Chapter-1

INTRODUCTION

Machine learning (ML) is a category of algorithm that allows software applications to become more accurate in predicting outcomes without being explicitly programmed. The basic premise of machine learning is to build algorithms that can receive input data and use statistical analysis to predict an output while updating outputs as new data becomes available. The processes involved in machine learning are similar to that of data mining and predictive modeling. Both require searching through data to look for patterns and adjusting program actions accordingly. Many people are familiar with machine learning from shopping on the internet and being served ads related to their purchase. This happens because recommendation engines use machine learning to personalize online ad delivery in almost real time. Beyond personalized marketing, other common machine learning use cases include fraud detection, spam filtering, network security threat detection, predictive maintenance and building news feeds. Machine learning platforms are among enterprise technology's most competitive realms, with most major vendors, including Amazon, Google, Microsoft, IBM and others, racing to sign customers up for platform services that cover the spectrum of machine learning activities, including data collection, data preparation, model building, training and application deployment. As machine learning continues to increase in importance to business operations and AI becomes ever more practical in enterprise settings, the machine learning platform wars will only intensify. Computer vision is a field of computer science that works on enabling computers to see, identify and process images in the same way that human vision does, and then provide appropriate output. It is like imparting human intelligence and instincts to a computer. Computer vision's goal is not only to see, but also process and provide useful results based on the observation. For example, a computer could create a 3D image from a 2D image, such as those in cars, and provide important data to the car and/or driver. For example, cars could be fitted with computer vision which would be able to identify and distinguish objects on and around the road such as traffic lights, pedestrians, traffic signs and so on, and act accordingly. The intelligent device could provide inputs to the driver or even make the car stop if there is a sudden obstacle on the road.

1.1 Machine Learning

Machine learning is the science of getting computers to act without being explicitly programmed. Machine learning is closely related to (and often overlaps with) computational statistics, which also focuses on prediction - making through the use of computers. It has strong ties to mathematical optimization, which delivers methods, theory and application domains to the field. Machine learning is sometimes conflated with data mining, where the latter subfield focuses more on exploratory data analysis and is known as unsupervised learning.

1.11 Types Of Machine Learning

There are three types of Machine Learning:-

- Supervised
- Unsupervised
- Reinforcement

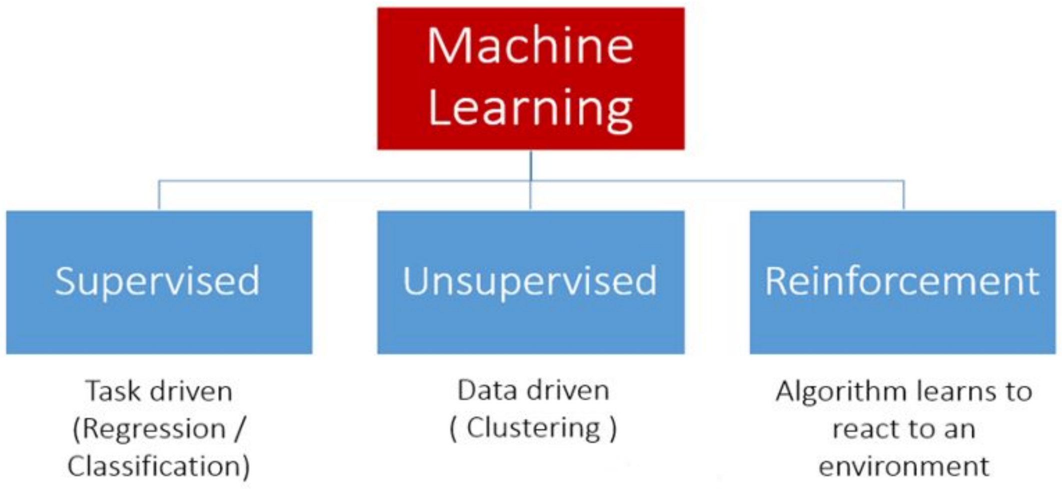

Machine Learning
Supervised
Unsupervised
Reinforcement
Task driven (Regression / Classification)
Data driven (Clustering)
Algorithm learns to react to an environment

Fig. 1.1 Types of Machine Learning

1.111 Supervised Learning

Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs. It infers a function from labeled training data consisting of a set of training examples. 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.

