# IMPLEMENTING CUSTOM NEURAL NETWORK FROM SCRATCH

**Student Name: Akshat Gosain**
Department Name: Computer Science
BITS Pilani K K Birla Goa Campus,
email id: f20220154@goa.bits-pilani.ac.in

**ABSTRACT**

In this assignment, a one hidden layer neural network model that adapts to the most suitable activation function according to the dataset is developed. The mathematical formulation of the activation function used in this method is $g(x) = k_0 + k_1 x$, where $x, g(x) \in \mathbf{R}$, and $k_0, k_1$ are learned from the data via gradient descent. The performance of the one hidden layer neural network model is evaluated on Breast Cancer Wisconsin, Iris, Bank Note Authentication, and MNIST datasets.

## 1 Background

In the Deep Learning literature, the selection of activation function is mostly based on brute-force stratergy. A popularly used activation function such as *ReLU*, *sigmoid*, etc. are incorporated in the model. If the activation function is not working, then repeat the whole process with a new activation function. This process is very time consuming especially when dealing with high dimensional data. Hence, it is important to develop a framework where a suitable activation function is discovered from the data - *Data Driven Discovery of Activation Function*. The possibilities of such an approach could not only save significant time and effort for tuning the model, but will also open up new ways for discovering essential features of not-so-popular activation functions.

## 2 Mathematical Framework

In this section, we shall describe the mathematical framework for the data driven discovery of activation function. For the sake of simplicity, we start with a binary classification problem. We shall develop a two layer neural network (one hidden layer and one output layer) for solving binary classification problem. Then we shall extend the framework for the multiclass classification problem.

### 2.1 Forward Pass and Backward Pass for a single data instance

In this section we concretely show the forward and backward pass for a single data instance. We consider a binary classification problem for mathematical easiness. Figure 1 depicts the neural network of our interest.

1. **Data:** Let the training data represented as $X$ contains $m$ data instances and two features ($n = 2$). The size of the matrix $X$ is $2 \times m$.
2. **Two Layer Neural Network Model:** We consider a two layer neural network with 2 nodes in the input layer, 2 nodes in the first hidden layer and 2 nodes in the output layer (Refer Figure 1).
3. **Forward Pass** To begin with, we shall show the forward and backward pass for a single data instance $x^{(i)} = \begin{bmatrix} x_1^{(i)} \\ x_2^{(i)} \end{bmatrix}$.

   The following are the steps in the forward pass:

$$Z^{[1](i)} = \begin{bmatrix} z_1^{[1](i)} \\ z_2^{[1](i)} \end{bmatrix} = \begin{bmatrix} w_{11}^{[1]} & w_{12}^{[1]} \\ w_{21}^{[1]} & w_{22}^{[1]} \end{bmatrix} \begin{bmatrix} x_1^{(i)} \\ x_2^{(i)} \end{bmatrix} + \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \end{bmatrix} \tag{1}$$

$$A^{[1](i)} = \begin{bmatrix} a_1^{[1](i)} \\ a_2^{[1](i)} \end{bmatrix} = \begin{bmatrix} z_1^{[1](i)} & 1 \\ z_2^{[1](i)} & 1 \end{bmatrix} \begin{bmatrix} k_1^{[1]} \\ k_0^{[1]} \end{bmatrix}, \tag{2}$$

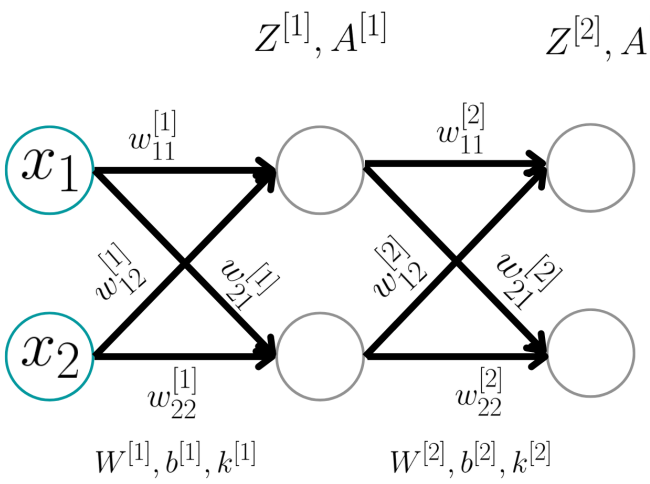



Figure 1: Two layer neural network with two neurons in the input layer, two neurons in the hidden layer and two neurons in the output layer. Ignore $K^{[2]}$ provided in the figure.

