**1. INTRODUCTION**

The project work is a practical experience of the knowledge one has. The documentation leads a way to the concept to present the thinking and the upgradation of various techniques into the project. This project entitled “DIGIT RECOGNITION” is a practical project based on some trends of computer science. Every day the world is searching new techniques in the field of computer science to upgrade the human limitations into machines to get more and more accurate and meaningful data. The way of machine learning and artificial intelligence has no negative slope it has only the slope having positive direction. This project is a very basic idea of those concepts. This project deals with the very popular learning process called Neural Network. There are various ways by which one can achieve the goal to a desired output, but in machine learning Neural network gives a way that machine learns the way to reach the output. This project has come through the concepts of statistical modelling, the computer vision and machine learning libraries which includes a lot of study about these concepts. I tried to lead these projects to the end of some updated techniques, upgradation and application of some new algorithms. This project has a good explanation and this project can be enhanced further into some complex applications of machine learning.

Machine learning and deep learning plays an important role in computer technology and artificial intelligence. With the use of deep learning and machine learning, human effort can be reduced in recognizing, learning, predictions and many more areas.

To make machines more intelligent, the developers are diving into machine learning and deep learning techniques. A human learns to perform a task by practicing and repeating it again and again so that it memorizes how to perform the tasks. Then the neurons in his brain automatically trigger and they can quickly perform the task they have learned. Deep learning is also very similar to this. It uses different types of neural network architectures for different types of problems. For example – object recognition, image and sound classification, object detection, image segmentation, etc. The handwritten digit recognition is the ability of computers to recognize human handwritten digits. It is a hard task for the machine because handwritten digits are not perfect and can be made with many different flavours. The handwritten digit recognition is the solution to this problem which uses the image of a digit and recognizes the digit present in the image.

**1.1 PURPOSE**

Digit recognition system is the working of a machine to train itself or recognizing the digits from different sources like emails, bank cheque, papers, images, etc. and in different real-world scenarios for online handwriting recognition on computer tablets or system, recognize number plates of , numeric entries in forms filled up by hand and so on.

**1.2 PROJECT SCOPE**

The system will be used as the application that serves hospitals, clinic, dispensaries or other health institutions. The intention of the system is to increase the number of patients that can be treated and managed properly. If the hospital management system is file based, management of the hospital has to put much effort on securing the files. They can be easily damaged by fire, insects, and natural disasters. Also, could be misplaced by losing data and information.

**1.3 Problem Statement:**

The goal of this project is to create a model that will be able to recognize and determine the handwritten digits from its image by using the concepts of Convolution Neural Network. Though the goal is to create a model which can recognize the digits, it can 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 handwritten recognition system.

**2.WORK DONE IN RELATED AREA**

**2.1 Basic steps in constructing a Machine Learning model:**

**2.1.1 - Data Collection**

• The quantity & quality of your data dictate how accurate our model is

• The outcome of this step is generally a representation of data (Guo simplifies to specifying a table) which we will use for training

• Using pre-collected data, by way of datasets from Kaggle, UCI, etc., still fits into this step

**2.1.2 - Data Preparation**

• Wrangle data and prepare it for training

• Clean that which may require it (remove duplicates, correct errors, deal with missing values, normalization, data type conversions, etc.)

• Randomize data, which erases the effects of the particular order in which we collected and/or otherwise prepared our data

• Visualize data to help detect relevant relationships between variables or class imbalances (bias alert!), or perform other exploratory analysis

• Split into training and evaluation sets

**2.1.3 - Choose a Model**

• Different algorithms are for different tasks; choose the right one

**2.1.4 - Train the Model**

• The goal of training is to answer a question or make a prediction correctly as often as possible

• Linear regression example: algorithm would need to learn values for m (or W) and b (x is input, y is output)

• Each iteration of process is a training step

**2.1.5 - Evaluate the Model**

• Uses some metric or combination of metrics to "measure" objective performance of model

• Test the model against previously unseen data 17

• This unseen data is meant to be somewhat representative of model performance in the real world, but still helps tune the model (as opposed to test data, which does not)

• Good train/eval split? 80/20, 70/30, or similar, depending on domain, data availability, dataset particulars, etc. Parameter Tuning

• This step refers to hyperparameter tuning, which is an "artform" as opposed to a science

• Tune model parameters for improved performance

• Simple model hyperparameters may include: number of training steps, learning rate, initialization values and distribution, etc.

**2.1.7 - Make Predictions**

• Using further (test set) data which have, until this point, been withheld from the model (and for which class labels are known), are used to test the model; a better approximation of how the model will perform in the real world

**2.2 Methodologies for Handwritten Digit Recognition System**

We used MNIST as a primary dataset to train the model, and it consists of 70,000 handwritten raster images from 250 different sources out of which 60,000 are used for training, and the rest are used for training validation. Our proposed method mainly separated into stages, preprocessing, Model Construction, Training & Validation, Model Evaluation & Prediction. Since the loading dataset is necessary for any process, all the steps come after it

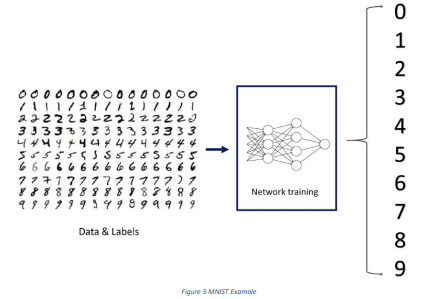
**2.3 Loading The Data Set:**

**2.3.1 MNIST Data Set:**

Modified National Institute of Standards and Technology (MNIST) is a large set of computer vision dataset which is extensively used for training and testing different systems. It was created from the two special datasets of National Institute of Standards and Technology (NIST) which holds binary images of handwritten digits. The training set contains handwritten digits from 250 people, among them 50% training dataset was employees from the Census Bureau and the rest of it was from high school students. However, it is often attributed as the first datasets among other datasets to prove the effectiveness of the neural networks



The database contains 60,000 images used for training as well as few of them can be used for crossvalidation purposes and 10,000 images used for testing. All the digits are grayscale and positioned in a fixed size where the intensity lies at the center of the image with 28×28 pixels. Since all the images are 28×28 pixels, it forms an array which can be flattened into 28\*28=784-dimensional vector. Each component of the vector is a binary value which describes the intensity of the pixel



**2.4 Pre-Processing**

Data pre-processing plays an important role in any recognition process. Data preprocessing is a data mining technique which is used to transform the raw data in a useful and efficient format. To shape the input images in a form suitable for segmentation pre-processing is used. Data preprocessing is a necessary step before building a model with these features. It usually happens in stages.

