A Comparative Study between Machine Learning (Neural Network) Algorithms

DONE BY:

Smriti Sivakumar
PES1UG20CS428

Utkarsh Bagaria
PES1UG20CS477

Abstract

Artificial Intelligence and Machine Learning through different neural networks have profoundly altered how human beings would solve problems in recent years. People have also come up with multiple different types of neural networks. Each neural network comes with its own set of advantages and disadvantages. This paper will take up a few of the neural networks and compare them based on their different characteristics to determine which neural network is better for a given problem, including the implementation for a few.

Introduction

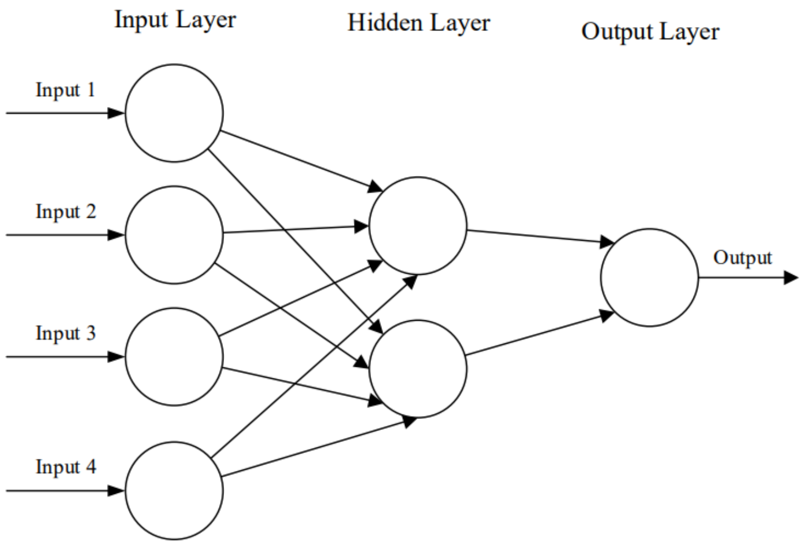
The neural networks taken up in this research paper to perform the comparative analysis are Graph Convolutional Networks, Artificial Neural Networks and Convolutional Neural Networks. Simple Artificial Neural Networks form the basis of Neural Networks being similar to the human neuron. They try to replicate the human neural network through various structures and algorithms. The Convolutional Neural Network comes under Artificial Neural Networks and is a more advanced form of the simple ANN. CNN reduces the complexity of ANNs and makes it more accurate and efficient to run on large datasets as well. The GCN algorithm is another variant of Convolutional Neural Networks which operate directly on graphs.

Literary Review

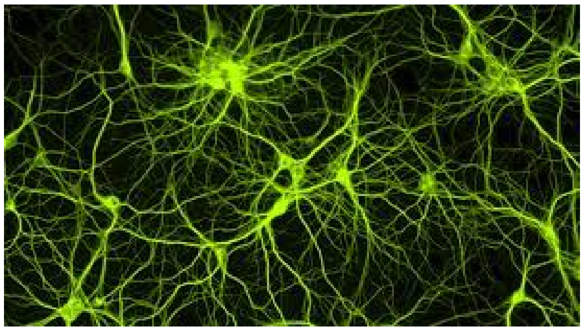
Algorithm

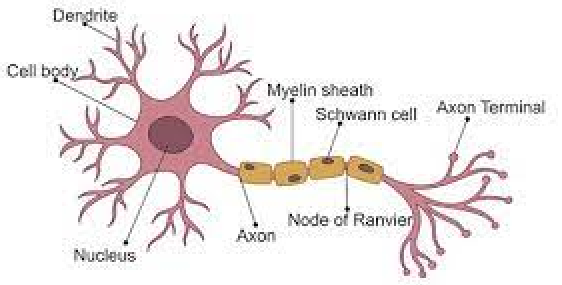
Artificial neural networks

Artificial neural networks, usually simply called neural networks or neural nets, are computing systems inspired by the biological neural networks that constitute animal brains. An ANN is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain.



A simple three layer feedforward neural network, comprising an input layer, a hidden layer and an output layer. This usually forms the basis of an Artificial Neural Network. This structure shown above is what is used to mimic the human neuron. The human neural network is a web of these interconnected neurons with the neurons being millions in number. These neurons allow parallel processing in the human body and make it the best example of parallel processing. These neurons try to process information from one neuron to the other and this is what Artificial Neural Networks try to achieve through the structure shown above.





The neural network in Human Body.   Structure of a neuron in the human body.

