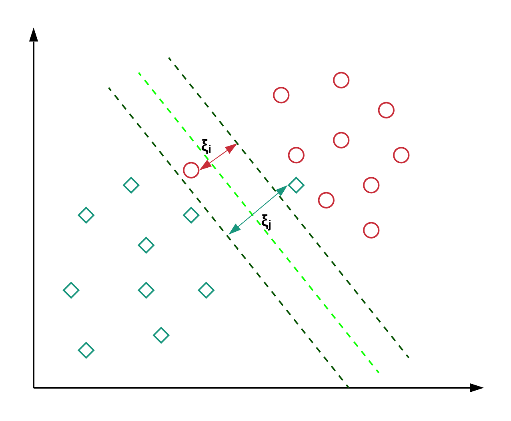
# Project 2 (a)

**SVM Classifier Implementation**

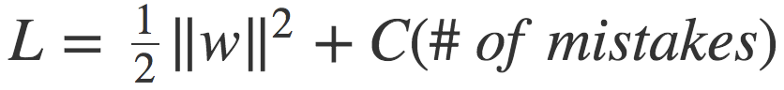
We are given a MNIST data of handwritten numbers ( 0-9). Data has 60000 images and SVM Model needs to be trained to identify these characters.

***SVM Working Principal***



The principal of SVM algorithm is based on maximizing the separation between two ( or more classes) by constructing a imaginary hyperplane.

In this new setting, we would aim to minimize the following objective:



A Kernel score can be used to converge this. Among the several kernels available we used Linear kernel as our data is seperable linearly. However for other data sets a different kernel can be used

***Implementation***

Implemented the Support Vector Machine ( SVM ) implementation in python. The output of the implementation is the test\_label. This array has the labels that are generated by the SVM code.

Tested with varying number of PCA components considered. Increasing the number of PCA components can improve the model accuracy. Blue is the model with 120 PCA components.

***Requirements***

Python 3.6

MNIST package [ pip install python-mnist ] . This package has the capability to read in the training and test data set if these are kept in a directory. (Please see code for details)

***Results***

Accuracy is calculated by All the 10000 images of testing array have been taken to generate the plot below. Model shows over 87% accuracy. The figure on the lower right shows the Pricipal Components that explain the variance of the original data set

A screenshot of a cell phone

Description automatically generatedA screenshot of a social media post

Description automatically generated

Cumulative Variance explained by PCA components

Fig Above: Model Accuracy[predicted – actual labels] increases when we take more PCA components. Accuracy also improves by adding more training data

# Project 2 (b)

**Deep Learning Implementation**

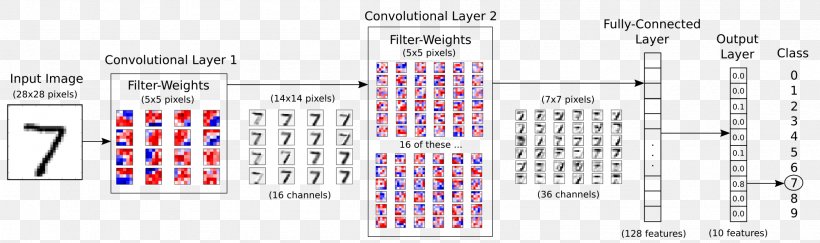
A deep learning algorithm has to be implemented to classify handwritten numbers (0-9).

**Deep Learning Convolutional Network**

The flow of the deep learning convolution network is depicted below. The goal is to reduce the images into a form which is easier to process without losing features critical for classification. In the example below every sub set of 5x5 pixels is converted into a single number using a kernel. This kernel is then applied successively to adjacent set of 5x5 till the whole image is covered. This process dramatically reduces image dimensions and helps in identifying high level features such as edges.

*Maxpooling*

In addition to the kernel, we also use a pooling ( max pooling in the code) that reduces the spatial size of the convolved feature.



Epochs:

Model learns

***Requirements***

Python 3.6

Tensorflow 2.0

MNIST Package

**Implementation**

Implemented a Deep Learning Algorithm that uses CONV2D in Tensorflow to identify.

**Model outputs**

10000/10000 [==============================] - 6s 578us/sample - loss: 0.4502 - accuracy: 0.8662

Epoch 2/10

10000/10000 [==============================] - 5s 531us/sample - loss: 0.2001 - accuracy: 0.9390

Epoch 3/10

10000/10000 [==============================] - 5s 521us/sample - loss: 0.1346 - accuracy: 0.9599

Epoch 4/10

10000/10000 [==============================] - 5s 541us/sample - loss: 0.1001 - accuracy: 0.9678

Epoch 5/10

10000/10000 [==============================] - 5s 541us/sample - loss: 0.0725 - accuracy: 0.9786

Epoch 6/10

10000/10000 [==============================] - 6s 591us/sample - loss: 0.0637 - accuracy: 0.9800

Epoch 7/10

10000/10000 [==============================] - 7s 679us/sample - loss: 0.0406 - accuracy: 0.9870

Epoch 8/10

10000/10000 [==============================] - 5s 526us/sample - loss: 0.0341 - accuracy: 0.9893

Epoch 9/10

10000/10000 [==============================] - 5s 550us/sample - loss: 0.0314 - accuracy: 0.9910

Epoch 10/10

10000/10000 [==============================] - 6s 603us/sample - loss: 0.0248 - accuracy: 0.9925

10000/10000 [==============================] - 2s 172us/sample - loss: 0.1286 - accuracy: 0.9657

Epoch = 10

A test pattern image was given and the model correctly identified it. The accuracy of model increases with each epoch of the Convolution. Each Epoch pass leads to a higher accuracy.

A screenshot of a cell phone

Description automatically generated

The CNN approach shows a faster rise in accuracy than the SVM for the MNIST data set

Fig above: Shows comparison of predicted and a training label

Fig above: Shows accuracy increases with epochs