

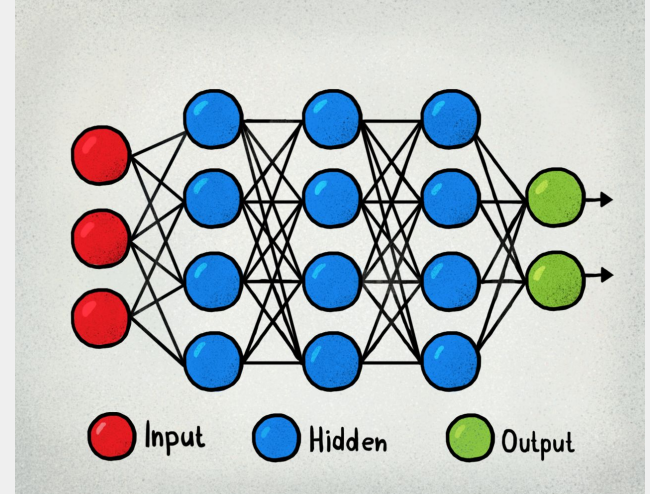
# DS 303 Project

## Topic:

Handwritten digit recognition using multiple Machine Learning algorithms

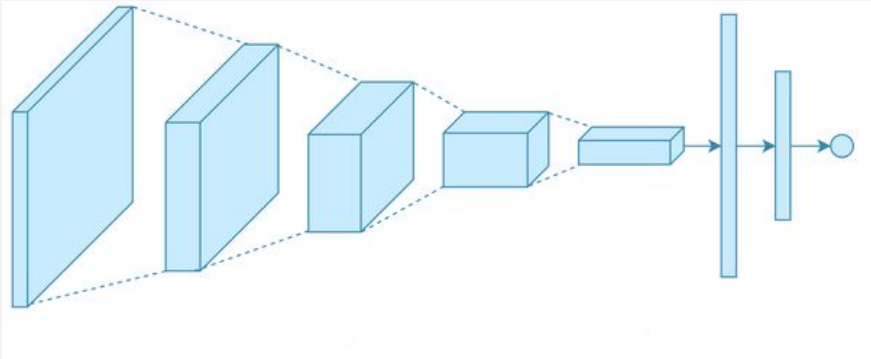
## Team Members:

1. Sanket Ghyar - 190110023 - MEMS
2. Arif Ahmad - 190110010 - EE
3. Naresh Balamurugan - 190110051 - MEMS
4. Yash Shah - 190020136 - CheE



# Problem Statement:

Analyzing Machine Learning Algorithms of **Support Vector Machines** (both **linear Kernel** and **RBF Kernel**), **Neural Networks** and **Convolutional Neural Networks** for the task of **Digit Classification**. Make a **comparison of the accuracy** of these models and the **time required to train** the models.



# Background:

Lets, briefly have a discussion on the Machine Learning methods used in this project one-by-one.

## 1. Support Vector Machines:

The objective of the support vector machine algorithm is to find a hyperplane in an  $N$ -dimensional space ( $N$  — the number of features) that distinctly classifies the data points.

It uses the concept of **Hyperplanes**, **Support Vectors** and **Support Vector Margin Maximization** as the basic concepts to train the Machine Learning model as well as finally in classification during testing.

SVM also uses Kernel functions to account for non-linearity in the data set and for a better classification while still using a simple hypothesis of separation by linear functions.

As SVM uses Kernel functions, the SVM model can generate very different results based on the Kernel Function used. We are using two very famous Kernel functions Linear Kernel and Radial

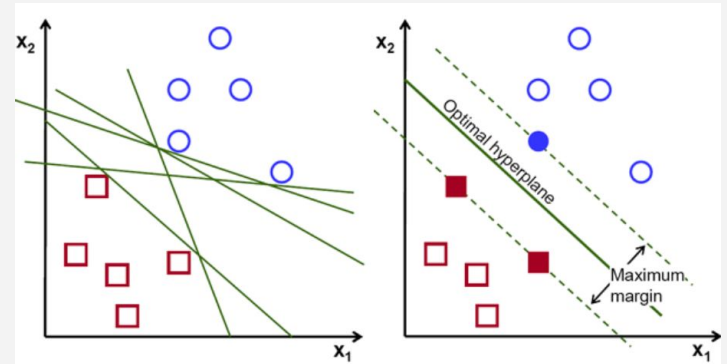
Basis Kernel Functions in this project and analyzing the accuracy and training time.