$$Z^{[2](i)} = \begin{bmatrix} z_1^{[2](i)} \\ z_2^{[2](i)} \end{bmatrix} = \begin{bmatrix} w_{11}^{[2]} & w_{12}^{[2]} \\ w_{21}^{[2]} & w_{22}^{[2]} \end{bmatrix} \begin{bmatrix} a_1^{[1](i)} \\ a_2^{[1](i)} \end{bmatrix} + \begin{bmatrix} b_1^{[2]} \\ b_2^{[2]} \end{bmatrix}, \tag{3}$$

$$A^{[2](i)} = \begin{bmatrix} a_1^{[2](i)} \\ a_2^{[2](i)} \end{bmatrix} = \begin{bmatrix} \frac{\exp^{Z_1^{[2](i)}}}{\exp^{Z_1^{[2](i)}} + \exp^{Z_2^{[2](i)}}} \\ \frac{\exp^{Z_2^{[2](i)}}}{\exp^{Z_1^{[2](i)}} + \exp^{Z_2^{[2](i)}}} \end{bmatrix}, \tag{4}$$

$A^{[2](i)}$ is passed to the categorical cross entropy loss function. Loss function calculates the error in classification. The categorical cross entropy (CCE) is defined as follows:

$$CCE^{(i)} = -\sum_{j=1}^{C} y_j^{(i)} \log(a_j^{[2](i)}), \tag{5}$$

where $C$ is the total number of classes (in this case $C = 2$), $y^{(i)}$ is the true label of the $i$-th data instance in one hot encoding format. For example, if $C = 2$, $y^{(i)} = \begin{bmatrix} y_1^{(i)} \\ y_2^{(i)} \end{bmatrix}$. If the true label of $i$-th data instance is 1 then $y_1^{(i)} = 1$ and $y_2^{(i)} = 0$. If the true label of $i$-th data instance is 2 then $y_1^{(i)} = 0$ and $y_2^{(i)} = 1$. The output of a loss function is a scalar. The aim is to minimise the loss with respect to $W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}, K^{[1]}$. (Note: $K^{[1]} = \begin{bmatrix} k_1^{[1]} \\ k_0^{[1]} \end{bmatrix}$.

This involves computing the derivative with respect to above variables. Following that we shall apply gradient descent algorithm to find the optimum parameter values.

(Note: The activation function defined as $g(x) = k_0 + k_1 x$ is applied only to nodes in the hidden layers. This activation function is not applied in the output layer. This formulation is different from what we have seen in the class. )

The categorical cross entropy (CCE) for the entire data is defined as follows:

$$CCE = -\frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{C} y_j^{(i)} \log(a_j^{[2](i)}), \tag{6}$$

4. **Backward Pass:** We shall find the partial derivatives of the loss function with respect to the learnable parameters. This involves the usage of chain rule.

$$\frac{\partial L}{\partial w_{11}^{[2]}} = -\frac{1}{m}.\sum_{i=1}^{m}\sum_{j=1}^{2} y_j^{(i)}.(1-a_j^{[2](i)})a_1^{[1](i)}, \tag{7}$$

$$\frac{\partial L}{\partial w_{12}^{[2]}} = -\frac{1}{m}.\sum_{i=1}^{m}\sum_{j=1}^{2} y_j^{(i)}.(1-a_j^{[2](i)})a_2^{[1](i)}, \tag{8}$$

$$\frac{\partial L}{\partial w_{21}^{[2]}} = -\frac{1}{m}.\sum_{i=1}^{m}\sum_{j=1}^{2} y_j^{(i)}.(1-a_j^{[2](i)})a_1^{[1](i)}, \tag{9}$$

$$\frac{\partial L}{\partial w_{22}^{[2]}} = -\frac{1}{m}.\sum_{i=1}^{m}\sum_{j=1}^{2} y_j^{(i)}.(1-a_j^{[2](i)})a_2^{[1](i)}, \tag{10}$$

The above can be compactly represented in the matrix formulation (refer class notes) as follows:

$$\boxed{\frac{\partial L}{W^{[2]}} = \frac{1}{m}.Y_{2x1}.((A^{[2]})^T_{1x2}-1).A^{[1]}_{2x1}} \tag{11}$$

where '$\cdot$' represents matrix multiplication, $T$ represents transpose and $y$ represents the one hot label representation of the data instance $x$.

$$\frac{\partial L}{\partial b_1^{[2]}} = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{2} y_j^{(i)}.(1-a_j^{[2](i)}), \tag{12}$$

$$\frac{\partial L}{\partial b_2^{[2]}} = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{2} y_j^{(i)}.(1-a_j^{[2](i)}), \tag{13}$$

The above can be compactly represented in matrix form as follows:

$$\boxed{\frac{\partial L}{\partial b^{[1]}} = \frac{1}{m}.Y_{2x1}.((A^{[2]})^T_{1x2}-1).I_{2x1}} \tag{14}$$

$$\frac{\partial L}{\partial k_1^{[1]}} = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{C}\frac{y_j^{(i)}}{a_j^{[2](i)}}\cdot a_j^{[2](i)}(1-a_j^{[2](i)})\cdot w_{j1}^{[2]}\cdot k_1^{[1]}\cdot z_1^{[1](i)}, \tag{15}$$

