

MNIST DIGITS CLASSIFICATION WITH CONVOLUTION NEURAL NETWORK

BY

AZLAN KAKAR (2021123)

ALI IFTIKHAR (2022341)



DEPARTMENT OF COMPUTER SCIENCE

9 MAY - 2024

ABSTRACT

To build a model which can identify the MNIST digits by Recognition and classify them. Humans can see and visually sense the world around them by using their eyes and brains. Computer vision works on enabling computers to see and process images in the same way that human vision does. Several algorithms have developed in computer vision to recognize images. The goal of our work will be to create a model that will be able to identify and determine the MNIST digit from its image with better accuracy. Aim to complete this by using the concepts of Convolutional Neural Network. Through this work, aim to learn and practically apply the concepts of Convolutional Neural Networks. Created our own data images of 1000 Images which are used to train and test.

The issue of transcribed digit acknowledgment has for some time been an open issue in the field of example order. A few examinations have demonstrated that a Neural Network has an incredible execution in information arrangement. The fundamental target of this paper is to give effective and solid procedures to acknowledge transcribed numerical by looking at different existing arrangement models. This paper is about the exhibition of Convolutional Neural Network (CNN). Results demonstrate that CNN classifier beat over Neural Network with critical improved computational effectiveness without relinquishing execution. MNIST digit recognition can be performed using the Convolutional neural network from Machine Learning. Basically, to perform the model required some libraries such as NumPy, Pandas', TensorFlow, Kera's.

This discussion is about the Convolutional Neural Network and its importance. It also covered how a dataset is divided into training and test dataset. A dataset was taken to make predictions of MNIST digits from 0 to 9. The dataset was cleaned, scaled, and shaped. Using TensorFlow, a CNN model was created and was eventually trained on the training dataset. Finally, predictions classification was made using the trained model with an accuracy of 98.6%.

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE NUMBER
	ABSTRACT	i
	INTRODUCTION	1
	1.1 INTRODUCTION OF MNIST DIGITS CLASSIFICATION	1
	1.2 PROBLEM STATEMENT & OBJECTIVE	2
2	LITERATURE SURVEY	5
3	EXISTING SYSTEM	8
	3.1 SUPPORT VECTOR MACHINE	8
	3.2 METHODOLOGY	9
	3.3 DISADVANTAGES	10
4	PROPOSED SYSTEM	11
	4.1 CONVOLUTION NEURAL NETWORK	13
	4.2 ADVANTAGES	15
	4.3 CNN ALGORITHM	15
	4.4 BLOCK DIAGRAM & WORKFLOW	16
	4.6 POOLING	19
	4.5 NORMALIZATION	18
	4.7 FULLY CONNECTED LAYER	21
	4.8 CNN WORKING	23

5	RESULTS AND DISCUSSIONS	24
	5.1 PREDICTED VALUES	25
	5.2 LIBRARIES AND DATA	28
	5.3 DIVIDING DATA INTO TRAINING & TESTING PART	29
	5.4 BUILDING THE MODEL	30
	5.5 ACCURACY	31
6	CONCLUSION & FUTURE SCOPE	32
	REFERENCES	33

CHAPTER - 1

INTRODUCTION

1.1 INTRODUCTION OF MNIST DIGITS CLASSIFICATION

Crane's early work culminated in a first patent in 1964 for a special pen device. These initial efforts took a more advanced form in the early 1970s with a system that used what was generally referred to as the “SRI pen” as a method of inputting characters into a computer. Commercial products incorporating handwriting recognition as a replacement for keyboard input were introduced in the early 1980s.

In the current age of digitization, handwriting recognition plays an important role in information processing. There is a lot of information available on paper and less processing of digital files than the processing of traditional paper files. The purpose of the handwriting recognition system is to convert MNIST letters into machine-readable formats. Major applications include vehicle licence-plate identification, postal paper-sorting services, historical document preservation in the check truncation system (CTS) scanning and archaeology departments, old document automation in libraries and banks, and more.

All of these areas deal with large databases and therefore require high identification accuracy, low computational complexity, and consistent performance of the identification system. Over time, the number of fields that can implement deep learning is increasing. In deep learning, convolutional neural networks (CNN) are being used for visual image analysis. CNN can be used in object detection, facial recognition, robotics, video analysis, segmentation, pattern recognition, natural language processing, spam detection, topical gradation, regression analysis, speech recognition, image classification.

Detection of MNIST numbers, including accuracy in these areas, has reached human perfection using deep convolutional neural networks (CNNs). Recently CNN has become one of the most attractive approaches and has been the ultimate factor in recent success and in several challenging machine learning applications. Considering all the factors stated above have chosen CNN for our challenging tasks of image classification.Used it to identify MNIST numbers, which is one of the higher education and business transactions. There are many applications of MNIST digit recognition for our real-life purposes. Hence using the Convolutional Neural Network (CNN).

1.2 PROBLEM STATEMENT & OBJECTIVE

Handwriting recognition has been the main subject of research for almost the last forty years. This research work analyzes the behavior of classification techniques (CNN) in a large handwriting dataset to predict a digit. Machine-learning techniques, particularly when applied to Neural Networks like CNN or ANN, have played an increasingly important role in the design of these recognition systems.

Several methods have been developed in MNIST digit recognition and these methods have been classified into categories: knowledge-based methods, feature-based methods, template-based methods and appearance-based methods. Errors in Digit recognition cause severe problems like digits written on a bank cheque if recognized erroneously could result in unfortunate consequences.

The goal of our work is to create a model that will be able to recognize and classify the MNIST digits from images by using concepts of Convolution Neural Network. Though the goal of our research is to create a model for digit recognition and classification, it can also be extended to letters and an individual's handwriting. The major goal of the proposed system is understanding Convolutional Neural Network, and applying it to the MNIST digit recognition system by working on the dataset created.

There have already been significant advancements in this area previously. Tried to form a model around the Conventional Neural Network as our own dataset so that the model has high accuracy and has been trained and tested on a large dataset. Also consider developing a robust test harness for estimating the performance of the model and then exploring improvements to the model. With high accuracy rates, the model can solve a lot of real-life problems. In the field of digital image processing, pattern recognition is the research area that studies the operation and design of systems that recognize patterns in data. Its primary goal is the classification of objects into several categories or classes. Depending on the use, these objects can be images, signal waveforms or any other type of measurements that need to be classified. Common applications of pattern recognition are automatic speech recognition, classification of text into several categories (e.g. spam/non-spam email messages) and the automatic recognition of MNIST postal codes on postal envelopes.

A human brain can easily interpret a sophisticated image and can extract data from it, but for a computer, an image is a collection of pixels which are nothing but a list of combinations of numbers ranging from 0 to 255, blends being RGB, B.W., greyscale, etc. Information in an image can be extracted from the pixel values. A human eye contains light-sensitive cells that can segment the image into partitions which help to interpret the information like shape, size, and color which are called features, these features are later sent to the visual cortex that analyses the characteristics and maps it to our memory for identification. Thus, our brain is capable of identifying an unseen image.



Figure: MNIST Digits

Figure shows Different MNIST Digits from Digit 0 to 9.

