Table of Contents

[Project Description 3](#_Toc510274840)

[Types of Neural Networks and Details 3](#_Toc510274841)

[MLP 3](#_Toc510274842)

[Application 3](#_Toc510274843)

[Datasets 3](#_Toc510274844)

[Dataset description 3](#_Toc510274845)

[CNN 4](#_Toc510274846)

[Application 4](#_Toc510274847)

[Dataset 4](#_Toc510274848)

[Dataset Description 4](#_Toc510274849)

[RNN 4](#_Toc510274850)

[Applications 4](#_Toc510274851)

[Dataset 4](#_Toc510274852)

[Dataset Description: 4](#_Toc510274853)

[Hyperparameters selected for tuning 4](#_Toc510274854)

[RBM 5](#_Toc510274855)

[Application 5](#_Toc510274856)

[Dataset: 5](#_Toc510274857)

[Dataset Description 5](#_Toc510274858)

[Hyperparameters selected for tuning 5](#_Toc510274859)

[GAN 5](#_Toc510274860)

[Application 5](#_Toc510274861)

[Dataset 5](#_Toc510274862)

[Dataset Description 5](#_Toc510274863)

[Hyperparameters selected for tuning 5](#_Toc510274864)

[Autoencoders 5](#_Toc510274865)

[Application 5](#_Toc510274866)

[Dataset 5](#_Toc510274867)

[Hyperparameters selected for tuning 6](#_Toc510274868)

[Detail Description 6](#_Toc510274869)

[Stretch Goals 6](#_Toc510274870)

[Project Progress 7](#_Toc510274871)

[Research Study of Deep Neural Networks 7](#_Toc510274872)

[Changes Proposed during Review 16](#_Toc510274873)

[Steps Ahead 16](#_Toc510274874)

[Code attachment details 16](#_Toc510274875)

[Progress Summary 17](#_Toc510274876)

[References 18](#_Toc510274877)

Jaini Bhansali

Big Data Intelligence Analytics

# Project Progress Report

# Project Description

My primary aim is to understand the neural networks in a bottom up approach. The project would encompass understanding a neural network, then delving into Deep Neural Networks. I will be exploring each of the following neural networks:

1. Multi Layer Perceptron Neural Network (MLP)
2. Convolutional Neural Network (CNN)
3. Recurrent Neural Network (RNN)
4. Restricted Boltzmann Machine (RBM)
5. Generative Adversarial Network (GAN)
6. Autoencoders

I have selected Hello World data sets for each of these neural networks (Except MNIST) based on their applications and would be exploring these using the below datasets:

# Types of Neural Networks and Details

## MLP

Application: MLP’s are very good classifier algorithm. Hence, the Iris dataset is a popular Hello World Dataset that is used for classification.

Datasets: Iris dataset

Dataset description: The Iris Dataset is a multivariate dataset. It consists of 3 species of Iris flowers (Iris Setosa, Iris Virginica, Iris Versicolor) of 50 samples each. The dataset consists of petal and sepal width and length respectively for each specie. Based on the 4 features we would be using the MLP Classifier to classify the Iris Specie type.

<https://archive.ics.uci.edu/ml/machine-learning-databases/iris/>

**Hyperparameters selected for tuning**

|  |  |
| --- | --- |
| Hyperparameter Name | Hyperparameter Values |
| Activation Function | Tanh, ReLU,sigmoid |
| Learning Rate | 0.1,0.001,0.0001,0.00001 |
| Number of Epochs | 100,1000,2000 |
| Loss | Logistic,MSE,MAE,MAPE |
| Number of Hidden Units | 5,10,20 |

## CNN

Application: CNN are popularly used for image and video recognition. A hello world dataset Cifar10 will be used for Object Detection and Recognition.

Object Detection and Recognition

Dataset: Cifar 10

Dataset Description: The Cifar 10 dataset is a collection of images that are commonly used to train machine learning and computer vision algorithms. It is majorly used for machine learning research. The CIFAR-10 dataset contains 60,000 32x32 color images in 10 different classes. The 10 different classes represent airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships and trucks. There are 6000 images for each class. This hello world dataset is popularly used for Object Detection and Recognition.

