University of Guelph

Using Artificial Neural Networks to Classify Swipe Gestures in Internet Browsers

---

Dr. Simon Yang

Dylan MacKenzie (0892889)

Colin Raper (0895611)

March 31st, 2019

Abstract – Touchscreens have become an essential asset in enriching the daily life of users in devices ranging from laptops, tablets, and smartphones. Its simple approach for reading user input has allowed increased accessibility to individuals with physical or mental disabilities, increased performance, and improved user satisfaction. As such, many personal electronic manufacturers build these devices with touchscreen in mind. Currently, touch is primarily recognized by operating system calls in many personal computing devices such as Android and iOS.

In this report, we are proposing 3 configurations for fully connected Artificial Neural Network (ANN) systems for recognizing and classifying swipe gestures in an internet browser, independent of the operating system (OS). The proposed design uses supervised learning via Error Backpropagation (EBP) with the least-mean square algorithm (LMS) to train the ANN. Currently, sample JavaScript programs for generating artificial swipe data have been created. The model was trained using TensorFlow.js [1] and is suitable to be deployed to a production environment running in the browser.

All of the models were trained for classifying using the procedurally generated data that simulated up, down, left and right swipes and achieved an accuracy above 86%. Regardless of configuration, the accuracy between the neural network architectures were very similar. The final proposed neural network architecture for classifying simple swipes is a 2-layer feedforward network with 4 input neurons, 4 hidden neurons and 4 one-hot encoded output neurons. The activation function of the hidden layer is a sigmoid function and the activation function of the output layer is a softmax function to produce a probability of the gesture that was recognized.

Key Terms – Operating System (OS), Artificial Neural Network (ANN), Error Backpropagation (EBP), Feed-forward Network (FFN), Multi-Layered Neural Network (MLNN

1 Introduction

Touchscreens have become an increasingly integral part of daily life in devices ranging from laptops, tablets, smartphones, ATM's, self-checkouts and many others. Its intuitive approach to recognizing user input allows easy interactions without the need for a mouse and keyboard. This ease of interaction can be extended to provide more accessibility to individuals with physical or mental disabilities by combining touchscreen capability with other accessibility services provided by electronic devices. Touchscreen support also provides versatile control and accuracy for many applications which speeds up and improves user satisfaction. For this reason, many personal electronics manufacturers build these devices with touchscreen capability specifically in mind.

Touchscreens in devices like tablets, laptops, and smartphones can recognize multiple different gestures including tap, double-tap, hold, drag, two-finger zoom, and swipe. Currently, touch is primarily recognized through the operating system; companies like Google (for Android) and Apple (for iOS) provide software development kits to assist in recognizing gestures for use in native applications for their devices [2] [3]. In this report, we are proposing a system for recognizing and classifying swipe gestures in an internet browser, independent of the operating system, using an Artificial Neural Network (ANN).

2 Literature Review

A. Artificial Neural Networks

Developing the swipe classification model will primarily focus on knowledge learned in-class and understanding existing methods for swipe tracking. Some key topics to focus on for our proposal include: Artificial Neural Networks, basics neuron models, multi-layer networks, activation functions, error backpropagation, and current swipe tracking systems.

ANNs are primarily used to create regression and classification models through various training techniques. The basic artificial neuron model is as follows:

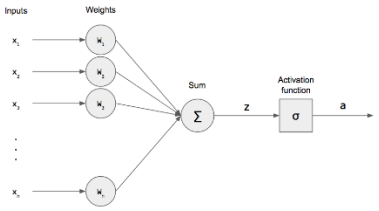


Figure 1 Basic artificial neuron model [4]

This model takes an input vector and calculates a weighted sum. This sum is then passed through an activation function which mimics the firing of a motor neuron the human brain [5]. The weights serve as a means for evaluating the importance of different combinations of inputs and are the main target for training the neuron to learn a pattern/function.

$$y = f(\sum_{i=0}^{n} x_i w_i)$$

Equation 1 Mathematical model of an artificial neuron

The power in this model comes when many neurons are interconnected. A single artificial neuron is only capable of classifying a linearly separable classification model [6].

When multiple neurons are connected in layers, much more complex functions can be modeled. A classic feed-forward ANN is a collection of interconnected neuron models that form a system capable of approximating very complex non-linear functions.