• Data quality assessment

• Data cleaning

• Data transformation

• Data reduction

**2.4.1 Data quality assessment:**

A Data Quality Assessment is a distinct phase within the data quality life-cycle that is used to verify the source, quantity and impact of any data items that breach pre-defined data quality rules. The Data Quality Assessment can be executed as a one-off process or repeatedly as part of an ongoing data quality assurance initiative.

The quality of your data can quickly decay over time, even with stringent data capture methods cleaning the data as it enters your database. People moving house, changing phone numbers and passing away all mean the data you hold can quickly become out of date.

A Data Quality Assessment helps to identify those records that have become inaccurate, the potential impact that inaccuracy may have caused and the data’s source. Through this assessment, it can be rectified and other potential issues identified.

**2.4.2 Data cleaning**

Data cleaning is one of the important parts of machine learning. It plays a significant part in building a model. It surely isn’t the fanciest part of machine learning and at the same time, there aren’t any hidden tricks or secrets to uncover. However, proper data cleaning can make or break your project. Professional data scientists usually spend a very large portion of their time on this step. Because of the belief that, “Better data beats fancier algorithms”. If we have a well-cleaned dataset, we can get desired results even with a very simple algorithm, which can prove very beneficial at times. Obviously, different types of data will require different types of cleaning. However, this systematic approach can always serve as a good starting point

**2.4.3 Data transformation**

In fact, by cleaning and smoothing the data, we have already performed data modification. However, by data transformation, we understand the methods of turning the data into an appropriate format for the computer to learn from. Data transformation is the process in which data is taken from its raw, siloed and normalized source state and transform it into data that’s joined together, dimensionally modelled, de-normalized, and ready for analysis. Without the right technology stack in place, data transformation can be time-consuming, expensive, and tedious. Nevertheless, transforming the data will ensure maximum data quality which is imperative to gaining accurate analysis, leading to valuable insights that will eventually empower data-driven decisions. Building and training models to process data is a brilliant concept, and more enterprises have adopted, or plan to deploy, machine learning to handle many practical applications. But for models to learn from data to make valuable predictions, the data itself must be organized to ensure its analysis yield valuable insight

**2.4.4 Data reduction:**

Data reduction is a process that reduced the volume of original data and represents it in a much smaller volume. Data reduction techniques ensure the integrity of data while reducing the data. The time required for data reduction should not overshadow the time saved by the data mining on the reduced data set

**Data Reduction Techniques:**

Techniques of data deduction include dimensionality reduction, numerosity reduction and data compression

1. Dimensionality Reduction:

a. Wavelet Transform

b. Principal Component Analysis

c. Attribute Subset Selection

2. Numerosity Reduction:

a. Parametric

b. non-Parametric

3. Data Compression:

When you work with large amounts of data, it becomes harder to come up with reliable solutions. Data reduction can be used to reduce the amount of data and decrease the costs of analysis. After loadingthe data, we separated the data into X and y where X is the image, and y is the label corresponding to X. The first layer/input layer for our model is convolution. Convolution takes each pixel as a neuron, so we need to reshape the images such that each pixel value is in its own space, thus converting a 28x28 matrix of greyscale values into 28x28x1 tensor. With the right dimensions for all the images, we can split the images into train and test for further steps. After loading the data, we separated the data into X and y where X is the image, and y is the label corresponding to X. The first layer/input layer for our model is convolution. Convolution takes each pixel as a neuron, so we need to reshape the images such that each pixel value is in its own space, thus converting a 28x28 matrix of greyscale values into 28x28x1 tensor. With the right dimensions for all the images, we can split the images into train and test for further steps.

**2.5 Data Encoding:**

This is an optional step since we are using the cross-categorical entropy as loss function. We have to specify the network that the given labels are categorical in nature. The raw data can contain various different types of data which can be both structured and unstructured and needs to be processed in order to bring to form that is usable in the Machine Learning models. Since machine learning is based on mathematical equations, it would cause a problem when we keep categorical variables as is. Many algorithms support categorical values without further manipulation, but in those cases, it’s still a topic of discussion on whether to encode the variables or not. After the identification of the data types of the features present in the data set, the next step is to process the data in a way that is suitable to put to Machine Learning models. The three popular techniques of converting Categorical values to Numeric values are done in two different methods.

1. Label Encoding.

2. One Hot Encoding.

3. Binary Encoding.

Encoding variability describes the variation of encoding of individually inside a category. When we talk about the variability in one hot encoding, the variability depends on the time of implementation in which it decides the number of categories to take that do have sufficient impact on the target. Other encoding methodologies do show a significant variability which is identified at the time of validation

**2.5 Model Construction**

**Now**, comes the fun part where we finally get to use the meticulously prepared data for model building. Depending on the data type (qualitative or quantitative) of the target variable (commonly referred to as the Y variable) we are either going to be building a classification (if Y is qualitative) or regression (if Y is quantitative) model.

**2.6.1** **Learning Algorithms:**

Machine learning algorithms could be broadly categorized to one of three types: 1. Supervised learning — In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal). A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. It is a machine learning task that establishes the mathematical relationship between input X and output Y variables. Such X, Y pair constitutes the labeled data that are used for model building in an effort to learn how to predict the output from the input. Supervised learning problems can be further grouped into regression and classification problem

**• Classification**: A classification problem is when the output variable is a category, such as “red” or “blue” or “disease” and “no disease”.