1.112 Unsupervised Learning

Unsupervised machine learning is the machine learning task of inferring a function that describes the structure of unlabeled data (i.e. data that has not been classified or categorized). Since the examples given to the learning algorithm are unlabeled, there is no straightforward way to evaluate the accuracy of the structure that is produced by the algorithm.

1.113 Reinforcement Learning

Reinforcement Learning is a type of Machine Learning, and thereby also a branch of Artificial Intelligence. It allows machines and software agents to automatically determine the ideal behaviour within a specific context, in order to maximize its performance. Simple reward feedback is required for the agent to learn its behaviour; this is known as the reinforcement signal. There are many different algorithms that tackle this issue. As a matter of fact, Reinforcement Learning is defined by a specific type of problem, and all its solutions are classed as Reinforcement Learning algorithms. In the problem, an agent is supposed decide the best action to select based on his current state. When this step is repeated, the problem is known as a Markov Decision Process.

1.12 Machine Learning Techniques

The following are the Machine Learning Techniques:-

- Regression

- Classification
- Clustering

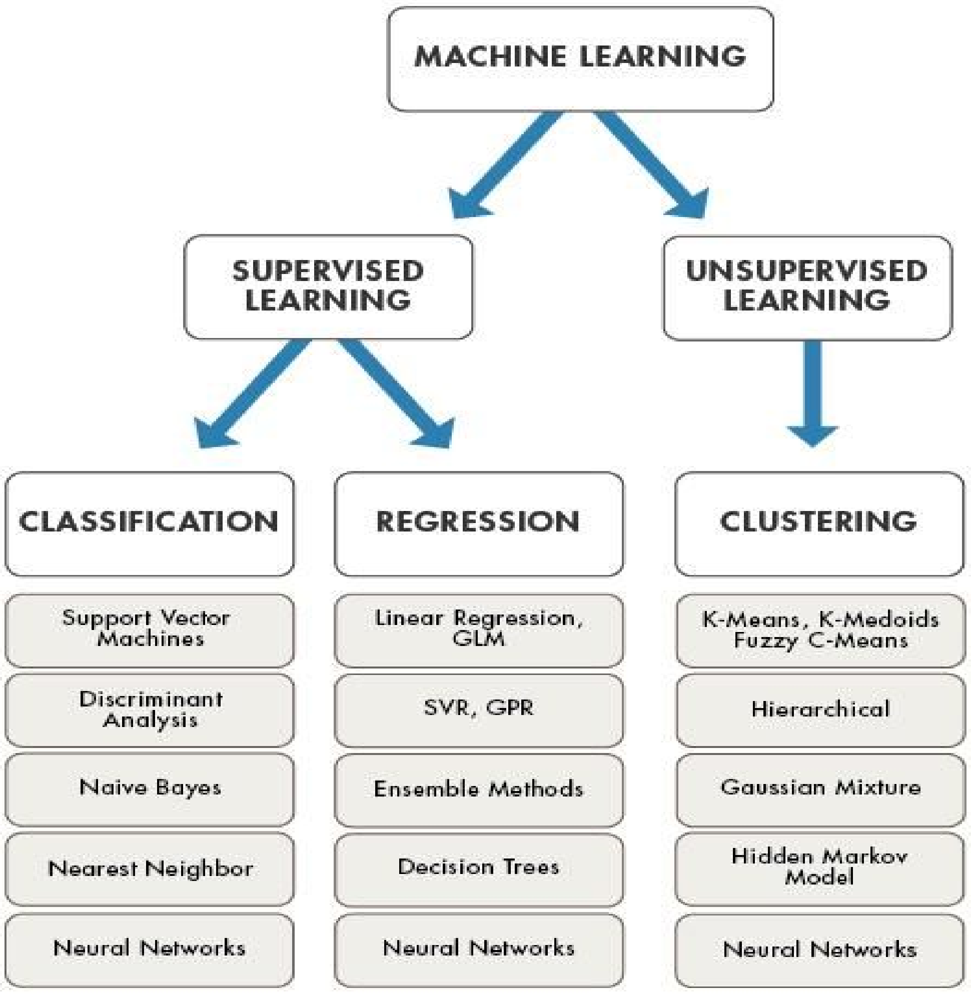


Fig. 1.2 Machine Learning Techniques

1.121 Regression

It predicts continuous valued output. The Regression analysis is the statistical model which is used to predict the numeric data instead of labels. It can also identify the distribution trends based on the available data or historical data. Predicting a person's income from their age, education is example of regression task.

