"Handwriting Recognition"

A
Project Report
Submitted towards the fulfillment for the Degree of

Bachelors of Technology
in
Information Technology

by

Mr. Shantanu Bakare (Reg No. 2014BIT031)
Mr.Anurag Gawai (Reg No. 2014BIT047)
Mr.Mayank Singh(Reg No. 2014BIT014)

Under the Valuable Guidance of
Mr. B.S.Shetty
(Assistant Professor, Department of Information Technology)

Department of Information Technology

Shri Guru Gobind Singhji Institute of Engineering and Technology, Nanded-431606
Maharashtra, India.

May-2018

'This page left intentionally blank'

Certificate

This is to certify that the Project report entitled

"Handwriting Recogniton"

has been successfully completed by

Mr. Shantanu Bakare(Reg No. 2014BIT031)
Mr.Anurag Gawai(Reg No. 2014BIT047)
Mr.Mayank Singh(Reg No. 2014BIT014)

In partial fulfillment for awarding degree of
Bachelors of Technology
In Information Technology
Under Shri Guru Gobind Singhji Institute of Engineering and Technology, Nanded, Maharashtra, India.

Prof. B.S. Shetty
(Guide, Dept. of IT)

Dr. M. V. Vaidya
(Head, Dept. of IT)

Date: 29/10 /2018
Place: Nanded

Department of Information Technology
Shri Guru Gobind Singhji Institute of Engineering and Technology, Nanded-431606
Maharashtra, India. www.sggs.ac.in
2017-2018

"Handwriting Recognition"

A

Project Report

Submitted towards the fulfillment for the Degree of

Bachelors of Technology

in

Information Technology

by

Mr. Shantanu Bakare (Reg No. 2014BIT031)
Mr. Anurag Gawai (Reg No. 2014BIT047)
Mr.Mayank Singh (Reg No. 2014BIT014)

Under the Valuable Guidance of

Mr. B.S.Shetty

(Assistant Professor, Department of Information Technology)

Department of Information Technology
Shri Guru Gobind Singhji Institute of Engineering and Technology, Nanded-431606
Maharashtra, India.

May-2018

PROJECT APPROVAL SHEET

The Dissertation entitled *"Handwriting Recognition"* by *Mr.Shantanu Bakare* (Reg.No.2014BIT031), Mr.Anurag Gawai (Reg.No.2014BIT047), Mr.Mayank Singh(Reg.No.2014BIT014) is approved for the degree of Bachelors of Technology in Information Technology from Shri Guru Gobind Singhji Institute of Engineering and Technology, Nanded, Maharashtra.

Guide:

.....

Asst. Prof. B.S. Shetty
Department of Information Technology,
SGGSIE&T, Nanded.

Examiner(s):

1. **Sign** ……..

   Name ……

2. **Sign** …….

   Name ……

Date: 29/10/2018

Place: Nanded

DECLARATION

I hereby declare that, the dissertation entitled *"Handwriting Recognition"*, which is being submitted for the award of Bachelors of Technology in Information Technology to Shri Guru Singhji Institute of Engineering and Technology, Nanded, Maharashtra is my original and independent work.

.....

1. Mr. Shantanu Bakare

Reg. No. 2014BIT031

Department of Information Technology,

SGGSIE&T, Nanded.

…….

2. Mr. Anurag Gawai

Reg. No. 2014BIT047

Department of Information Technology,

SGGSIE&T, Nanded.

…….

3. Mr. Mayank Singh

Reg. No. 2014BIT014

Department of Information Technology,

SGGSIE&T, Nanded.

ACKNOWLEDGMENT

I am pleased to present this project report entitled ***"Handwriting Recognition"*** to my institute as part of academic activity. I would like to express my deep sense of gratitude to my guide ***Mr. B. S Shetty,*** Assistant Professor, Information Technology. I would like to thank Head of the Department***,*** ***Dr. M. V. Vaidya*** for their valuable guidance and kind co-operation throughout the project work. I feel honored to present my work under their guidance.

Also, I am very thankful to all my B. Tech. (IT) colleagues for their support and help. I am very much thankful to all my seniors, juniors and friends. Thanks a lot to all.

Last but not the least I cannot forget the support and encouragement received from my family, without which this work would not be possible.

Date: 29/10/2018

Place: Nanded

.....

Mr. Shantanu Bakare

(Reg. No. 2014BIT031)

.....

Mr. Anurag Gawai

(Reg. No. 2014BIT047)

.....

Mr. Mayank Singh

(Reg. No. 2014BIT014)

ABSTRACT

Handwriting recognition is one of area pattern recognition. The purpose of pattern recognition is to categorize or classify data or object to one of the classes or categories. Handwriting recognition is defined as the task of transforming a language represented in its spatial form of graphical marks into its symbolic representation. Each script has a set of icons, which are known as characters or letters, which have certain basic shapes. The goal is to identify input characters or image correctly then analyze through automated process system. The development of handwriting is more sophisticated, which is found various kinds of handwritten character such as digit, numeral, cursive script, symbols. The automatic recognition of handwritten text can be extremely useful in many applications where it is necessary to process large volumes of handwritten data, such as recognition of addresses and postcodes on envelopes, interpretation of amounts on bank checks, document analysis, and verification of signatures. Therefore, computer is needed to be able to read document or data for ease of document processing.

In this project we are using three different technology for recognizing the character using a data set. Main Recognition model used is the CNN model, which is more focused in the project. We are using Tensorflow to build CNN. Analysis is performed based on the output of all three programs based on their accuracy and Time of execution. Conceptual discrimination is also stated for all the concepts used in the Project along with referred images and the algorithm used in the program. We have programmed all the algorithms in Python language which can run on any of the python IDE. The same program is tested by Training MNIST data set of digits, with the ability to learn alphabetical or different characters. Though this program is suitable for any character data sets as defined above.