• Regression: A regression problem is when the output variable is a real value, such as “dollars” or “weight”.

**2. Unsupervised learning** — is a machine learning task that makes use of only the input X variables. Such X variables are unlabeled data that the learning algorithm uses in modeling the inherent structure of the data. Unsupervised learning problems can be further grouped into clustering and association problems

**• Clustering**: A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior. 25

**• Association**: An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.

**4. Reinforcement learning** — Reinforcement learning is an area of Machine Learning. It is about taking suitable action to maximize reward in a particular situation. It is employed by various software and machines to find the best possible behavior or path it should take in a specific situation. Reinforcement learning differs from the supervised learning in a way that in supervised learning the training data has the answer key with it so the model is trained with the correct answer itself whereas in reinforcement learning, there is no answer but the reinforcement agent decides what to do to perform the given task. In the absence of a training dataset, it is bound to learn from its experience. It is a machine learning task that decides on the next course of action and it does this by learning through trial and error in an effort to maximize the reward.

• Input: The input should be an initial state from which the model will start

• Output: There are many possible outputs as there are variety of solution to a particular problem

• Training: The training is based upon the input; The model will return a state and the user will decide to reward or punish the model based on its output.

• The model keeps continues to learn.

• The best solution is decided based on the maximum reward.

**2.6.** **MODELS THAT CAN BE USED FOR THE PROJECT:**

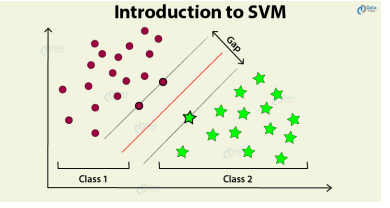
**1.** **SUPPORT VECTOR MACHINE:**

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane. 26 SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. SVM algorithm can be used for Face detection, image classification, text categorization, etc

**SVM can be of two types: o Linear SVM:**

**Linear SVM** is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

**o Non-linear SVM**: Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

**Example**: SVM can be understood with the example that we have used in the KNN classifier. Suppose we see a strange cat that also has some features of dogs, so if we want a model that can accurately identify whether it is a cat or dog, so such a model can be created by using the SVM algorithm. We will first train our model with lots of images of cats and dogs so that it can learn about different features of cats and dogs, and then we test it with this strange creature. So as support vector creates a decision boundary between these two data (cat and dog) and choose extreme cases (support vectors), it will see the extreme case of cat and dog. On the basis of the support vectors, it will classify it as a cat. 

The followings are important concepts in SVM –

**• Support Vectors** − Datapoints that are closest to the hyperplane is called support vectors. Separating line will be defined with the help of these data points.

**• Hyperplane** − As we can see in the above diagram, it is a decision plane or space which is divided between a set of objects having different classes.

**• Margin** − It may be defined as the gap between two lines on the closet data points of different classes. It can be calculated as the perpendicular distance from the line to the support vectors. Large margin is considered as a good margin and small margin is considered as a bad margin.

The main goal of SVM is to divide the datasets into classes to find a maximum marginal hyperplane (MMH) and it can be done in the following two steps –

• First, SVM will generate hyperplanes iteratively that segregates the classes in best way.

• Then, it will choose the hyperplane that separates the classes correctly.

**Pros of SVM classifiers**

• SVM classifiers offers great accuracy and work well with high dimensional space. SVM classifiers basically use a subset of training points hence in result uses very less memory.

**Cons of SVM classifiers**

• They have high training time hence in practice not suitable for large datasets. Another disadvantage is that SVM classifiers do not work well with overlapping classes.

**2.** **K-NN ALGORITHM:**

o K-Nearest Neighbor is one of the simplest Machine Learning algorithms based on Supervised Learning technique.

o K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.

o K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.

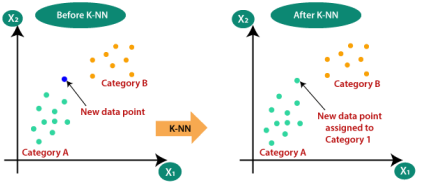
o K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems

o K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data.

o It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.

o KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

o Example: Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category

****Figure 7 KNN

The K-NN working can be explained on the basis of the below algorithm:

o Step-1: Select the number K of the neighbours

o Step-2: Calculate the Euclidean distance of K number of Neighbours

o Step-3: Take the K nearest Neighbours as per the calculated Euclidean distance.

o Step-4: Among these k Neighbours, count the number of the data points in each category.

o Step-5: Assign the new data points to that category for which the number of the neighbor is maximum.

o Step-6: Our model is ready.

**Advantages of KNN Algorithm:**

o It is simple to implement.

o It is robust to the noisy training data.

o It can be more effective if the training data is large.

**Disadvantages of KNN Algorithm:**

o Always needs to determine the value of K which may be complex some time.

o The computation cost is high because of calculating the distance between the data points for all the training samples.

**Steps to implement the K-NN algorithm:**

o Data Pre-processing step

o Fitting the K-NN algorithm to the Training set

o Predicting the test result

o Test accuracy of the result(Creation of Confusion matrix)

o Visualizing the test set result. This is pseudocode for implementing the KNN algorithm from scratch:

1. Load the training data.

2. Prepare data by scaling, missing value treatment, and dimensionality reductioon as required.

3. Find the optimal value for K:

4. Predict a class value for new data:

1. Calculate distance (X, Xi) from i =1, 2, 3,….,n. where X= new data point, Xi= training data, distance as per your chosen distance metric.

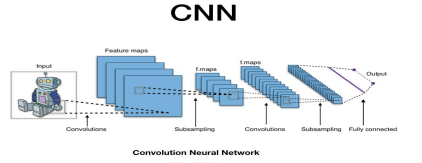
2. Sort these distances in increasing order with corresponding train data.