CHAPTER - 2

LITERATURE SURVEY

Ahamed Hafiz (2018) developed a Support Vector Machine SVM Based Real Time MNIST Digits Classification using Support vector Machine but it is Not suitable for Large Data sets. Hence it failed.

Denker J.S (2019) implemented MNIST Digits Recognition to reduce the error rate as much as possible in handwriting recognition. In one research, an error rate of 1.19% is achieved using 3-NN trained and tested on MNIST Coherence recurrent convolutional network (CRCN) is a multimodal neural architecture. It is being used in recovering sentences in an image. Some researchers are trying to come up with new techniques to avoid drawbacks of traditional convolutional layers. NCFM (No combination of feature maps) is a technique which can be applied for better performance using MNIST datasets.

Haider (2020) developed a new challenging digit Arabic dataset collected from different study levels of schools. A large dataset is collected after paying vast effort for distributing and collecting digit forms over hundreds of primary, high, college students. After he found that there were few and not challenging Arabic digit dataset, He paid vast effort for preparing such a challenging dataset.

Mukesh N (2018) implemented a Classification of MNIST Characters Using k-means clustering Machine Learning Algorithms But it is Not suitable for Large Data sets, and it has low accuracy and High Baseline Error.

Nitin Kali Raman (2021) proposed MNIST Digit Recognition using Artificial Neural Network. An ANN is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal to other neurons. An artificial neuron receives a signal then processes it and can signal neurons connected to it. But he got a High Baseline Error.

Plamondon, R (2018) developed a MNIST digits recognition and dataset that has been trained on artificial neural networks and on convolutional neural networks. The average error of both networks was calculated using the accuracy evaluation metric. The average error of CNN is less than an artificial neural network on the CPU. Though while training CNN over CPU took more time than artificial neural network. But image classification is better performed in the case of CNN. It can be concluded that as the model is trained with CNN, the accuracy of recognition increases respectively but if trained on GPU can get optimum results for the classification with CNN.

Saeed Mansoori (2020) proposed a Multilayer Perceptron (MLP) Neural Network to recognize and predict MNIST digits from 0 to 9. A multilayer perceptron (MLP) is a fully connected class of feedforward artificial neural network (ANN). The term MLP is used ambiguously, sometimes loosely to mean *any* feedforward ANN, sometimes strictly to refer to networks composed of multiple layers of perceptrons (with threshold activation). The proposed neural system was trained and tested on a dataset achieved from MNIST.

Shamim S.M (2018) introduced MNIST Digit Recognition using Machine Learning Algorithms using Gaussian Naive Bayes. Gaussian Naive Bayes is a variant of

Naive Bayes that follows Gaussian normal distribution and supports continuous data. The drawback of this is Low accuracy.

Shyam R (2017) implemented MNIST Digits Classification using MNIST dataset that deep nets perform better when they are trained by simple back-propagation. But, Their architecture results in the lowest error rate on MNIST compared to NORB and CIFAR10.

Sonia flora (2016) developed a MNIST Digits Classification using support vector machine (SVM). Support vector machine is supervised learning models with associated learning algorithms that analyze data for classification and regression analysis. But got less accuracy compared to convolution neural network (CNN). Since it is a large dataset it gives less accuracy.

Vinjit B.M (2020) implemented MNIST Character Recognition Methods and Techniques Using keras with Theano and Tensorflow gives the lowest accuracy in comparison with the most widely used machine learning algorithms. Because of its lowest accuracy, Convolutional Neural Network (CNN) is being used on a large scale in image classification, video analysis, etc.

CHAPTER - 3

EXISTING SYSTEM

3.1 SUPPORT VECTOR MACHINE

“Support Vector Machine” (SVM) is a supervised machine learning algorithm that can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, plot each data item as a point in n -dimensional space (where n is a number of features you have) with the value of each feature being the value of a particular coordinate. Then performed classification by finding the hyper-plane that differentiates the two classes very well.

Support Vectors are simply the coordinates of individual observation. The SVM classifier is a frontier that best segregates the two classes (hyper-plane/ line).

At support vector machines and a few examples of their working here. All values for z would be positive always because z is the squared sum of both x and y

In the original plot, red circles appear close to the origin of x and y axes, leading to lower value of z and stars relatively away from the origin result in higher value of z .

Support Vector Machine (SVM) is a supervised machine learning algorithm. In this generally plot data items in n -dimensional space where n is the number of features, a particular coordinate represents the value of a feature performing the classification by finding the hyperplane that distinguishes the two classes. It will choose the hyperplane that separates the classes correctly. SVM chooses the extreme vectors that help in creating the hyperplane. These extreme cases are called support vectors, and hence the algorithm is termed as Support Vector Machine. There are mainly two types of SVMs, linear and non-linear SVM. Linear SVM for MNIST digit recognition.

3.2 METHODOLOGY

SVM is capable of doing both classification and regression. In this report we'll focus on using SVM for classification. In particular we'll be focusing on non-linear SVM, or SVM using a non-linear kernel. Non-linear SVM means that the boundary that the algorithm calculates doesn't have to be a straight line. The benefit is that you can capture much more complex relationships between your data points without having to perform difficult transformations on your own. The downside is that the training time is much longer as it's much more computationally intensive.

DISADVANTAGES

1) SVM algorithm is not suitable for large data sets.

2) SVM does not perform very well when the data set has more noise i.e. target classes

are overlapping.

3) In cases where the number of features for each data point exceeds the number of training data samples, the SVM will underperform.

4) As the support vector classifier works by putting data points, above and below the classifying hyperplane there is no probabilistic explanation for the classification.

Hence we need to propose a new and optimized system for this problem.

CHAPTER - 4

PROPOSED SYSTEM

4.1 CONVOLUTION NEURAL NETWORK (CNN)

A convolutional neural network (CNN) is a type of artificial neural network used in image recognition and processing that is specifically designed to process pixel data. A convolutional neural network consists of an input layer, hidden layers and an output layer. In any feed-forward neural network, any middle layers are called hidden because their inputs and outputs are masked by the activation function and final convolution. In a convolutional neural network, the hidden layers include layers that perform convolutions. Typically, this includes a layer that performs a dot product of the convolution kernel with the layer's input matrix. This product is usually the Frobenius inner product, and its activation function is commonly ReLU. As the convolution kernel slides along the input matrix for the layer, the convolution operation generates a feature map, which in turn contributes to the input of the next layer. This is followed by other layers such as pooling layers, fully connected layers, and normalization layers.

Feed Forward neural network and Designed mostly for Image classification Consists of an Input layer , multiple hidden layers , output layers In deep learning, a convolutional neural network (CNN/ConvNet) is a class of deep neural networks, most commonly applied to analyze visual imagery. Now when thinking of a neural network think about matrix multiplications but that is not the case with ConvNet. It uses a special technique called Convolution. Now in mathematics convolution is a mathematical operation on two functions that produces a third function that expresses how the shape of one is modified by the other. The role of the ConvNet is to reduce the images into a form that is easier to process, without losing features that are critical for getting a good prediction.