The fundamental concept in ANN lies in the presence of neurons with multiple inputs, and a single output.

$$o = f\left(\sum_{i=1}^{m} w_i\, x_i\right) = f\left(\mathbf{w}^t \mathbf{x}\right),$$

Here, x is the input vector ,and w is the weight vector. The output is equated to the activation function on the RHS. The goal is to minimize loss, or by extension, the loss function, or, the loss that is back propagated through the inner layers. The regularization function is as follows:

$$\lambda(w) = \sum_{p=1}^{n} \sum_{k=1}^{K} \left(d_{kp} - o_{kp}\right)^2 .$$

Performance measures, such as Root Mean Square Error, Mean Average Error, etc, can be applied on the Regularization Function. The models are structured and trained with an aim of minimizing the Regularization Function.
Investigations on Training Tolerances were carried out by researchers to measure the above stated performance metrics.

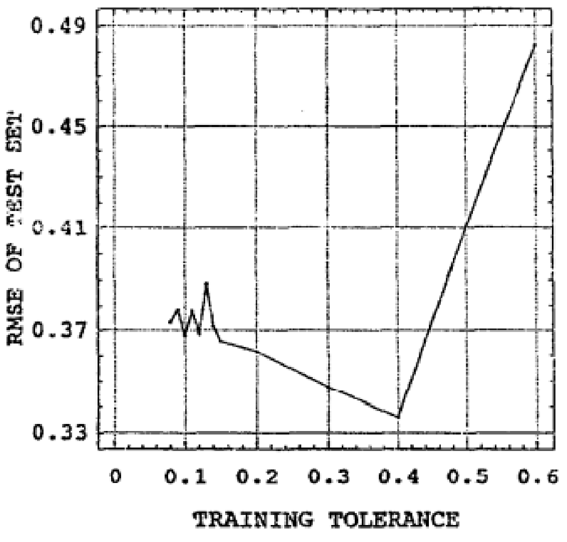


Figure 2. Test set results for Root Mean Squared Error (RMSE) vs. training tolerance.

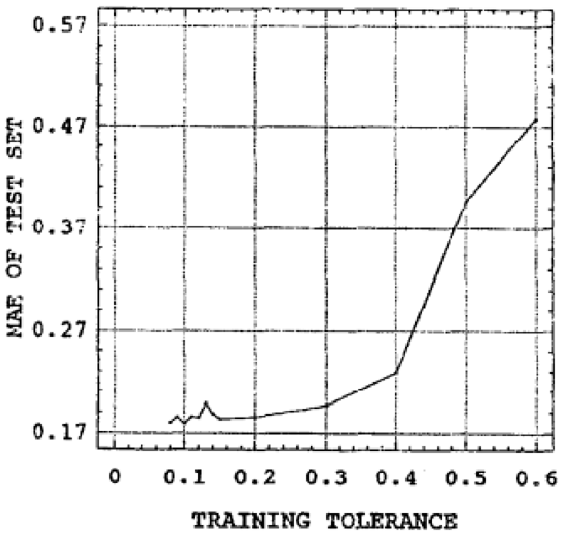


Figure 3. Test set results for Mean Absolute Error (MAE) vs. traning tolerance.

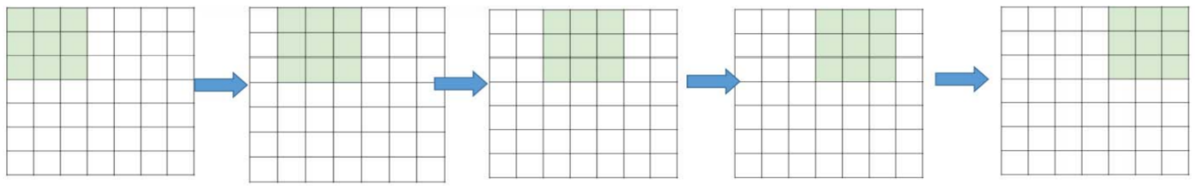
Convolutional Neural Networks

Convolutional Neural Networks is one of the various types of Artificial Neural Networks which are used for different applications and data types.The most beneficial aspect of CNNs is reducing the number of parameters in ANNs. This achievement has prompted both researchers and developers to approach larger models in order to solve complex tasks which were not possible using simple ANNs. A very important assumption about problems to be solved by CNN is that they should not have features which are spatially dependent. This means that we do not have to worry about the location of the feature. Detecting the feature regardless of its location.