On simplification, we get the following:

$$\frac{\partial L}{\partial k_1^{[1]}} = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{C} y_j^{(i)}(1-a_j^{[2](i)})\cdot w_{j1}^{[2]}\cdot k_1^{[1]}\cdot z_1^{[1](i)}, \tag{16}$$

$$\frac{\partial L}{\partial k_0^{[1]}} = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{C}\frac{y_j^{(i)}}{a_j^{[2](i)}}\cdot a_j^{[2](i)}(1-a_j^{[2](i)})\cdot w_{j1}^{[2]}\cdot k_1^{[1]}, \tag{17}$$

On simplification, we get the following:

$$\frac{\partial L}{\partial k_0^{[1]}} = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{C} y_j^{(i)}(1-a_j^{[2](i)})\cdot w_{j1}^{[2]}\cdot k_1^{[1]} \tag{18}$$

The above can be compactly represented in matrix form as follows:

$$\boxed{\frac{\partial L}{\partial K^{[1]}} = \frac{1}{m}\sum_{i=1}^{m}\left(Z^{[1](i)} * (W^{[2]})^T * (A^{[2](i)}-Y)\right)}. \tag{19}$$

where $sum_e$ represents the sum of all elements, and $*$ represents element-wise multiplication.

$$\frac{\partial L}{w_{11}^{[1]}} = -y_1(1-a_1^{[2]})w_{11}^{[2]}k_1^{[1]}x_1^{(i)} - y_2(1-a_2^{[2]})w_{21}^{[2]}k_1^{[1]}x_1^{(i)}, \tag{20}$$

$$\frac{\partial L}{\partial w_{12}^{[1]}} = -y_1(1-a_1^{[2]})w_{11}^{[2]}k_1^{[1]}x_2^{(i)} - y_2(1-a_2^{[2]})w_{21}^{[2]}k_1^{[1]}x_2^{(i)}, \tag{21}$$

$$\frac{\partial L}{\partial w_{21}^{[1]}} = -y_1(1-a_1^{[2]})w_{12}^{[2]}k_2^{[1]}x_1^{(i)} - y_2(1-a_2^{[2]})w_{22}^{[2]}k_2^{[1]}x_1^{(i)}, \tag{22}$$

$$\frac{\partial L}{\partial w_{22}^{[1]}} = -y_1(1-a_1^{[2]})w_{12}^{[2]}k_2^{[1]}x_2^{(i)} - y_2(1-a_2^{[2]})w_{22}^{[2]}k_2^{[1]}x_2^{(i)}. \tag{23}$$

The matrix form for the weights is:

$$\boxed{\frac{\partial L}{\partial W^{[1]}} = -\frac{1}{m}\sum_{i=1}^{m}\left(X^{(i)} * (W^{[2]})^T * (A^{[2](i)} - Y)\right)}. \tag{24}$$

where $sum_c$ represents the column sum.

$$\frac{\partial L}{\partial b_1^{[1]}} = -\frac{1}{m}\sum_{i=1}^{m}\left(y_1^{(i)}(1-a_1^{[2](i)})w_{11}^{[2]} + y_2^{(i)}(1-a_2^{[2](i)})w_{21}^{[2]}\right), \tag{25}$$

$$\frac{\partial L}{\partial b_2^{[1]}} = -\frac{1}{m}\sum_{i=1}^{m}\left(y_1^{(i)}(1-a_1^{[2](i)})w_{12}^{[2]} + y_2^{(i)}(1-a_2^{[2](i)})w_{22}^{[2]}\right). \tag{26}$$

The above can be compactly represented as follows:

$$\boxed{\frac{\partial L}{\partial b^{[1]}} = -\frac{1}{m}\sum_{i=1}^{m}\left((A^{[2](i)} - Y)\cdot(W^{[2]})^T\right)}. \tag{27}$$

5. **Gradient Descent Algorithm**: Now, we shall use the gradient descent algorithm for updating the learnable parameters of the two-layer neural network. The following parameters are updated:
   - $W^{[2]} := W^{[2]} - \alpha \cdot \frac{\partial L}{\partial W^{[2]}}$
   - $b^{[2]} := b^{[2]} - \alpha \cdot \frac{\partial L}{\partial b^{[2]}}$
   - $K^{[1]} := K^{[1]} - \alpha \cdot \frac{\partial L}{\partial K^{[1]}}$
   - $W^{[1]} := W^{[1]} - \alpha \cdot \frac{\partial L}{\partial W^{[1]}}$
   - $b^{[1]} := b^{[1]} - \alpha \cdot \frac{\partial L}{\partial b^{[1]}}$

   where $\alpha$ is the learning rate.