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The benefit of using CNNs is their ability to develop an internal representation of a two-dimensional image. This allows the model to learn position and scale invariant structures in the data, which is important when working with images. CNN is mainly used in image analysis tasks like Image recognition, Object detection & Segmentation. CNNs are powerful image processing, artificial intelligence (AI) that use deep learning to perform both generative and descriptive tasks, often using machine vision that includes image and video recognition, along with recommender systems and natural language processing (NLP). Faster R-CNN is a single-stage model that is trained end-to-end. It uses a novel region proposal network (RPN) for generating region proposals, which save time compared to traditional algorithms like Selective Search. It uses the ROI Pooling layer to extract a fixed-length feature vector from each region proposal. A convolutional neural network (CNN/ConvNet) is a class of deep neural networks, most commonly applied to analyze visual imagery. Now when thinking of a neural network, think about matrix multiplications but that is not the case with ConvNet. It uses a special technique called Convolution. In a Convolutional neural network, the kernel is nothing but a filter that is used to extract the features from the images. The kernel is a matrix that moves over the input data, performs the dot product with the sub-region of input data, and gets the output as the matrix of dot products.

CNNs are a class of Deep Neural Networks that can recognize and classify particular features from images and are widely used for analyzing visual images. Their applications range from image and video recognition, image classification, medical image analysis, computer vision and natural language processing. Neurons in a convolutional layer that cover the entire input and look for one feature are called filters. These filters are 2 dimensional (they cover the entire image). However, having the whole convolutional layer looking for just one feature (such as a corner) would massively limit the capacity of your network. The main special technique in CNNs is

convolution, where a filter slides over the input and merges the input value + the filter value on the feature map. In the end, our goal is to feed new images to our CNN so it can give a probability for the object it thinks it sees or describe an image with text. The output of the CNN is also a 4D array. Where batch size would be the same as input batch size but the other 3 dimensions of the image might change depending upon the values of filter, kernel size, and padding use. 2D CNNs use 2D convolutional kernels to predict the segmentation map for a single slice. Segmentation maps are predicted for a full volume by taking predictions one slice at a time. The 2D convolutional kernels are able to leverage context across the height and width of the slice to make predictions.

In any neural network, a dense layer is a layer that is deeply connected with its preceding layer which means the neurons of the layer are connected to every neuron of its preceding layer. This layer is the most commonly used layer in artificial neural network networks.

4.2 ADVANTAGES

- 1) Very High accuracy in image recognition problems.
- 2) Automatically detects the important features without any human supervision.
- 3) Weight sharing.
- 4) It automatically detects the important features without any human supervision. For example, given many pictures of cats and dogs it learns distinctive features for each class by itself.
- 5) CNN is also computationally efficient.

4.3 CNN ALGORITHM

- 1) Images (100x100x3 images)
- 2) Convolution Layer
- 3) ReLU (Normalization)
- 4) Max pooling

5) Fully connected Output

4.4 BLOCK DIAGRAM & WORKFLOW

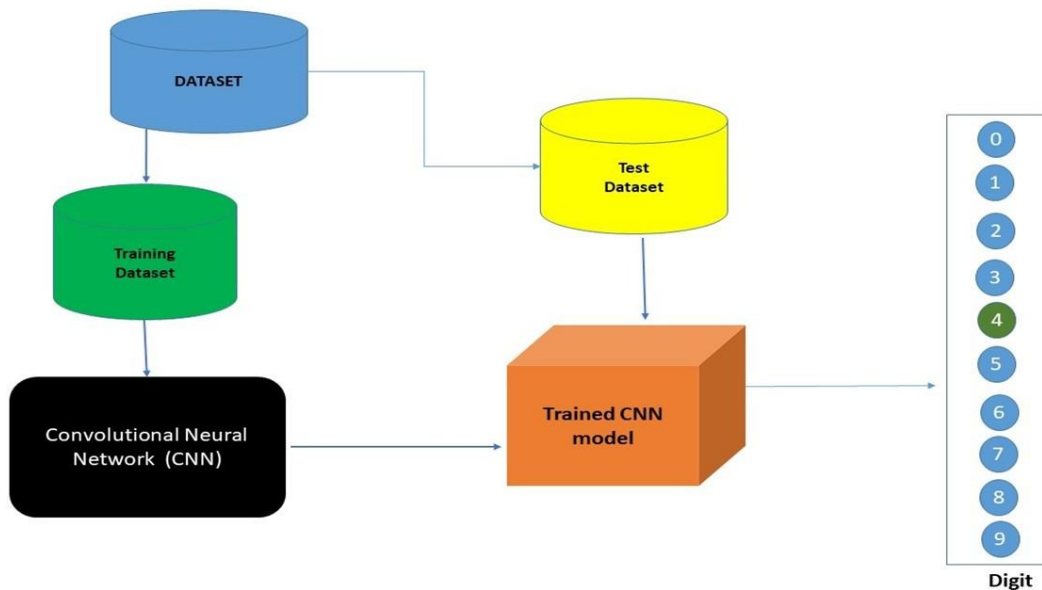


Figure : CNN WorkFlow

Figure represents Convolutional Neural Network workflow. MNIST Digits Dataset is created and then Dataset will be trained and tested from training dataset it will go to cnn model. Test dataset is sent to Trained CNN model from that digits were classified and predicted.

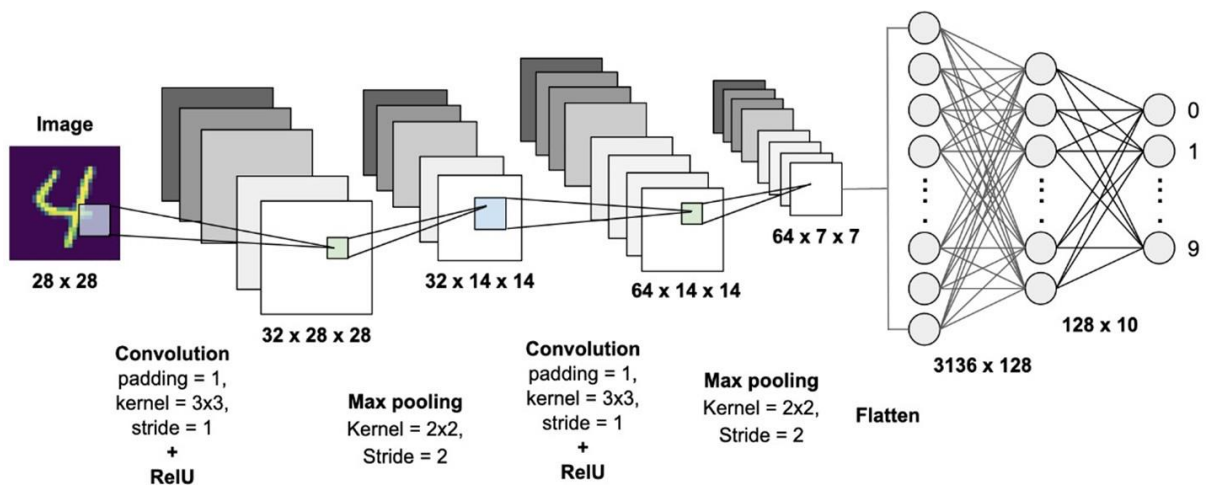


Figure: CNN Block Diagram

Figure shows CNN Block Diagram. From 28 x 28 pixel Image using convolution and max pooling and relu techniques images are predicted and classified.