<https://www.kaggle.com/c/cifar-10/data>

<https://www.cs.toronto.edu/~kriz/cifar.html>

**Hyperparameters selected for tuning**

|  |  |
| --- | --- |
| Hyperparameter Name | Hyperparameter Values |
| Number of Epochs | 1500,10000,15000 (Might differ based on time to process the dataset) |
| Batch Size | 32,64,256,516 |
| Activation | Sigmoid, Tanh, ReLU,softplus |
| Drop Out | 0.5,0.7,0.8 |
| Number of Filters  Neurons in the Fully Connected Layer | 32,64  2,3,5 |
| Filters in the Convolution Layers | 512,1024 |

## RNN

Applications: RNN are popularly used for Handwriting Recognition. HasyV2 is a dataset similar to the Hello World Dataset MNIST for Handwriting recognition.

Dataset: HasyV2 (similar to MNIST)

Dataset Description: This is a dataset of single symbols similar to MNIST. It contains 32px x 32px images of 168233 instances of 369 classes. In total, the dataset contains over 150,000 instances of handwritten symbols. This is used as a Hello world dataset for Handwriting recognition.

<https://www.kaggle.com/adityaecdrid/hasyv2-dataset-friend-of-mnist/data>

### Hyperparameters selected for tuning

|  |  |
| --- | --- |
| Hyperparameter Name | Hyperparameter Values |
| Number of Epochs | 500,1000,2000,4000 (Might differ based on time to process the dataset) |
| Batch Size | 4,2,20 |
| Number of Neurons | 1,2,3 |
| Regularization | L1, L2 |
| Learning Rate | 0.1,0.001,0.0001,0.00001 |
| Activation | Softsign,softmax,tanh |
| Optimizers | RMSProp, Adagrad, Momemtum |

## RBM

Application– RBM’s are used for classification. Hence, a dataset used for classification is the PIMA Indian Diabetes Dataset.

Dataset: PIMA Indian diabetes dataset

Dataset Description: This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict if a patient has diabetes, based on certain diagnostic measurements included in the dataset. All patients are females at least 21 years old of Pima Indian heritage. The features include number of pregnancies, glucose levels, blood pressure levels to name a few.

### Hyperparameters selected for tuning

|  |  |
| --- | --- |
| Hyperparameter Name | Hyperparameter Values |
| Activation | ReLU,Sigmoid |
| Learning Rate | 0.1,0.001,0.0001 |
| Number of Epochs | 20,40,80 (Might differ based on time to process the dataset) |
| Cost Function | Kullback-Leiber Divergence, Quadratic Cost, Cross Entropy |
| Loss Function | MSE,MAE,MAPE |
| Number of Layers | 1,2,3,4 |

Stretch Goal: Visible layer size will be tuned as well.

## GAN

Application: GAN’s are used popularly used to generate images. Hence, we have used the FMNIST dataset to generate images.

Dataset: FMNIST dataset

Dataset Description**:** The Fashion MNIST dataset consists of training set of 60000 examples and a test set of 10000 examples. Each example is a 28X28 gray scale image associated with a label of 10 classes. The FMNIST Dataset is similar to the MNIST dataset and will be used to generate images using GAN.

<https://www.kaggle.com/abhishekyana/fmnist-dataset-with-cnns-tensorflow/notebook>

### Hyperparameters selected for tuning

|  |  |
| --- | --- |
| Hyperparameter Name | Hyperparameter Values |
| Network Initialization | Random Normal Initialization, Xavier |
| Activation (Generator and Discriminator) | ReLU, Leaky ReLU, Sigmoid |
| Number of Epochs | 15,20, 30 |
| Gradient Estimation | Stochastic Gradient Descent, Momentum |

Batch Normalization will also be experimented with as a part of hyper parameter tuning.

