Explainable Clustering in a Classification Neural Network

Dhruvit Kothari dhruvit.kothari@gmail.com

The code for the following experiment is bundled with the paper.

Abstract

Neural networks are supervised learning algorithms which accept input from the user, apply numerous transformations on the input and then spit out an output. Different layers of the neural network apply different transformations based on the weights associated with the neurons in that given layer. A weight is a real number associated with a neuron that decides the strength of that neuron. Neural networks have seen a recent surge in their usage and deployment due to the quantity of data available. Mathematically speaking, the neural network architecture can map any arbitrary function f(x) to their output 'y'. (given enough data and by employing the right neural network architecture) This is a consequence of the Universal Approximation Theorem i.e. no matter what the function f(x) is, there exists a network that will approximately approach the result.

Training of the neural network is what makes them so special and easy to use. The network is trained through prior examples, where a function of the difference in the output produced by the network and the true output is used to adjust the weights of the neuron. This function is known as the loss function. The neural network through the backpropagation algorithm adjusts the weights of the neural network in such a way that the output processed by the network is similar to the true output i.e. the loss function is minimised. After the given criteria for accuracy is met, the training can be terminated.

Neural Networks are usually treated as a black box after the network is trained. It accepts input, applies magical calculations and then spits out an output. The probability in their prediction and why they output a certain prediction cannot be understood clearly. Overall we can only look at the values of the weights that are associated with the neurons and make an assumption about what the neural network is trying to learn.
What does the Neural network learn when we train it to classify things? Does it make a nonlinear/linear boundary like the other machine learning algorithms? These are some questions that have troubled my mind from the moment I started reading about them. To effectively use neural networks in our daily life we should be confident of their prediction and be able to give a value to that confidence. The uncertainty of the prediction by the network makes

them unreliable to use in situations where expensive equipment or lives are at stake. The experiment ahead aims to help us understand the working of the neural network further.

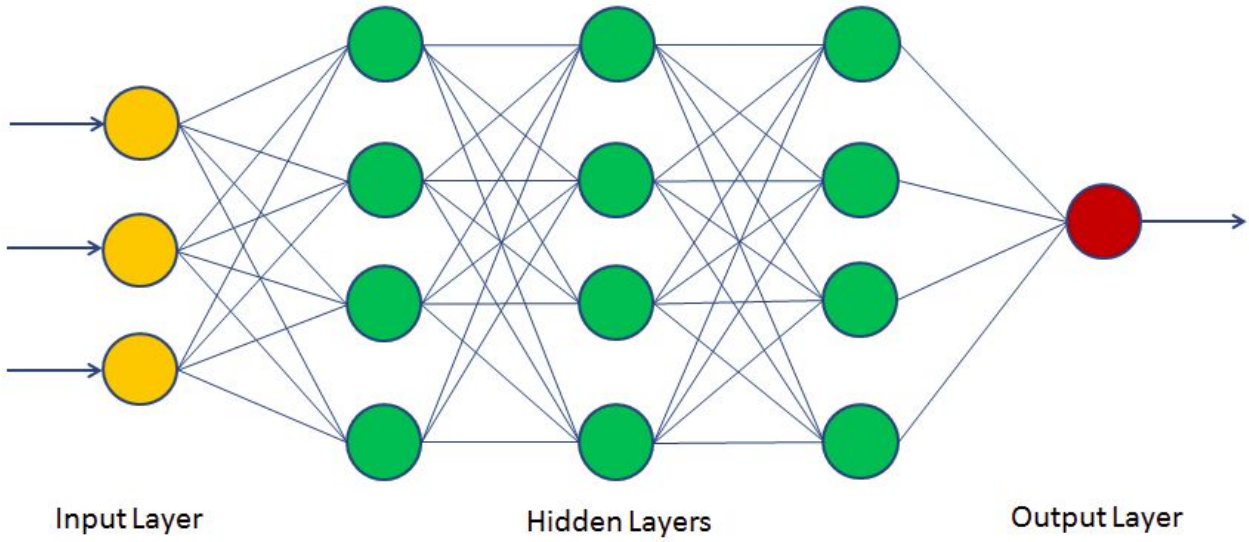


Figure 0: A typical neural network architecture.

