Project Presentation

ENEL 610 - Biometric Technologies and Systems

Yadvendar Singh
Jemish Goti

System Goals/Objectives

Objectives for the Project are as below

- An implementation of Convolution Neural Networks (CNNs) to achieve the classification of a very famous MNIST database, which consists of 60,000 training samples and 10,000 test samples.
- Learning Usage of Deep NN, CNN, Pooling, Keras, Tensor Flow, and OpenCV.
- Training CNN for classifying handwritten digit images in the MNIST database with 10 different Labels ranging from 0-9.
- Use of plots of loss and accuracy for both training and validation process for determining the efficiency of CNN.
- Achieving high accuracy on validation samples and achieving least possible loss of training samples.

Experimental Setup

System Specifications

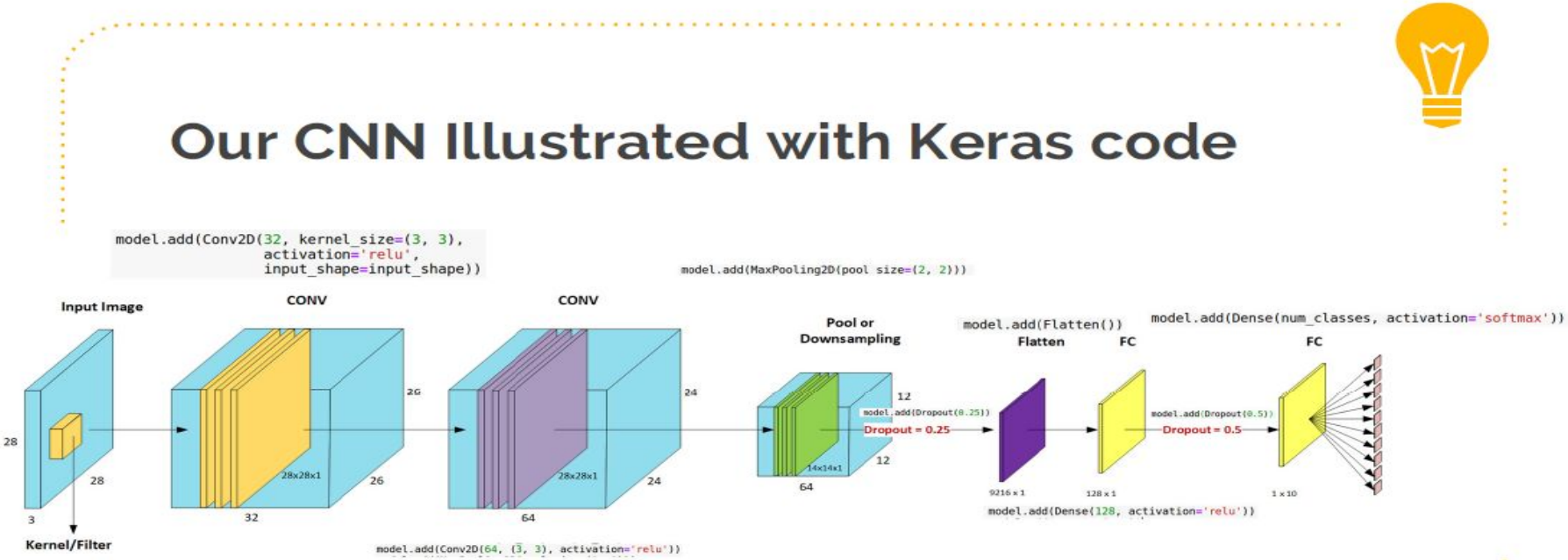
- Processor : Intel(R) Core i7-9750H
- RAM : 16.0 GB
- System type : 64-bit
- GPU : Nvidia 1660 Ti

Setting up the System

- Download Anaconda with Python 3.6+
- Create Environment
- Install Jupyter Notebook
- Install Dependencies
- Downloading the MNIST dataset

➢ Detailed Setup steps are included in the project report.

Pipeline [preprocessing, processing, classification, result]



Software Dependencies

Here, we are providing a basic information about the software dependencies involved in our project.

- Keras
- TensorFlow
- OpenCV
- MatplotLib

Keras

Keras is a high-level neural networks API, written in Python and capable of running on top of Tensorflow. It was developed with a focus on enabling fast experimentation. Keras supports both convolutional networks and recurrent networks, as well as combinations of the two. Also, it runs seamlessly on CPU and GPU.

TensorFlow

TensorFlow is a Python-friendly open source library for numerical computation that makes machine learning faster and easier. TensorFlow uses a dataflow graph to represent the computation in terms of the dependencies between individual operations. This leads to a low-level programming model in which firstly the dataflow graph is defined followed by creation of a TensorFlow session to run parts of the graph across a set of local and remote devices.

OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

Matplotlib

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUItoolkits like Tkinter, wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged. SciPy makes use of Matplotlib.

Hardware Dependencies

Anaconda

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows users to launch applications and manage conda packages, environments and channels without using command-line commands. It is available for Windows, macOS and Linux.