The most significant ways in which CNNs better the performance of ANNs is through the convolutional layer. Assuming a simple ANN with one neuron in the hidden layer. One takes a coloured image with 32x32 pixels and depth of 3 for the RGB channel. If one wants to add another neuron in the hidden layer in a simple ANN, one would have to make 32x32x3 weight connections doubling the number of parameters. This makes it more than 6000 parameters to just connect the input to the hidden layer of 2 neurons. Here is where CNN is significantly different from ANN. A more efficient deployment by CNN is that instead of a full connection, looking for local regions in the picture instead of in the whole image. Considering the above example, if one only sends a 5x5 pixel data to the next layer one will only have 5x5x3 weight connections. Thus in a complete neural network CNN would be almost 41 times more efficient than simple ANN for the example taken.

If one fixes the weights for the local connections it would become similar to a sliding window of 5x5x3 in the input neuron. This allows the ability to detect the feature regardless of its location in the image. This is the feature because of which the algorithm gets the name Convolution.

Another way CNN helps in reducing the number of parameters is the stride for the sliding window. For example if we have a 7x7 image and move or slide the filter by one node each time we would have a 5x5 output. If we move and make every stride 2, then the output will be 3x3. Put simply, not only overlap, but also the size of the output will be reduced.

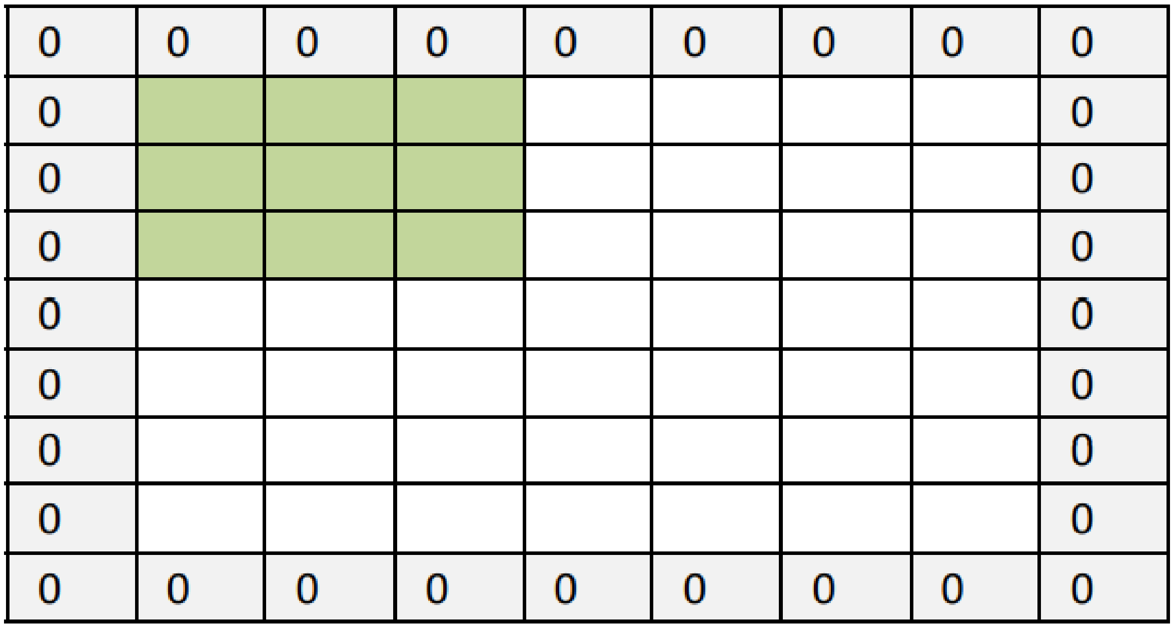


Formula to calculate this stride is :

O = 1 + (N-F)/S,

where O is the output size, N is the input size, F is the filter size and S is the stride size.

One of the drawbacks of the convolution step is the loss of information that might exist on the border of the image. Because they are only captured when the filter slides, they never have the chance to be seen. A very simple, yet efficient method to resolve the issue, is to use zero-padding. The other benefit of zero padding is to manage the output size.



This would change the formula stated above as

O = 1 + (N-F+2P)/S,

where P is the amount of padding on one side. P=1 in the image shown above.

The process Pooling is down-sampling in order to reduce the complexity for further layers. In the image processing domain, it can be considered as similar to reducing the resolution. Pooling does not affect the number of filters. Max-pooling is one of the most common types of pooling methods.