The Experiment.

The backpropagation algorithm is truly the backbone of the neural network. As training proceeds, it adjusts the weights of the neural network so that the processed output is similar to the true output. The neural network can learn any complex function given enough data.
Till now gradient descent was only applied to the weights of the neural network and not the input data. But what if the network was allowed to modify the input data?

Step 1.
Generate 500 random data points with two features ie x and y. Each data point is generated randomly from a uniform distribution and has a value between 1000 and -1000.
The data points are assigned classes randomly from a range between 0 and 2. Eg
Data point 1 - [52,-100] assigned class - [0]
Data point 2 - [487,200] assigned class - [1]
Data point 3 - [529,-255] assigned class - [2]
.

.
.

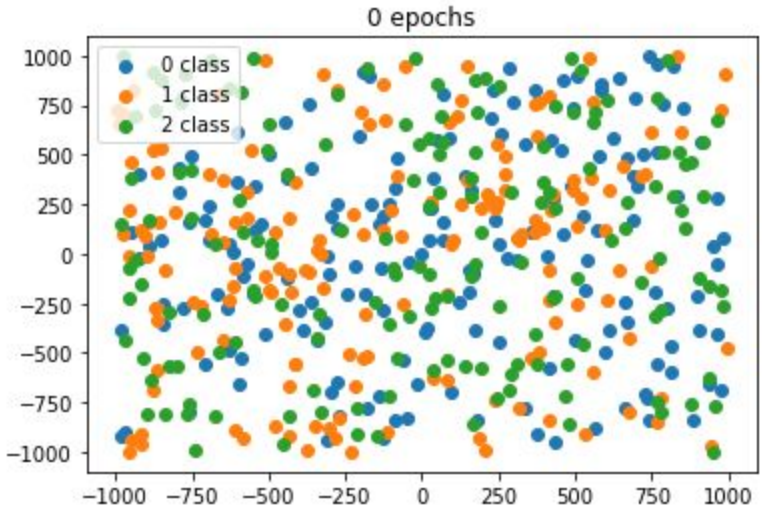
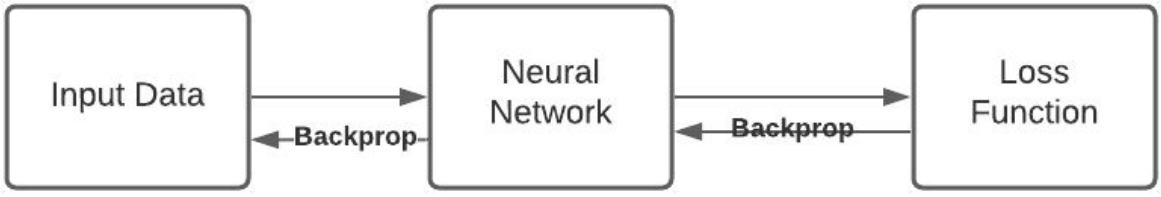


Figure 1: Data points plotted
As we can see the data points are scattered all over the graph randomly.

Step 2:
Train the neural network on the data like the normal training process, with the only exception that the gradients are also calculated for the input data points with respect to the loss function and then applied with a different learning rate than the neural network. An epoch refers to one cycle through the full training dataset.
The data points are plotted at different epochs steps.



Gradient Descent Step for Input Data

$$\mathbf{I = I - \alpha \cdot \nabla J(I)}$$

Where :
I: Input Data, α : transformation learning rate,
∇J(θ): Derivative of loss w.r.t. Input Data