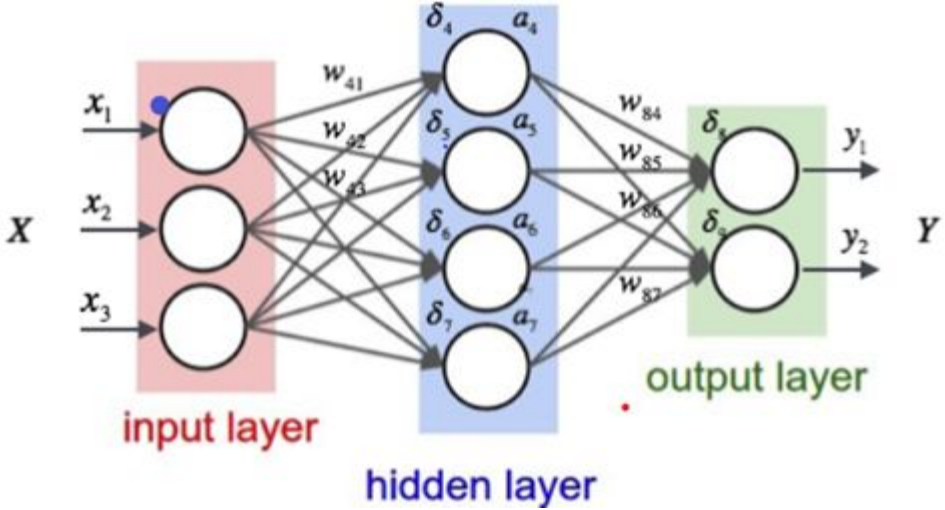
Dataset used - MNIST

- The MNIST database of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples.
- It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.
- The MNIST training set is composed of 30,000 patterns from SD-3 and 30,000 patterns from SD-1. The test set was composed of 5,000 patterns from SD-3 and 5,000 patterns from SD-1. The 60,000 pattern training set contained examples from approximately 250 writers.
- It was made sure that the sets of writers of the training set and test set were disjoint.

Methodology Used

Neural Networks

In information technology (IT), a neural network is a system of hardware and/or software patterned after the operation of neurons in the human brain. ... Commercial applications of these technologies generally focus on solving complex signal processing or pattern recognition problems.



We will be using Convolutional Neural Networks for the proposed project

Convolutional Neural Networks

Convolutional Neural Networks (CNN) is one of the variants of neural networks used heavily in the field of Computer Vision. It derives its name from the type of hidden layers it consists of. The hidden layers of a CNN typically consist of convolutional layers, pooling layers, fully connected layers, and normalization layers. Here it simply means that instead of using the normal activation functions, convolution and pooling functions are used as activation functions.

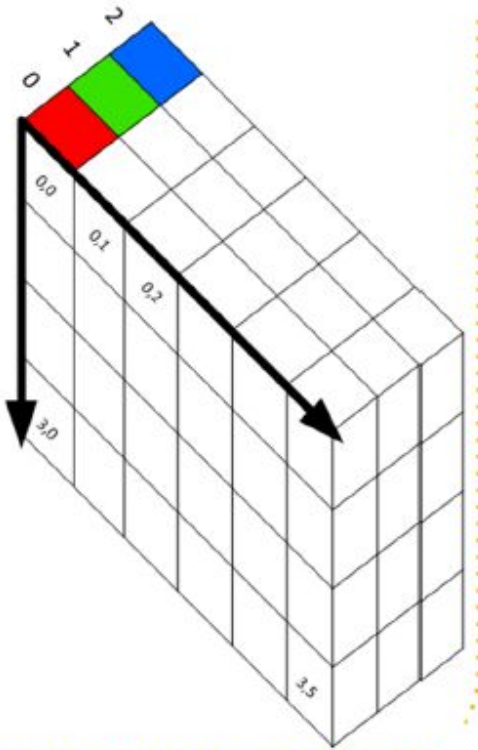
Convolutional Neural Networks

- Introduction to Convolutional Neural Networks
- Convolution & Image Features
- Depth, Stride and Padding
- ReLU
- Pooling
- The Fully Connected Layer
- Training CNNs
- Final Output

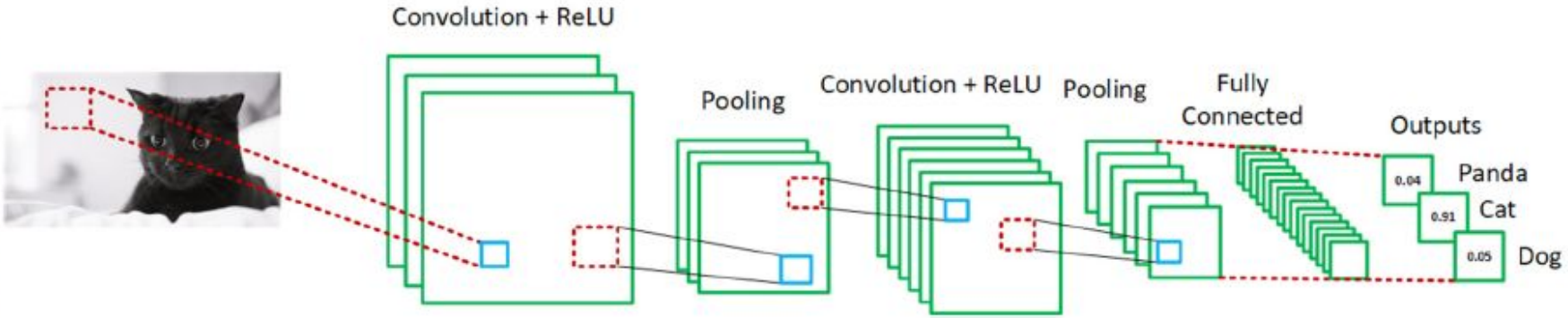
How do Computers Store Images?