# 3 Dataset Description

## 3.1 Breast Cancer Wisconsin

The *Breast Cancer Wisconsin* ( [1, 2]) dataset consists of two classes: class 'M' (refers to a malignant infection level) and class 'B' (refers to a benign infection level). There are a total of 569 data instances[12] with 30 attributes[12] such as radius, perimeter, texture, smoothness, etc. for each cell nucleus[12]. The class distribution for training and testing is provided in Table 1.

## 3.2 Iris

The *Iris* ( [3, 4]) dataset is a three-class classification problem of three types of iris plants: Iris Setosa, Iris Versicolour, and Iris Virginica. There are 150 data instances[3] in this dataset with four attributes in each data instance: sepal length, sepal width, petal length, and petal width[3]. All attributes are in *cms*. The specified class distribution provided in Table 1 is in the following order: (Setosa, Versicolour, Virginica).

## 3.3 Bank Note Authentication

The *Bank-note Authentication* ( [1, 5]) dataset is a binary classification problem involving the classification of Genuine and Forgery banknotes. A Genuine class refers to an authentic banknote denoted by '0', while a Forgery class refers to a forged banknote denoted by '1'. The dataset was curated by extracting four features[1] from the images of banknotes belonging to the two classes. These features are obtained through wavelet transformation[1]. There are a total of 1372 data instances[1]. The class distribution for training and testing is provided in Table 1.

### 3.4 MNIST

MNIST is a publicly available computer vision dataset corresponding to handwritten digits from 0 to 9. This is a 10 class classification task. Each image in their respective classes has a dimension of 28 pixels $\times$ 28 pixels. There is a total of 70,000 images in the MNIST database. The class distribution for training and testing is provided in Table 1.

Table 1: Train-Test split in experiments. High training sample regime corresponds to an 80% and 20% split in training and testing respectively.

| **Dataset** | **Classes** | **Features** | **Training samples /class** | **Testing samples /class** |
|---|---|---|---|---|
| Breast Cancer Wisconsin | 2 | 30 | (227, 227) | (57, 57) |
| Iris | 3 | 4 | (40, 40, 40) | (10, 10, 10) |
| Bank Note Authentication | 2 | 4 | (549, 549) | (137, 137) |
| MNIST | 10 | 784 | (5600,5600, 5600, 5600, 5600, 5600, 5600, 5600, 5600, 5600) | (1400, 1400, 1400, 1400, 1400, 1400, 1400,1400, 1400, 1400) |

I've assumed all datasets to be having evenly distributed classes, just for simplicity and since entirely accurate data isn't available for every single dataset.

I know the actual data can vary, as for example the Breast Cancer Wisconsin is actually split into 212 M and 357 B samples so that way it would be (170,286) training and (42,71) testing samples instead of me evenly distributing it assuming (227,227) and (57,57) respectively.

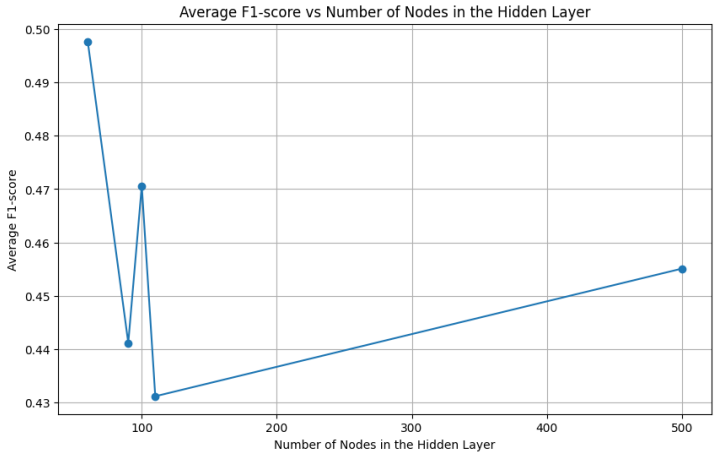
## 4 Experiments and Results

A three-fold cross-validation on the training set to determine the best hyperparameters (number of nodes in the hidden layer) for the two-layer neural network is carried out. For this, train data is split into training and validation instances. The average F1-Score of three-fold cross-validation is found. We choose the hyperparameter which gives maximum average F1-Score. Now, we retrain the model with the entire train data. The model is then tested on test data (the test data is visited only once). The system configuration used to perform the experiments are the following: a standard PC with an AMD Ryzen 9 processor and 16GB RAM. The platform used to implement the code is Python with libraries such as NumPy, Pandas, and Scikit-learn.