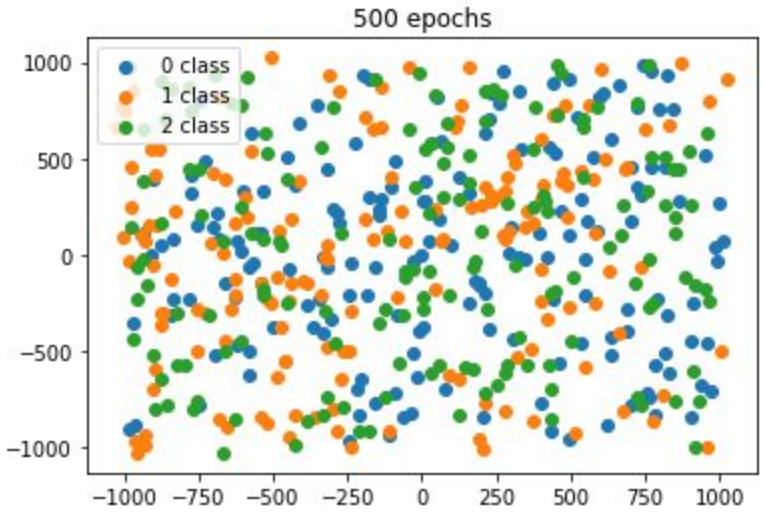


Figure 2: Data points plotted after 500 epochs.
Not much has changed in the graph after training the network for 500 epochs.

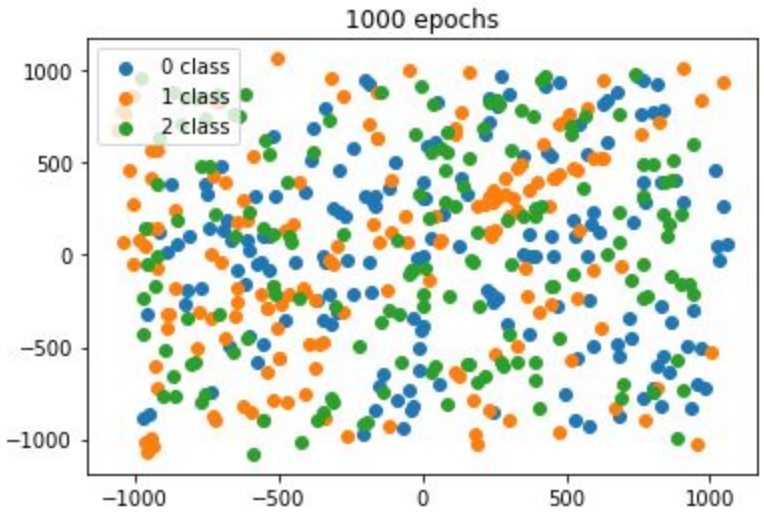


Figure 3: Data points plotted after 1000 epochs
Clusters of various sizes seem to be forming in the graph.