- A Color image would consist of 3 channels (RBG) - Right
- And a Grayscale image would consist of one channel - Below

255	255	255	255	255	255	220	146	237	255	255	255	255	255
255	255	255	255	255	255	91		170	255	255	255	255	255
255	255	255	255	255	253	68		220	255	255	255	255	255
255	255	255	255	255	238	10	62	241	255	255	255	255	255
255	255	255	255	255	218		64	241	255	255	255	255	255
255	255	255	255	255	145		64	241	255	255	255	255	255
255	255	255	255	255	81		72	244	255	255	255	255	255
255	255	255	255	255	72		128	255	255	255	255	255	255
255	255	255	255	255	73		146	255	255	255	255	255	255
255	255	255	255	255	64		184	255	255	255	255	255	255
255	255	255	255	255	63		188	255	255	255	255	255	255
255	255	255	255	255	71		187	255	255	255	255	255	255
255	255	255	255	255	68		190	255	255	255	255	255	255
255	255	255	255	255	83		160	255	255	255	255	255	255
255	255	255	255	255	181	48	184	255	255	255	255	255	255

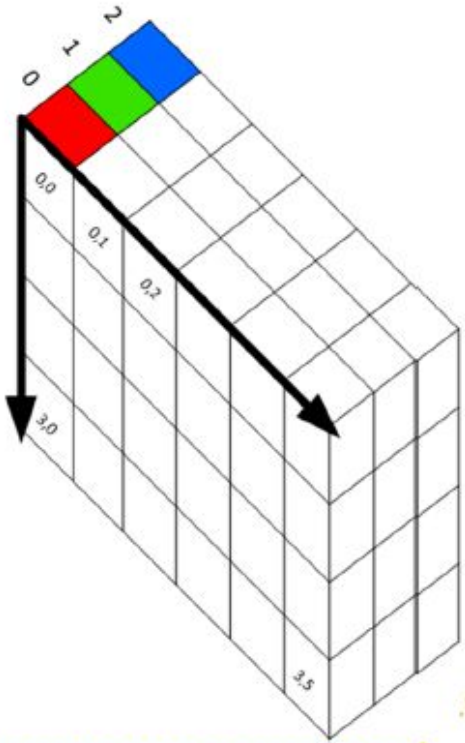


Introducing CNNs



CNN's use a 3D Volume Arrangement for it's Neurons

- Because our input data is an image, we can constrain or design our Neural Network to better suit this type of data
- Thus, we arrange our layers in **3 Dimensions. Why 3?**
- Because of image data consists of:
  1. Height
  2. Width
  3. Depth (RGB) our colors components

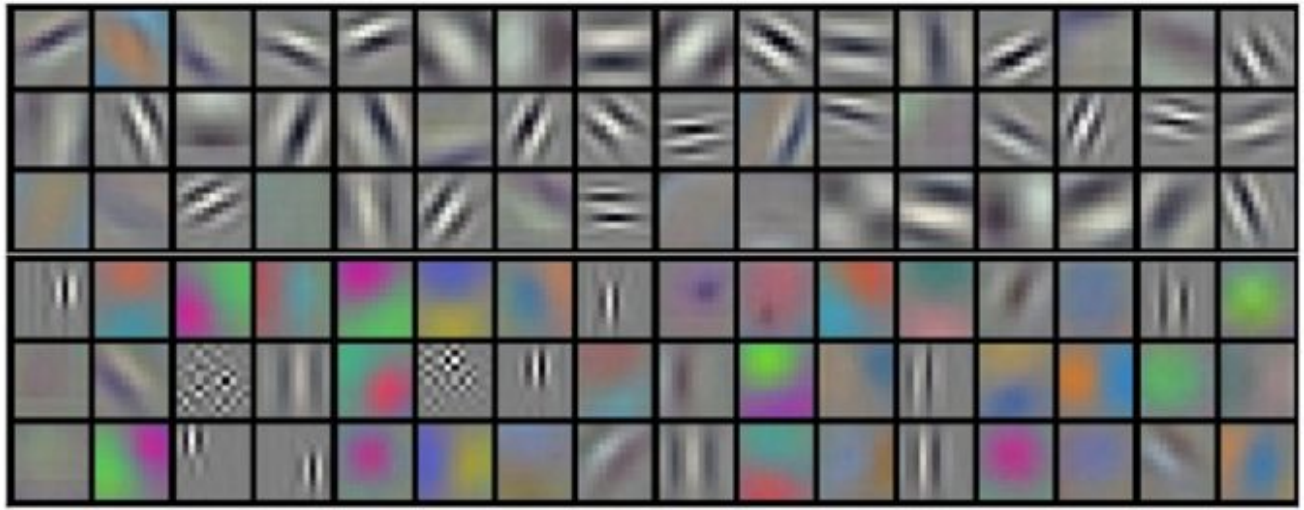


Convolution and Image Features

Image Features

- Image Features are simply interesting areas of an image. Examples:
  - Edges
  - Colors
  - Patterns/Shapes

Examples of Image Features

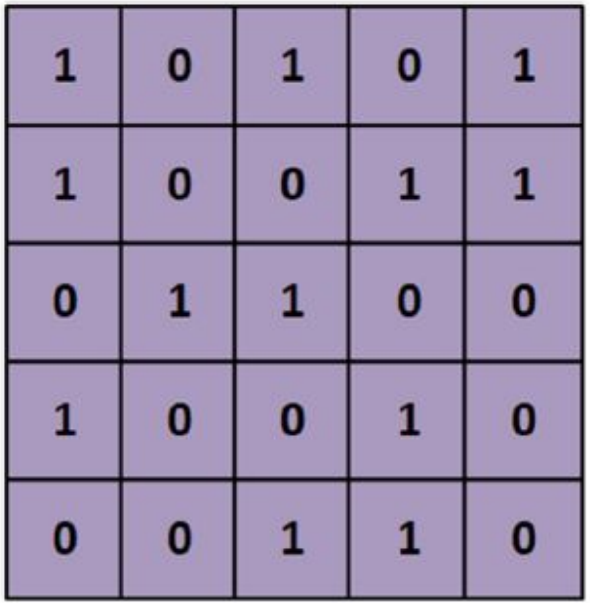


- Example filters learned by Krizhevsky et al.