## Autoencoders

Application**:** Autoencoders are used to generate images. Hence, selected the Optical Recognition of Handwritten Digits Dataset used for Optical recognition

Dataset: Optical Recognition of Handwritten Digits Dataset

**Dataset Description**: The dataset was created by preprocessing programs made available by NIST to extract normalized bitmaps of handwritten digits from a preprinted form. From a total of 43 people, 30 contributed to the training set and different 13 to the test set. 32x32 bitmaps are divided into nonoverlapping blocks of 4x4 and the number of on pixels are counted in each block. This generates an input matrix of 8x8 where each element is an integer in the range 0 to 16. This reduces dimensionality and gives invariance to small distortions. An autoencoder will be used recognize the handwritten dataset.

<https://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits>

### Hyperparameters selected for tuning

|  |  |
| --- | --- |
| Hyperparameter Name | Hyperparameter Values |
| Activation of Output Layer | Sigmoid, ReLU |
| Loss | Cross Entropy, MSE |
| Number of Layers | 3,4,5 |
| Optimizer | Adam, RMSProp, Adadelta |

## Detail Description

1. Exploring each of these neural networks using the Tensorflow Library
2. Identifying the parameters and hyperparameters for the Neural Networks
3. Tuning the parameters and hyperparameters to improve accuracy

# Stretch Goals

1. Use of Keras
2. Incorporate use of Big Data

# Project Progress

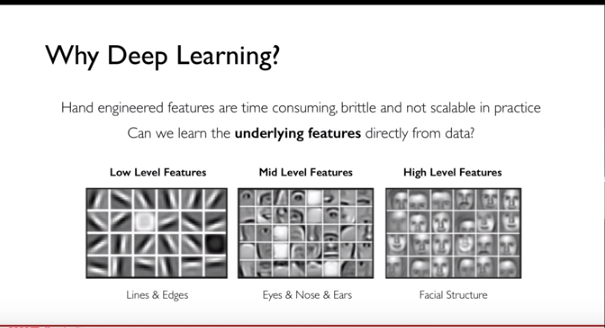
1. **Finalized datasets based on the applications of the various Neural Network**
2. **Research Study of Deep Neural Networks**
3. **Researched on the various Kinds of Neural Networks**
4. **Used the IRIS dataset on Databricks and performed EDA**
5. **Used Tensorflow to implement a MLP Neural Network. I have a achieved an accuracy of 97%**
6. **Experimented with Parameters and Hyperameters**

## Research Study of Deep Neural Networks

The first step was to identify the basics of a neural network.

The breakthrough for deep learning came with the image net challenge, where AI researches were tasked with building image recognition to detect objects. As a current achievement Deep Learning Accuracy has surpassed Human Accuracy with the Imagenet challenge.

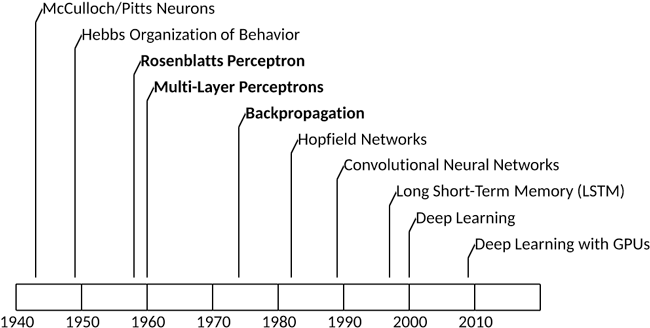
The key differentiator between Traditional Machine Learning and Deep Learning is that Deep Learning can be used learn brittle features. Deep Learning uses the features provided and can apply to practical scenarios as well.