1.122 Classification

It predicts discrete number of values. In classification the data is categorized under different labels according to some parameters and then the labels are predicted for the data. Classifying emails as either spam or not spam is example of classification problem.

1.123 Clustering

Clustering is the task of partitioning the dataset into groups, called clusters.The goal is to split up the data in such a way that points within single cluster are very similar and points in different clusters are different. It determines grouping among unlabeled data.

1.13 Machine Learning Algorithms

Here is the list of some machine learning algorithms. These algorithms can be applied to almost any data problem: :-

- KNN
- K-Means
- Decision Tree
- Random Forest

1.131 KNN

It can be used for both classification and regression problems. However, it is more widely used in classification problems in the industry. K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases by a majority vote of its k neighbors. The case being assigned to the class is most common amongst its K nearest neighbors measured by a distance function. These distance functions can be Euclidean, Manhattan, Minkowski and Hamming distance. First three functions are used for continuous function and fourth one (Hamming) for categorical variables. If $K = 1$, then the case is simply assigned to the class of its nearest neighbor. At times, choosing K turns out to be a challenge while performing KNN modeling.

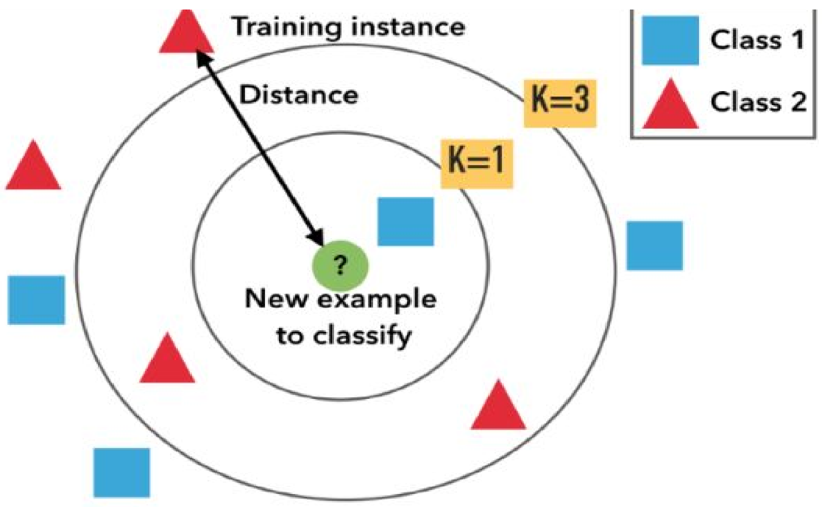


Fig. 1.3 KNN

1.132 K-Means

It is a type of unsupervised algorithm which solves the clustering problem. Its procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters). Data points inside a cluster are homogeneous and heterogeneous to peer groups.