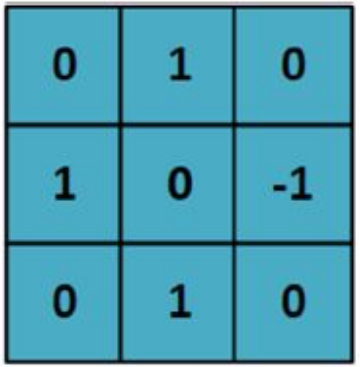
Convolutions

- Convolution is a mathematical term used to describe the process of combining two functions to produce third function.
- Third function is called feature map.
- Convolution is the action of applying a filter or kernel to our input.
- Convolution is executed by sliding the filter or kernel over the input image.
- This sliding process is a simple matrix multiplication or dot product.

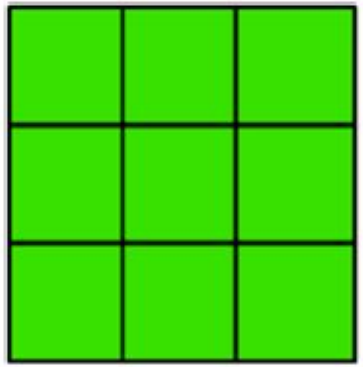
The Convolution Process



- Input



- Convolution



- Output or Feature Map

Applying out Kernel / Filter - Sliding

1	0x0	1x1	0x0	1
1	0x1	0x0	1x-1	1
0	1x0	1x1	0x1	0
1	0	0	1	0
0	0	1	1	0

(0x0)+(1x1)+(0x0)+
(0x1)+(0x0)+(1x-1)+
(1x0)+(1x1)+(0x1)

0+1+0+
0+0-1+
0+1+0 = 1

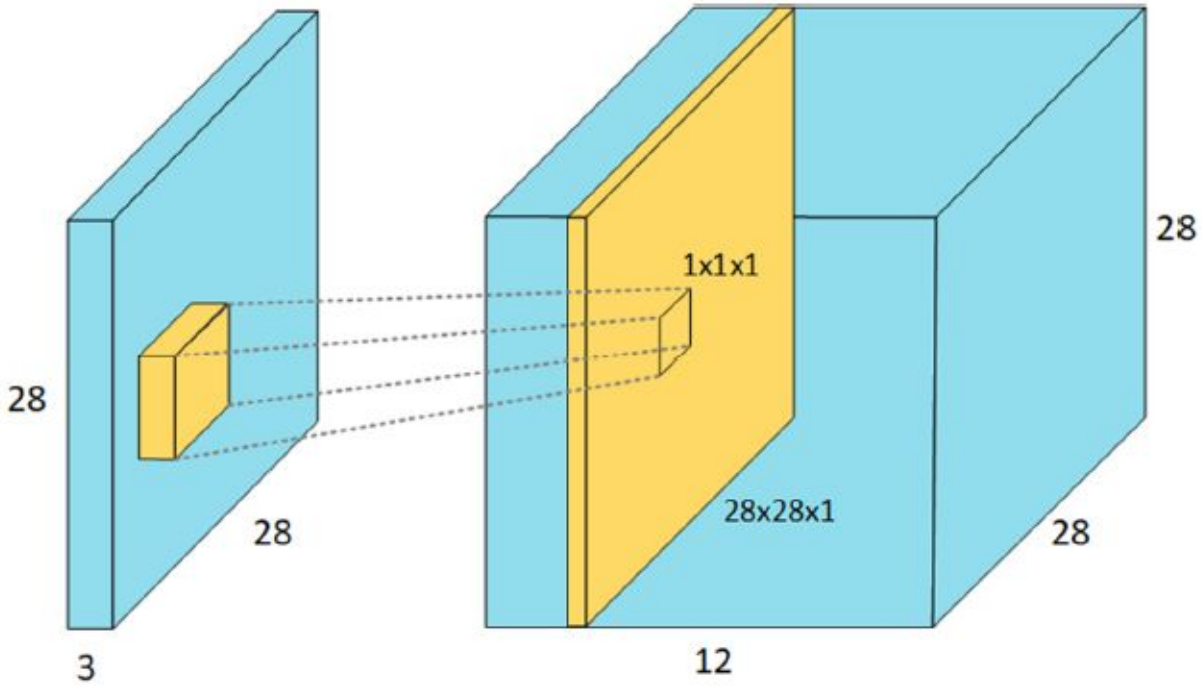
2	1	

What Are the Effects of the Kernel?

- Depending on the values on the kernel matrix, we produce different feature maps. Applying our kernel produces scalar outputs as we just saw.

- Convolving with different kernels produces interesting feature maps that can be used to detect different features.
- Convolution keeps the spatial relationship between pixels by learning image features over the small segments we pass over on the input image.