Table of Contents

List of Figures

List of Abbreviations

ANN	Artificial Neural Network
AI	Artificial Intelligence
CNN	Convolutional Neural Network
CONVO	Convolutional Layer
DTC	Decision Tree Classifier
LC	Linear Classifier
ReLU	Rectified Linear unit

1. INTRODUCTION

1.1. Introduction

Handwritten character recognition is a field of research in artificial intelligence, computer vision, and pattern recognition. A computer performing handwriting recognition is said to be able to acquire and detect characters in paper documents, pictures, touch-screen devices and other sources and convert them into machine-encoded form. Its application is found in optical character recognition and more advanced intelligent character recognition systems. Most of these systems nowadays implement machine learning mechanisms such as neural networks

Machine learning is a branch of artificial intelligence inspired by psychology and biology that deals with learning from a set of data and can be applied to solve wide spectrum of problems. A supervised machine learning model is given instances of data specific to a problem domain and an answer that solves the problem for each instance. When learning is complete, the model is able not only to provide answers to the data it has learned on, but also to yet unseen data with high precision. Neural networks are learning models used in machine learning. Their aim is to simulate the learning process that occurs in an animal or human neural system. Being one of the most powerful learning models, they are useful in automation of tasks where the decision of a human being takes too long, or is imprecise. A neural network can be very fast at delivering results and may detect connections between seen instances of data that human cannot see.

1.2. Problem statement

Our main purpose is to develop a model that will recognize handwritten characters. We Aim to achieve approx. 100 percent accuracy. Once developed, the model can be used to convert printed pages to digital content, Help blind people by converting handwritten text to audio and for forensic purposes.

2. OVERVIEW

In order to reach the analysis of the used learning model and the specification and implementation of the algorithms. we had to review existing approaches to the problem of character recognition. In this chapter, we describe the project and compare the methods that have been considered as candidates for this work.

2.1 Project Description

The MNIST problem is a dataset developed by Yann LeCun, Corinna Cortes and Christopher Burges for evaluating machine learning models on the handwritten digit classification problem. The dataset was constructed from a number of scanned document dataset available from the National Institute of Standards and Technology (NIST). This is where the name for the dataset comes from, as the Modified NIST or MNIST dataset. Images of digits were taken from a variety of scanned documents, normalized in size and centered. This makes it an excellent dataset for evaluating models, allowing the developer to focus on the machine learning with very little data cleaning or preparation required. Each image is a 28 by 28 pixel square (784 pixels total). A standard spit of the dataset is used to evaluate and compare models, where 60,000 images are used to train a model and a separate set of 10,000 images are used to test it. It is a digit recognition task. As such there are 10 digits (0 to 9) or 10 classes to predict. Results are reported using prediction error, which is nothing more than the inverted classification accuracy. Excellent results achieve a prediction error of less than 1%. State-of-the-art prediction error of approximately 0.2% can be achieved with large Convolutional Neural Networks.

2.2. Models Used

The Models used for character recognition in this project can be divided in three parts: Linear Classifier, Decision Tree Classifier, Convolutional Neural Network Model. Which are mainly used for training and testing the dataset by different algorithms. This three different models gives

us different accuracy and prediction error, Main moto of the project is to get robust python code with higher accuracy and this can be achieved by using the CNN model.

2.2.1. Linear Classifier

A linear classifier does classification decision based on the value of a linear combination of the characteristics. Imagine that the linear classifier will merge into it's weights all the characteristics that define a particular class.

f(x□,W□,b□)=∑j(Wjxj)+b [x: input vector; W: Weight matrix; b: Bias vector]

The weight matrix will have one row for every class that needs to be classified, and one column for ever element(feature) of x.On the picture above each line will be represented by a row in our weight matrix

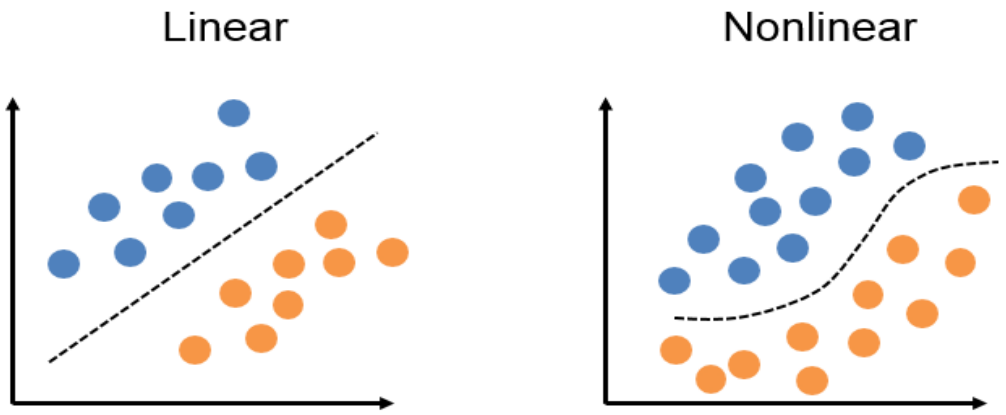


Fig. Linear Classifier

2.2.2. Decision Tree Classifier

Decision tree builds classification or regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node (e.g., Outlook) has two or more branches (e.g., Sunny, Overcast and Rainy). Leaf

node (e.g., Play) represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data.

Predictors | Target

Outlook	Temp	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

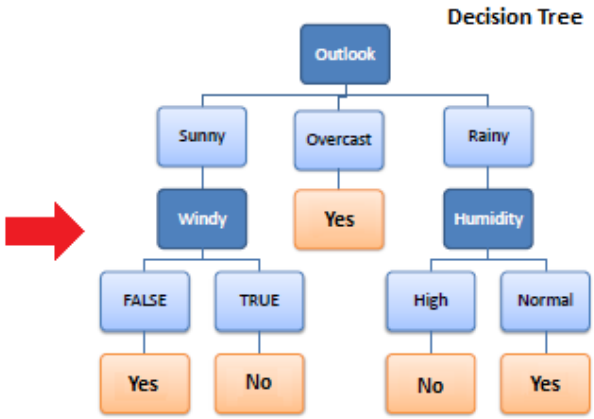


Fig. Decision Tree Classifier

2.2.3. CNN

In machine learning, a network (CNN, or ConvNet) is a class of deep, feed-forward artificial neural networks, most commonly applied to analyzing visual imagery. CNNs use a variation of multilayer perceptrons designed to require minimal preprocessing. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics.

Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.

CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage.

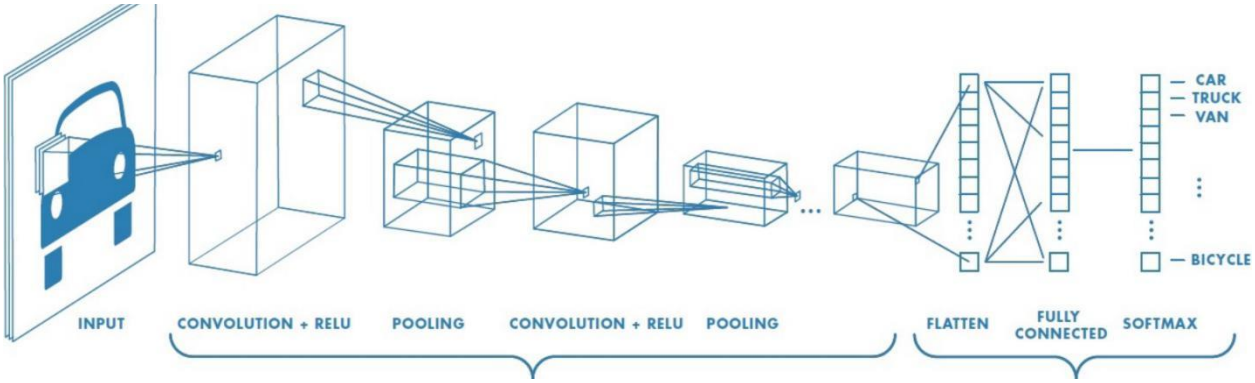


Fig. CNN

2.3 Artificial Neural Network

A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. Neural networks can adapt to changing input so the network generates the best possible result without needing to redesign the output criteria. The conception of neural networks is swiftly gaining popularity in the area of trading system development.

2.3.1. Types Of Artificial Neural Network

There are several kinds of artificial neural networks. These type of networks are implemented based on the mathematical operations and a set of parameters required to determine the output. Lets look at some of the neural networks:

1. Feed forward Neural Network
2. Radial basis function Neural Network
3. Kohonen Self Organizing Neural Network
4. Recurrent Neural Network
5. Convolutional Neural Network
6. Modular Neural Network

Here we are going to study only the 5th i.e Convolutional Neural Network, Since the main focus of the project is to use the CNN's Algorithms.

3. MACHINE LEARNING

Machine learning is an application of Artificial Intelligence(AI) that provides system the ability to automatically learn and improve from the experience without explicitly programmed. Machine learning focuses on the developing of computer programs that can access data and use it learn from themselves. Primary aim of machine learning is to allow the computer learn automatically without human intervention or assistance and adjust action accordingly.

Machine learning enables analysis of massive quantity of data, while it generally delivers faster, more accurate results in order to identify profitable opportunities or dangerous risk. It may also require additional time and resources to train it properly. Combining machine learning with AI and cognitive technology can make it even more effective in processing large volume of information.

Methods of machine learning are as below:

3.1. Supervised Machine Learning

These algorithm can apply what has been learned in the past to new data using labeled examples to predict future events. Starting from the analysis of a known training dataset, the learning algorithm produces an inferred function to make prediction about the output values.

System provide targets for any new input after sufficient training. This algorithm also compares its output with the correct intended output and find errors to modify the model accordingly.

3.2. Unsupervised Machine Learning:

These algorithms are used when the information used to train train is neither classified nor labeled. Unsupervised learning studies from system, how it can infer a function to describe a hidden structure from unlabeled data. The system doesn't figure out the right output, but it explores the data and can draw inferences from dataset to describe hidden structures from unlabeled data.

3.3. Semi Supervised Machine learning

Some where in between supervised and unsupervised they are both labeled and unlabeled data for training. Its analysis typically a small amount of the labeled data and large amount of unlabeled data. System using this method are able to improve learning accuracy. Usually semi

supervised algorithm are used when the acquired labeled data requires skilled and relucent resources in order to train it. Acquiring unlabeled data doesn't acquire additional resources.

3.4. Reinforcement Machine learning

Learning algorithm that interacts with its environment by producing actions and discovers error or rewards. Trial and error search and delayed rewards are the most relevant characteristics of Reinforcement learning. Method allows machine and software agents to automatically determine the ideal behaviour within a specific content to maximize the performance.

Simple reward feedback is required for the agent to learn which action is best, I.e the Reinforcement Signal.