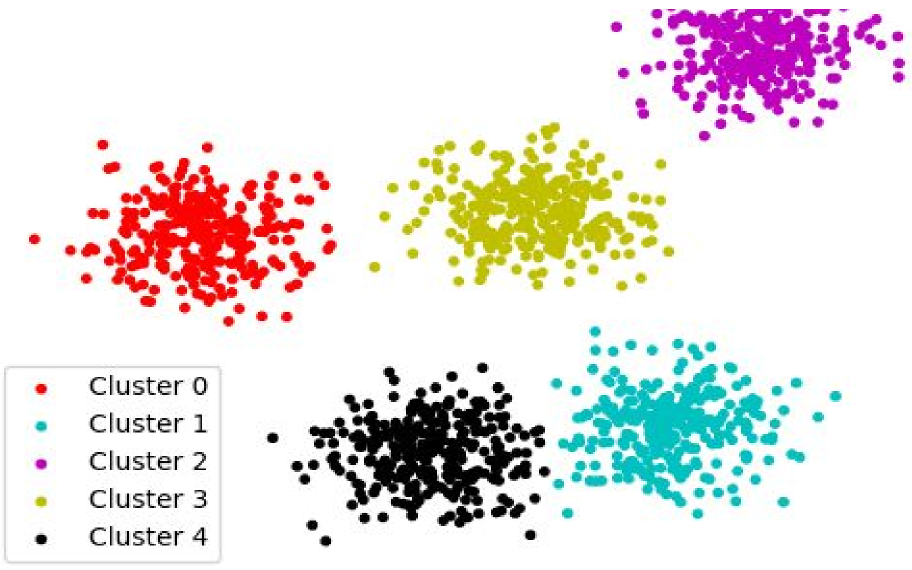


Fig. 1.4 K-Means

1.133 Decision Trees

This is one of my favorite algorithms and I use it quite frequently. It is a type of supervised learning algorithm that is mostly used for classification problems. Surprisingly, it works for both categorical and continuous dependent variables. In this algorithm, we split the population into two or more homogeneous sets. This is done based on most significant attributes/ independent variables to make as distinct groups as possible.

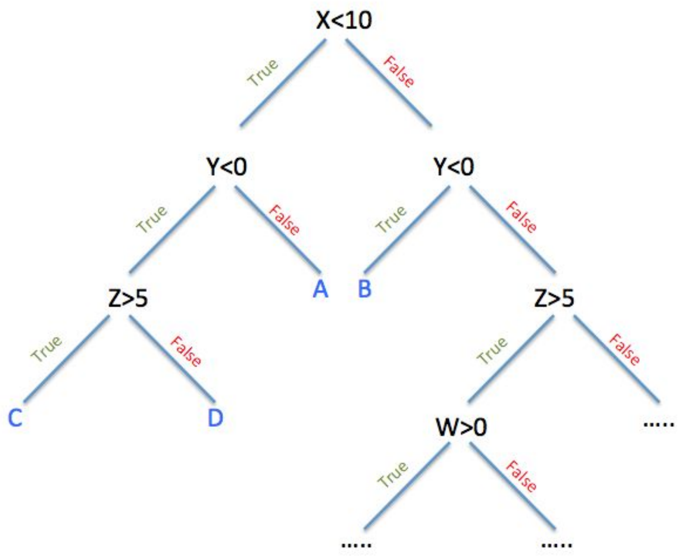


Fig. 1.5 Decision Tree

1.134 Random Forest

Random Forest is a trademark term for an ensemble of decision trees. In Random Forest, we've collection of decision trees (so known as "Forest"). To classify a new object based on attributes, each tree gives a classification and we say the tree "votes" for that class. The forest chooses the classification having the most votes (over all the trees in the forest).