Illustration of our Multiple 3D Kernels/Filters



The Outputs of Our Convolutions

- From our last slide we saw by applying **12** filters (of size 3x3x3) to our input image (of size 28x28x3) we produced 12 **Feature Maps** (also called **Activation Maps**).
- These outputs are stacked together and treated as another 3D Matrix (in our example, our output was of **size 28x28x12**.
- This 3D output is the **input** to our next layer in our CNN

Depth, Stride and Padding

Designing Feature Maps

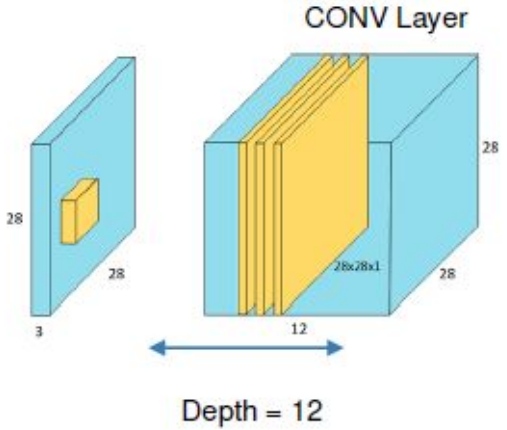
By tweaking the following **parameters** to control the size of our **feature maps.**

- Kernel Size (K x K)
- Depth
- Stride
- Padding

Depth

- Depth describes the **number of filters used**. It does not relate the image depth (3 channels) nor does it describe the number of hidden layers in our CNN.

- Each filter learns different feature maps that are activated in the presence of different image features (edges, patterns, colors, layouts)



Stride

- Stride simply refers to the step size we take when we slide our kernel the input image.

- In the last example we used a stride of 1

Zero-Padding Illustrated.

- We add a border of 0s around our input. Basically this is equivalent of adding a black border around an image
- We can set our Padding P to 2 if needed.

0	0	0	0	0	0	0
0	1	0	1	0	1	0
0	1	0	0	1	1	0
0	0	1	1	0	0	0
0	1	0	0	1	0	0
0	0	0	1	1	0	0
0	0	0	0	0	0	0

ReLU

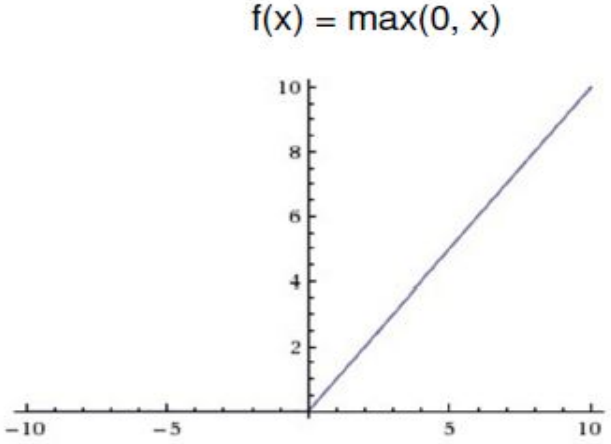
ReLU is the Activation Function of Choice

- ReLU or Rectified Linear Unit simply changes all the negative values to 0 while leaving the positives values unchanged.