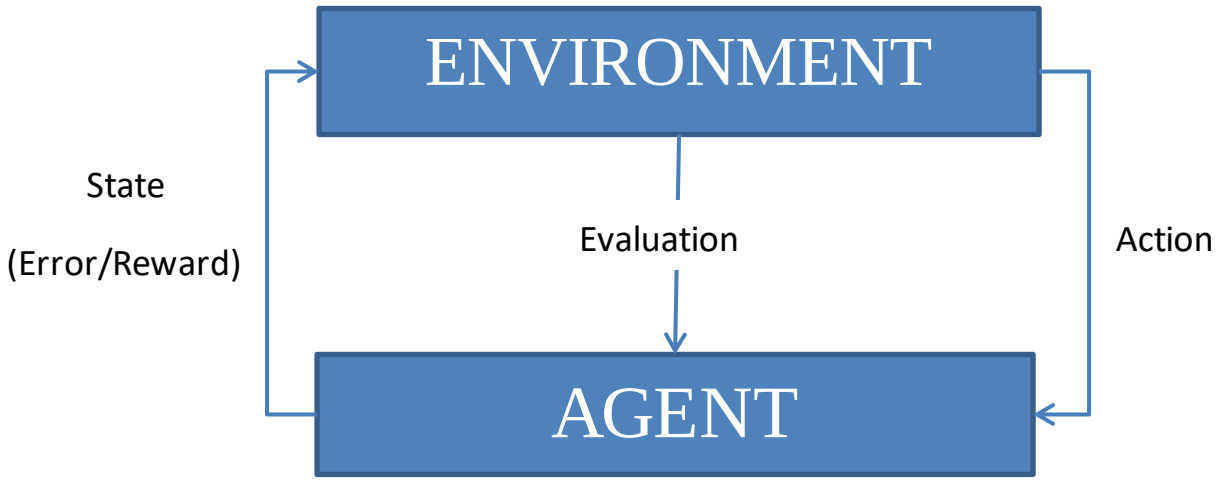
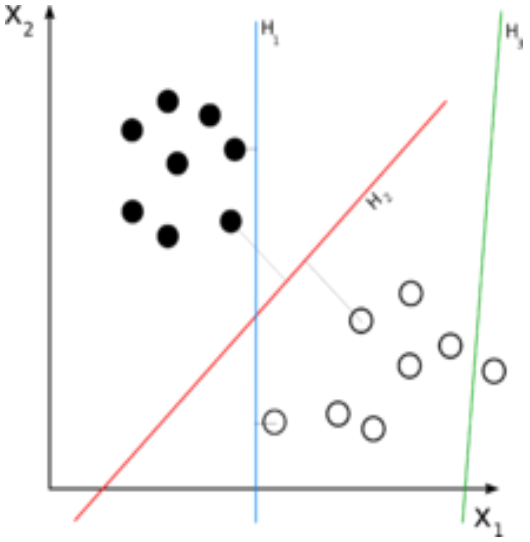


Fig. Reinforcement machine Learning

4. LINEAR CLASSIFIER

In the field of machine learning, the goal of statistical classification is to use an object's characteristics to identify which class (or group) it belongs to. A linear classifier achieves this by making a classification decision based on the value of a linear combination of the characteristics. An object's characteristics are also known as feature values and are typically presented to the machine in a vector called a feature vector. Such classifiers work well for practical problems such as document classification, and more generally for problems with many variables (features), reaching accuracy levels comparable to non-linear classifiers while taking less time to train and use.



In this case, the solid and empty dots can be correctly classified by any number of linear classifiers. H1 (blue) classifies them correctly, as does H2 (red). H2 could be considered "better" in the sense that it is also furthest from both groups. H3 (green) fails to correctly classify the dots.

Fig. Linear Classifier Model 2

$$y = f(\vec{w} \cdot \vec{x}) = f\left(\sum_j w_j x_j\right),$$

If the input feature vector to the classifier is a real vector x, then the output score is where w is a real vector of weights and *f* is a function that converts the dot product of the two vectors into the desired output. (In other words, w is a one-form or linear functional mapping x onto **R**.) The weight vector w is learned from a set of labeled training samples. Often *f* is a simple function that maps all values above a certain threshold to the first class and all other values to the second class. A more complex *f* might give the probability that an item belongs to a certain class.

For a two-class classification problem, one can visualize the operation of a linear classifier as splitting a high-dimensional input space with a hyperplane: all points on one side of the hyperplane are classified as "yes", while the others are classified as "no".

A linear classifier is often used in situations where the speed of classification is an issue, since it is often the fastest classifier, especially when x is sparse. Also, linear classifiers often work very well when the number of dimensions in x is large, as in document classification, where each element in x is typically the number of occurrences of a word in a document. In such cases, the classifier should be well-regularized.

4.1. Project Code Description

```
import numpy as np #package used for scientific computing
import matplotlib.pyplot as plt #plotting library for python
#%matplotlib inline
import tensorflow as tf #open source software library for data flow programming across a range of tasks


learn = tf.contrib.learn #contains contributed code that is to be merged in to core Tensor Flow
tf.logging.set_verbosity(tf.logging.ERROR) #sets the threshold for what messages will be logged

mnist = learn.datasets.load_dataset('mnist') #is used to load 'MNIST' datasets
data = mnist.train.images #vairable to load the train images from 'MNIST' data set
labels = np.asarray(mnist.train.labels, dtype=np.int32) #information of train images is copied in labels
test_data = mnist.test.images #variable to load the test images of 'MNIST' datasets
test_labels = np.asarray(mnist.test.labels, dtype=np.int32) #information of test images is copied in test_labels


max_examples = 20000 #Maximum range of index through the 'MNIST' train dataset
data = data[:max_examples] #data of the defined index is copied to data
labels = labels[:max_examples] #labels matrix is copied to labels variable
```

Fig. Linear Classifier Model 3

Here first we have loaded all the important package for python. Then we are importing tensorflow which is open source software library for data flow. contrib.learn() function includes the contributed code helpful for tensorflow to learn. logging function is used for intermediates logged values. Then we have loaded MNIST dataset and then train the images from the dataset and the labels corresponding to the features or train images are copied in the labels. Further the te8st images are loaded and array of test labels are created for the corresponding features of test dataset.