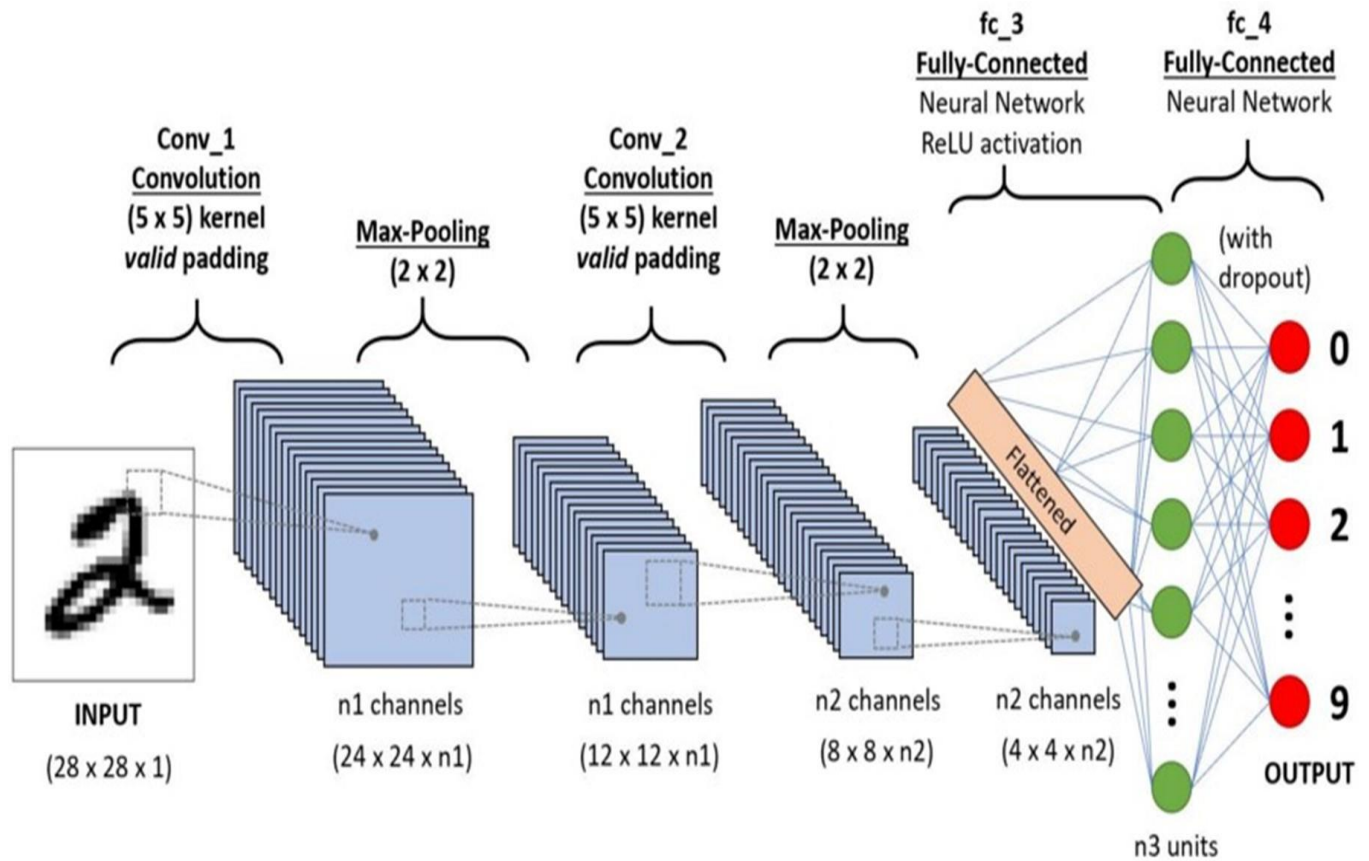


Figure : Layers in CNN

Figure depicts layers in CNN. From input image with 28x28x1 pixel modified into different layers and channels as shown in the above Figure.

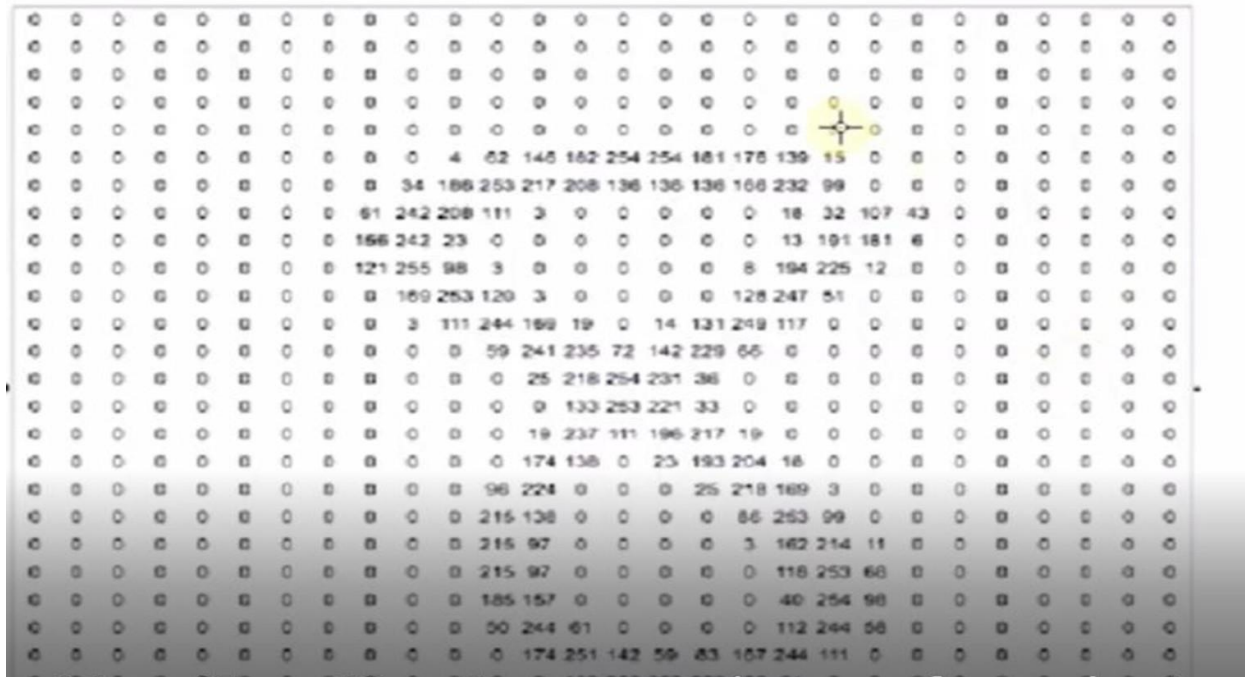


Figure: Digit in pixel view

Figure indicates Digit in pixel view. It is showing pixel values in black region and zeroes in white region. Based on this image are identified.