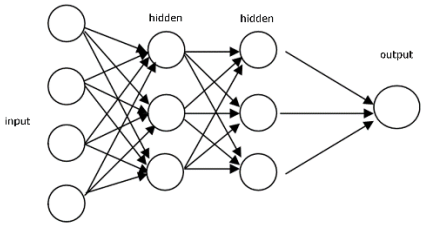


Figure 2 Example of feed-forward artificial neural network

B. Activation Functions

Two activation functions that will be examined in this proposal are the hard-limit and sigmoid transfer functions. Both activation functions can be altered to operate for positive values between 0 and 1 which could prove useful for our design proposal. As our swipe recognition will be making swipe direction predictions using pixel Cartesian coordinates of the display, our inputs can be bounded between 0 and 1 as negative values do not provide any additional information.

The hard-limit activation function is primarily used for making classifications of 0 and 1, while the sigmoid function provides a less strict curve within the same bound and is better at classifying floating point values.

$$y = hl(x) = \begin{cases} 1, if x \geq 0 \\ 0, if x < 0 \end{cases}$$

Equation 2 Hard-limit activation function

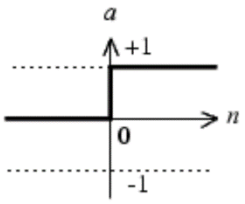
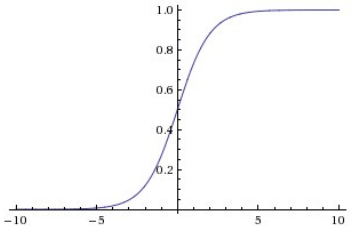


Figure 3 Hard-limit activation function [7]

The sigmoid activation function could be used for our hidden neuron layer(s). This will allow our input x and y coordinates to be better interpolated between the 0 and 1 bounds and we believe will provide better performance when testing/validation.



Equation 3 Sigmoid activation function [8]

$$y = f(x) = \frac{1}{1 + e^{-x}}$$

Figure 4 Sigmoid activation function

C. Supervised Learning

The main method for training a neural network in a supervised learning fashion is Error Backpropagation (EBP). As the error for the final output layer neurons will always be known, we can use EBP for tuning the weights in the neural network so that they can learn patterns for different swipe gestures. EBP can utilize the least-mean-square (LMS) algorithm to minimize the error created at the output layer. The LMS algorithm is designed to penalize the network proportionally to the error between the output and the expected target. This feature will help us create a robust supervised learning system for training our swipe recognition.

$$E = \frac{1}{2}\sum_{i=1}^{N}(t_j - y_j)^2$$

Equation 4 Least-mean square algorithm

A primary concern using this method is having too many or too few hidden neurons. As we increase the number of neurons, we may see overfitting problems. If we do not have enough neurons, we may see the opposite effect and experience underfitting. This will be evaluated during our experimentation.

$$\Delta w_{jk} = \eta f^{'} e_j h_k$$

Equation 5 Change in hidden weights in backpropagation

$$\Delta v_{ki} = \eta g^{'} e^{h}{}_{k} x_i$$

Equation 6 Change in input weights in backpropagation

D. Current Swipe Gesture Recognition Systems

In their current configuration, swipe gesture recognition systems are widely available at the operating system level for mobile devices such as Android and iOS [2] [3]. Gesture recognition systems for internet browsers are available, however, many of their implementations explicitly define the rules for recognizing each type of gesture [9]. Internet browsers are capable of firing events when the window is touched/clicked and JavaScript is natively capable of accessing the (x, y) coordinates relative to this window [10]. Contrary to the traditional explicitly defined rules for recognizing swipe gestures in the browser, we would like to train a neural network that can classify these gestures and possibly more.

3 The Proposed Method

To solve this problem, we are proposing creating various configured fully connected neural network. A series of (x, y) coordinates will be the inputs for the network and the classification of the swipe gesture will be the output. The screen swipes should be classified in real-time, therefore, a smaller neural network with minimal nodes is ideal for performance. A smaller neural network with fewer nodes with yield higher performance when loading the pre-trained weights when the webpage loads and when classifying the swipes.

The first consideration in the design is how to collect the (x, y) coordinates on the device's display. First, using JavaScript in the browser, the design will sample the (x, y) coordinates at 30-60 Hz while the mouse/finger is dragged on the display. This will yield an array of (x, y) coordinates once the mouse/finger is released. To achieve this effect, the design will use the built-in event listeners that JavaScript provides for modern web browsers.