3. From this sorted list, select the top ‘K’ rows.

• Find the most frequent class from these chosen ‘K’ rows. This will be your predicted class. 30 After data encoding, the images and labels are ready to be fitted into our model. We need to define a Convolutional Neural Network Model

**3.****CONVOLUTION NEURAL NETWORK:**

In simpler words, CNN is an artificial neural network that specializes in picking out or detect patterns and make sense of them. Thus, CNN has been most useful for image classification. A CNN model has various types of filters of different sizes and numbers. These filters are essentially what helps us in detecting the pattern. The convolutional neural network, or CNN for short, is a specialized type of neural network model designed for working with two-dimensional image data, although they can be used with one-dimensional and three-dimensional data. Central to the convolutional neural network is the convolutional layer that gives the network its name. This layer performs an operation called a “convolution“. A CNN model generally consists of convolutional and pooling layers. It works better for data that are represented as grid structures, this is the reason why CNN works well for image classification problems. The dropout layer is used to deactivate some of the neurons and while training, it reduces offer fitting of the model. Our model is composed of feature extraction with convolution and binary classification. Convolution and max pooling are carried out to extract the features in the image, and a 32 3x3 convolution filters are applied to a 28x28 image followed by a max-pooling layer of 2x2 pooling size followed by another convolution layer with 64 3x3 filters

****

In the end, we obtain 7x7 images to flatten. Flatten layer will flatten the 7x7 images into a series of 128 values that will be mapped to a dense layer of 128 neurons that are connected to the categorical output layer of 10 neurons. The filter is smaller than the input data and the type of multiplication applied between a filter-sized patch of the input and the filter is a dot product. A dot product is the element-wise multiplication between the filter-sized patch of the input and filter, which is then summed, always resulting in a single value. Because it results in a single value, the operation is often referred to as the “scalar product” Using a filter smaller than the input is intentional as it allows the same filter (set of weights) to be multiplied by the input array multiple times at different points on the input. Specifically, the filter is applied systematically to each overlapping part or filter-sized patch of the input data, left to right, top to bottom. The output from multiplying the filter with the input array one time is a single value. As the filter is applied multiple times to the input array, the result is a two-dimensional array of output values that represent a filtering of the input. As such, the two-dimensional output array from this operation is called a “feature map”

**Convolution in Computer Vision:**

The idea of applying the convolutional operation to image data is not new or unique to convolutional neural networks; it is a common technique used in computer vision.

Historically, filters were designed by hand by computer vision experts, which were then applied to an image to result in a feature map or output from applying the filter then makes the analysis of the image easier in some way. The network will learn what types of features to extract from the input. Specifically, training under stochastic gradient descent, the network is forced to learn to extract features from the image that minimize the loss for the specific task the network is being trained to solve, e.g. extract features that are the most useful for classifying images as dogs or cats.

Worked Example of Convolutional Layers The Keras deep learning library provides a suite of convolutional layers. We can better understand the convolution operation by looking at some worked examples with contrived data and handcrafted filters

The one-dimensional convolutional layer and a two-dimensional convolutional layer example to both make the convolution operation concrete and provide a worked example of using the Keras layers.

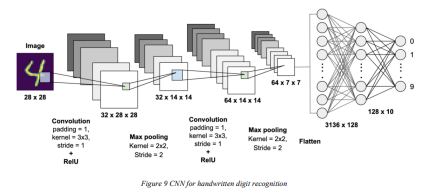
• Convolutional neural networks apply a filter to an input to create a feature map that summarizes the presence of detected features in the input.

• Filters can be handcrafted, such as line detectors, but the innovation of convolutional neural networks is to learn the filters during training in the context of a specific prediction problem.

**WORKING OF CNN:**

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 outputs an activation value.

The behaviour of each neuron is defined by its weights. When fed with the pixel values, the artificial neurons of a CNN pick out various visual features. When you input an image into a ConvNet, each of its layers generates several activation maps. Activation maps highlight the relevant features of the image. Each of the neurons takes a patch of pixels as input, multiplies their colour values by its weights, sums them up, and runs them through the activation function.

****

The first (or bottom) layer of the CNN usually detects basic features such as horizontal, vertical, and diagonal edges. The output of the first layer is fed as input of the next layer, which extracts more complex features, such as corners and combinations of edges. As you move deeper into the convolutional neural network, the layers start detecting higher-level features such as objects, faces, and more.

The operation of multiplying pixel values by weights and summing them is called “convolution” (hence the name convolutional neural network). A CNN is usually composed of several convolution layers, but it also contains other components. The final layer of a CNN is a classification layer, which takes the output of the final convolution layer as input (remember, the higher convolution layers detect complex objects).

Based on the activation map of the final convolution layer, the classification layer outputs a set of confidence scores (values between 0 and 1) that specify how likely the image is to belong to a “class.” For instance, if you have a ConvNet that detects cats, dogs, and horses, the output of the final layer is the possibility that the input image contains any of those animals. After selecting the model the following process is done:

The model type that we will be using is Sequential. Sequential is the easiest way to build a model in Keras. It allows you to build a model layer by layer. We use the ‘add()’ function to add layers to our model. Our first 2 layers are Conv2D layers. These are convolution layers that will deal with our input images, which are seen as 2-dimensional matrices. 64 in the first layer and 32 in the second layer are the number of nodes in each layer. This number can be adjusted to be higher or lower, depending on the size of the dataset. In our case, 64 and 32 work well, so we will stick with this for now. Kernel size is the size of the filter matrix for our convolution. So a kernel size of 3 means we will have a 3x3 filter matrix. Refer back to the introduction and the first image for a refresher on this. Activation is the activation function for the layer. The activation function we will be using for our first 2 layers is the ReLU, or Rectified Linear Activation. This activation function has been proven to work well in neural networks Our first layer also takes in an input shape. This is the shape of each input image, 28,28,1 as seen earlier on, with the 1 signifying that the images are greyscale. In between the Conv2D layers and the dense layer, there is a ‘Flatten’ layer. Flatten serves as a connection between the convolution and dense layers. ‘Dense’ is the layer type we will use in for our output layer. Dense is a standard layer type that is used in many cases for neural networks. We will have 10 nodes in our output layer, one for each possible outcome (0–9). The activation is ‘softmax’. Softmax makes the output sum up to 1 so the output can be interpreted as probabilities. The model will then make its prediction based on which option has the highest probability.