4.5 NORMALIZATION

ReLU (Rectified Linear Units) Activation Function The main reason ReLU is used is because it is simple, fast, and empirically it seems to work well. The usage of ReLU helps to prevent the exponential growth in the computation required to operate the neural network. If the CNN scales in size, the computational cost of adding extra ReLUs increases linearly. The rectified linear activation function, or ReLU activation function, is perhaps the most common function used for hidden layers. It is common because it is both simple to implement and effective at overcoming the limitations of other previously popular activation functions, such as Sigmoid and Tanh.

4.6 POOLING

Reduces the Dimension of matrix Max pooling, Min pooling, Average pooling
Max pooling is mostly used Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data through dimensionality reduction. Furthermore, it is useful for extracting dominant features which are rotational and positional invariant, thus maintaining the process of effectively training the model. There are two types of Pooling: Max Pooling and Average Pooling. Max Pooling returns the maximum value from the portion of the image covered by the Kernel. On the other hand, Average Pooling returns the average of all the values from the portion of the image covered by the Kernel.

3.0	3.0	3.0
3.0	3.0	3.0
3.0	2.0	3.0

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

Figure: Pooling

Figure shows pooling. From convolution matrix it converts and takes as 5x5 matrix and calculates its values of that particular matrix and then it forms a 3x3 new matrix. Similarly for all matrices as shown in the above Figure.

Max Pooling also performs as a Noise Suppressant. It discards the noisy activations altogether and also performs de-noising along with dimensionality reduction. On the other hand, Average Pooling simply performs dimensionality reduction as a noise suppressing mechanism. Hence Max Pooling performs a lot better than Average Pooling. The Convolutional Layer and the Pooling Layer, together form the i-th layer of a Convolutional Neural Network. Depending on the complexities in the images, the number of such layers may be increased for capturing low-level details even further, but at the cost of more computational power.

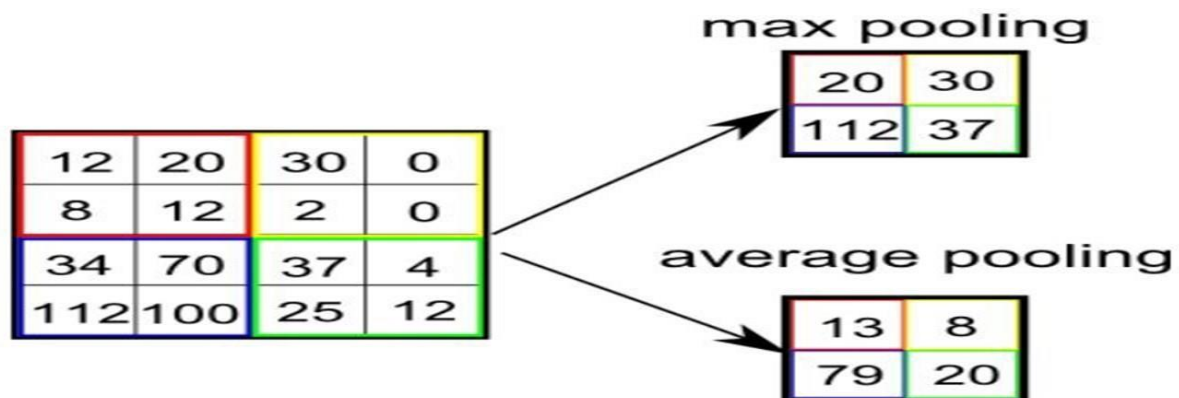


Figure: Max & Average Pooling

Figure explains about max & Average pooling. In max pooling it will take max number in every matrix. In Average pooling it will take Average number in every matrix.

4.7 FULLY CONNECTED LAYER

Adding a Fully-Connected layer is a (usually) cheap way of learning non-linear combinations of the high-level features as represented by the output of the convolutional layer. The Fully-Connected layer is learning a possibly non-linear function in that space. Now convert our input image into a suitable form for our MultiLevel Perceptron, shall flatten the image into a column vector. The flattened output is fed to a feed-forward neural network and backpropagation applied to every iteration of training. Over a series of epochs, the model is able to distinguish between dominating and certain low-level features in images and classify them using the Softmax Classification technique.

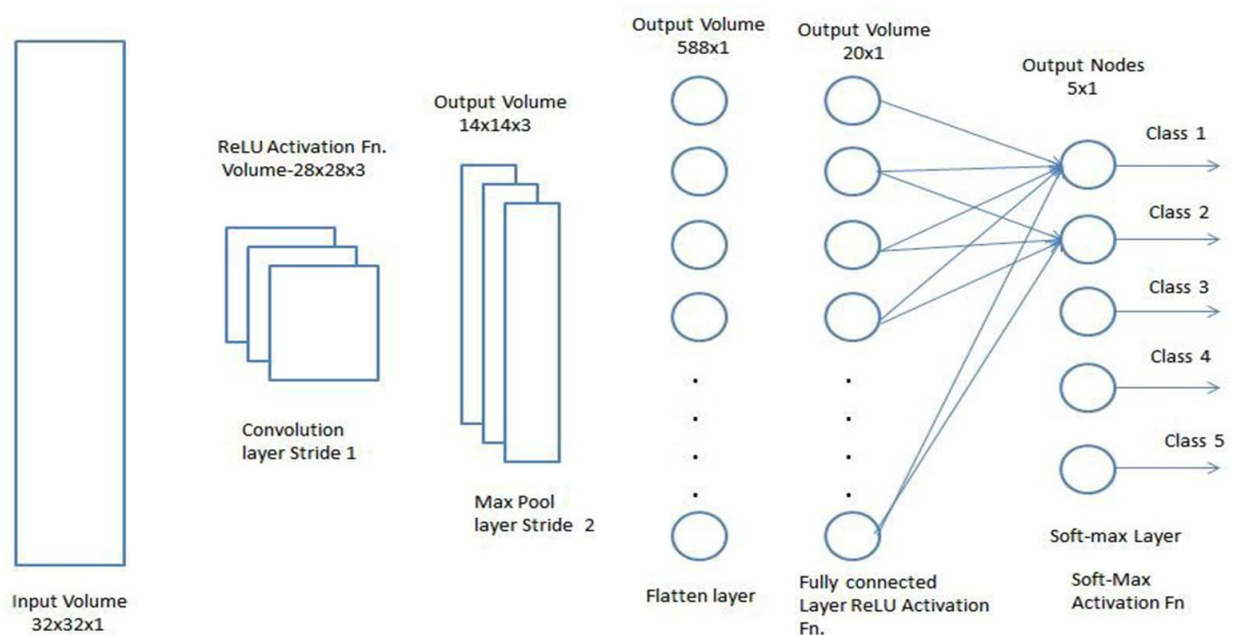


Figure: Fully Connected Layer Block Diagram

Figure represents Fully Connected Layer Block Diagram. In this layer from input volume $32 \times 32 \times 1$ converts into convolution layer stride 1 and then converts into max pooling layer stride 2. Using flatten layer and fully connected layer with relu activation function converts into soft-max layer.

4.8 CNN WORKING

An RGB image is nothing but a matrix of pixel values having three planes whereas a grayscale image is the same but it has a single plane. Take a look at this image to understand more. For simplicity, let's stick with grayscale images to understand CNNs working.



Figure: Three Colour Channels

Figure indicates Three Colour Channels. Three colours are red, green, blue with width 4 units pixels and height 4 units pixels with different matrices.