Here we are using the max 20000 images from the MNIST train dataset, Data of defined index and label matrix is copied to ‘data’ and ‘labels’ variable respectively.

```
# we are going to train a linear classifier for our data
# here in classifier we are taking 10 features for 10 digits
# feature_columns informs classifier about features we are going to use
feature_columns = learn.infer_real_valued_columns_from_input(data)
# declaring the model type to fit the data in
classifier = learn.LinearClassifier(feature_columns = feature_columns, n_classes=10)
# fitting the data into declared model
# it takes input of data, labels, batch size and number of steps to perform
classifier.fit(data, labels, batch_size=100, steps=10000)

# we will use evaluate function to compare our predicted values and values of labels
# and then use this to calculate our classifier accuracy
# here evaluate and accuracy both are inbuilt in classifier
x = classifier.evaluate(test_data, test_labels)
#print(classifier.evaluate(test_data, test_labels)["accuracy"])
print("Results: ",x)
print("Accuracy:",x["accuracy"]*100)
```

Fig. Linear Classifier Model 4

We are going to train linear classifier for our data, we are taking 10 features for 10 digits, here feature columns inform the features we are using for Linear classification. Model type is declared to fit the data in to the model which takes different attributes as input. Evaluate function is used to compare the prediction values and labeled values. And use this prediction to calculate the accuracy. The function evaluate and accuracy are both inbuilt in classifier. Finally we print the result and the accuracy.

Here accuracy achieved is 92.11% and the loss is 0.2911 for each data element which can be further enhanced by using different model for training and testing the dataset.

5. DECISION TREE CLASSIFIER

Decision Tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning algorithms, decision tree algorithm can be used for solving regression and classification problems too. The general motive of using Decision Tree is to create a training model which can use to predict class or value of target variables by learning decision rules inferred from prior data(training data).

The understanding level of Decision Trees algorithm is so easy compared with other classification algorithms. The decision tree algorithm tries to solve the problem, by using tree representation. Each internal node of the tree corresponds to an attribute, and each leaf node corresponds to a class label.

Decision Tree Algorithm Pseudo Code

1. Place the best attribute of the data set at the root of the tree.
2. Split the training set into subsets. Subsets should be made in such a way that each subset contains data with the same value for an attribute.
3. Repeat step 1 and step 2on each subset until you find leaf nodes in all the branches of the tree.

In decision trees, for predicting a class label for a record we start from the root of the tree. We compare the values of the root attribute with record's attribute. On the basis of comparison, we follow the branch corresponding to that value and jump to the next node. We continue comparing our record's attribute values with other internal nodes of the tree until we reach a leaf node with predicted class value. As we know how the modeled decision tree can be used to predict the target class or the value. Now let's understanding how we can create the decision tree model.

Assumptions while creating Decision Tree

The below are the some of the assumptions we make while using Decision tree:

1. At the beginning, the whole training set is considered as the **root.**
2. Feature values are preferred to be categorical. If the values are continuous then they are discretized prior to building the model.
3. Records are distributed recursively on the basis of attribute values.

4. Order to placing attributes as root or internal node of the tree is done by using some statistical approach.

The primary challenge in the decision tree implementation is to identify which attributes do we need to consider as the root node and each level. Handling this is know the attributes selection. We have different attributes selection measure to identify the attribute which can be considered as the root node at each level.

1. Information gain (Entropy Function)
2. Gini Index(Classification)

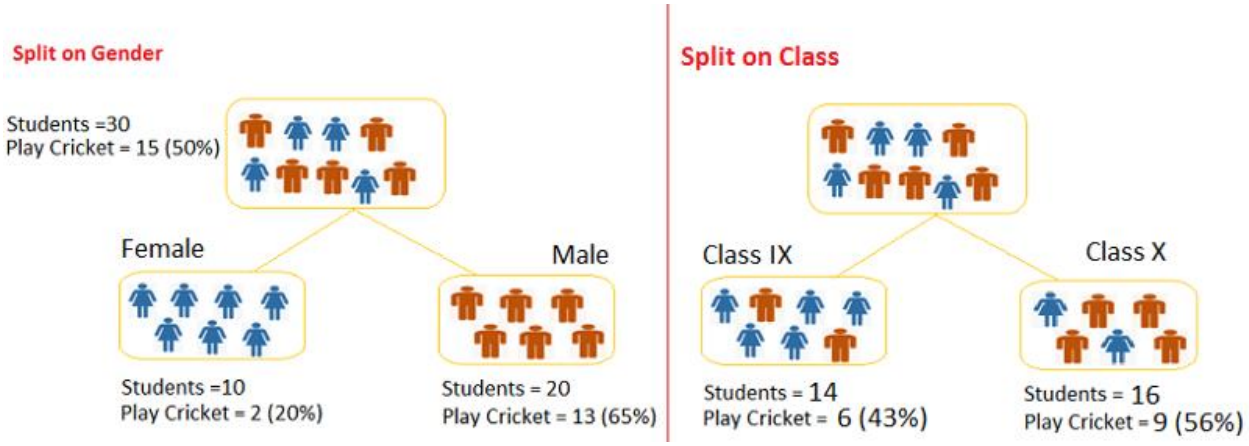


Fig. Decision tree classifier 3

In above fig. We can observe that on both side of table the students are splits on the two Components, Split on Gender and Split on class. Where on the gender basis there are two subclass of male and Female and on class basis the students are either of class IX or class X. Similarly for the handwriting recognition dataset is labeled on the basis of their features, and after training the dataset through the program it is compared by the Decision Tree classifier model imported from sklearn Library.

5.1. Project Code Description

```
import numpy as nm #Package used for scientific computing
import matplotlib.pyplot as pt #plotting library for python
import pandas as pd #fast, flexible and expressive datastructure designed to make working with 'relational' or 'labeled' data
from sklearn.tree import DecisionTreeClassifier #scikit learn,a machine lerning library

data= pd.read_csv("datasets/train/train.csv").as_matrix() # for reading the train.csv file(csv format to read tabular data)
clf=DecisionTreeClassifier()

#training datasets to train the program for supervisied learning

xtrain=data[0:21000,1:]
train_label=data[0:21000,0]

clf.fit(xtrain,train_label)

# testing data, includes only the input data not the corresponding expected output, used to assess how well the algo is trained
```

Fig. Decision Tree Classifier 4

We are using python language for coding the program, first three line the use full packages are imported to the notebook, and fourth line give the code for importing decision tree classifier from the skit learn(sklearn) machine learning library. Further we are using pd.read_csv() function from the particular destination as defined.