2	1	-5
2	0	4
-5	3	-1

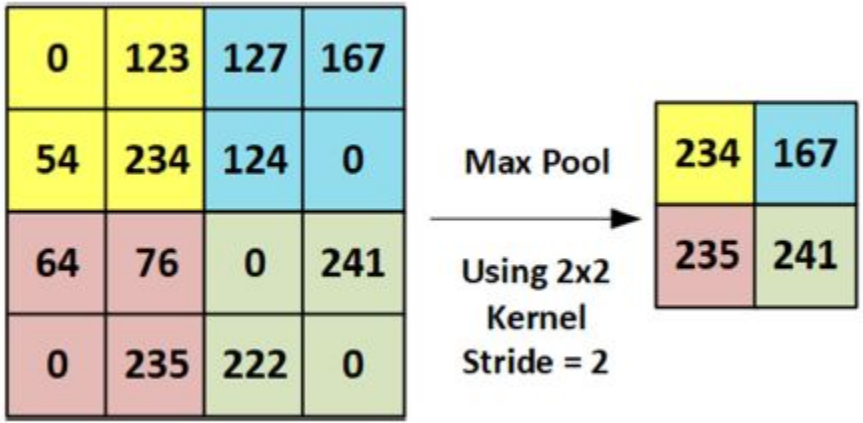
Applying ReLU →

2	1	0
2	0	4
0	3	0



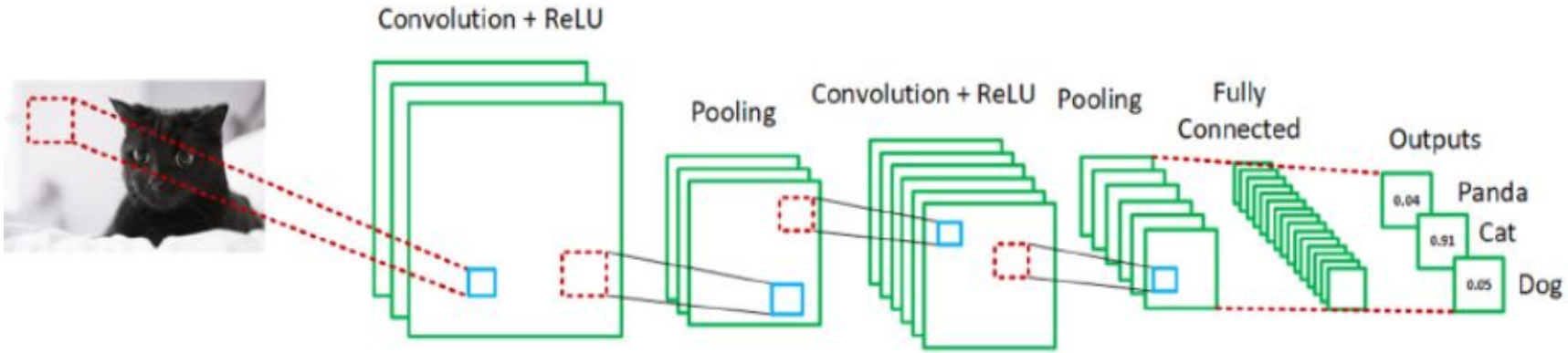
Pooling

- Pooling, also known as subsampling or downsampling is a process where we reduce the size or dimensionality of the feature map.
- The purpose of this reduction is to reduce the number of parameters needed to train our model.