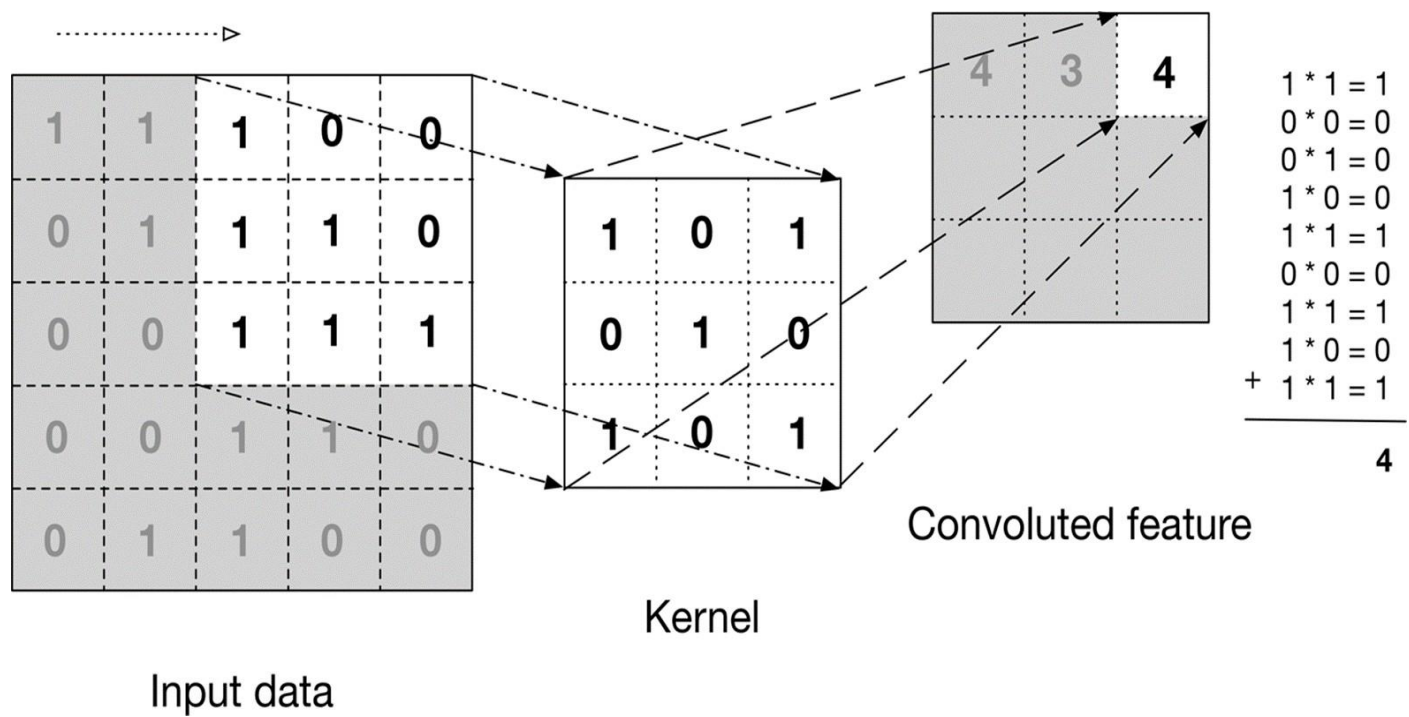


Figure : Kernel View

Figure 4.9 shows a convolution. Take a filter/kernel(3×3 matrix) and apply it to the input image to get the convolved feature. This convolved feature is passed on to the next layer.

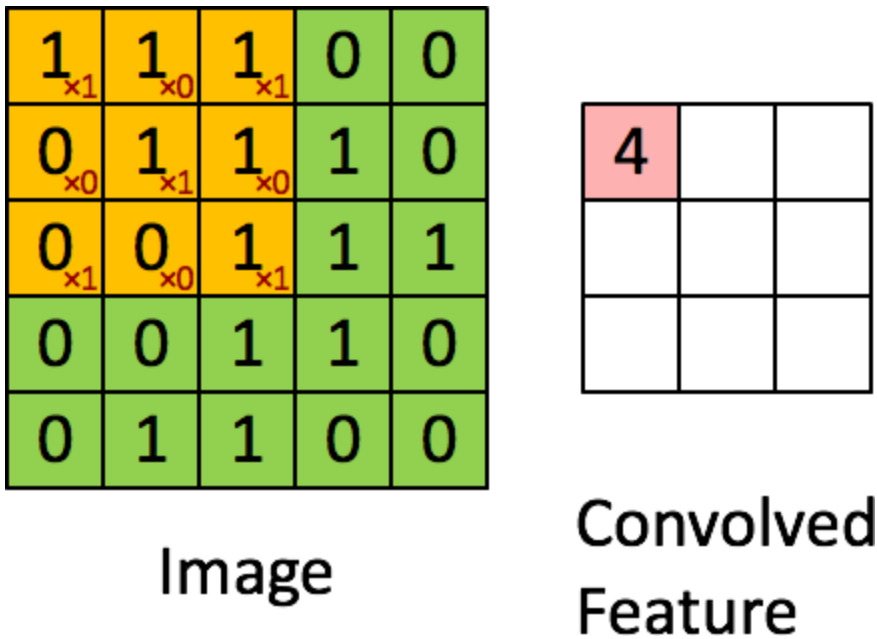


Figure: Convolved Feature

Figure indicates convolved features. From Image matrix converted into convolved feature. It converts and takes as 5x5 matrix and calculates its values of that matrix and then it forms a 3x3 new matrix. Similarly for all matrices as shown in the above Figures.

In the case of RGB color, channels look at this animation to understand how it's working. Convolutional neural networks are composed of multiple layers of artificial neurons. Artificial neurons, a rough imitation of their biological counterparts, are mathematical functions that calculate the weighted sum of multiple inputs and output an activation value.

When given an input image in a ConvNet, each layer generates several activation functions that are passed onto the next layer. The first layer usually extracts basic features such as horizontal or diagonal edges. This output is passed on to the next layer which detects more complex features such as corners or combinational edges. As it

moves deeper into the network it can identify even more complex features such as objects, faces, etc.....