### 4.1 Breast Cancer Wisconsin

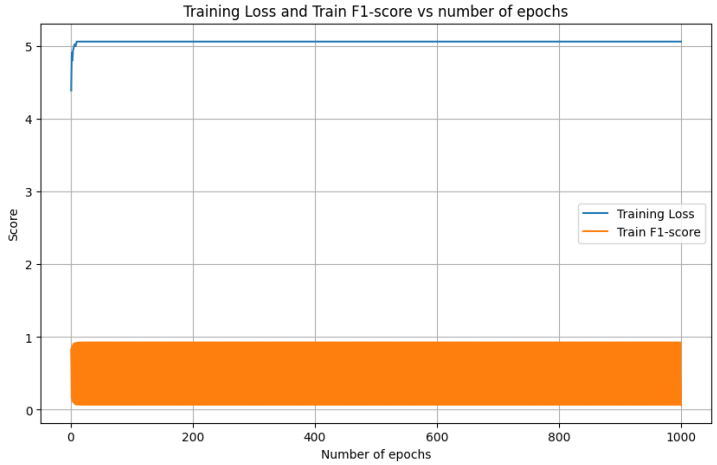
#### 4.1.1 Three fold crossvalidation

For the following number of nodes in the hidden layer = $[60, 90, 100, 110, 500]$ and learning rate ($\alpha$) = 0.21, three-fold cross-validation has been performed on the training set. A maximum average macro F1-score = 0.49766129185136804 has been obtained for the validation data in the three-fold cross-validation experiment, when the number of nodes in the hidden layer is 60. For testing, we choose $\alpha = 0.21$, and $n_h = 60$ (number of nodes in the hidden layer). Figure 4.1.1 depicts the three-fold cross-validation on breast cancer Wisconsin data for varying number of nodes in the hidden layer.




### 4.1.2 Training and Testing

The specifics of the two-layer neural network architecture is the following: the number nodes in the input layer is 30 (as there are 30 features in the Breast Cancer Wisconsin dataset), the number of nodes in the first hidden layer is 60 (as determined from the cross-validation), the number of nodes in the output layer is 2 (represented by two classes in this dataset). The train-test distribution is provided in Table 1. Training was done for 1000 epochs with $\alpha = 0.21$. The training loss is provided in Figure 4.1.2.

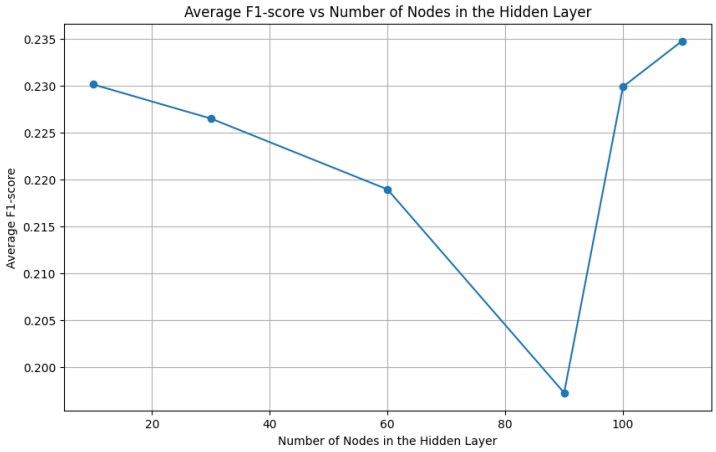



We choosed the parameters of the model $(W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}, K^{[1]})$ corresponding to the best training F1-score. From Figure 4.1.2, we choose the parameters of the network corresponding to epoch 1000. Table 2 provides the training accuracy and F1-score of Breast cancer Wisconsin data.

## 4.2 Iris

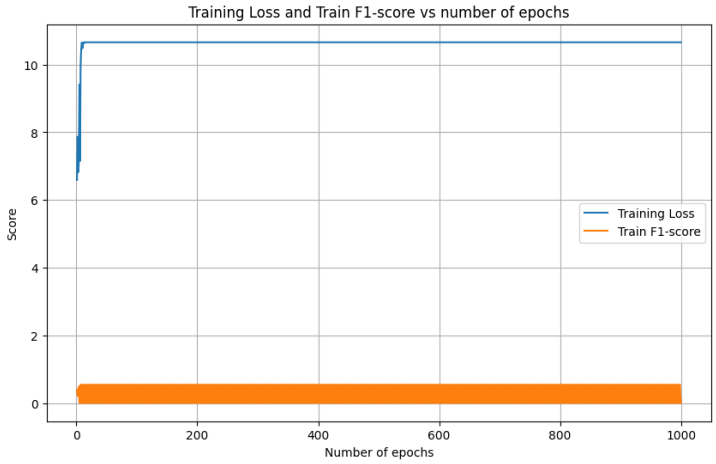
### 4.2.1 Three fold crossvalidation

For the following number of nodes in the hidden layer = $[10, 30, 60, 90, 100, 110]$ and learning rate ($\alpha$) = 0.21, three-fold cross-validation has been performed on the training set. A maximum average macro F1-score = 0.2347276270545683 has been obtained for the validation data in the three-fold cross-validation experiment, when the number of nodes in the hidden layer is 110. For testing, we choose $\alpha = 0.21$, and $n_h = 110$. Figure 3 depicts the three-fold cross-validation on Iris data for varying number of nodes in the hidden layer.