Metrics such as Global Accuracy, IoU, Lesion, etc. can be used to evaluate CNNs.

$$global\ accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$accuracy = \frac{(TP/TP + FN) + (TN/TN + FP)}{2}$$

$$IoU = \frac{Lesion + Background}{2}$$

Where:

$$Lesion = \frac{TP}{TP + FN + FP}$$

$$Background = \frac{TN}{TN + FN + FP}$$

Graph Convolutional Networks

Exploring further on the concept of Convolutional Neural Networks, we arrive at Graph Neural Networks, a new form of Artificial Neural Network, based on graph models. As a result, a GNN model can be much more similar to people's minds because it uses graph models to represent the relationships between objects. A graph, put simply, can be thought of as a collection of edges and vertices, where the vertices are linked by edges, possibly representing some relation. Graphs have great expressive power, thus they can denote a large number of complex systems across various domains.

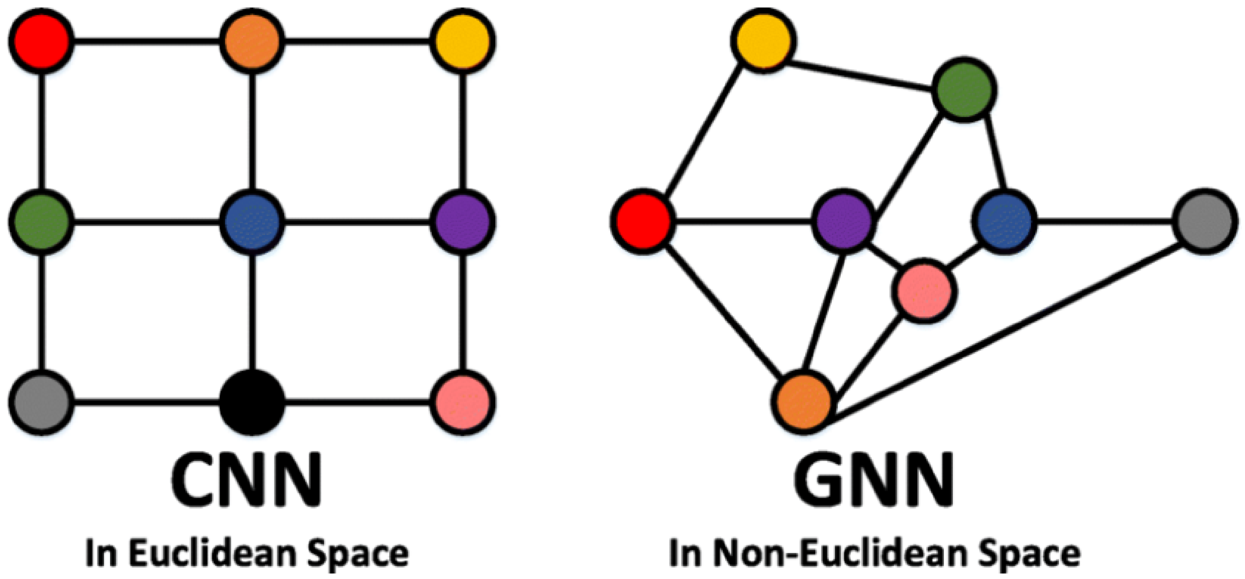
On datasets with a large number of nodes and long node features, GNN models do not make good use of the parallelism of GPU training. The data loading time for tasks of graph level is a major part of training time because batching multiple graphs into a single large graph is pretty time-consuming, which also leads to the low GPU utilization of GNNs.

TYPES OF GNN MODELS:

1. **GCN**
2. GatedGCN
3. GIN
4. GAT
5. GraphSAGE
6. MoNet

GCN is their simplest form that works by performing a linear approximation to spectral graph convolutions.

In addition to the graph structure and node features, GCN can capture the complex interactions between nodes by aggregating neighboring information and performing nonlinear transformations on feature dimensions to produce new features. By utilizing such capabilities, GCN is able to achieve state-of-the-art performance on graph-related applications. A significant difference between CNNs and GCNs arise on the data they operate on; CNN operates on Euclidean/regularly ordered data whereas GCN(essential GNN, but extends to GCN as well) operates on graph data (non- Euclidean space), involving nodes and edges, with no constraints on ordering.



Source: ResearchGate

For GCNs, the goal is to learn a function of signals/features on a graph which takes the input as follows:

- A feature description x_i for every node i;summarized in a N * D feature matrix X(N:number of nodes, D: number of input features).
- A representation description of the graph structure in matrix form; typically in the form of an adjacency matrix A (or some function thereof).