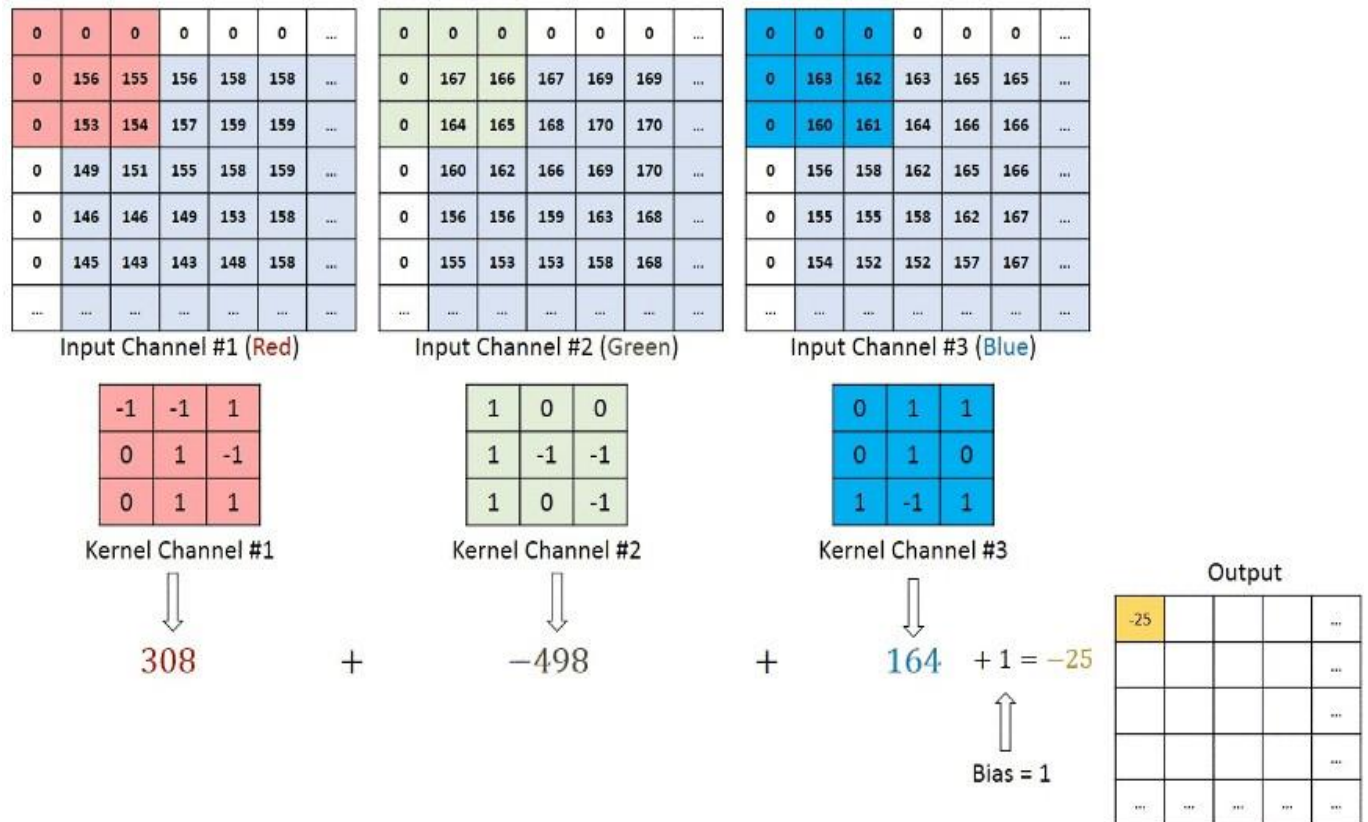


Figure: Kernel Channel

Figure shows the Kernel Channel of CNN. Considering three color channels as input channel and converting into kernel channel as 3x3 matrix and predicting the output.

CHAPTER - 5

RESULTS AND DISCUSSION

5.1 RESULTS WITH THE PREDICTED VALUES

Predicted values



Figure : PREDICTED VALUES

Figure : Captured Image Function Code

```
✓ 2s # predicted targets of each images
# (labels above the images are predicted labels)
fig, ax = plt.subplots(figsize=(18, 12))
for ind, row in enumerate(test[:15]):
    plt.subplot(3, 5, ind+1)
    plt.title(y_pred[ind])
    img = row.reshape(28, 28)
    fig.suptitle('Predicted values', fontsize=24)
    plt.axis('off')
    plt.imshow(img, cmap='cividis')
```

The given code snippet visualizes a set of images with their corresponding predicted labels. Here is a breakdown of what the code is doing:

- It creates a Figure and a set of subplots (Figure, `ax = plt.subplots(figsize=(18, 12))`) to hold multiple smaller plots (images in this case).
- The code iterates over a subset of 15 elements from a dataset (`test[:15]`), which are expected to be images.
- For each of these 15 images:
 - A subplot is created within the main plot layout. The subplot is arranged in a 3-row by 5-column grid (`plt.subplot(3, 5, ind+1)`).
 - The title for the subplot is set to the corresponding predicted label (`plt.title(y_pred[ind])`).
 - The image is reshaped into a 28x28 format (`img = row.reshape(28, 28)`) - likely because these images originate from a dataset such as the MNIST dataset, where each image is 28x28 pixels.
 - The plot title for the entire Figure is set to "Predicted values" (Figure.suptitle('Predicted values', fontsize=24)) to indicate the nature of the displayed images.
 - The image is displayed without showing the axes (`plt.axis('off')`), to focus on the image content.
 - The image is then displayed using a colormap of 'cividis' (`plt.imshow(img, cmap='cividis')`), which is a yellow-to-blue color gradient often used for better visibility.

- The purpose of this code snippet is to visualize a set of 15 test images, with each subplot showing the predicted label for the corresponding image. This kind of plot is typically used to assess or present the results of a machine learning model's predictions on a dataset of images, providing a visual context for the predicted outcomes.

5.2 Libraries and DATA import

▾ Libraries

```
# for numerical analysis
import numpy as np
# to store and process in a dataframe
import pandas as pd

# for plotting graphs
import matplotlib.pyplot as plt
# advanced plotting
import seaborn as sns

# image processing
import matplotlib.image as mpimg

# train test split
from sklearn.model_selection import train_test_split
# model performance metrics
from sklearn.metrics import confusion_matrix, classification_report

# utility functions
from tensorflow.keras.utils import to_categorical
# sequential model
from tensorflow.keras.models import Sequential
# layers
from tensorflow.keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout
```

▾ Data

```
✓ [2] # import train and test dataset
6s   train = pd.read_csv("/content/train.csv")
      test = pd.read_csv("/content/test.csv")
```

Figure : Create Function

Figure indicates Create function. In this data is created and loaded. Once created, it will shuffle the data randomly.

5.3 DIVIDING DATA INTO TRAINING AND TESTING PART

```
[16] # train test split
# =====

# random seed
random_seed = 2

# train validation split
X_train, X_val, y_train_enc, y_val_enc = train_test_split(X, y_enc, test_size=0.3)

# shape
for i in [X_train, y_train_enc, X_val, y_val_enc]:
    print(i.shape)

(29400, 28, 28, 1)
(29400, 10)
(12600, 28, 28, 1)
(12600, 10)
```

Figure : Data Dividing Code

Figure shows Data Dividing Code. Data divided into training and testing part.