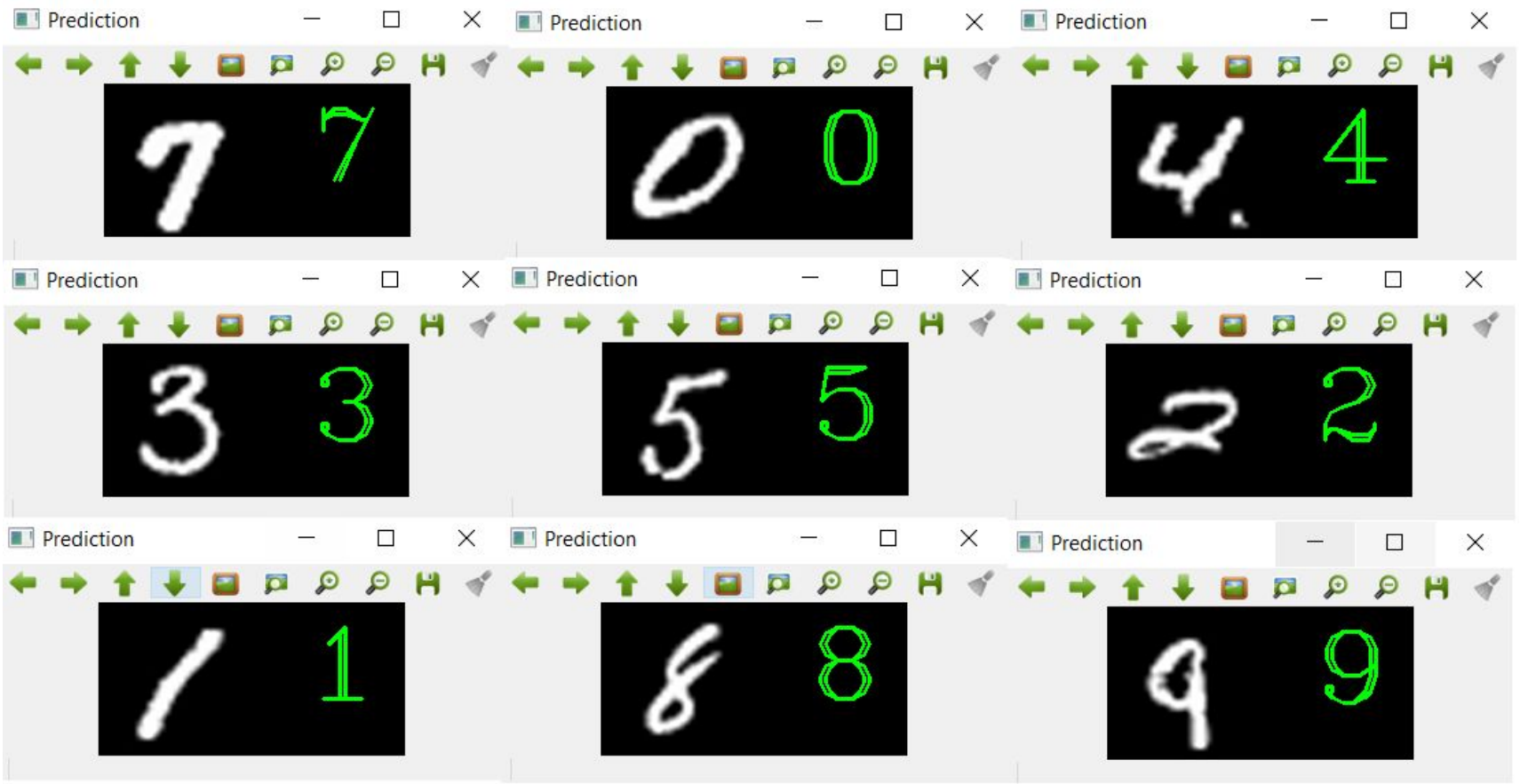


Fully Connected Layer

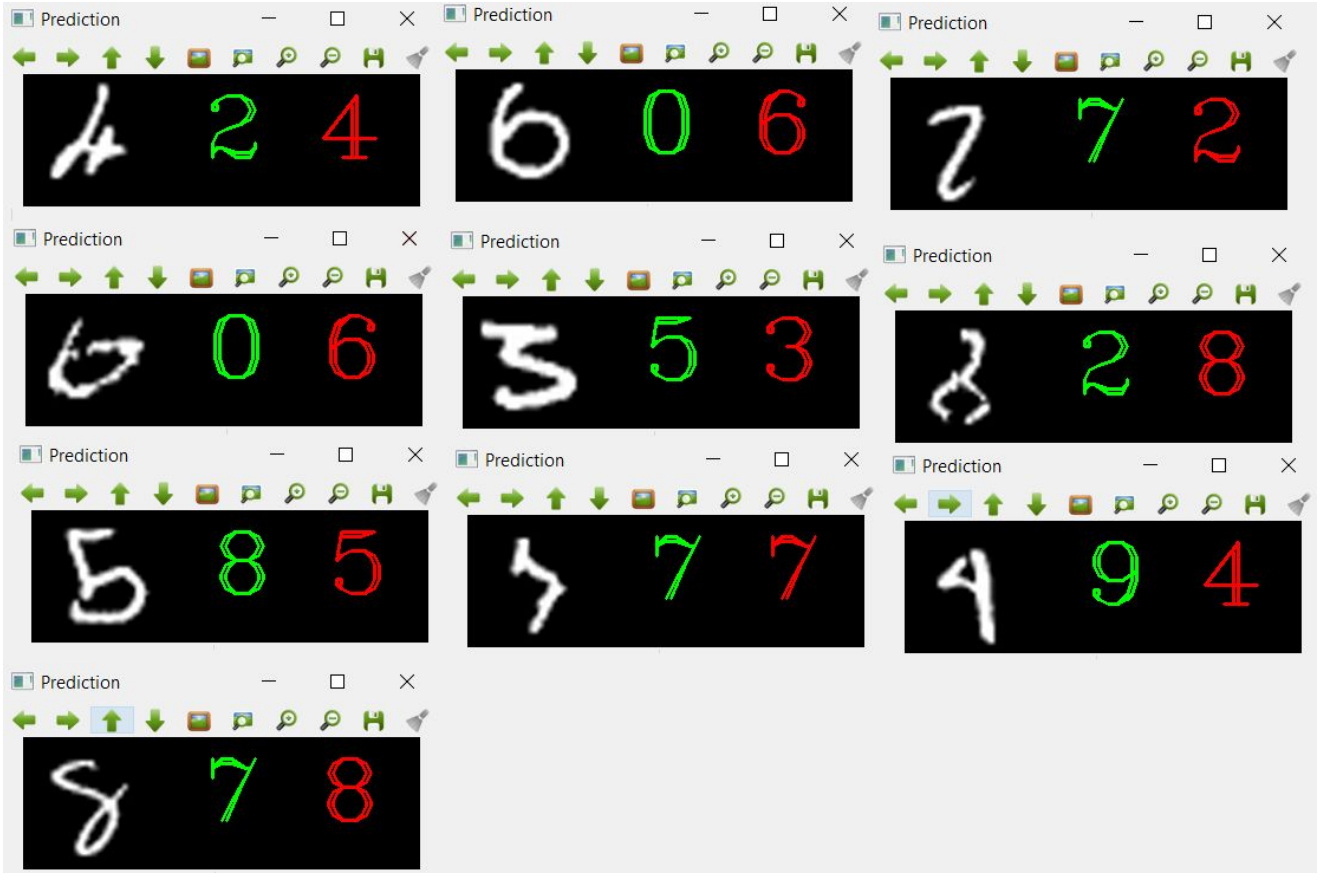
- Fully connected layer means all nodes in one layer are connected to the output of next layer.