## 2. Neural Networks:

A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a

process that mimics the way the human brain operates. A basic Neural Network consists of one or more input nodes, one or more output nodes and one or more hidden layers with each layer having a certain number of nodes.

Neural networks use a technique called 'back propagation' to fit the Model to the Training Data set.



In the hidden layers, we can have many different Activation Functions, but Activation functions commonly used are the ReLU, the Sigmoid and the SoftPlus Function. In this Project we are using the ReLU Activation Function.

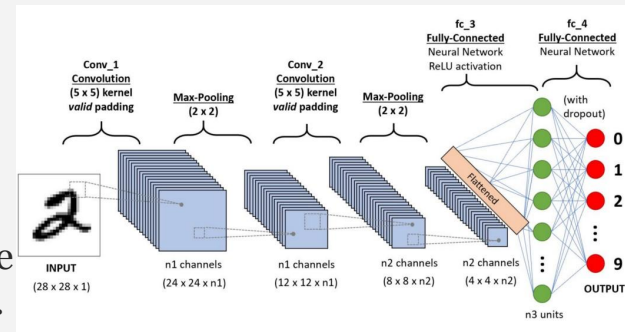
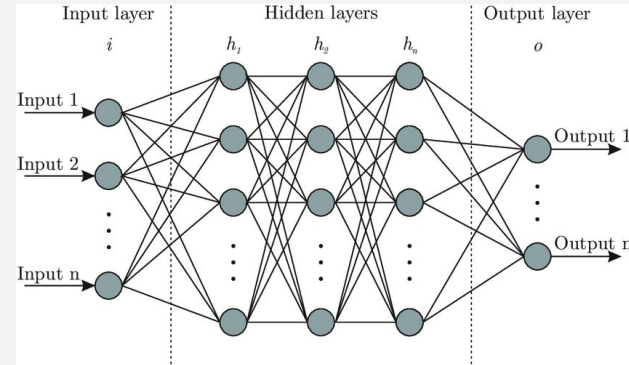
### 3. Convolutional Neural Networks:

#### Convolutional Neural Network (ConvNet/CNN)

is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and bias) to various aspects/objects in the image and be able to

differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

CNN uses Kernels to extract high-level feature such as edges from the input image so that working with images becomes more manageable.



# Our Methods and Implementation:

The aim of this Project as described in the Problem Statement is to use four different models namely Linear Kernel SVM, RBF Kernel SVM, Neural Networks and Convolutional Neural Networks and to compare them and find the most efficient algorithm in terms of accuracy and training time.

## a. Data Preprocessing and Loading

We have used the **MNIST database** for building our model in each of the above methods. Then we have split this dataset into 60k training and 10k testing samples.

The images consist of 28 x 28 pixel values. As the images are black and white their pixel values are on a gray scale ranging between 0 and 255, we have normalized the pixel values to lie between 0 and 1. The labels {0,1,2,...,8,9} have also been one hot encoded in the case of Neural Networks for ease of modelling and interpretation of outcomes.

## **b. Building the Machine Learning Models for each case.**


### **1. Neural Network**

The Neural Network is made using Sequential Model in the Keras Library. It has an input layer of size  $28 \times 28 = 784$  followed by two hidden layers and a final densely connected layer consisting of 10 nodes for the 10 class labels 0,1,2,...,8,9.

Both the Hidden Layers consist of 512 nodes. And use the ReLU activation function. This is a dense Neural Network. We also fixed few parameter values and didn't update them in model iterations using the Dropout function in Keras; to ensure that the model doesn't overfit the training dataset.

### **2. Convolutional Neural Network**

The Convolutional Neural Network is made using Sequential Model in the Keras Library. It has an input layer of size  $28 \times 28 = 784$  followed by three convolutional layers, followed by three fully connected layers and finally connected to dense layer Consisting of 10 nodes for the 10 class labels. All 3 convolutional layers uses  $3 \times 3$  Filters with a stride of 1. We have used max pooling which takes the maximum value of each local clusters of neurons with tiling size of  $2 \times 2$ .