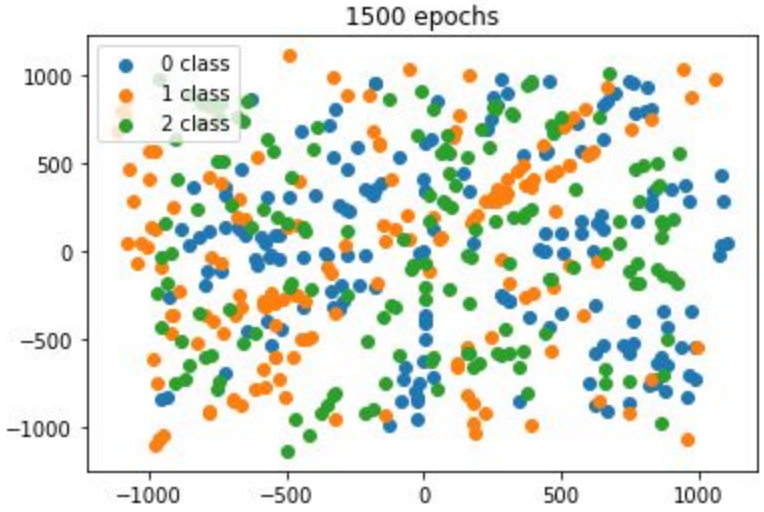


Figure 4: Data points plotted after 1500 epochs

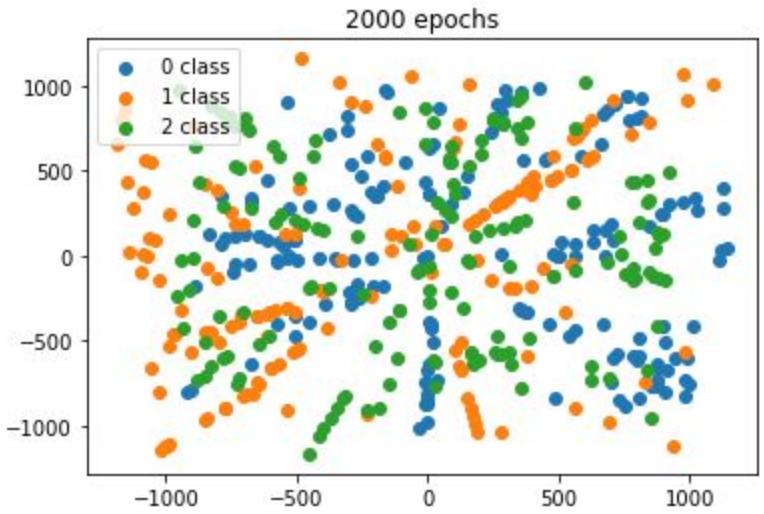


Figure 5: Data points plotted after 2000 epochs
A visible orange cluster can be seen in the graph. Small blue and green are also noticeable

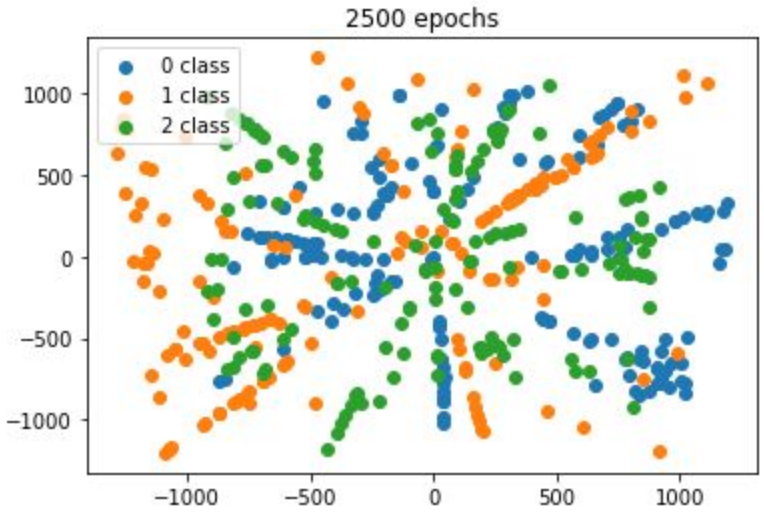


Figure 6: Data points after 2500 epochs.
Clusters are distinct and unique now.

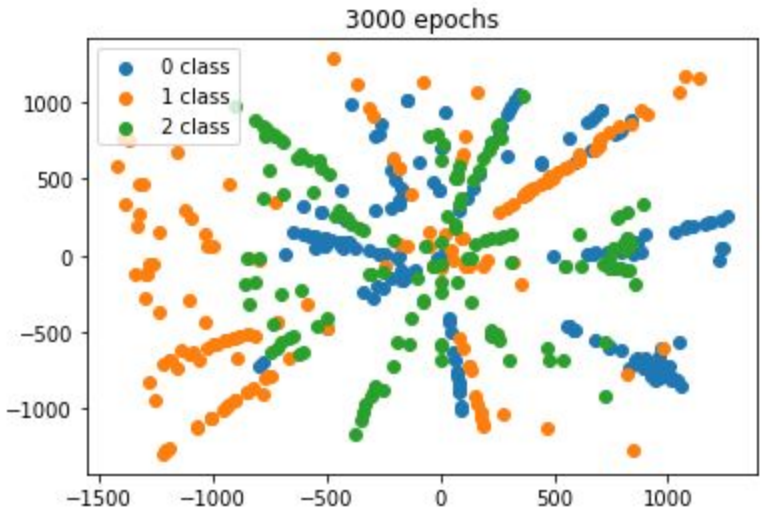


Figure 7: Data points after 3000 epochs.