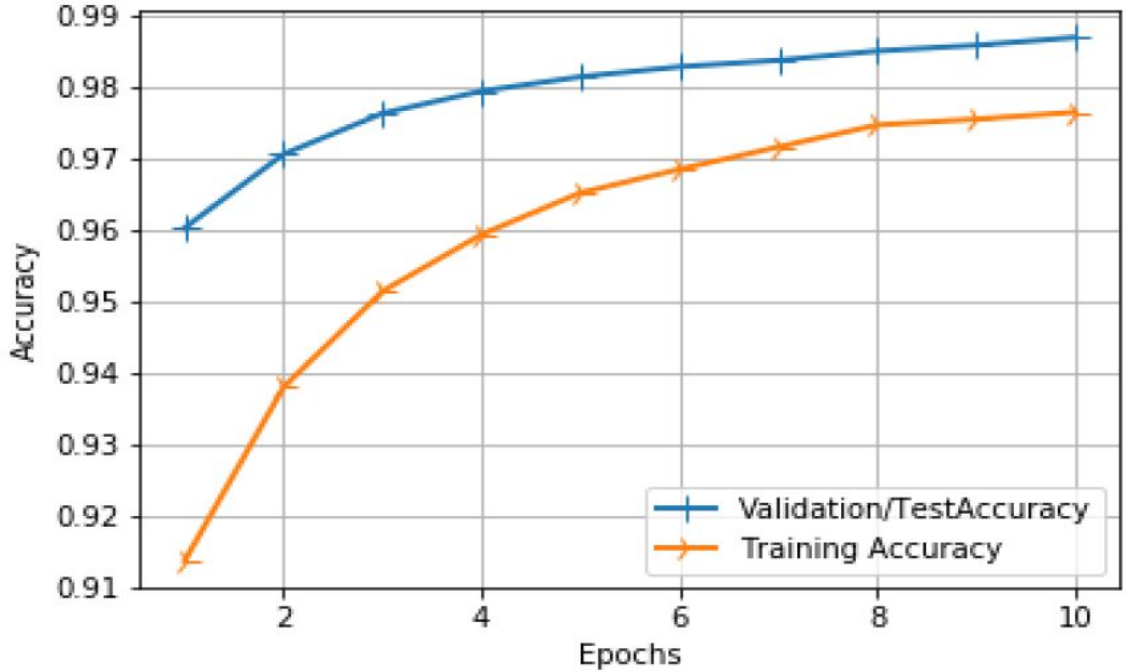
Predicted Values



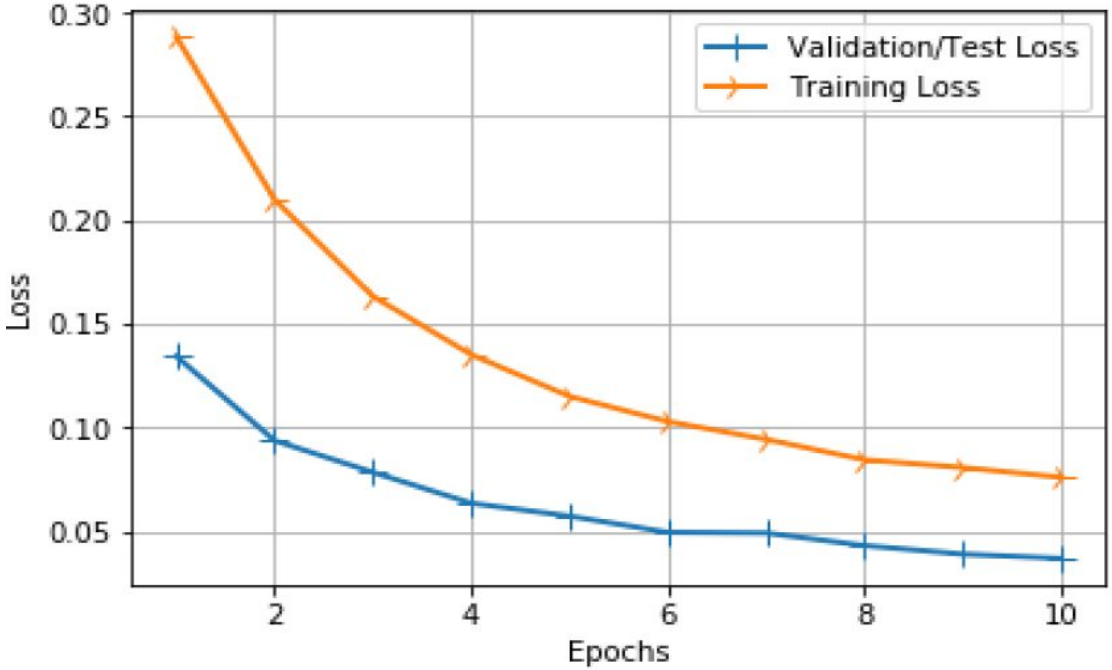
Misclassified Values



Accuracy Plot



Loss Plot


Validation/Test Loss
Training Loss
Loss
Epochs
0.30
0.25
0.20
0.15
0.10
0.05
2
4
6
8
10

Accuracy

	precision	recall	f1-score	support
0	0.97	1.00	0.98	980
1	0.99	0.99	0.99	1135
2	0.98	0.99	0.99	1032
3	0.99	0.98	0.99	1010
4	0.99	0.99	0.99	982
5	0.98	0.99	0.99	892
6	0.99	0.98	0.99	958
7	0.99	0.98	0.98	1028
8	0.99	0.98	0.98	974
9	0.99	0.98	0.98	1009
accuracy			0.99	10000
macro avg	0.99	0.99	0.99	10000
weighted avg	0.99	0.99	0.99	10000

Confusion Matrix

- Accuracy of the classifier is evaluated using a confusion matrix.
- Confusion matrix tells us the number of observations known to be in true values and predicted to be in predicted values. Confusion matrix for our digit classifier is shown below in figure.

```
[[ 976    1    0    0    0    0    1    1    1    0]
 [   0 1127    3    1    0    1    2    0    1    0]
 [   3    2 1022    1    1    0    0    3    0    0]
 [   0    0    3  994    0    8    0    3    2    0]
 [   0    1    1    0  968    0    4    0    2    6]
 [   2    0    0    4    0  882    2    1    1    0]
 [   8    2    0    0    1    4  942    0    1    0]
 [   2    1   10    1    0    0    0 1011    1    2]
 [   7    1    2    1    2    1    1    3  952    4]
 [   5    4    0    2    3    1    0    4    1  989]]
```

Conclusion

- In the project, we trained a 2 layer CNN on the MNIST dataset for the handwritten digits classification task.
- Accuracy can be increased by introducing additional layers of CNN, which in turn will extract more features.
- After tuning some hyperparameters, our model achieves the test accuracy of 98.62%.
- We believe the performance can be further improved by choosing other optimizers instead of SGD as it has a constant learning rate.

References

[1]. Convolutional Neural Network. http://cs231n.github.io/convolutional-networks/
[2]. MNIST Dataset. http://yann.lecun.com/exdb/mnist/
[3]. Stochastic Gradient Descent. https://www.geeksforgeeks.org/ml-stochastic-gradient-descent-sgd/
[4]. Dropout. https://www.tensorflow.org/api_docs/python/tf/nn/dropout
[5]. Activation Maximization on MNIST. https://github.com/raghakot/keras-vis/blob/master/examples/mnist/activation_maximization.ipynb (https://github.com/raghakot/keras-vis/blob/master/examples/mnist/activation_maximization.ipynb)
[6]. One Hot Encoder. https://towardsdatascience.com/categorical-encoding-using-label-encoding-and-one-hot-encoder-911ef77fb5bd and Udemy.
[7]. Confusion Matrix. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html
[8]. Deep learning Course Udemy. Deep Learning Computer Vision™ CNN, OpenCV, YOLO, SSD & GANs