### 4.2.2 Training and Testing

The specifics of the two-layer neural network architecture is the following: the number nodes in the input layer is 4 (as there are 4 features in the Iris dataset), the number of nodes in the first hidden layer is 100 (as determined from the cross-validation), the number of nodes in the output layer is 3 (as this is a multi-class classification problem with 3 classes). The train-test distribution is provided in Table 1. Training was done for 1000 epochs with $\alpha = 0.21$. The training loss is provided in Figure 4.2.2.

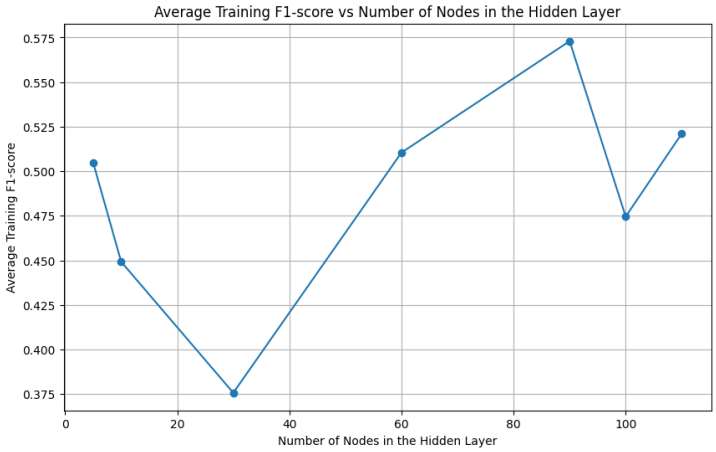



We choosed the parameters of the model corresponding to the best training F1-score. From Figure 4.2.2, we choose the parameters of the network corresponding to epoch 1000. Table 2 provides the training accuracy and F1-score of Iris data.

## 4.3 Bank Note Authentication

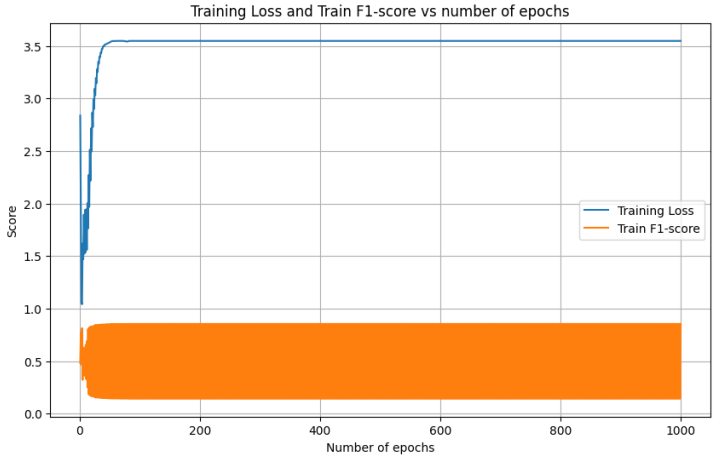
### 4.3.1 Three fold crossvalidation

For the following number of nodes in the hidden layer = $[5, 10, 30, 60, 90, 100, 110]$ and learning rate ($\alpha$) = 0.21, three-fold cross-validation has been performed on the training set. A maximum average macro F1-score = 0.5729794911151888 has been obtained for the validation data in the three-fold cross-validation experiment, when the number of nodes in the hidden layer is 90. For testing, we choose $\alpha = 0.21$, and $n_h = 90$. Figure 4.3.1 depicts the three-fold cross-validation on Bank Note Authentication data for varying number of nodes in the hidden layer.




### 4.3.2 Training and Testing

The specifics of the two-layer neural network architecture is the following: the number nodes in the input layer is 4 (as there are 4 features in the Bank Note Authentication dataset), the number of nodes in the first hidden layer is 10 (as determined from the cross-validation), the number of nodes in the output layer is 2 (represented by two classes in this datase). The train-test distribution is provided in Table 1. Training was done for 1000 epochs with $\alpha = 0.21$. The training loss is provided in Figure 4.3.2.




We choosed the parameters of the model corresponding to the best training F1-score. From Figure 4.3.2, we choose the parameters of the network corresponding to epoch 1000. Table 2 provides the training accuracy and F1-score of Bank Note Authentication data.

## 4.4 MNIST

### 4.4.1 Three fold crossvalidation

For the following number of nodes in the hidden layer = $[790, 1024, 4000]$ and learning rate ($\alpha$) = 0.21, three-fold cross-validation has been performed on the training set. A maximum average macro F1-score = 0.024003561052894404 has been obtained for the validation data in the three-fold cross-validation experiment, when the number of nodes in the hidden layer is 4000. For testing, we choose $\alpha = 0.21$, and $n_h = 4000$. Figure 2 depicts the three-fold cross-validation on MNIST data for varying number of nodes in the hidden layer.