CNNs are governed by its propagation rules, i.e. Forward and Backward Propagation Rule. GCNs, however, have its propagation rule governed by the following formula:

$$h_i^{k+1} = \sigma(W^k \frac{1}{deg_i} \sum_{j \in N_i} h_j^k),$$

Here, h^k_i is a vector of activations of node i in the kth neural network layer, N_i is the set of nodes connected to node i on the graph, $deg_i = |N_i|$ is the degree of node i, $\sigma(\cdot)$ denotes a differentiable, non-linear activation function, and W^k is a layer-specific weight matrix.

Application Differences

ANN has a broad field of applications. They can do classifications, clustering, experimental design, modeling, mapping, etc. The ANNs are quite flexible for adaptation to different types of problems and can be custom-designed to almost any type of data representations. ANNs are not a good choice if the data do not represent or are not correlated well enough to the information sought, secondly, if the user does not know exactly what should be achieved, and third, if other standard methods have not been tried as well.

CNN is specifically used for image recognition and tasks that involve the processing of pixel data. There are other types of neural networks in deep learning, but for identifying and recognizing objects, CNNs are the network architecture of choice.
Being such a great choice for image classification and object recognition it is extensively used in medical image computing. CNN medical image classification detects anomalies in X-ray and MRI images with better accuracy than the human eye.

GCN is applicable in solving various problems related to research operations and combinatorial optimisation problems. They can be used to solve various problems such as:

- Traveling Salesman
- Quadratic Assignment Problem etc.

With the help of the input graph, it can outclass traditional complex algorithms.

Implementation:

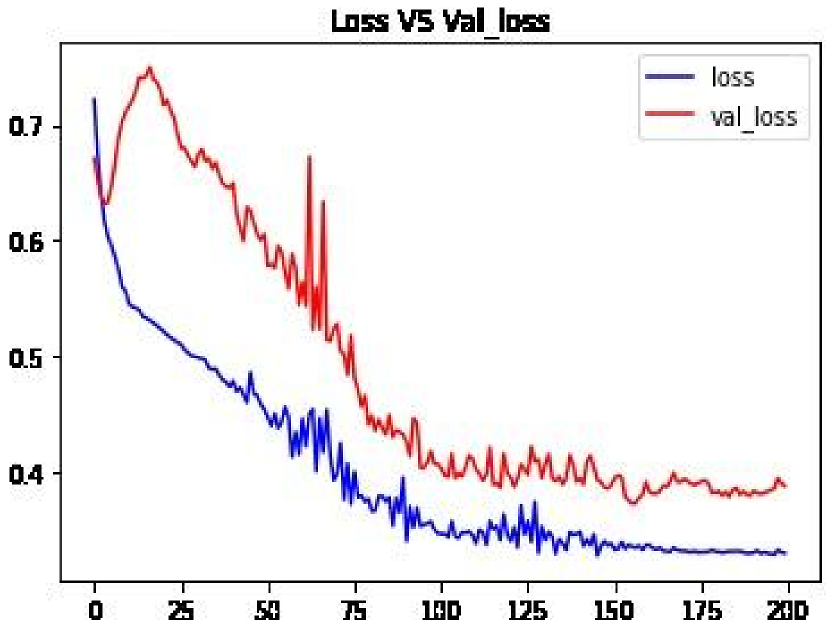
To try and substantiate our findings, we sought to implement these algorithms on a problem, to try and classify it as best solved by a particular NN algorithm.
Our test problem, was to conduct predictive stock analysis, on **Indian Penny Stocks**.
This was a problem that we previously attempted to solve using LSTM, CNN and XGBoost, but this time, we sought to run it separately using ANN, CNN and GCN.

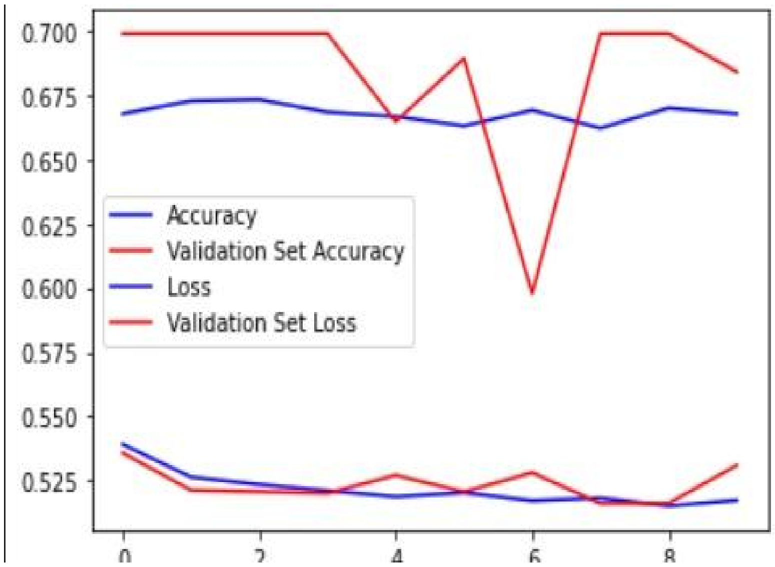
The dataset was obtained through sources such as Screener, Yahoo Finance, etc.