**2.6 .1.Training & Validation**

After the construction of the model the model has to be compiled to train it with the available data set. Optimizers are used to compile the model. Compiling the model takes three parameters: optimizer, loss and metrics. Optimizers are algorithms or methods used to change the attributes of the neural network such as weights and learning rate to reduce the losses. Optimizers are used to solve optimization problems by minimizing the function.

The optimizer controls the learning rate. We will be using ‘adam’ as our optmizer. Adam is generally a good optimizer to use for many cases. The adam optimizer adjusts the learning rate throughout training. The learning rate determines how fast the optimal weights for the model are calculated. A smaller learning rate may lead to more accurate weights (up to a certain point), but the time it takes to compute the weights will be longer.

We will use ‘categorical\_crossentropy’ for our loss function. This is the most common choice for classification. A lower score indicates that the model is performing better. To make things even easier to interpret, we will use the ‘accuracy’ metric to see the accuracy score on the validation set when we train the model. The idea behind training and testing any data model is to achieve maximum learning rate and maximum validation. Better Learning rate and better validation can be achieved by increasing the train and test data respectively.

**2.7 Model Evaluation & Prediction:**

For real-world image classification prediction, we need to do a little image pre-processing on the real-world images as model training was done with greyscale raster images. The steps of image pre-processing are :

1. Loading image

2. Convert the image to greyscale

3. Resize the image to 28x28

4. Converting the image into a matrix form

5. Reshape the matrix into 28x28x1

After preprocessing, we predict the label of the image by passing the pre-processed image through the neural network. The output we get is a list of 10 activation values 0 to 9, respectively. The position having the highest value is the predicted label for the image.

These structures are called as Neural Networks. It teaches the computer to do what naturally comes to humans. Deep learning, there are several types of models such as the Artificial Neural Networks (ANN), Autoencoders, Recurrent Neural Networks (RNN) and Reinforcement Learning. But there has been one particular model that has contributed a lot in the field of computer vision and image analysis which is the Convolutional Neural Networks (CNN) or the ConvNet. 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.

Methods for evaluating a model’s performance are divided into 2 categories: namely, holdout and Crossvalidation. Both methods use a test set (i.e data not seen by the model) to evaluate model performance. It’s not recommended to use the data we used to build the model to evaluate it. This is because our model will simply remember the whole training set, and will therefore always predict the correct label for any point in the training set. This is known as overfitting.

**Holdout:**

The purpose of holdout evaluation is to test a model on different data than it was trained on. This provides an unbiased estimate of learning performance. In this method, the dataset is randomly divided into three subsets

1. Training set is a subset of the dataset used to build predictive models.

2. Validation set is a subset of the dataset used to assess the performance of the model built in the training phase. It provides a test platform for fine-tuning a model’s parameters and selecting the best performing model. Not all modeling algorithms need a validation set.

3. Test set, or unseen data, is a subset of the dataset used to assess the likely future performance of a model. If a model fits to the training set much better than it fits the test set, overfitting is probably the cause.

The holdout approach is useful because of its speed, simplicity, and flexibility. However, this technique is often associated with high variability since differences in the training and test dataset can result in meaningful differences in the estimate of accuracy.

**Cross-Validation:**

Cross-validation is a technique that involves partitioning the original observation dataset into a training set, used to train the model, and an independent set used to evaluate the analysis. The most common cross-validation technique is k-fold cross-validation, where the original dataset is partitioned into k equal size subsamples, called folds. The k is a user-specified number, usually with 5 or 10 as its preferred value. This is repeated k times, such that each time, one of the k subsets is used as the test set/validation set and the other k-1 subsets are put together to form a training set. The error estimation is averaged over all k trials to get the total effectiveness of our model.

**3. CNN ARCHITECTURE**

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. The term ‘Convolution” in CNN denotes the mathematical function of convolution which is a special kind of linear operation wherein two functions are multiplied to produce a third function which expresses how the shape of one function is modified by the other. In simple terms, two images which can be represented as matrices are multiplied to give an output that is used to extract features from the image. Technically, deep learning CNN models to train and test, each input image will pass it through a series of convolution layers with filters (Kernals), Pooling, fully connected layers (FC) and apply Softmax function to classify an object with probabilistic values between 0 and 1. The below figure is a complete flow of CNN to process an input image and classifies the objects based on values.

**3.1 Basic Architecture**

There are two main parts to a CNN architecture • A convolution tool that separates and identifies the various features of the image for analysis in a process called as Feature Extraction • A fully connected layer that utilizes the output from the convolution process and predicts the class of the image based on the features extracted in previous stages.

**3.2 CNN Layers:**

The multiple occurring of these layers shows how deep our network is, and this formation is known as the deep neural network.

● Input: raw pixel values are provided as input.

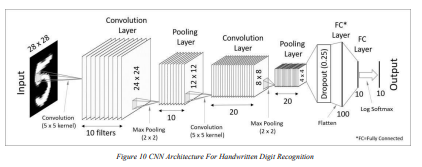
● Convolutional layer: Input layers translates the results of the neuron layer. There is a need to specify the filter to be used. Each filter can only be a 5\*5 window that slides over input data and gets pixels with maximum intensities

. ● Rectified linear unit [ReLU] layer: provided activation function on the data taken as an image. In the case of back propagation, ReLU function is used which prevents the values of pixels from changing.

● Pooling layer: Performs a down-sampling operation in volume along the dimensions (width, height). 40

● Fully connected layer: score class is focused, and a maximum score of the input digits is found.

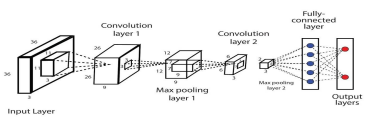
As we go deeper and deeper in the layers, the complexity is increased a lot. But it might be worth going as accuracy may increase but unfortunately, time consumption also increases.