Next we create empty classifier clf using Decision tree classifier. Now we have to give training data set, for this we use two list one the list of feature and another the corresponding label of each feature. List of feature will contain the intensity value of each pixel, and the corresponding label will contain the digits. Here we are training the first 21000 dataset from the column 1 to remaining and further their corresponding level for first 21000 with only the 0^{th} column which contains label.

We are training the classifier using the fit method, it take the features list and labels to train the classifier.

```
# testing data, includes only the input data not the corresponding expected output, used to assess how well the algo is trained

xtest=data[21000:,1:]
actual_label=data[21000:,0]

d=xtest[543]
d.shape=(28,28) #for as plotting the number
pt.imshow(255-d, cmap='gray')

print(clf.predict([xtest[543]]))
pt.show

p=clf.predict(xtest)

count=0
for i in range(0,21000):
    count = count+1 if p[i]==actual_label[i] else 0

print("acurracy is ", (count/21000)*100)
```

Fig. Decision Tree Classifier 5

In above fig. We are creating testing data for first 21000 dataset and actual label is used to checking the accuracy after getting the prediction from the machine learning model. Now we take sample of 543th element. The row vector is converted to 28X28 matrix and plotting of the graph is shown by using pt.imshow() function. (255-d) is used to show white background and black color, we are using gray instead of RGB. Further our learning model is ready for the prediction with the feature. Further we take p as prediction of testing dataset xtest. Further we use count method to give the prediction data of each element in the range and accuracy is calculated based on the predictions.

Here we can found the accuracy of 83.77 % which can be enhanced by using the CNN model.

6 . CONVLUTIONAL NEURAL NETWORKS

Convolutional neural networks (CNNs) are the current state-of-the-art model architecture for image classification tasks. CNNs apply a series of filters to the raw pixel data of an image to extract and learn higher-level features, which the model can then use for classification. CNNs contains three components:

- **Convolutional layers**, which apply a specified number of convolution filters to the image. For each subregion, the layer performs a set of mathematical operations to produce a single value in the output feature map. Convolutional layers then typically apply a ReLU activation function to the output to introduce non linearities into the model.

- **Pooling layers**, which downsample the image data extracted by the Convolutional layers to reduce the dimensionality of the feature map in order to decrease processing time. A commonly used pooling algorithm is max pooling, which extracts subregions of the feature map (e.g., 2x2-pixel tiles), keeps their maximum value, and discards all other values.

- **Dense (fully connected) layers**, which perform classification on the features extracted by the Convolutional layers and down sampled by the pooling layers. In a dense layer, every node in the layer is connected to every node in the preceding layer.

Network Flow in The Model

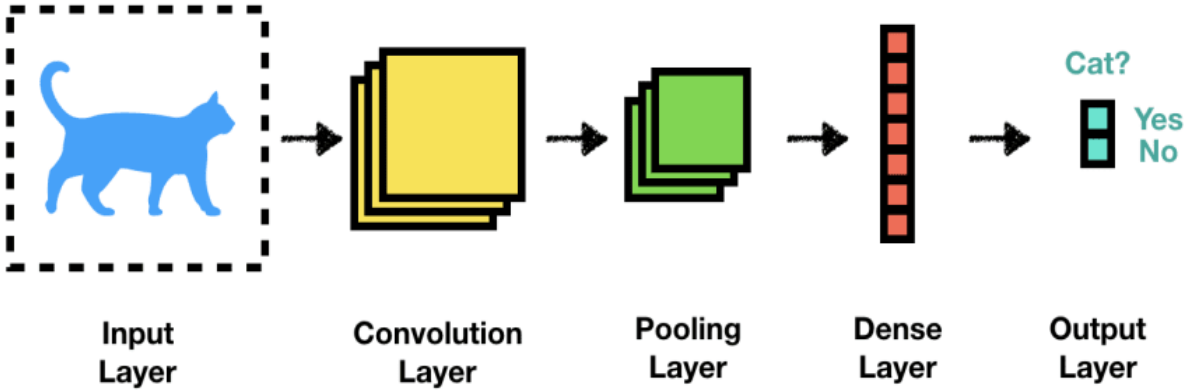
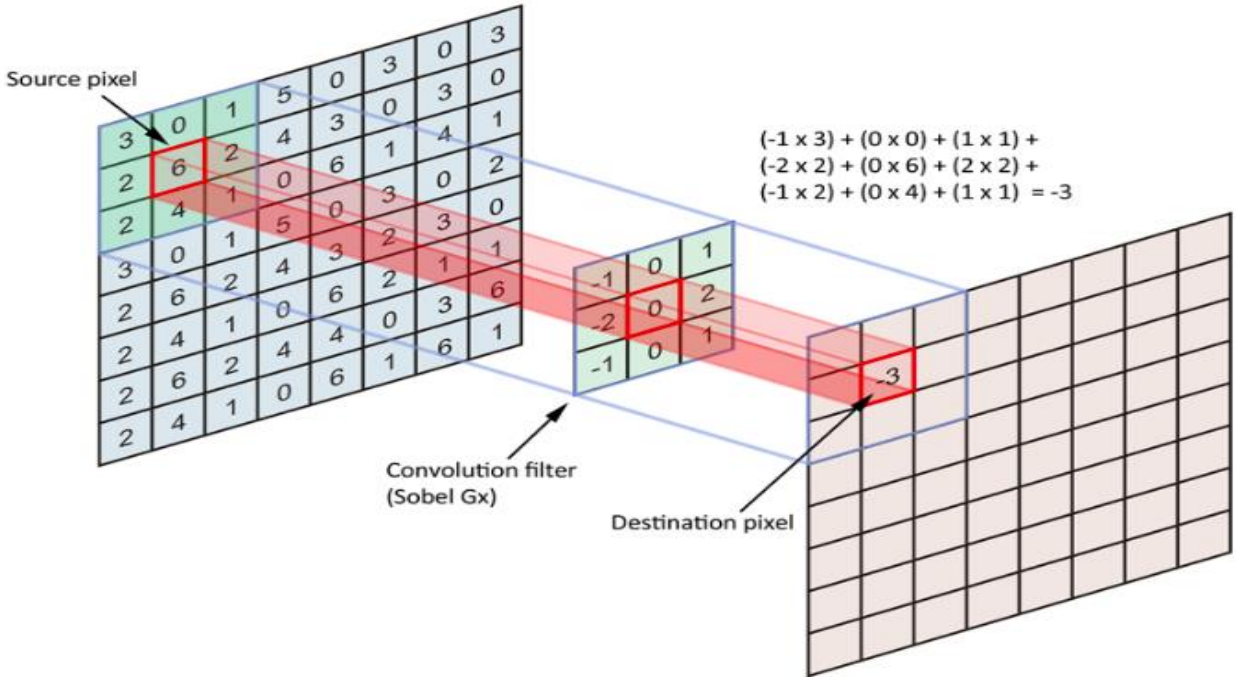