Reference: https://www.youtube.com/watch?v=JN6H4rQvwgY&feature=youtu.be

**The Resurgence of Neural Networks and Deep Learning**

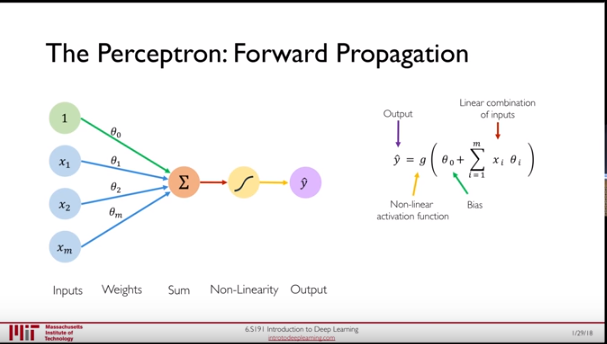
As a part of understanding Neural Networks, I learnt that neural networks existed decades ago but resurfaced due to the popularity and ease of use of Big Data, Cheaper and more computationally user friendly. Hardware and great software that aided the popularity of Deep Learning.



**The Perceptron**

The Perceptron is the fundamental building block of a Neural Network. A perceptron is a single Neuron in a Neural Network.

**Feed Forward Propagation**



Reference: <https://www.youtube.com/watch?v=JN6H4rQvwgY&feature=youtu.be>

In simple words, a feed forward neural network consists of a number of inputs (say ‘m’) which has weights associated with it (Sa y Theta 0 to Theta m). The Inputs are multiplied with the weights summed up and passed through a sigmoid function to product output y.

**Bias**

A bias is used to ensure that even if there are no input features to yet provide a positive output

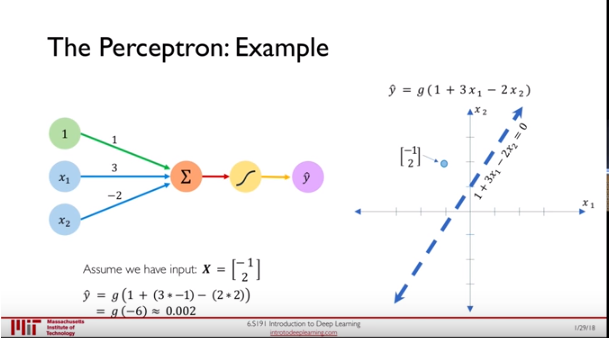
Hence, the inputs and Weights are treated as Vectors and used as a dot product mathematically.

**Activation Function**

There are various kinds of activation functions like Sigmoid, Tanh (Hyperbolic Tangent), ReLU (Rectified Linear Unit) to name a few. Each of the activation functions have various applications in different scenarios. For example, Sigmoid functions are popularly used as they transform the input features as an output that lies between 0 and 1 , making it very popular for to represent probabilities.

Activation functions are used to introduce nonlinearity into the network for better computation of the output.

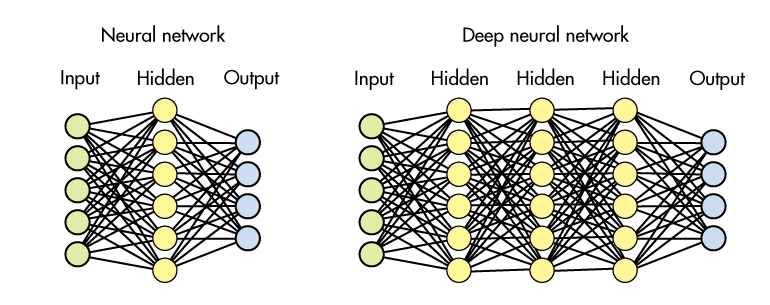
A Mathematical example for a perceptron with the application of an activation function.



In Deep Learning each time for we utilize inputs, do a dot product of the inputs and weight, apply a bias and non linearity. This is done for each node, each neuron to form a Neural Network.

**Shallow Neural Network**

A Shallow Neural Network is one with a single layer in the Neural Network

**A Deep Neural Network** A Neural Network with more than Hidden Layer . 

**Loss in a Neural netwrok**

In Simple words the loss function is used to tell us how wrong our predicted value is from the ground truth. The empirical Loss is the Loss calculated over the entire dataset. The loss sometimes referred using different names such as Objective Function, Cost Function or Empirical Risk.