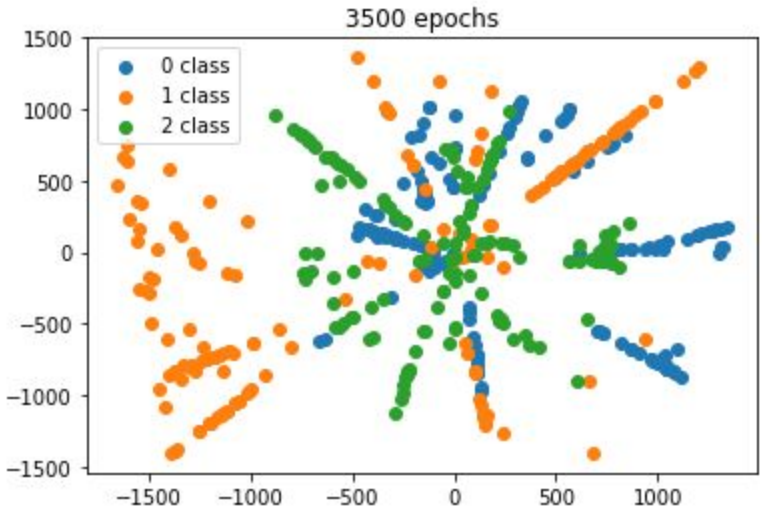


Figure 8: Data point after 3500 epochs

The orange cluster in the bottom left of the graph seems to move away from the other clusters.

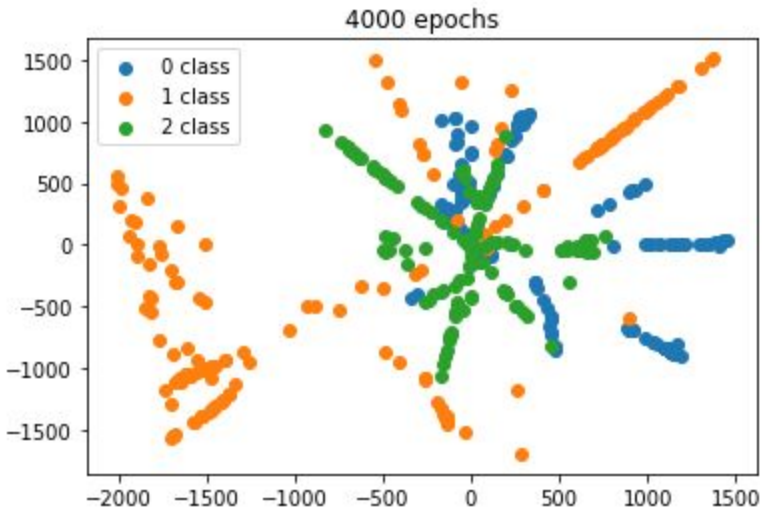


Figure 9: Data points after 4000 epochs.
The green class has formed a huge cluster in the center of the graph. The orange clusters seem to move away from the center.

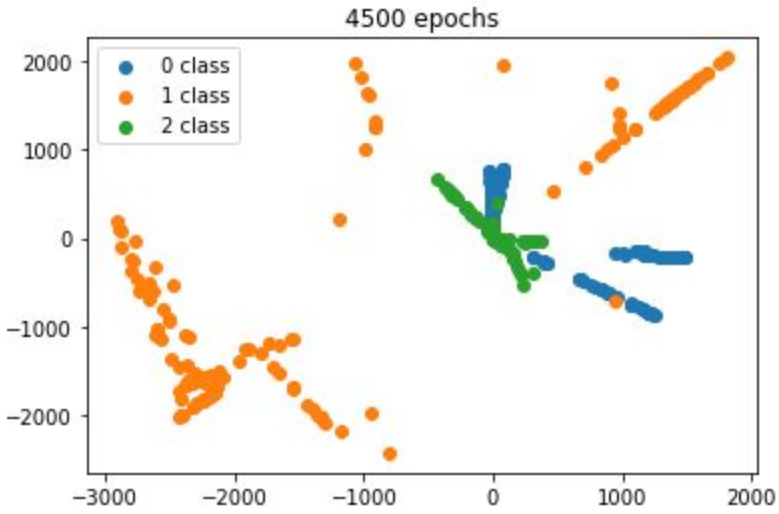


Figure 10: Data point after 4500 epochs.
Distinct clusters of each colour can be seen in the graph.