Fig. Network Flow

6.1. CONVOLUTIONAL LAYER



The Convolutional layer is the core building block of a Convolutional Network that does most of the computational heavy lifting. The CONV layer's parameters consist of a set of learnable filters. Every filter is small spatially (along width and height), but extends through the full depth of the input volume. For example, a typical filter on a first layer of a ConvNet might have size 5x5x3 (i.e. 5 pixels width and height, and 3 because images have depth 3, the color channels).

During the forward pass, we slide (more precisely, convolve) each filter across the width and height of the input volume and compute dot products between the entries of the filter and the input at any position. As we slide the filter over the width and height of the input volume we will produce a 2-dimensional activation map that gives the responses of that filter at every spatial position. Intuitively, the network will learn filters that activate when they see some type of visual feature such as an edge of some orientation or a blotch of some color on the first layer, or eventually entire honeycomb or wheel-like patterns on higher layers of the network. Now, we will have an entire set of filters in each CONV layer (e.g. 12 filters), and each of them will produce a separate 2-dimensional activation map. We will stack these activation maps along the depth dimension and produce the output volume.

6.2. Pooling Layer

It is common to periodically insert a Pooling layer in-between successive Conv layers in a ConvNet architecture. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting. The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the MAX operation. The most common form is a pooling layer with filters of size 2x2 applied with a stride of 2 downsamples every depth slice in the input by 2 along both width and height, discarding 75% of the activations. Every MAX operation would in this case be taking a max over 4 numbers (little 2x2 region in some depth slice). The depth dimension remains unchanged.

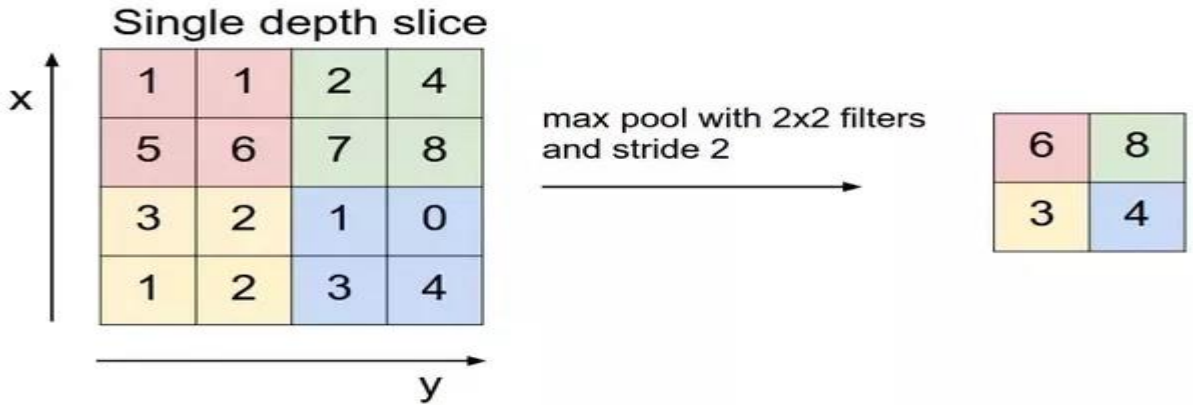
Max pooling is a sample-based discretization process. The objective is to down-sample an input representation (image, hidden-layer output matrix, etc.), reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned.

This is done to in part to help over-fitting by providing an abstracted form of the representation. As well, it reduces the computational cost by reducing the number of parameters to learn and provides basic translation invariance to the internal representation.

Max pooling is done by applying a *max filter* to (usually) non-overlapping subregions of the initial representation.

Let's say we have a 4x4 matrix representing our initial input. Let's say, as well, that we have a 2x2 filter that we'll run over our input. We'll have a **stride** of 2 (meaning the (dx, dy) for stepping over our input will be (2, 2)) and won't overlap regions. For each of the regions represented by the filter, we will take the ***max*** of that region and create a new, output matrix where each element is the max of a region in the original input.

In order to make this super easy, with a nice pictorial representation - I give you this:



For a real example (note that the z dimension, the number of layers, remains unchanged in the pooling operation)

6.3. Dense (Fully Connected) Layer

Dense layer connect every neuron from the previous layer to the next one. It is more complex in terms of parameter fitting then Convolutional layer. The layer is followed by non linear activation function. There can be a multiple Dense layer before Output layer.