**Cross Entropy Loss**

This is mainly used with models that output a probability between 0 and 1 , i.e. mainly classification problems. Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label. A perfect model will have a log loss of 0. As the predicted probability approached 1 the log loss decreases.

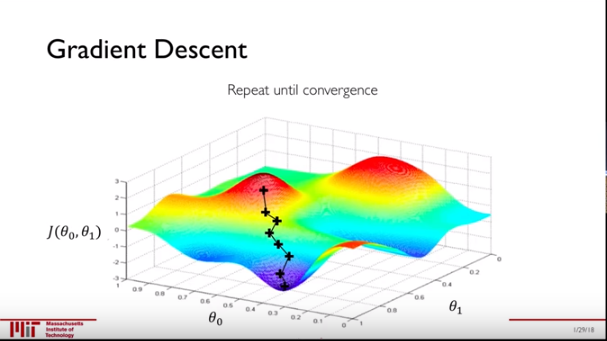
**Mean Squared Error Loss**

This is mainly used in regression models that output continuous variables. ( In Simple words as the values are further from the predicted value the Mean Squared Error Loss Increases)

Hence, the Loss depicts the neural network and how well our Neural Network is doing. The aim would be to reduce the loss and make it minimum. Additionally, the loss is a function of Network Weights.

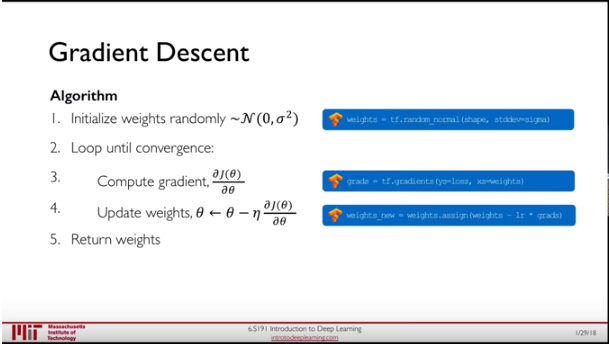
**Loss Optimization**

Since, the loss is a function of Network Weights, we can find the weights that contribute to minimum loss.

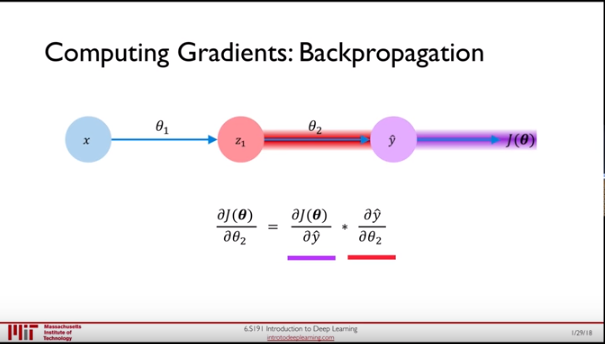


Theta 0 and Theta 1 are weights and the points are the various weights selected to test the least loss. Since, we move downwards it’s a descent.

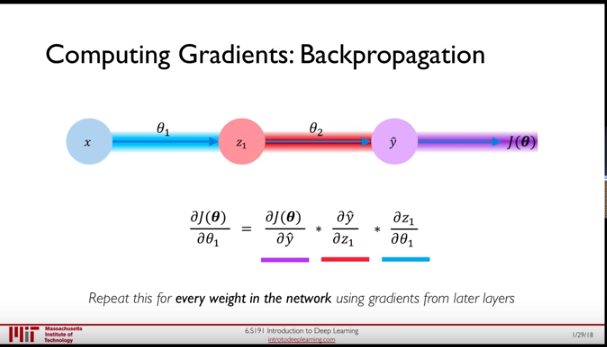
**Pseudocode**



**This is calculated using Back Propagation.** Additionally, computing this gradient at every point can be computationally expensive, contributing to various other algorithms such as use of Stochastic Gradient Descent.