The (x, y) coordinates collected by the event listener will then have to be normalized. This allows the network to classify swipes independently of the viewport or tracking region dimensions. To do this, the x coordinates will pass through *Equation 7* and the y coordinates will pass through *Equation 8*. This turns the array of coordinates into a series of ratios relative to the dimensions of the area in which they were recorded. This way, the classification model can be used with different screen dimensions such as smartphones, tablets or desktop monitors.

$w = width\ of\ the\ tracking\ region\ (pixels)$
$l = leftmost\ x\ coordinate\ of\ tracking\ region\ (pixels)$
$x_{raw} = x\ coordinate\ (pixels)$
$x_{normalized} = normalized\ x\ coordinate$

$$x_{normalized} = \frac{x_{raw} - l}{w - l}$$

Equation 7 Normalization equation for x coordinates

$h = height\ of\ the\ tracking\ region\ (pixels)$
$u = top\ y\ coordinate\ of\ tracking\ region\ (pixels)$
$y_{raw} = y\ coordinate\ (pixels)$
$y_{normalized} = normalized\ y\ coordinate$

$$y_{normalized} = \frac{y_{raw} - u}{h - u}$$

Equation 8 Normalization equation for y coordinates

$$Coordinate : (500, 200)$$
$$w = 700px$$
$$l = 200px$$

$$x_{normalized} = \frac{500px - 200px}{700px - 200px} = 0.6$$

Equation 9 Example of normalizing x coordinate

Next, the outputs need to be encoded in order to classify which type of swipe gesture took place on the display. To do this, the final design will use a one-hot encoded output. With one-hot encoding, the neural network can use a softmax activation function on the output layer [11]. This way, the classification model's output will be a probability ranging from 0 – 1 and all of the output nodes will sum to 1. For simplicity, the design will initially only classify four different types of swipes:

Table 1 One-hot encoding of swipe types as model outputs

Swipe Type	One-hot Encoding for Output
Up	[1, 0, 0, 0]
Down	[0, 1, 0, 0]
Left	[0, 0, 1, 0]
Right	[0, 0, 0, 1]

To train the neural network, artificial swipe data will be procedurally generated and used for supervised learning. The proposed method for procedurally generating an artificial swipe to the right is in *Figure 5*. Alternative methods for generating up, down and left swipes are listed in the appendix.

n = desired number of coordinates in swipe array
m = minimum length of a swipe $(pixels)$
w = width of the tracking region $(pixels)$
h = height of the tracking region $(pixels)$

1. Select random x_1 coordinate between 0 and w
2. Select random y_1 coordinate between 0 and h
3. Select random x_2 coordinate between m and $w - x_1$
4. Add x_1 to x_2
5. Select random y_2 coordinate between $\frac{-h}{m}$ and $\frac{h}{m}$
6. Add y_1 to y_2
7. Linearly interpolate n points between (x_1, y_1) and (x_2, y_2)

Figure 5 Procedure for generating an artificial right swipe

Once the inputs and outputs have been collected, normalized, and encoded, we will transform them to be compatible with the neural network and shuffle the dataset to remove variance in the data [12]. We have decided that the upper input nodes of the neural network will be responsible for feeding the x coordinates and the lower inputs nodes will be responsible for the y coordinates. We would like to evaluate the performance difference between different architectures, such as the number of layers and hidden nodes for our final design.

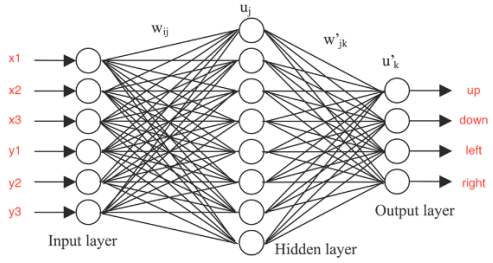


Figure 6 Proposed architecture for swipe classification neural network

To test the performance of the neural network, the data will be split into a training dataset and a validation dataset. The training set will consist of 80% of the data, and the validation will be the remaining 20%. We will use the validation dataset to get an unbiased performance metric of the neural networks. The three network architectures that were decided to test are as follows. All networks have ten input neurons to represent five x-coordinates and five y-coordinates of the swipe. All networks also feature four output neurons to represent the one-hot encodings of the swipe types.

1. A two-layer feedforward network with ten input neurons, four hidden neurons, and four output neurons.
2. A two-layer feedforward network with ten input neurons, sixteen hidden neurons, and four output neurons.
3. A three-layer feedforward network with ten input neurons, 32 hidden neurons in the first hidden layer, 16 hidden neurons in the second hidden layer, and four output neurons.