**1. Convolutional Layer**

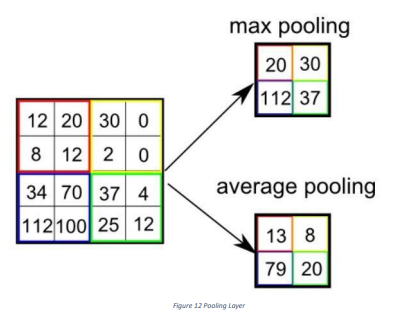
This layer is the first layer that is used to extract the various features from the input images. In this layer, the mathematical operation of convolution is performed between the input image and a filter of a particular size MxM. By sliding the filter over the input image, the dot product is taken between the filter and the parts of the input image with respect to the size of the filter (MxM)

The output is termed as the Feature map which gives us information about the image such as the corners and edges. Later, this feature map is fed to other layers to learn several other features of the input image



**2. Pooling Layer**

In most cases, a Convolutional Layer is followed by a Pooling Layer. The primary aim of this layer is to decrease the size of the convolved feature map to reduce the computational costs. This is performed by decreasing the connections between layers and independently operates on each feature map. Depending upon method used, there are several types of Pooling operations. In Max Pooling, the largest element is taken from feature map. Average Pooling calculates the average of the elements in a predefined sized Image section. The total sum of the elements in the predefined section is computed in Sum Pooling. The Pooling Layer usually serves as a bridge between the Convolutional Layer and the FC Layer



**3. Fully Connected Layer**

The Fully Connected (FC) layer consists of the weights and biases along with the neurons and is used to connect the neurons between two different layers. These layers are usually placed before the output layer and form the last few layers of a CNN Architecture.

**4.** **Activation Functions**

An activation function in a neural network defines how the weighted sum of the input is transformed into an output from a node or nodes in a layer of the network. Sometimes the activation function is called a “transfer function.” If the output range of the activation function is limited, then it may be called a “squashing function.” Many activation functions are nonlinear and may be referred to as the “nonlinearity” in the layer or the network design. The choice of activation function has a large impact on the capability and performance of the neural network, and different activation functions may be used in different parts of the model. Technically, the activation function is used within or after the internal processing of each node in the network, although networks are designed to use the same activation function for all nodes in a layer. A network may have three types of layers: input layers that take raw input from the domain, hidden layers that take input from another layer and pass output to another layer, and output layers that make a prediction. All hidden layers typically use the same activation function. The output layer will typically use a different activation function from the hidden layers and is dependent upon the type of prediction required by the model. Activation functions are also typically differentiable, meaning the first-order derivative can be calculated for a given input value. This is required given that neural networks are typically trained using the backpropagation of error algorithm that requires the derivative of prediction error in order to update the weights of the model. There are many different types of activation functions used in neural networks, although perhaps only a small number of functions used in practice for hidden and output layers. Finally, one of the most important parameters of the CNN model is the activation function. They are used to learn and approximate any kind of continuous and complex relationship between variables of the network. In simple words, it decides which information of the model should fire in the forward direction and which ones should not at the end of the network. It adds non-linearity to the network. There are several commonly used activation functions such as the ReLU, Softmax, tanH and the Sigmoid functions. Each of these functions have a specific usage. For a binary classification CNN model, sigmoid and softmax functions are preferred an for a multi-class classification, generally softmax us used.

• A convolution tool that separates and identifies the various features of the image for analysis in a process called as Feature Extraction

• A fully connected layer that utilizes the output from the convolution process and predicts the class of the image based on the features extracted in previous stages.

**Epochs Running:**

7/7 [==============================] - 0s 50ms/step - loss: 0.6243 - accuracy: 0.8138 - val\_loss: 0.4445 - val\_accuracy: 0.9050

Epoch 2995/3000

7/7 [==============================] - 0s 59ms/step - loss: 0.6753 - accuracy: 0.8238 - val\_loss: 0.4442 - val\_accuracy: 0.9050

Epoch 2996/3000

7/7 [==============================] - 0s 64ms/step - loss: 0.6569 - accuracy: 0.8275 - val\_loss: 0.4439 - val\_accuracy: 0.9050

Epoch 2997/3000

7/7 [==============================] - 0s 61ms/step - loss: 0.6449 - accuracy: 0.8338 - val\_loss: 0.4436 - val\_accuracy: 0.9050

Epoch 2998/3000

7/7 [==============================] - 0s 61ms/step - loss: 0.6386 - accuracy: 0.8213 - val\_loss: 0.4434 - val\_accuracy: 0.9050

Epoch 2999/3000

7/7 [==============================] - 0s 61ms/step - loss: 0.6547 - accuracy: 0.8175 - val\_loss: 0.4431 - val\_accuracy: 0.9050

Epoch 3000/3000

7/7 [==============================] - 0s 59ms/step - loss: 0.6729 - accuracy: 0.8138 - val\_loss: 0.4428 - val\_accuracy: 0.9050

The model has successfully trained

7/7 [==============================] - 0s 9ms/step - loss: 0.4428 - accuracy: 0.9050

Test loss: 0.44283050298690796

Test accuracy: 0.9049999713897705

Saving the model as mnist.h5

**3. SYSTEM ANALYSIS**

**3.1 HARDWARE REQUIREMENTS**

* Processor : Snapdragon Helio 2200 or Later
* Main Memory (RAM) :8GB
* Cache Memory :1MB
* Monitor :50-inch Color Monitor
* Keyboard : Mechanical Keyboard
* Mouse : Optical Mouse
* Hard Disk :2048 GB

**3.2 SOTWARE REQUIREMENT**

**Import the libraries**:

Libraries required are Keras, Tensor flow, NumPy, Pillow, Tkinkter.

**3.2.1 python**

I python is something different. It provides a rich architecture for interacting computing with: A powerful shell A kernel for jupyter Support for data visualization like matrix values flexible interpreters for easy use. Python is an interpreted, high-level, general purpose programming language created by Guido Van Rossum and first released in 1991, Python's design philosophy emphasizes code Readability with its notable use of significant White space. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed and garbage collected. It supports multiple programming paradigms, including procedural, objectoriented, and functional programming.