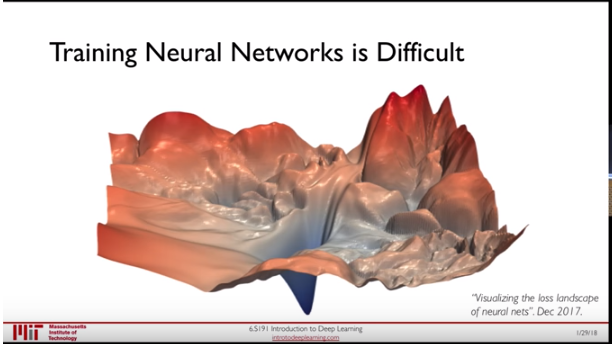


Then for Theta 1



**Actual Loss Landscape**

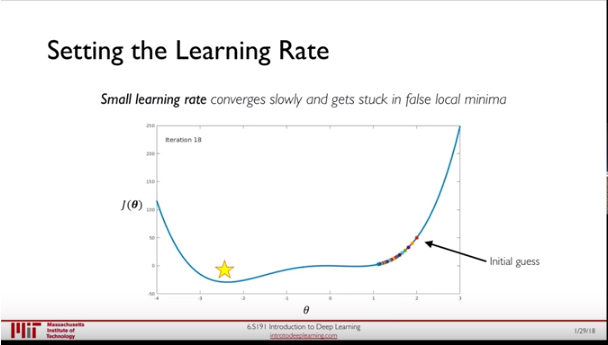
Represents various Local Minimas



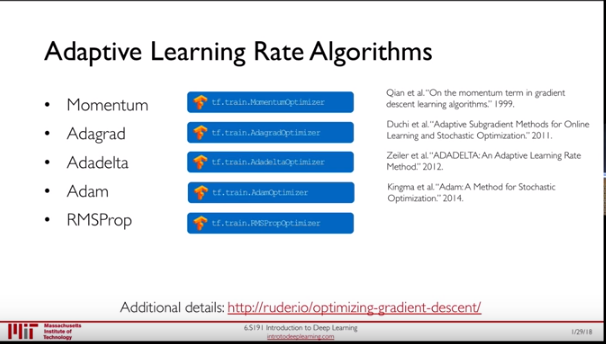
**Learning Rate**

In the context of the above loss landscape, the learning rate measures how large the step is towards the descent.

Setting the learning rate can be a challenge as it must not be too low that it gets lost in the local minima, also must not be too large such that the model would diverge and blow up.



This can be set using a adaptive learning rate. There are various algorithms that contribute to this.



More accurate the gradient estimation, smoother convergences and larger learning rates. Giving rise to parallelized computation and use of GPU’s.

**Regularization**

Regularization constrains our optimization problem to discourage complex models and used to generalize our model on unseen data.

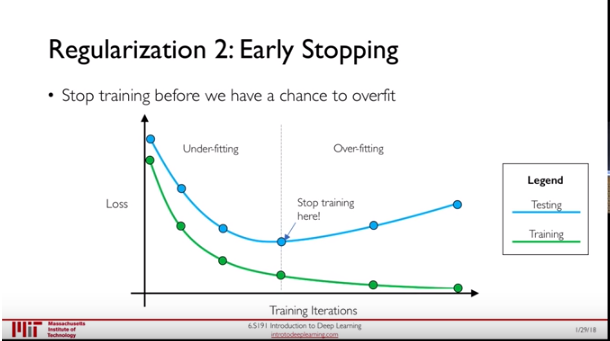
**Drop Outs**

Process used to randomly drop neurons such that activation becomes 0. This ensures the the network does not randomly rely on a few neuron or provide larger weights to neurons.

**Early Stopping**

Stop training before we have a chance to overfit.

In [machine learning](https://en.wikipedia.org/wiki/Machine_learning), early stopping is a form of [regularization](https://en.wikipedia.org/wiki/Regularization_(mathematics)) used to avoid [overfitting](https://en.wikipedia.org/wiki/Overfitting) when training a learner with an iterative method. Such methods update the learner so as to make it better fit the training data with each iteration



1. **Researched on the various Kinds of Neural Networks**

**CNN**

In machine learning, a convolutional neural network (CNN, or ConvNet) is a class of deep, [feed-forward](https://en.wikipedia.org/wiki/Feedforward_neural_network) [artificial neural networks](https://en.wikipedia.org/wiki/Artificial_neural_network) that has successfully been applied to analyzing visual imagery.