### 4.4.2 Training and Testing

The specifics of the two-layer neural network architecture is the following: the number nodes in the input layer is 784 (as there are 784 features in the MNIST dataset), the number of nodes in the first hidden layer is 1024 (as determined from the cross-validation), the number of nodes in the output layer is 10 (as this is a multi-class classification problem with 10 classes). The train-test distribution is provided in Table 1. Training was done for 1000 epochs with $\alpha = 0.21$. The training loss is provided in Figure 3.

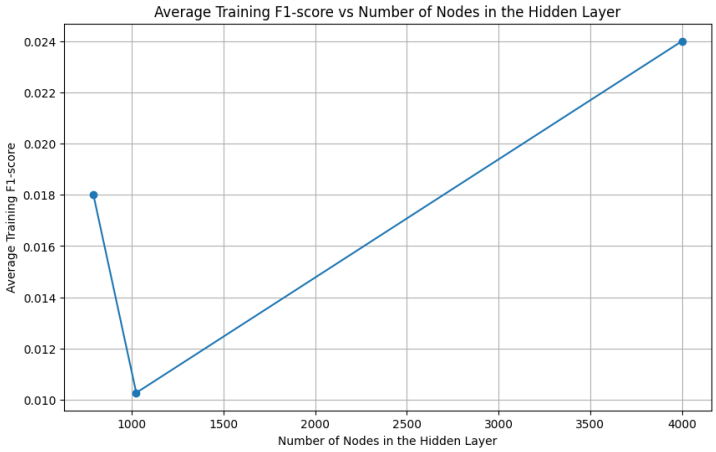



Figure 2: MNIST: Three-fold crossvalidation.

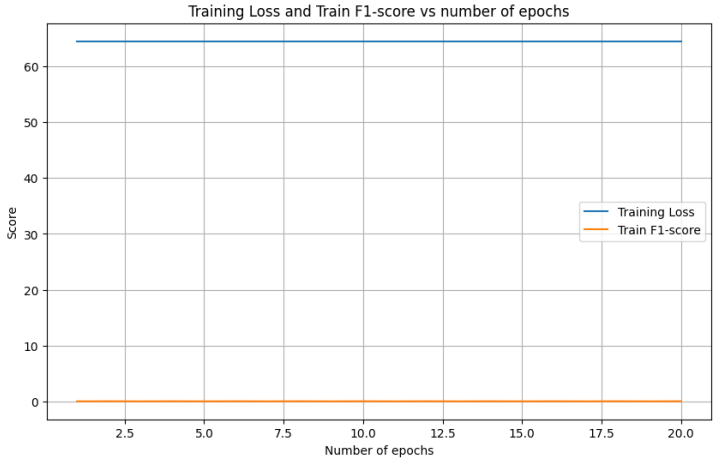



Figure 3: MNIST: Training Loss and Train F1-score vs number of epochs.

We chose the parameters of the model corresponding to the best training F1-score. From Figure 3, we choose the parameters of the network corresponding to epoch 100. Table 2 provides the training accuracy and F1-score of MNIST data.

## 5 Summary and Future Directions

## References

[1] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

Table 2: Testdata results for the two layer neural network.

| Dataset | Train Accuracy | Train F1-score | Test Accuracy | Test F1-score |
|---|---|---|---|---|
| Breast Cancer Wisconsin | 5.921052631578947% | 0.49766129185136804 | 96.49122807017544% | 0.9626596790042581 |
| Iris | 33.33333333333333% | 0.2347276270545683 | 70% | 0.5698924731182796 |
| Bank Note Authentication | 60.34083858476672% | 0.5729794911151888 | 60.3570626543903% | 0.5729794911151888 |
| MNIST | 12.52946389709044% | 0.024003561052894404 | 7.58214493471959% | 0.024003561052894404 |

[2] W Nick Street, William H Wolberg, and Olvi L Mangasarian. Nuclear feature extraction for breast tumor diagnosis. In *Biomedical image processing and biomedical visualization*, volume 1905, pages 861–870. SPIE, 1993.

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[4] R. A. FISHER. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936.

[5] Eugen Gillich and Volker Lohweg. Banknote authentication. 11 2010.

## Supplementary Material

The intermediate steps used in computing the derivatives of the loss function with respect to learnable parameters is provided below:

$$\frac{\partial L}{\partial a_1^{[2](i)}} = -\frac{y_1^{(i)}}{a_1^{[2](i)}}. \tag{28}$$

$$\frac{\partial L}{\partial a_2^{[2](i)}} = -\frac{y_2^{(i)}}{a_2^{[2](i)}}. \tag{29}$$

$$\frac{\partial a_1^{[2](i)}}{\partial z_1^{[2](i)}} = a_1^{[2](i)}(1-a_1^{[2](i)}). \tag{30}$$

$$\frac{\partial a_2^{[2](i)}}{\partial z_2^{[2](i)}} = a_2^{[2](i)}(1-a_2^{[2](i)}). \tag{31}$$