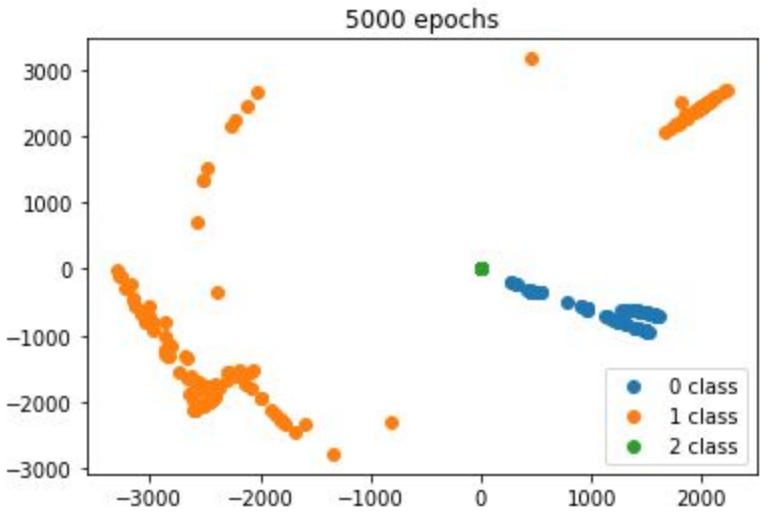


Figure 11: Data point after 5000 epochs

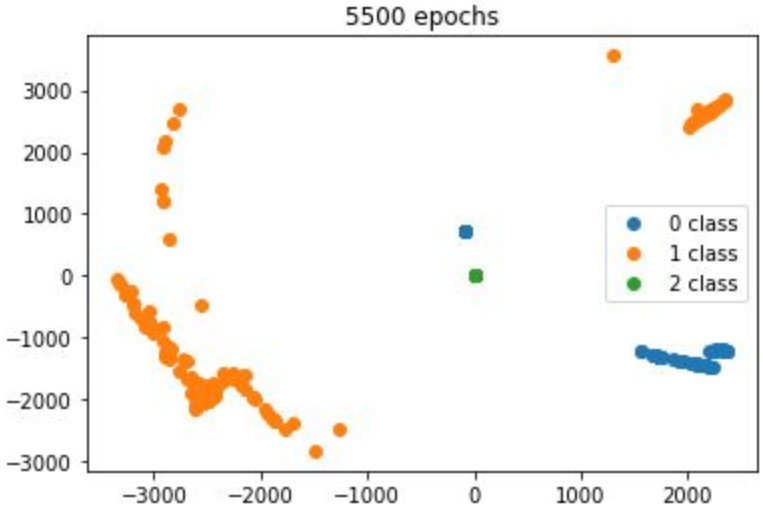


Figure 11: Data point after 5500 epochs.