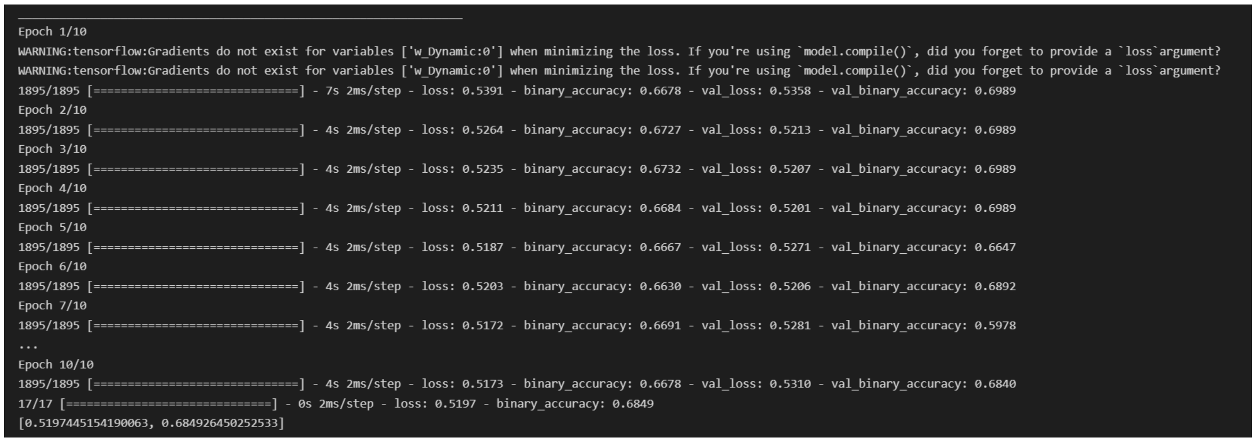
Once passed through the three networks separately, we were able to get good results through ANN and GCN. Our CNN solution seemed to be hit-and-miss, however, we saw that better results were obtained using GCN as compared to ANN, though the competition was close.

Github Link : https://github.com/smriti0302/comparison-of-nn-algorithms

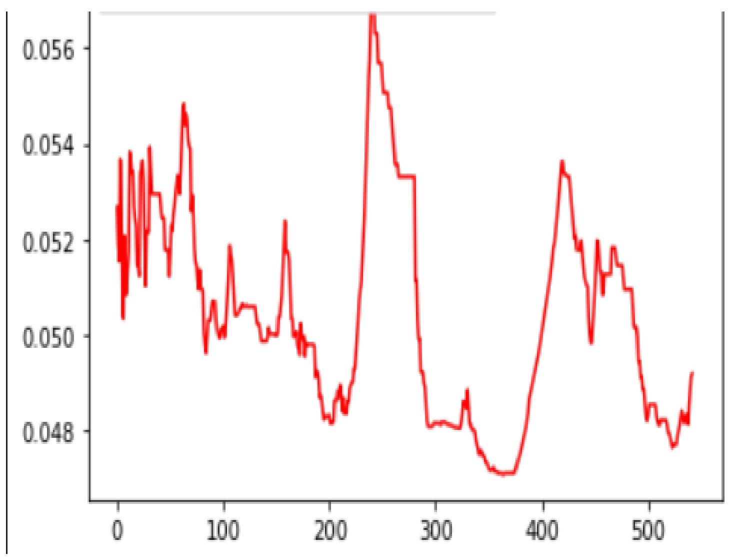
OUTPUT SCREENSHOTS:

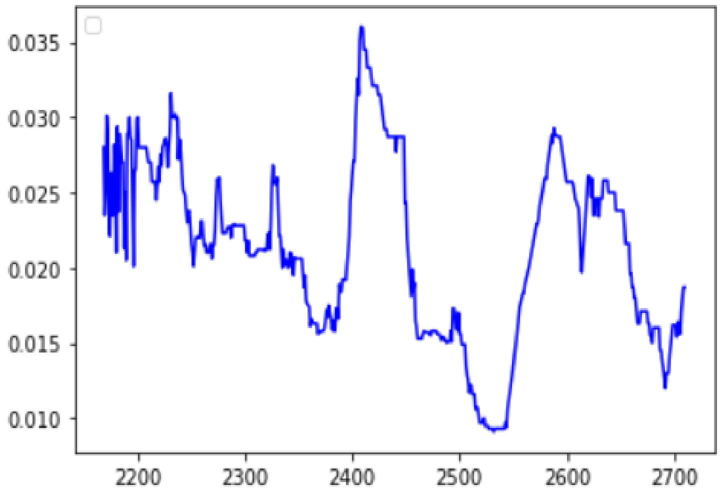


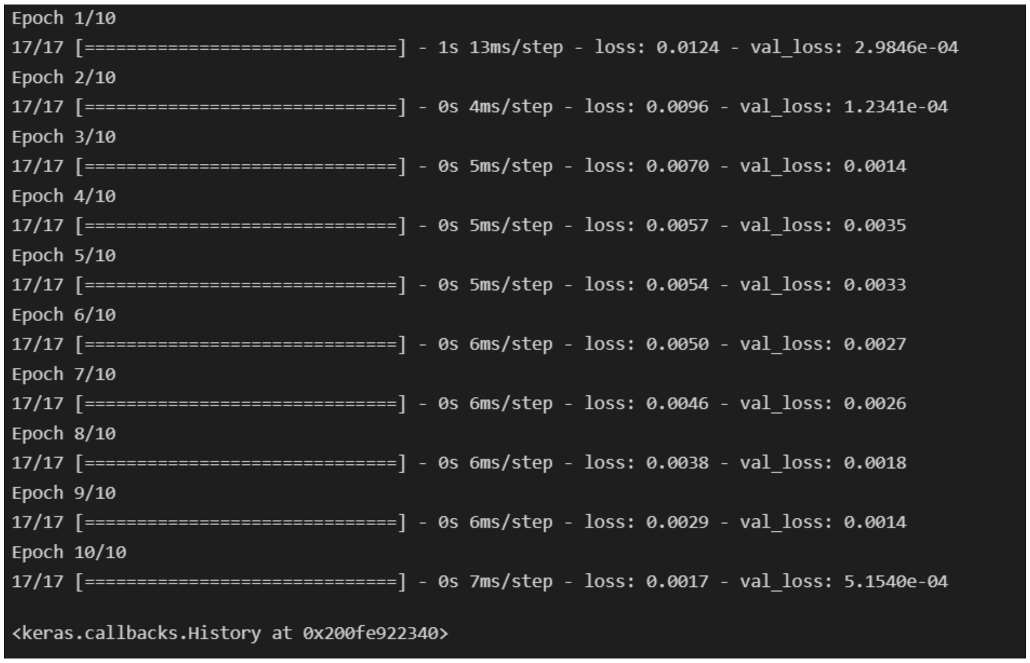




```
Epoch 1/10
WARNING:tensorflow:Gradients do not exist for variables ['w_Dynamic:0'] when minimizing the loss. If you're using `model.compile()`, did you forget to provide a `loss`argument?
WARNING:tensorflow:Gradients do not exist for variables ['w_Dynamic:0'] when minimizing the loss. If you're using `model.compile()`, did you forget to provide a `loss`argument?
1895/1895 [==============================] - 7s 2ms/step - loss: 0.5391 - binary_accuracy: 0.6678 - val_loss: 0.5358 - val_binary_accuracy: 0.6989
Epoch 2/10
1895/1895 [==============================] - 4s 2ms/step - loss: 0.5264 - binary_accuracy: 0.6727 - val_loss: 0.5213 - val_binary_accuracy: 0.6989
Epoch 3/10
1895/1895 [==============================] - 4s 2ms/step - loss: 0.5235 - binary_accuracy: 0.6732 - val_loss: 0.5207 - val_binary_accuracy: 0.6989
Epoch 4/10
1895/1895 [==============================] - 4s 2ms/step - loss: 0.5211 - binary_accuracy: 0.6684 - val_loss: 0.5201 - val_binary_accuracy: 0.6989
Epoch 5/10
1895/1895 [==============================] - 4s 2ms/step - loss: 0.5187 - binary_accuracy: 0.6667 - val_loss: 0.5271 - val_binary_accuracy: 0.6647
Epoch 6/10
1895/1895 [==============================] - 4s 2ms/step - loss: 0.5203 - binary_accuracy: 0.6630 - val_loss: 0.5206 - val_binary_accuracy: 0.6892
Epoch 7/10
1895/1895 [==============================] - 4s 2ms/step - loss: 0.5172 - binary_accuracy: 0.6691 - val_loss: 0.5281 - val_binary_accuracy: 0.5978
...
Epoch 10/10
1895/1895 [==============================] - 4s 2ms/step - loss: 0.5173 - binary_accuracy: 0.6678 - val_loss: 0.5310 - val_binary_accuracy: 0.6840
17/17 [==============================] - 0s 2ms/step - loss: 0.5197 - binary_accuracy: 0.6849
[0.5197445154190063, 0.684926450252533]
```