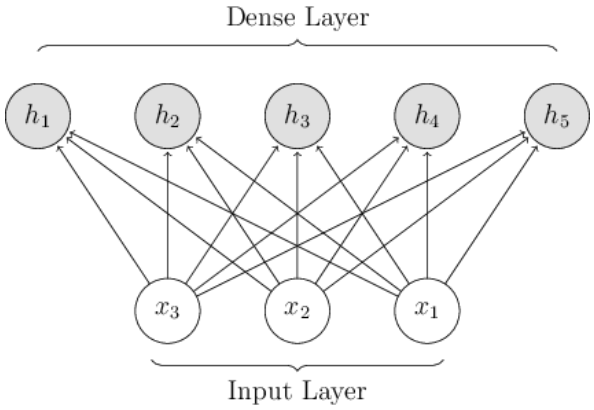


Fig. Dense Layer

6.4. Dropout

Dropout is a regularization technique for Neural Network models. A simple way to prevent neural network from over fitting. Dropout is a technique where randomly selected neurons are ignored during training. They are "dropped-out" randomly.

6.5. ReLU activation function

ReLU is the abbreviation of Rectified Linear Units. This layer applies the non-saturating activation function $f(x) = max(0,x)$. It increases the nonlinear properties of the decision function and of the overall network without affecting the receptive fields of the convolution layer.

Other functions are also used to increase nonlinearity, for example the saturating hyperbolic tangent $f(x) = tanh(x)$, $f(x) = |tanh(x)|$, and the sigmoid function $f(x) = (1+e^{-x})^{-1}$. ReLU is often preferred to other functions, because it trains the neural network several times faster without a significant penalty to generalization accuracy.

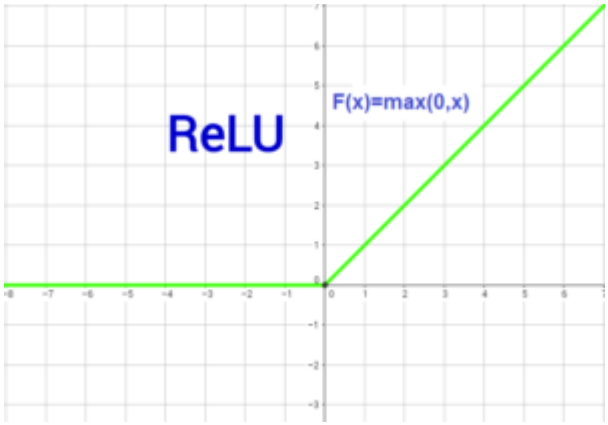


Fig. ReLU activation

6.6. Project code Description

In the project, CNN model consist of following sequence of layers:

1. **Convolutional Layer #1**: Applies 32 5x5 filters (extracting 5x5-pixel subregions), with ReLU activation function

2. **Pooling Layer #1**: Performs max pooling with a 2x2 filter and stride of 2 (which specifies that pooled regions do not overlap)

3. **Convolutional Layer #2**: Applies 64 5x5 filters, with ReLU activation function

4. **Pooling Layer #2**: Again, performs max pooling with a 2x2 filter and stride of 2

5. **Dense Layer #1**: 1,024 neurons, with dropout regularization rate of 0.4 (probability of 0.4 that any given element will be dropped during training)

6. **Dense Layer #2 (Logits Layer)**: 10 neurons, one for each digit target class (0–9).

```
14 def cnn_model_fn(features,labels,mode): #model function of cnn
15
16     input_layer = tf.reshape(features["x"],[-1,28,28,1])  #input layer
17
18     #convolutional Layer 1
19     conv1 = tf.layers.conv2d(
20     inputs = input_layer,
21     filters=32,
22     kernel_size=[5,5],
23     padding = "same",
24     activation = tf.nn.relu)
25
26
27
28     #pooling Layer 1
29     pool1 = tf.layers.max_pooling2d(inputs=conv1, pool_size=[2,2], strides=2)
30
31     #convolutional Layer 2
32     conv2 = tf.layers.conv2d(
33     inputs = pool1,
34     filters = 64,
35     kernel_size =[5,5],
36     activation = tf.nn.relu)
```

Fig. CNN Implementation

Above contains the part of the Convolutional layer and pooling layer used in the python code. Same there are total two CONVO layer and two pooling layer used in the program. Using CNN model for training and predicting the dataset gives the best result I.e highest accuracy to the implementation.

7. ANALAYSIS OF RESULT

After successfully running our dataset through all three models, we are in a position to compare the final results.

7.1. Linear classifier:

We will run the model for different number of iterations(n):

No. of Iterations	Accuracy(in %)
N = 1	26.08
N = 100	89.80
N = 1000	91.50
N = 10000	92.11

As we go on increasing the iteration on data, the accuracy will keep on increasing. After only a 100 iterations we have much better accuracy for our classifier.

After running the model through 10000 steps, we get an Accuracy of around 92.11% with overall loss for each data element of about 0.2911. This accuracy is pretty good in case of linear classifier.

If we keep on increasing the number of steps after this it will have little effect on our accuracy and loss.

7.2. Decision tree classifier:

For this model, we cannot run the classifier for different number of iterations. But instead of that we will try altering the size of our training data.

No matter what, our model is giving an Accuracy with variations between 80 – 85 %.

7.3. Convolutional Neural Network:

We will train model for different number of iterations and check its accuracy.

No. of Iteration	Accuracy(in %)
N = 1	15.1
N = 100	60.1
N = 1000	92.8
N = 10000	98.7

As we go on increasing the iteration on data, the accuracy will keep on increasing. After only a 100 iterations we have much better accuracy for our classifier.

Here this achieved accuracy is Maximum for our Designed CNN model. If we increase the size and complexity of our model this might go up to 99% or more.

We can see that CNN works like human brain. It keeps on learning with time. Also, that when working with CNN the accuracy is far much better as compared to linear classifier and decision tree classifier. We can conclude that with proper structure and time we can always have better results with CNN.

8. CONCLUSION

After using various models on our data, we have come to a conclusion that Convolutional Neural Network work best in case of pattern recognition. We can achieve near to a hundred percent accuracy with it. Also that we can use various types of data for purpose of recognition. This models if modified specifically can be used for various applications.

REFERENCES

1. www.wikipedia.org
2. www.youtube.com
3. Machine Learning by Andrew Ng on Coursera