4 Results

After creating a model for each network architecture in TensorFlow.js [1], each model was trained and validated using the procedurally generated data. The performance of the three network architectures was graphed and can be found in Figure 7, Figure 8, and Figure 9. From these figures, we can see that all architectures obtained similar final training and validation accuracy (shown in Table 2).

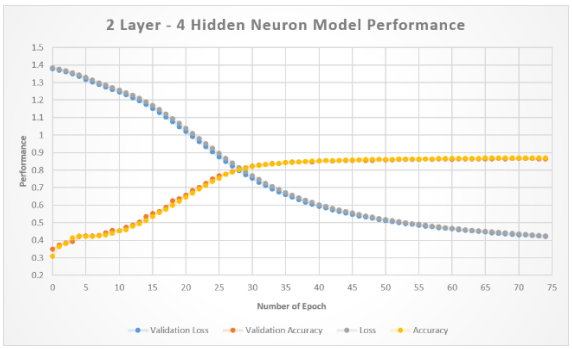


Figure 7: Performance of the 2 Layer Artificial Neural Network with 4 hidden neurons

Table 2: Final Training and Validation Accuracy based on Neural Network Architecture.

Neural Network Architecture	Training Accuracy (%)	Validation Accuracy (%)
2 Layer – 4 Hidden Neuron	87	86.5
2 Layer – 16 Hidden Neuron	86.9	87.8
3 Layer – 32 and 16 Hidden Neuron	87.5	86.9

5 Discussion

From the results, all of the networks that were generated were mostly successful in classifying simple swipe gestures as they all have a validation accuracy above 86%. The 2 layer network with 4 hidden neurons was selected as the best architecture for this application. As most of the neural networks that were created have similar final accuracy, choosing the network architecture with the least amount of nodes is important for performance reasons. As the network will be running in a web browser, a smaller network will be higher performance when transmitting across the internet, as it has fewer weights and thus less information about its architecture to submit. Also, a smaller network does not have to do as much

computation to classify a swipe while the network is running on the device, which means it would be more responsive to swipes.

As the data that was used to train the model was procedurally generated, it does not perfectly portray real-world data that would be created in real-time by users swiping on displays with their fingers. The model's accuracy may be high when it is tested on other procedurally generated data, however, a better approach to test the true performance of the model is to use live testing with real people.

The model is also very limited in the number of swipes that it was configured to classify. The same methodology could be used and the network architecture could be updated to add more swipe gestures to the classification, such as swiping diagonally or swiping to draw shapes. For prototyping purposes, only four gestures were used, however, future modifications to the model could be made to support more complex swipes.

6 Conclusion

In conclusion, the final trained artificial neural network has proven to be a viable option for classifying simple swipes in an internet browser. Since the architecture is small and simple, the performance suitable for internet browsers. The model is currently limited in the number of swipe gestures that it is capable of classifying, however, using the same modeling methodology should yield satisfactory results if more swipe gestures were added to the model for training.

References

[1] "A JavaScript library for training and deploying ML models in the browser and on Node.js," TensorFlow, [Online]. Available: https://js.tensorflow.org/. [Accessed 30 March 2019].

[2] "Track touch and pointer movements," Android, 2019. [Online]. Available: https://developer.android.com/training/gestures/movement. [Accessed 22 February 2019].

[3] "UISwipeGestureRecognizer - UIKit | Apple Developer Documentation," Apple Inc, 2019. [Online]. Available: https://developer.apple.com/documentation/uikit/uiswipegesturerecognizer. [Accessed 22 February 2019].

[4] M. Deshpande, "The First Neural Networks," [Online]. Available: https://pythonmachinelearning.pro/perceptrons-the-first-neural-networks/. [Accessed 2019 Febrauary 22].

[5] G. Di Pellegrino, "Understanding Motor Events: a Neurophysiological Study," *Experimental Brain Research,* vol. 91, no. 1, pp. 176-190.

[6] S. Knerr, L. Personnas and G. Dreyfus, ""Single-Layer Learning Revisited: a Stepwise Procedure for Building and Training a Neural Network," *Neurocomputing,* pp. 41-50, 1990.

[7] "Hard-Limit Transfer Function," [Online]. Available: https://edoras.sdsu.edu/doc/matlab/toolbox/nnet/model22.html. [Accessed 22 February 2019].

[8] "Sigmoid Function," Towards Data Science, [Online]. Available: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6. [Accessed 22 Febuary 2019].