0.056
0.054
0.052
0.050
0.048
0
100
200
300
400
500


0.035
0.030
0.025
0.020
0.015
0.010
2200
2300
2400
2500
2600
2700



```
Epoch 1/10
17/17 [==============================] - 1s 13ms/step - loss: 0.0124 - val_loss: 2.9846e-04
Epoch 2/10
17/17 [==============================] - 0s 4ms/step - loss: 0.0096 - val_loss: 1.2341e-04
Epoch 3/10
17/17 [==============================] - 0s 5ms/step - loss: 0.0070 - val_loss: 0.0014
Epoch 4/10
17/17 [==============================] - 0s 5ms/step - loss: 0.0057 - val_loss: 0.0035
Epoch 5/10
17/17 [==============================] - 0s 5ms/step - loss: 0.0054 - val_loss: 0.0033
Epoch 6/10
17/17 [==============================] - 0s 6ms/step - loss: 0.0050 - val_loss: 0.0027
Epoch 7/10
17/17 [==============================] - 0s 6ms/step - loss: 0.0046 - val_loss: 0.0026
Epoch 8/10
17/17 [==============================] - 0s 6ms/step - loss: 0.0038 - val_loss: 0.0018
Epoch 9/10
17/17 [==============================] - 0s 6ms/step - loss: 0.0029 - val_loss: 0.0014
Epoch 10/10
17/17 [==============================] - 0s 7ms/step - loss: 0.0017 - val_loss: 5.1540e-04

<keras.callbacks.History at 0x200fe922340>
```

Conclusion

ANNs can be used for a wide variety of data types; they are best used for speech recognition, machine translation, and medical diagnosis. They work best on Tabular data, text data and speech data. Although one has to be careful when to apply ANN as mentioned before.

CNNs are most popular for image classification and working with pixels because of their outstanding ability to reduce the number of parameters without any loss of data. This is what makes them the ideal choice even in the medical industries compared to the typical ANN lacking the efficiency and accuracy of CNN.

GCN works with semi-supervised graph data. If the input is in graphical form or the data provided can be converted to a graphical form using the various connections between the classes or attributes GCN would be the ideal choice for solving such problems. Thus this paper gives a deep insight about the Machine Learning algorithms ANN, CNN and GCN and would enable one to be more efficient and have more information when choosing a Machine Learning algorithm from the three discussed above for the problem to be solved by them.

References

1. https://ieeexplore.ieee.org/document/9408211
2. https://ieeexplore.ieee.org/document/9258821
3. https://ieeexplore.ieee.org/document/542681
4. https://ieeexplore.ieee.org/document/9377860
5. https://ieeexplore.ieee.org/document/9860228
6. https://ieeexplore.ieee.org/document/9221771
7. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9455286/
8. https://sleasian.github.io/
9. https://ai.plainenglish.io/graph-convolutional-networks-gcn-baf337d5cb6b
10. https://www.researchgate.net/figure/Comparison-of-CNN-and-graph-neural-network-GNN-CNN-is-applied-to-a-graph-in-Euclidean_fig16_343821039
11. https://jonathan-hui.medium.com/applications-of-graph-neural-networks-gnn-d487fd5ed17d
12. https://ieeexplore.ieee.org/abstract/document/7012785
13. https://ieeexplore.ieee.org/abstract/document/8859190
14. https://ieeexplore.ieee.org/abstract/document/8308186
15. Introduction to Artificial Neural Network (ANN) Methods: What They Are and How to Use Them Fadi Thaer
16. An Introduction to Convolutional Neural Networks Keiron O'Shea and Ryan Nash