Model fitting

```
[23] history = model.fit(X_train, y_train_enc,
                        epochs=EPOCHS,
                        batch_size=BATCH_SIZE,
                        verbose=VERBOSE,
                        validation_split=0.3)
```

Epoch 1/10
161/161 - 26s - loss: 0.6022 - accuracy: 0.8050 - val_loss: 0.1487 - val_accuracy: 0.9508 - 26s/epoch - 159ms/step
Epoch 2/10
161/161 - 20s - loss: 0.1576 - accuracy: 0.9526 - val_loss: 0.0760 - val_accuracy: 0.9756 - 20s/epoch - 127ms/step
Epoch 3/10
161/161 - 20s - loss: 0.1047 - accuracy: 0.9690 - val_loss: 0.0666 - val_accuracy: 0.9798 - 20s/epoch - 126ms/step
Epoch 4/10
161/161 - 20s - loss: 0.0828 - accuracy: 0.9753 - val_loss: 0.0570 - val_accuracy: 0.9830 - 20s/epoch - 126ms/step
Epoch 5/10
161/161 - 20s - loss: 0.0659 - accuracy: 0.9805 - val_loss: 0.0535 - val_accuracy: 0.9840 - 20s/epoch - 125ms/step
Epoch 6/10
161/161 - 21s - loss: 0.0533 - accuracy: 0.9832 - val_loss: 0.0564 - val_accuracy: 0.9819 - 21s/epoch - 132ms/step
Epoch 7/10
161/161 - 20s - loss: 0.0453 - accuracy: 0.9859 - val_loss: 0.0531 - val_accuracy: 0.9842 - 20s/epoch - 125ms/step
Epoch 8/10
161/161 - 20s - loss: 0.0393 - accuracy: 0.9875 - val_loss: 0.0509 - val_accuracy: 0.9861 - 20s/epoch - 123ms/step
Epoch 9/10
161/161 - 23s - loss: 0.0290 - accuracy: 0.9911 - val_loss: 0.0489 - val_accuracy: 0.9874 - 23s/epoch - 141ms/step
Epoch 10/10
161/161 - 20s - loss: 0.0282 - accuracy: 0.9914 - val_loss: 0.0460 - val_accuracy: 0.9875 - 20s/epoch - 126ms/step

Figure : Executing Epoch

Figure represents Executing Epoch. Epoch is the number of times that the learning algorithm will work through the entire training dataset.

5.4 PREVIEW OF IMAGE

```
preview of one image using matplotlib

In [21]: %matplotlib inline
import matplotlib.pyplot as plt
import cv2
idx = 248
img = X.loc[idx].values.reshape(28,28)
print(Y[idx])
plt.imshow(img)

2

Out[21]: <matplotlib.image.AxesImage at 0x1ef5a4b1100>
```

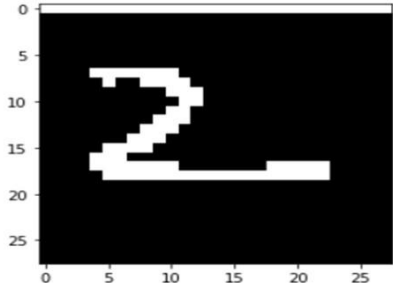


Figure 5.8 : Preview of Image

Figure depicts preview of Image using idx from dataset.

5.5 ACCURACY

```
15 [44] # prompt: write a code to measure the accuracy of the model in percents

y_pred = model.predict(X_val)
y_true = np.argmax(y_val_enc, axis=1)
y_pred_classes = np.argmax(y_pred, axis=1)
accuracy = 100 * np.sum(y_pred_classes == y_true) / len(y_true)
print(f"Accuracy of the model on the validation set: {accuracy:.2f}%")

394/394 [=====] - 9s 22ms/step
Accuracy of the model on the validation set: 98.69%
```

Figure : Accuracy

Figure shows the 98.6 % accuracy of the MNIST Digits Classification using Convolutional Neural Network.

CHAPTER - 6

CONCLUSION & FUTURE SCOPE

In this discussion about the Convolutional Neural Network and its importance. It also covered how a dataset is divided into training and test dataset. Here considered 80% data for training and 20% data for testing. It means that over all 1000 Images from that took 800 for training and 200 images for testing. A dataset was taken to make predictions of MNIST digits from 0 to 9. The dataset was cleaned, scaled, and shaped. Using TensorFlow, a CNN model was created and was eventually trained on the training dataset. Finally, predictions were made using the trained model with an accuracy of 98.6%. In this successfully predicted all digits from 0 to 9.

In the future, plan to observe the variation in overall classification accuracy by varying the number of hidden layers and toggling them with other parameters of the model. The plan to make our model more optimized in this direction also makes our model recognize the rotated 6 and 9 image(s).

Plan to achieve better accuracy results on the colored images also as there can be various boundary conditions that may exist in colored images. Also try to implement our models on letters and a double-digit(s) database so that can extend our implementation and can diversify our model's application. In future look forward to working with Double Digit and Triple Digit Numbers and will plan to predict characters also.

REFERENCES

1. Arif R.B, Siddique A.B, M. M. R. Khan and Z. Ashrafi, "Study and Observation of the Variations of Accuracies for MNIST Digits Recognition with Various Hidden Layers and Epochs using Neural Network Algorithm," in 2018 4th International Conference on Electrical Engineering and Information & Communication Technology (iCEEICT), 2018, pp. 118-123: IEEE.
2. Adwait Dixit , Ashwini Navghane, Yogesh Dandawate, "MNIST Devnagari character Recognition using Wavelet Based Feature Extraction and Classification Scheme", 2014, Annual IEEE India Conference(INDICON).
3. Ashutosh Aggarwal ,Karamjeet Singh, kamalpreet Singh, "Use of Gradient Technique for extracting features from MNIST Gurmukhi Characters and Numerals", 2014, International Conference on Information and Communication Technologies (ICICT 2014).
4. Akanksha Gaur, Sunita Yadav, "MNIST Hindi character Recognition using K-Means Clustering and SVM", 978-1-4799- 5532-9/15/\$31.00 ©2015 IEEE.
5. Archana N.Vyas, Mukesh m.Goswami, "Classification of MNIST Gujarati Numerals", 978-1-4799-8792- 4/15/\$31.00 ©2015 IEEE.
6. Denker J. S Gardner W. R., Graf, H. P., Henderson, D., Howard, R. E., Hubbard, W., Jackal, L. D., Baird, H. S., and Guyon, I. (1989). Neural Network Recognizer for Hand-Written Digits. In Touretzky, D., editor, Neural Information Processing Systems, volume 1, pages 323-331, Denver, 1988.

7. Fabien Lauer, Ching Y. Suen, and Gerard Bloch "A trainable feature extractor for MNIST digit recognition", Journal Pattern Recognition, Elsevier, 40 (6), pp.1816-1824, 2007.
8. Gil Levi and Tal Hassner, "OFFLINE MNIST DIGIT RECOGNITION USING NEURAL NETWORK", International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, vol. 2, no. 9, pp. 4373-4377, 2013
9. Gunjan Singh et al, "Recognition of MNIST Hindi Characters using Backpropagation Neural Network" 2012 / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 3 (4) ,4892-4895
10. Gurpreet Singh, Manoj Sachan, "Multi-Layer Perceptron(MLP) neural Network Technique for Offline MNIST gurmukhi Character Recognition", 978-1-4799-3975-6/14/\$31.00 ©2014 IEEE
11. Haider A. Alwazzy¹, Hayder M. Albehadili², Younes S. Alwan³, Naz E. Islam⁴, "MNIST Digit Recognition using Convolutional Neural Network", International Journal of Innovative Research in Computer and Communication Engineering, vol. 4, no. 2, pp. 1101-1106, 2016.
12. Ishani Patel, Viraj Jagtap and Ompriya Kale. "A Survey on Feature Extraction Methods for MNIST Digits Recognition", International Journal of Computer Applications, vol. 107, no. 12, pp. 11-17, 2014.