The flattened matrix goes through a series of three fully connected Layers finally to the last fully connected layer to classify the digits. The fully connected layers are dense as 64 nodes in the first layer reducing the number of nodes by factor of 2 and eventually reducing the Number of nodes to 10 in the final output layer. We have used the ReLU as the activation function for non-linearity in the convolutional layers and also used the dropout function to avoid overfitting.

### **3. Support Vector Machine**

After Data Preprocessing SVM model can directly be built using the SVC model in the scikit-learn library. Both the Linear model as well as RBF kernel models were built this way.



## c. Training the model

### 1. Neural Networks

Using categorical cross entropy as the loss function and Stochastic Gradient Descent as the optimizer the Neural network model with two 512-node hidden layers is trained over the 60k training samples. A batch size of 128 is used for updating the parameters and we go over the complete 60k training samples 20 times to yield a better accuracy.

### 2. Convolutional Neural Networks

Here too we used the categorical cross entropy as the loss function and Adam as the optimizer. In this case we went over the complete 60k training dataset over and over 5 times. (i.e. epochs = 5).

### 3. SVM

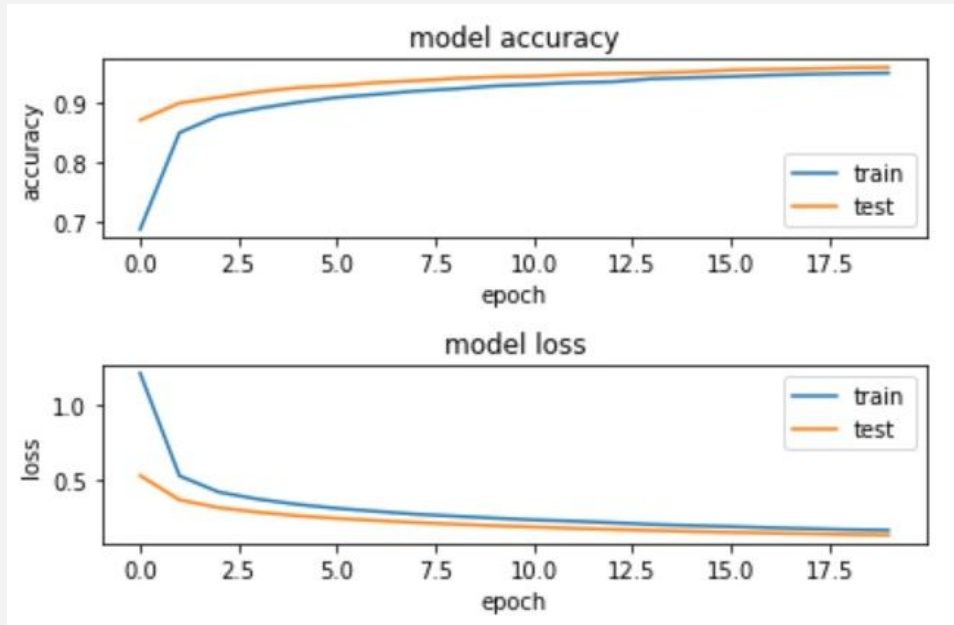
Here we used the squared hinge loss as the loss function. We did not go over the dataset repeatedly as we did in the last two cases, as the model training by SVM was very slow.

