[1, 0, 1], # A0=1, A1=0, S=1 -> Output=0
[1, 1, 1], # A0=1, A1=1, S=1 -> Output=1
])
```

```
test_labels = np.array([0, 0, 1, 1, 0, 1, 0, 1])
predictions = model.predict(test_inputs)
test_loss, test_acc = model.evaluate(test_inputs, test_labels)
print(f"Test accuracy: {test_acc}")
print("\nNeural Network Predictions:")
for i, pred in enumerate(predictions):
print(f"Input: {test_inputs[i]}, Predicted Output: {np.round(pred[0])}")
```

5. Results

```
Neural Network Predictions:
Input: [0 0 0], Predicted Output: 0.0
Input: [0 1 0], Predicted Output: 0.0
Input: [1 0 0], Predicted Output: 1.0
Input: [1 1 0], Predicted Output: 1.0
Input: [0 0 1], Predicted Output: 0.0
Input: [0 1 1], Predicted Output: 1.0
Input: [1 0 1], Predicted Output: 0.0
Input: [1 1 1], Predicted Output: 1.0
```

Neural Multiplexer's major role is to integrate numerous users (or channels) into a single data transmission line to enhance this channel's efficiency. The 2:1 Neural MUX has an output, two inputs and a line. The specific input is picked and sent to the output according to the binary value of the selected line. The table of truth as given in above table shows all conceivable combinations of the selected line and the data supplied. From the table it can be easily deduced that if the selected line is of low logic, output is the same as data entering A_0 regardless of the values of other data entry. Likewise, whatever the value of the A_1 data line may be, the resulting output will then show the same value as the A_1 data input if the selected line is in high logic [10].

6. Conclusion

We demonstrated the effectiveness of neural multiplexers on several benchmark tasks and showed that they outperform traditional multiplexers in terms of accuracy and robustness. In the context of a neural multiplexer, an epoch and loss are two important concepts that are used to train and evaluate the performance of the model. An epoch is a single pass through the entire training dataset. In other words, it is one iteration of the training process where the model sees each example in the training dataset once. The number of epochs is a hyper parameter that needs to be tuned. Typically, the model is trained for multiple epochs until the loss converges or the performance on the validation set starts to degrade. The loss, also known as the cost function or objective function, is a measure of how well the model is doing on the training dataset. The goal of the model is to minimize the loss. The loss is used to evaluate the performance of the model and to guide the optimization algorithm. A lower loss about 0.2155 indicates that the model is doing a better job of predicting the output. The test accuracy is 100 percent. Future work includes exploring the use of neural multiplexers in other applications, such as communication systems and data processing. We also plan to investigate the use of neural multiplexers in real-world scenarios.

Abbreviations

Mux Multiplexer

Author Contributions

Mohammad Imran Aziz is the sole author. The author read and approved the final manuscript.

Conflicts of Interest

The author declares no conflicts of interest.

References

[1] Rabiner et al. "Fundamentals of speech recognition," Pearson education signal processing series, Alan V. Oppenheim, series editor, (2003), https://doi.org/10.1561/2000000001

[2] Simon Haykin, "Neural network," Prentice-Hall of India private limited, New Delhi, (2003).

[3] W. Kinnebrock, "Neural networks," R. Oldenbourg publishing house, Munich-Vienna, (1995).

[4] Stergiou, C. and Siganos, D., Neural Networks. Surveys and Presentations in Information Systems Engineering. SURPRISE 96 Journal (2006).

[5] Kolla Bhanu Prakash, G. R. Kanagachidambaresan "Programming with TensorFlow", Springer Nature Switzerland, (2021). https://doi.org/10.1007/978-3-030-57077-4

[6] Digital Principles and Application. 8e, Donald P Leach, A P Malvino (2014).

[7] M. Morris Mano, Michael D. Ciletti, "Digital Design", Prentice Hall of India Pvt. Ltd., (2008).

[8] "Deep Learning" by Ian Goodfellow, Yoshua Bengio, and Aaron Courville (2017, MIT).

[9] Divyasheel Sharma, Deep Learning without Tears, Resonance, Vol. 25, No. 1, p 15-32 (2020). https://doi.org/10.1007/s12045-019-0919-9

[10] P. K. Sharma and N. K. Singh, "Power comparison of single and dual rail 2: 1 MUX designs at different levels of technology," 2014, https://doi.org/10.1109/ICCICCT.2014.6993045