$$\frac{\partial a_2^{[2](i)}}{\partial z_1^{[2](i)}} = -a_1^{[2](i)}a_2^{[2](i)}. \tag{32}$$

$$\frac{\partial a_1^{[2](i)}}{\partial z_2^{[2](i)}} = -a_1^{[2](i)}a_2^{[2](i)}. \tag{33}$$

$$\frac{\partial z_1^{[2](i)}}{\partial a_1^{[1](i)}} = w_{11}^{[2]}. \tag{34}$$

$$\frac{\partial z_1^{[2](i)}}{\partial a_2^{[1](i)}} = w_{12}^{[2]}. \tag{35}$$

$$\frac{\partial z_2^{[2](i)}}{\partial a_1^{[1](i)}} = w_{21}^{[2]}. \tag{36}$$

$$\frac{\partial z_2^{[2](i)}}{\partial a_2^{[1](i)}} = w_{22}^{[2]}. \tag{37}$$

$$\frac{\partial z_2^{[2](i)}}{\partial a_2^{[1](i)}} = w_{22}^{[2]}. \tag{38}$$

$$\frac{\partial a_1^{[1](i)}}{\partial k_1^{[1]}} = z_1^{[1]} = w_{11}^{[1]}x_1^{(i)} + w_{12}^{[1]}x_2^{(i)} + b_1^{[1]}. \tag{39}$$

$$\frac{\partial a_2^{[1](i)}}{\partial k_1^{[1]}} = 0. \tag{40}$$

$$\frac{\partial a_1^{[1](i)}}{\partial k_0^{[1]}} = 1. \tag{41}$$

$$\frac{\partial a_2^{[1](i)}}{\partial k_0^{[1]}} = 1. \tag{42}$$

$$\frac{\partial z_1^{[2](i)}}{\partial w_{11}^{[2]}} = a_1^{[1](i)}. \tag{43}$$

$$\frac{\partial z_1^{[2](i)}}{\partial w_{12}^{[2]}} = a_2^{[1](i)}. \tag{44}$$

$$\frac{\partial z_2^{[2](i)}}{\partial w_{21}^{[2]}} = a_1^{[1](i)}. \tag{45}$$

$$\frac{\partial z_2^{[2](i)}}{\partial w_{22}^{[2]}} = a_2^{[1](i)}. \tag{46}$$

$$\frac{\partial z_1^{[2](i)}}{\partial b_1^{[2]}} = 1. \tag{47}$$

$$\frac{\partial z_2^{[2](i)}}{\partial b_2^{[2]}} = 1. \tag{48}$$

$$\frac{\partial z_1^{[1](i)}}{\partial w_{11}^{[1]}} = x_1^{(i)}. \tag{49}$$

$$\frac{\partial z_1^{[1](i)}}{\partial w_{12}^{[1]}} = x_2^{(i)}. \tag{50}$$

$$\frac{\partial z_2^{[1](i)}}{\partial w_{21}^{[1]}} = x_1^{(i)}. \tag{51}$$

$$\frac{\partial z_2^{[1](i)}}{\partial w_{22}^{[1]}} = x_2^{(i)}. \tag{52}$$

$$\frac{\partial z_1^{[1](i)}}{\partial b_1^{[1]}} = 1. \tag{53}$$

$$\frac{\partial z_2^{[1](i)}}{\partial b_2^{[1]}} = 1. \tag{54}$$

$$\frac{\partial L}{\partial z_1^{[1](i)}} = \sum_{j=1}^{C} \frac{\partial L}{\partial a_j^{[2](i)}} \cdot \frac{\partial a_j^{[2](i)}}{\partial z_j^{[2](i)}} \cdot \frac{\partial z_j^{[2](i)}}{\partial a_1^{[1](i)}}, \tag{55}$$

On simplification we get,

$$\frac{\partial L}{\partial z_1^{[1](i)}} = -\frac{1}{m} \sum_{i=1}^{m} \left( y_1^{(i)} (1 - a_1^{[2](i)}) w_{11}^{[2]} + y_2^{(i)} (1 - a_2^{[2](i)}) w_{21}^{[2]} \right). \tag{56}$$

$$\frac{\partial L}{\partial z_2^{[1](i)}} = \sum_{j=1}^{C} \frac{\partial L}{\partial a_j^{[2](i)}} \cdot \frac{\partial a_j^{[2](i)}}{\partial z_j^{[2](i)}} \cdot \frac{\partial z_j^{[2](i)}}{\partial a_2^{[1](i)}}, \tag{57}$$

On simplification we get,

$$\frac{\partial L}{\partial z_2^{[1](i)}} = -\frac{1}{m} \sum_{i=1}^{m} \left( y_1^{(i)} (1 - a_1^{[2](i)}) w_{12}^{[2]} + y_2^{(i)} (1 - a_2^{[2](i)}) w_{22}^{[2]} \right). \tag{58}$$