[9] "HammerJS General API," HammerJS, [Online]. Available: http://hammerjs.github.io/api/. [Accessed 22 February 2019].

[10] "MouseEvent," Mozilla, [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/MouseEvent. [Accessed 22 February 2019].

[11] E. Jang, S. Gu and B. Poole, "Categorical Reparameterization with Gumbel-Softmax," 2016.

[12] H. Xiao, K. Rasul and R. Vollgraf, "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms," *arXiv:1708.07747,* 2017.

Appendix

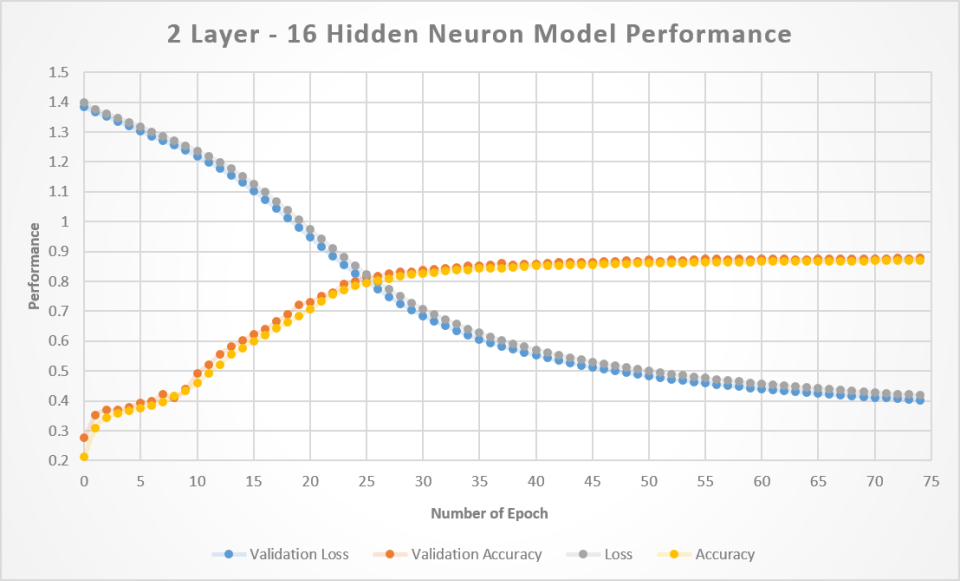


Figure 8: Performance of the 2 Layer Artificial Neural Network with 16 hidden neurons.

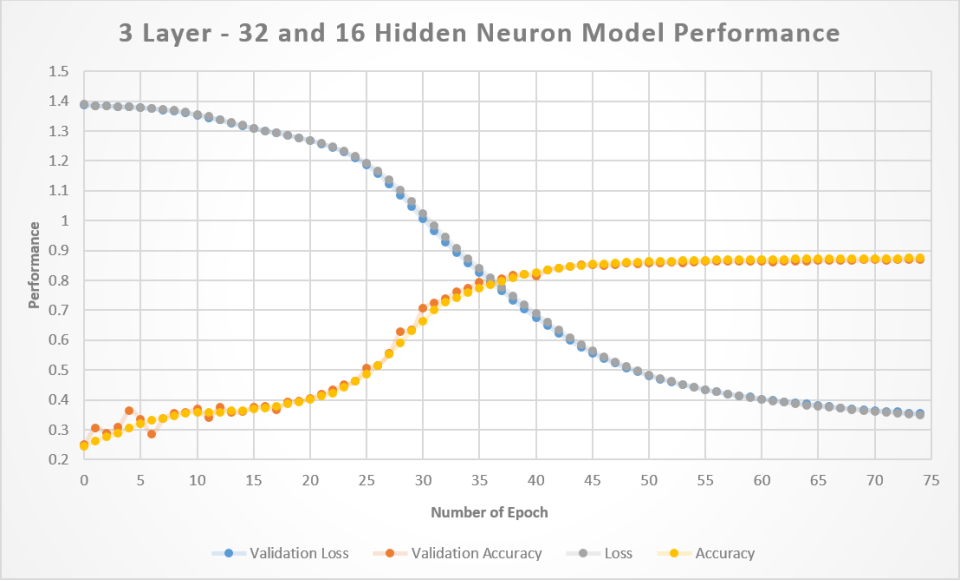


Figure 9: Performance of the 3 Layer Artificial Neural Network with 32 and 16 hidden neurons.