**Details of the implementation can be found in the code submitted alongside this presentation.**

# Results and Analysis of Results:

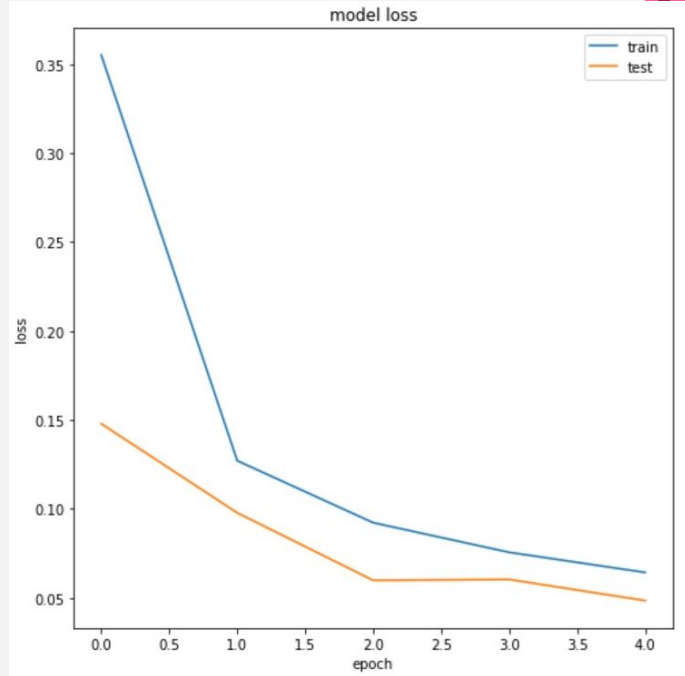
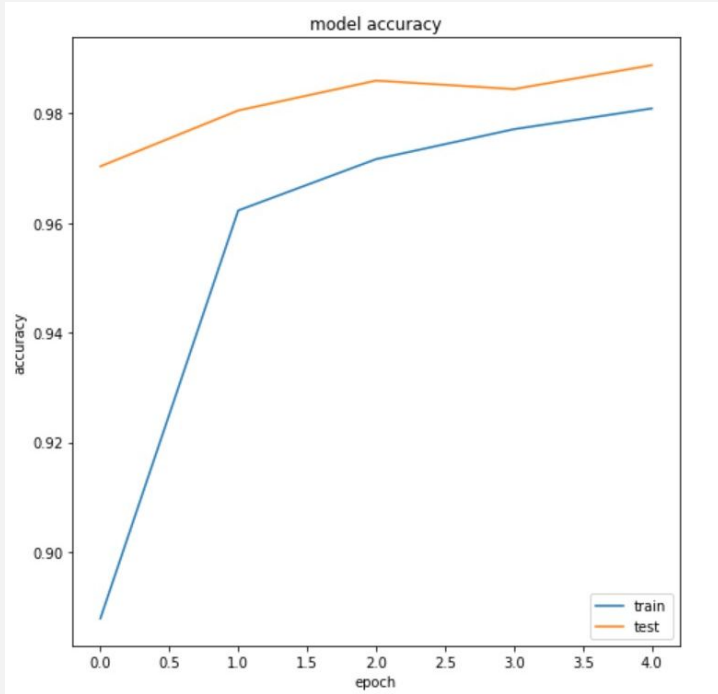
Below are the graphs of model accuracy and model loss for each of the method discussed:

## 1. Neural Network



As can be seen in the adjacent graphs, as we increase the number of iterations over the dataset (epochs) the model accuracy increases and the loss decreases. In fact the model accuracy increases the most in the first two iterations; indicating that our model is learning very rapidly.

## 2. Convolutional Neural Network



Here too the model accuracy increases very rapidly in first few iterations and loss also decrease rapidly.

### 3. Support Vector Machines:

We did not plot a graph in this model as there were no iterations over the complete batch; but made a confusion matrix to find out if the true positives are high or not.

Confusion Matrix for Linear Kernel SVM

[	[	953	0	6	2	1	8	6	2	1	1]
[	0	1118	7	2	0	1	2	1	4	0]	
[	9	12	956	11	9	4	5	5	18	3]	
[	7	1	15	940	0	17	1	6	19	4]	
[	3	2	18	1	928	0	3	6	3	18]	
[	7	6	7	40	5	791	12	1	20	3]	
[	14	3	17	1	9	19	892	0	3	0]	
[	2	8	23	14	11	2	0	945	2	21]	
[	11	7	10	29	8	23	8	6	860	12]	
[	9	7	6	11	38	5	0	23	12	898]]]	

Confusion Matrix for RBF Kernel SVM

[	[	967	0	2	1	0	3	3	2	2	0]
[	0	1125	5	0	0	1	2	0	2	0]	
[	5	1	996	2	2	0	1	15	9	1]	
[	0	0	3	980	1	7	0	12	7	0]	
[	0	0	13	0	945	2	3	7	2	10]	
[	2	0	2	11	1	857	6	5	6	2]	
[	6	2	0	0	4	8	927	6	5	0]	
[	1	6	13	3	3	0	0	989	0	13]	
[	3	0	6	5	6	10	3	12	926	3]	
[	4	5	6	11	13	2	0	21	3	944]]]	