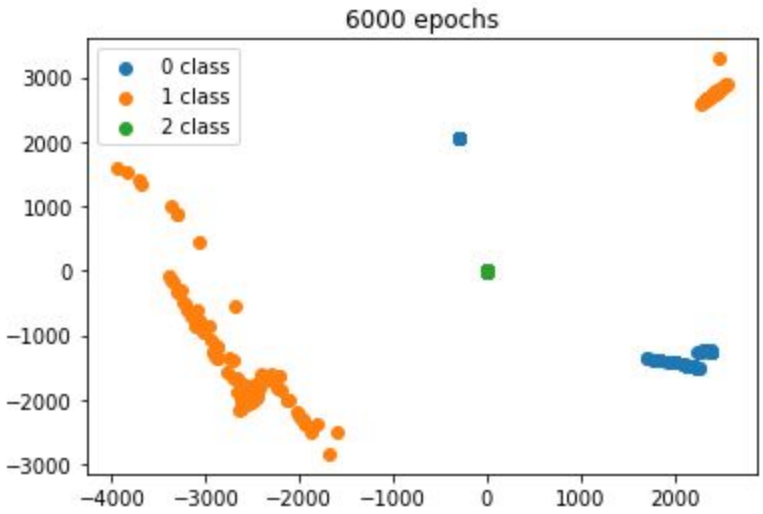


Figure 12: Data points after 6000 epochs

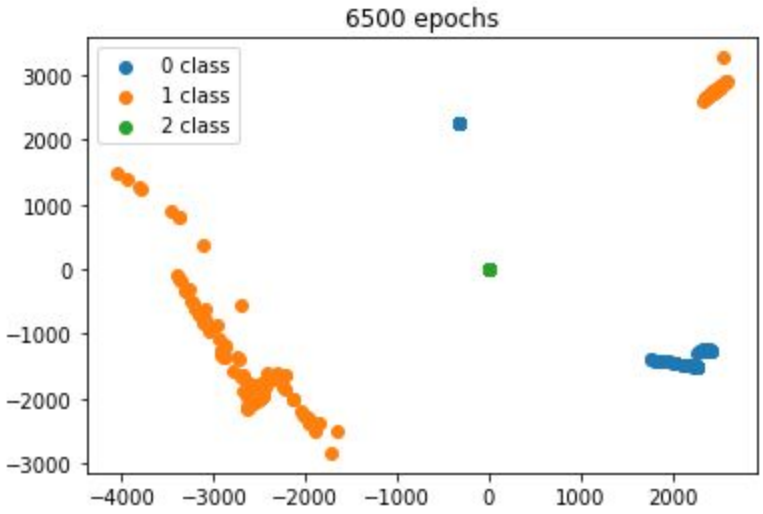


Figure 13: Data Points after 6500 epochs

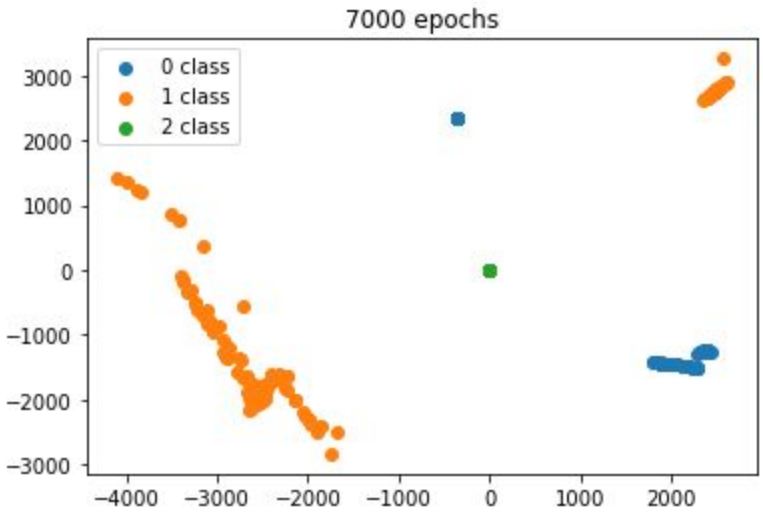


Figure 14: Data Points after 7000 epochs

A new set of data points was generated with three features(x1,x2,x3) and the same experiment was repeated. The graphs for the following experiment are as follows.

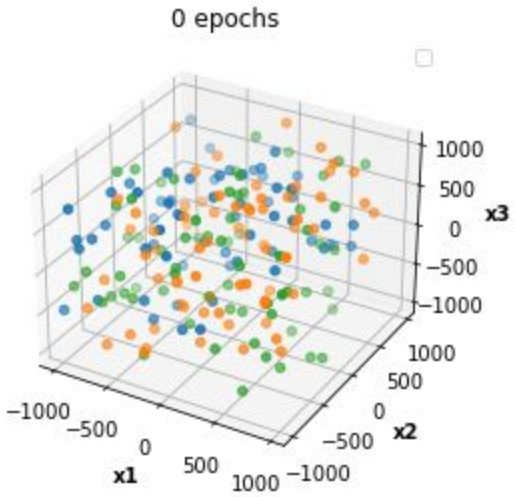


Figure 15: Plot of data points with three features.