**3.2.2 *OpenCV***

The OpenCV (Open-Source Computer Vision Library) is a python DIL library which is very updated in the work of Image processing. One image file and pixel values can easily come into surface by this library. This library provides a common infrastructure and module related to computer vision technologies. The most important thing about this tool is it is totally free and can be easily modified and changed respective to input by the programmer

**3.2.3 *NumPy***

NumPy is a library for the python programming language, adding support for large multidimensional array and metrices. This package contains a large collection of high-level mathematical functions to operate on those arrays. NumPy is open source and has many contributors. NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, Fourier transform, and matrices. Numpy which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. This tutorial explains the basics of NumPy such as its architecture and environment. It also discusses the various array functions, types of indexing, etc

**3.2.4 Keras**

Keras is a powerful and easy-to-use free open source Python library for developing and evaluating deep learning models. 18 It wraps the efficient numerical computation libraries Theano and TensorFlow and allows you to define and train neural network models in just a few lines of code. It uses libraries such as Python, C#, C++ or standalone machine learning toolkits. Theano and TensorFlow are very powerful libraries but difficult to understand for creating neural networks. Keras is based on minimal structure that provides a clean and easy way to create deep learning models based on TensorFlow or Theano. Keras is designed to quickly define deep learning models. Well, Keras is an optimal choice for deep learning applications.

**3.2.5 TensorFlow:** is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow. TensorFlow tutorial is designed for both beginners and professionals. Our tutorial provides all the basic and advanced concept of machine learning and deep learning concept such as deep neural network, image processing and sentiment analysis. TensorFlow is one of the famous deep learning frameworks, developed by Google Team. It is a free and open source software library and designed in Python programming language, this tutorial is designed in such a way that we can easily implements deep learning project on TensorFlow in an easy and efficient way.

**3.2.6. Pandas**

**pandas**is a Python package that provides fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-**level building** block for doing practical**, real world**data analysis in Python. Additionally, it has the broader goal of **becoming the most powerful and flexible open source data analysis / manipulation tool available in any language.** It is already well on its way towards this goal.

**3.2.7. matplotlib**

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

Create publication quality plots.

Make interactive figures that can zoom, pan, update.

Customize visual style and layout.

**3.2.8. Scikit-learn**

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

**3.2.9. tkinter**

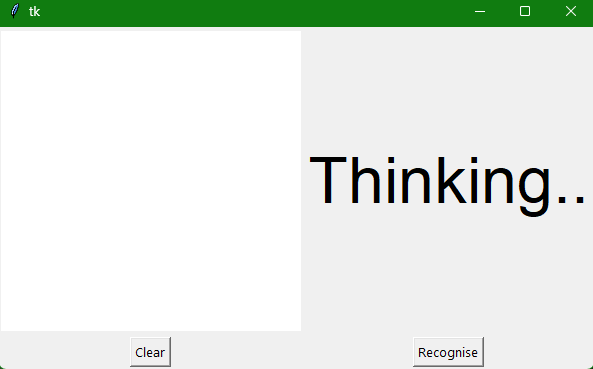
Tkinter is the inbuilt python module that is used to create GUI applications. It is one of the most commonly used modules for creating GUI applications in Python as it is simple and easy to work with. You don’t need to worry about the installation of the Tkinter module separately as it comes with Python already. It gives an object-oriented interface to the Tk GUI toolkit.

**4.SYSTEM DESIGN AND SPECIFICATIONS**

**4.1 FLOW CHART**

****

**4.2SCREEN SHOT DIAGRAM**

****

**5.CODING**

**Training The Model**

import pandas as pd

from sklearn.utils import shuffle

from sklearn.model\_selection import train\_test\_split

import numpy as np

import keras

from keras.datasets import mnist

from keras.models import Sequential

from keras.layers import Dense, Dropout, Flatten

from keras.layers import Conv2D, MaxPooling2D

from keras import backend as K

import matplotlib.pyplot as plt

data =pd.read\_csv('dataset.csv')

data=shuffle(data)

X = data.drop(["label"],axis=1)

Y= data["label"]

train\_x,test\_x,train\_y,test\_y = train\_test\_split(X,Y, test\_size = 0.2)

train\_x = np.array(train\_x)

test\_x = np.array(test\_x)

train\_y = np.array(train\_y)

test\_y = np.array(test\_y)

x\_train =train\_x.reshape(train\_x.shape[0], 28, 28, 1)

x\_test = test\_x.reshape(test\_x.shape[0], 28, 28, 1)

input\_shape = (28, 28, 1)

# convert class vectors to binary class matrices

y\_train = keras.utils.to\_categorical(train\_y, 10)

y\_test = keras.utils.to\_categorical(test\_y, 10)

x\_train = x\_train.astype('float32')

x\_test = x\_test.astype('float32')

print('x\_train shape:', x\_train.shape)

print(x\_train.shape[0], 'train samples')

print(x\_test.shape[0], 'test samples')

batch\_size = 128

num\_classes = 10

epochs = 3000

model = Sequential()

model.add(Conv2D(32, kernel\_size=(3, 3),activation='relu',input\_shape=input\_shape))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Conv2D(64, (3, 3), activation='relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Flatten())

model.add(Dense(128, activation='relu'))

#model.add(Dropout(0.3))

model.add(Dense(64, activation='relu'))

model.add(Dropout(0.3))

model.add(Dense(num\_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical\_crossentropy,optimizer=keras.optimizers.Adadelta(),metrics=['accuracy'])

hist = model.fit(x\_train, y\_train,batch\_size=batch\_size,epochs=epochs,verbose=1,validation\_data=(x\_test, y\_test))

print("The model has successfully trained")

score = model.evaluate(x\_test, y\_test, verbose=1)

print('Test loss:', score[0])

print('Test accuracy:', score[1])

model.save('mnist.h5')

print("Saving the model as mnist.h5")