# Accuracy, Time Analysis and Results:

Accuracy for the four methods discussed when validated on the test dataset is:

1. Accuracy of Neural Network Model: **95.8%**
2. Accuracy of Convolutional Neural Network Model: **98.49%**
3. Accuracy of Linear Kernel SVM: **92.81%**
4. Accuracy of RBF Kernel SVM: **96.56%**

Time taken to train the model (one iteration over dataset only):

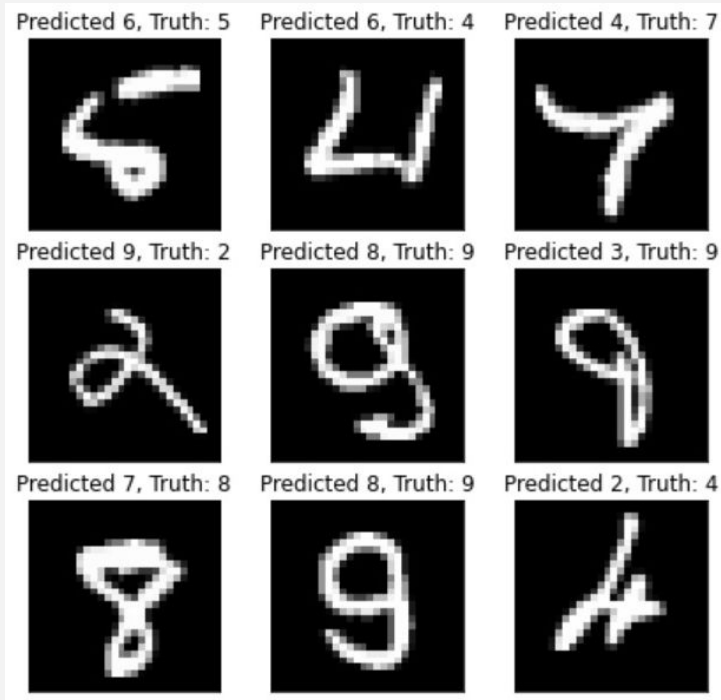
1. Neural Network: **4.6 seconds**
2. Convolutional Neural Networks: **100 seconds**
3. Linear Kernel SVM: **11 minutes**
4. RBF Kernel SVM: **16 minutes**

## Result:

Thus we can see that CNN has the highest accuracy while Neural Networks has the least training time. Thus if accuracy of model is a priority then one should use CNN for image classification while if time is a constraint then one should use Neural Network for image classification.

# Comment on Correct and Incorrect Classifications:

## Misclassifications:

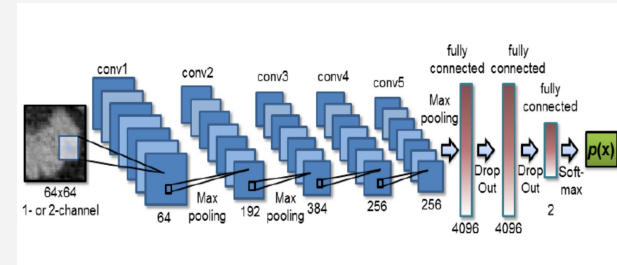


Of the 10k Test samples; Neural Network misclassified 420 samples, CNN misclassified 151 samples, Linear Kernel SVM misclassified 719 samples and RBF Kernel SVM misclassified 344 samples.

Some of these misclassifications can be seen to the left. These mostly correspond to ill-written digits. These digits are not written very clearly and can even be misclassified by a human; let alone our ML models.

# Contributions of each Team member:

1. Arif Ahmad (190110010) :
  - a. Making Slides
  - b. Code for Neural Network Model
  - c. Interpreting Results and reading relevant articles.
2. Sanket Ghyar (190110023):
  - a. Making Slides
  - b. Code for Convolutional Neural Network
  - c. Interpreting Results and reading relevant articles.
3. Yash Shah (190020136):
  - a. Making Slides
  - b. Code for Linear Kernel SVM and Neural Networks (partly)
  - c. Interpreting Results and reading relevant articles
4. Naresh Balamurugan (190110051):
  - a. Making Slides
  - b. Code for RBF Kernel SVM and CNN (partly)
  - c. Interpreting Results and reading relevant articles.





# Thank you.

End of Presentation