CNNs use relatively little pre-processing compared to other [image classification algorithms](https://en.wikipedia.org/wiki/Image_classification). This means that the network learns the [filters](https://en.wikipedia.org/wiki/Filter_(signal_processing)) that in traditional algorithms were [hand-engineered](https://en.wikipedia.org/wiki/Feature_engineering). This independence from prior knowledge and human effort in feature design is a major advantage.

They have applications in [image and video recognition](https://en.wikipedia.org/wiki/Computer_vision), [recommender systems](https://en.wikipedia.org/wiki/Recommender_system)and [natural language processing](https://en.wikipedia.org/wiki/Natural_language_processing).

**RNN**

A **recurrent neural network** (**RNN**) is a class of [artificial neural network](https://en.wikipedia.org/wiki/Artificial_neural_network) where connections between units form a directed graph along a sequence. Unlike [feedforward neural networks](https://en.wikipedia.org/wiki/Feedforward_neural_networks), RNNs can use their internal state (memory) to process sequences of inputs. This makes them applicable to tasks such as unsegmented, connected [handwriting recognition](https://en.wikipedia.org/wiki/Handwriting_recognition)[[1]](https://en.wikipedia.org/wiki/Recurrent_neural_network#cite_note-1) or [speech recognition](https://en.wikipedia.org/wiki/Speech_recognition).

A fully Recurrent Neural network is a network of [neuron-like](https://en.wikipedia.org/wiki/Artificial_neuron) nodes, each with a [directed (one-way) connection](https://en.wikipedia.org/wiki/Directed_graph) to every other node. Each node (neuron) has a time-varying real-valued activation. Each connection (synapse) has a modifiable real-valued [weight](https://en.wikipedia.org/wiki/Weighting). Nodes are either input nodes (receiving data from outside the network), output nodes (yielding results), or hidden nodes (that modify the data on route from input to output).

**RBM**

A restricted Boltzmann machine (RBM) is a [generative](https://en.wikipedia.org/wiki/Generative_model) [stochastic](https://en.wikipedia.org/wiki/Stochastic_neural_network) [artificial neural network](https://en.wikipedia.org/wiki/Artificial_neural_network) that can learn a [probability distribution](https://en.wikipedia.org/wiki/Probability_distribution) over its set of inputs. RBMs have found applications in [dimensionality reduction](https://en.wikipedia.org/wiki/Dimensionality_reduction), [classification](https://en.wikipedia.org/wiki/Statistical_classification), [collaborative filtering](https://en.wikipedia.org/wiki/Collaborative_filtering),[[4]](https://en.wikipedia.org/wiki/Restricted_Boltzmann_machine#cite_note-softCF-4) [feature learning](https://en.wikipedia.org/wiki/Feature_learning)[[5]](https://en.wikipedia.org/wiki/Restricted_Boltzmann_machine#cite_note-coates2011-5) and [topic modelling](https://en.wikipedia.org/wiki/Topic_model). They can be trained in either [supervised](https://en.wikipedia.org/wiki/Supervised_learning) or [unsupervised](https://en.wikipedia.org/wiki/Unsupervised_learning) ways, depending on the task.

**GAN**

Generative adversarial networks (GANs) are a class of [artificial intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence) algorithms used in [unsupervised machine learning](https://en.wikipedia.org/wiki/Unsupervised_machine_learning), implemented by a system of two [neural networks](https://en.wikipedia.org/wiki/Neural_network). This technique can generate photographs that look at least superficially authentic to human observers, having many realistic characteristics (though in tests people can tell real from generated in many cases).