**Predicting**

from keras.models import load\_model

from tkinter import \*

import tkinter as tk

import win32gui

from PIL import ImageGrab, Image

import numpy as np

import cv2

model = load\_model('mnist.h5')

def predict\_digit(img):

img = img.resize((28,28))

img = np.array(img)

img = cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

#img =cv2.GaussianBlur(im\_gray, (15,15), 0)

#Threshold the image

#ret, im\_th = cv2.threshold(im\_gray,100, 255, cv2.THRESH\_BINARY)

#reshaping to support our model input and normalizing

img = img.reshape(1,28,28,1)

img = img/255.0

img = (1 - img)

cv2.imshow("i", img[0])

#predicting the class

res = model.predict([img])[0]

return np.argmax(res), max(res)

class App(tk.Tk):

def \_\_init\_\_(self):

tk.Tk.\_\_init\_\_(self)

self.x = self.y = 0

# Creating elements

self.canvas = tk.Canvas(self, width=300, height=300, bg = "white", cursor="cross")

self.label = tk.Label(self, text="Thinking..", font=("Helvetica", 48))

self.classify\_btn = tk.Button(self, text = "Recognise", command = self.classify\_handwriting)

self.button\_clear = tk.Button(self, text = "Clear", command = self.clear\_all)

# Grid structure

self.canvas.grid(row=0, column=0, pady=2, sticky=W, )

self.label.grid(row=0, column=1,pady=2, padx=2)

self.classify\_btn.grid(row=1, column=1, pady=2, padx=2)

self.button\_clear.grid(row=1, column=0, pady=2)

#self.canvas.bind("<Motion>", self.start\_pos)

self.canvas.bind("<B1-Motion>", self.draw\_lines)

def clear\_all(self):

self.canvas.delete("all")

def classify\_handwriting(self):

HWND = self.canvas.winfo\_id() # get the handle of the canvas

rect = win32gui.GetWindowRect(HWND) # get the coordinate of the canvas

im = ImageGrab.grab(rect)

digit, acc = predict\_digit(im)

self.label.configure(text= str(digit)+', '+ str(int(acc\*100))+'%')

def draw\_lines(self, event):

self.x = event.x

self.y = event.y

r=15

self.canvas.create\_rectangle(self.x-r, self.y-r, self.x + r, self.y + r, fill='black')

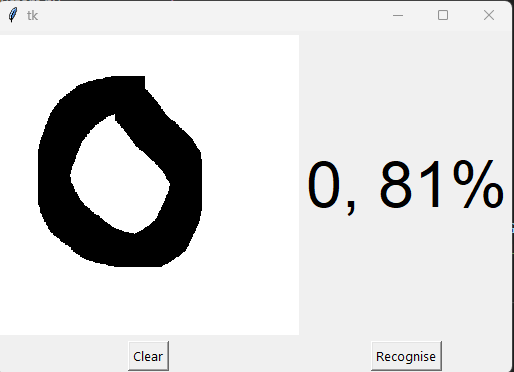
app = App()

mainloop()

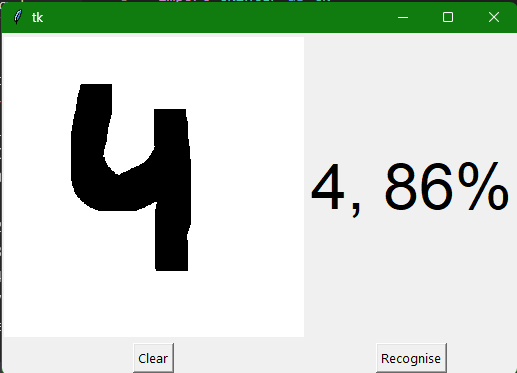
**6.TESTING**

**Input/output**

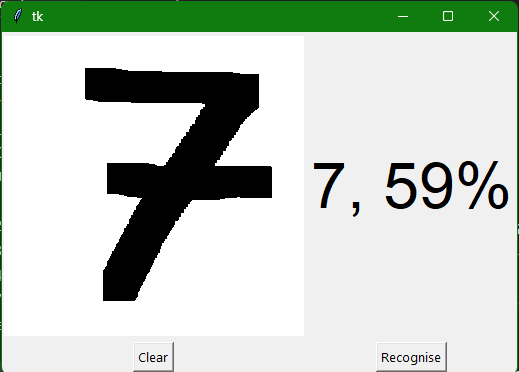
Given input: 0

****

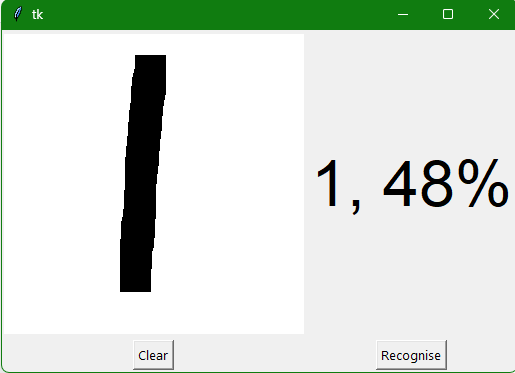
Given input:4

****

Given input:7

****

Given input:1

****

**7.CONCLUSION**

Our project HANDWRITTEN DIGIT RECOGNITION deals with identifying the digits. The main purpose of this project is to build an automatic handwritten digit recognition method for the recognition of handwritten digit strings. In this project, different machine learning methods, which are SVM (Support Vector Machine), ANN (Artificial Neural Networks), and CNN (Convolutional Neural Networks) architectures are used to achieve high performance on the digit string recognition problem.

In this project, the Handwritten Digit Recognizer has been implemented and is able to recognize the number digits of different handwriting flavors. The Convolutional Neural Network or CNN is one of the most widely used machine learning algorithms which has been trained and tested on the given dataset in order to compare and analyse.

**8.REFERENCE**

<https://www.youtube.com/watch?v=XUTkyHeMZPQ&t=853s>

<https://youtu.be/aqaD1wdeNeE>