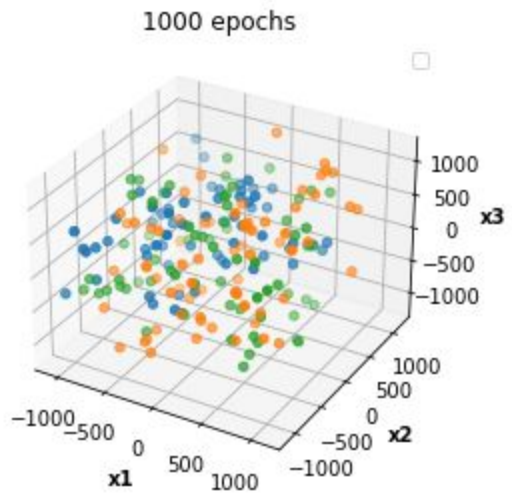


Figure 16: Data points plotted after 1000 epochs.

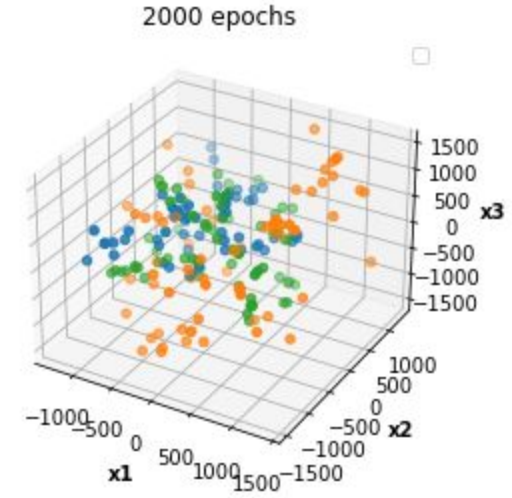


Figure 17: Data points plotted after 2000 epochs.

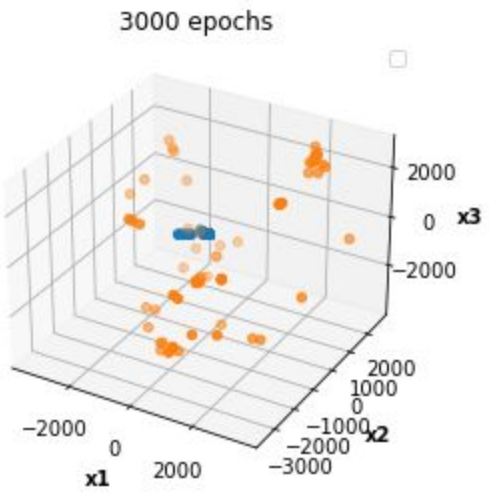


Figure 18: Data points plotted after 3000 epochs.

Results and Findings.

It is interesting to see how the graph transforms from a randomized scatter of points to distinct clusters as the training continues. The network clusters the points of the same class together so it can learn the classification task easily and with more accuracy. During the training process, the loss of the network keeps on decreasing albeit by a small value but the decrease is visible as the epochs compound. As you can see in Figure 11, the three classes have been separated completely from one another and now even a simple machine learning algorithm can classify them with high accuracy. Such applications were not possible with the initial data set where each point was generated randomly. This behaviour of neural networks is very interesting for classification tasks like in the case of vectors embedding generated by convolutional neural networks in facial recognition systems. More research should be done in this area of explainable artificial intelligence where the predictions made by the network are used for security and law enforcement purposes.

Clustering in a higher dimension would be easier for the network as it has more axes to differentiate the classes in different clusters. As we can see the network was able to separate the clusters of data points with three features in 3000 epochs of training vs 5000+ epoch training by the same network for data points with two features.Clustering of the data points by the network gives us an insight about what the network is trying to learn during a classification task.

References

1. Csáji, B.C., 2001. Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, *24*(48), p.7.

2. Hecht-Nielsen, R., 1992. Theory of the backpropagation neural network. In *Neural networks for perception* (pp. 65-93). Academic Press.

3. Anthony, M. and Bartlett, P.L., 2009. *Neural network learning: Theoretical foundations*. Cambridge University press.

4. Schroff, F., Kalenichenko, D. and Philbin, J., 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 815-823).