**Autoencoder**

An autoencoder is an [artificial neural network](https://en.wikipedia.org/wiki/Artificial_neural_network) used for [unsupervised learning](https://en.wikipedia.org/wiki/Unsupervised_learning) of [efficient codings](https://en.wikipedia.org/wiki/Feature_learning). The aim of an autoencoder is to learn a representation (encoding) for a set of data, typically for the purpose of [dimensionality reduction](https://en.wikipedia.org/wiki/Dimensionality_reduction). Recently, the autoencoder concept has become more widely used for learning [generative models](https://en.wikipedia.org/wiki/Generative_model) of data.

1. Used the IRIS dataset on Databricks and performed EDA
2. Used Tensorflow to implement a MLP Neural Network. I have a achieved an accuracy of 97%
3. Experimented with Parameters and Hyperparameters

## Changes Proposed during Review

1. Display hyperparameters to be used for tuning with their respective values.
2. Describe the datasets and provide details.

## Steps Ahead

1. Tuning parameters and hyperparameters of the MLP Neural Network (Tentative date by 30th March 2018)
2. Understand structures of CNN, RBM, RNN, GAN and Autoencoders (Tentative Date by April 3rd 2018)
3. Implement in Tensorflow perform tuning of parameters (April 15th 2018)
4. If possible find a better dataset to demonstrate CNN

## Code attachment details

The code is present in a ipynb that is attached in the blackboard submission. In order to compare accuracies, computation time and as a part of my assignment I have compared the computation time using Tensorflow with Spark and Tensorflow without Spark.

## Progress Summary

As a summarization, I firstly understood the basics of Neural Network and the basics of the selected different types of neural networks. After implementing the MLP Neural Network, I have analyzed that I have understood the network structures better while implementing in Tensorflow a little time consuming. Tuning of parameters and hyperparameters can be time consuming as a beginner. Though the datasets selected are Hello World Datasets, more research is required to find similar notebooks as a guide to implement Tensorflow. The Tensorflow library is extensive and extremely interesting to explore and learn.

## References

<https://www.youtube.com/watch?v=a5BUunInTQU&t=1227s>

<https://github.com/nikhilroxtomar/Iris-Data-Set-Classification-using-TensorFlow-MLP/blob/master/iris.py>

<https://github.com/Vikramank/Deep-Learning-/blob/master/Iris%20data%20classification.ipynb>

<https://www.google.com/search?hl=en&tbm=isch&sa=1&ei=7zakWv3DA-P25gLx5YPgAw&q=time+frame+of+neural+network+popularity&oq=time+frame+of+neural+network+popularity&gs_l=psy-ab.3...2489.15273.0.15614.42.39.1.2.2.0.109.3264.37j1.39.0....0...1c.1.64.psy-ab..0.16.1315.0..0j0i67k1j0i8i30k1j0i24k1.84.3561_Iisg6s#imgrc=WiGk8bn3rnboUM>:

<https://en.wikipedia.org/wiki/Convolutional_neural_network>

<https://en.wikipedia.org/wiki/Recurrent_neural_network#Fully_recurrent>

<https://en.wikipedia.org/wiki/Restricted_Boltzmann_machine>

<https://en.wikipedia.org/wiki/Generative_adversarial_network>

<https://deeplearning4j.org/restrictedboltzmannmachine>

<https://github.com/shekkizh/neuralnetworks.thought-experiments/tree/master/Generative%20Models/GAN>

<https://gertjanvandenburg.com/blog/autoencoder/>

<https://gertjanvandenburg.com/blog/autoencoder/>

<https://en.wikipedia.org/wiki/Iris_flower_data_set>

<https://en.wikipedia.org/wiki/CIFAR-10>

<https://zenodo.org/record/259444#.Wr_N1IjwbIU>

<https://www.kaggle.com/uciml/pima-indians-diabetes-database>

<https://archive.ics.uci.edu/ml/machine-learning-databases/optdigits/optdigits.names>

<https://www.kaggle.com/abhishekyana/fmnist-dataset-with-cnns-tensorflow/data>

<https://www.kaggle.com/uciml/pima-indians-diabetes-database/data>