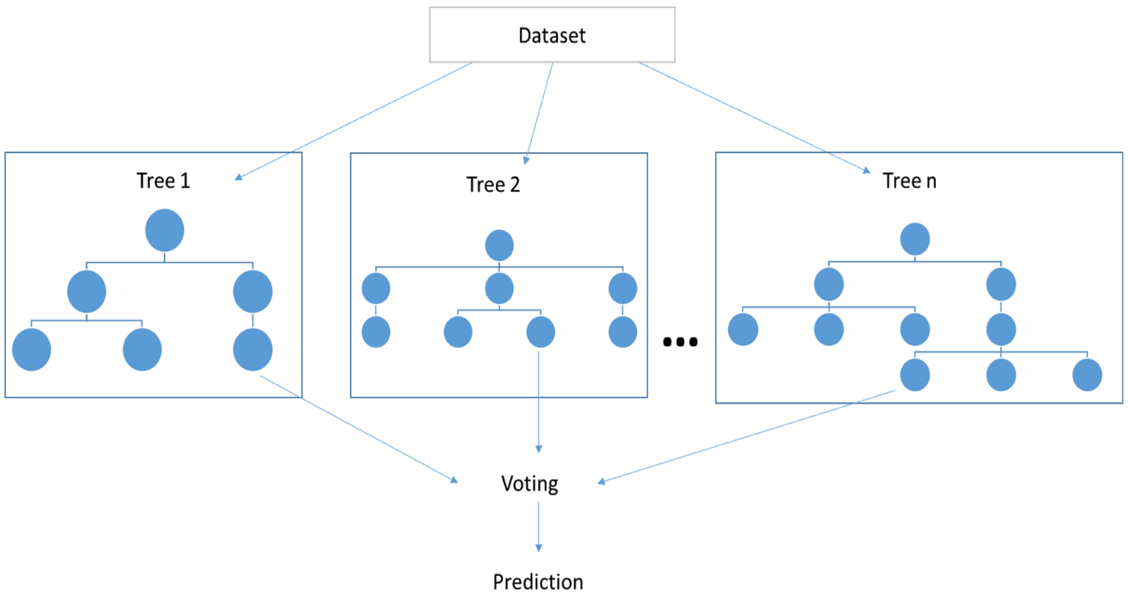


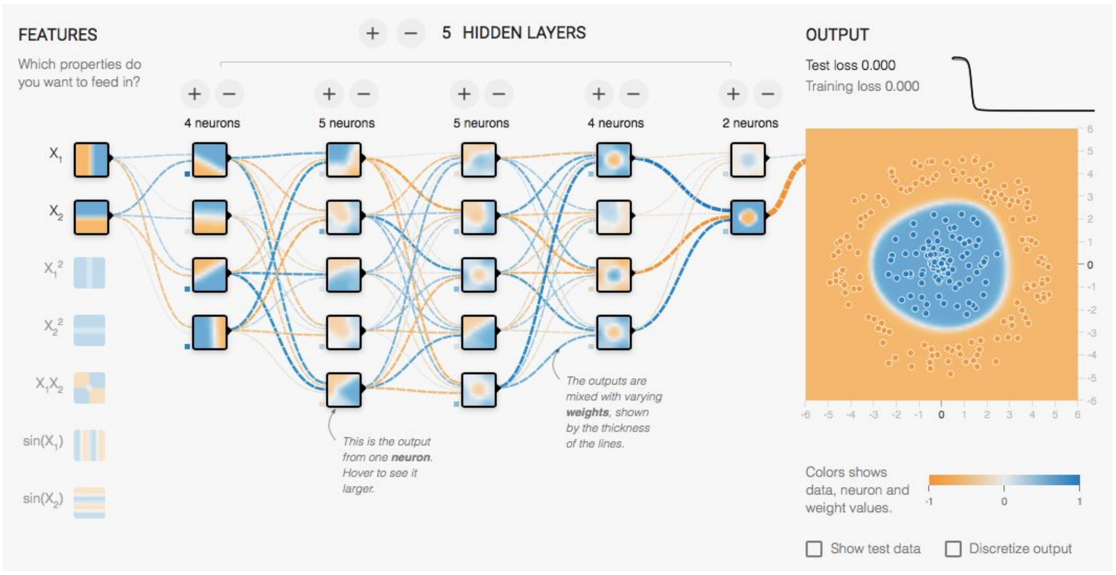
Fig. 1.6 Random Forest

1.2 Computer Vision

Computer vision is field that deals with how computers can be made for gaining high-level understanding from digital images or videos. It seeks to automate tasks that the human visual system can do. Computer vision is concerned with the automatic extraction, analysis and understanding of useful information from a single image or a sequence of images. It involves the development of a theoretical and algorithmic basis to achieve automatic visual understanding. As a scientific discipline, computer vision is concerned with the theory behind artificial systems that extract information from images. The image data can take many forms, such as video sequences, views from multiple cameras, or multidimensional data from a medical scanner.

1.3 Deep Learning

Deep learning (also known as deep structured learning or hierarchical learning) is part of a broader family of machine learning methods based on learning data representations, as opposed to task-specific algorithms. Learning can be supervised, semi-supervised or unsupervised. Deep learning architectures such as deep neural networks, deep belief networks and recurrent neural networks have been applied to fields including computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics, drug design and board game programs, where they have produced results comparable to and in some cases superior to human experts. Deep learning models are vaguely inspired by information processing and communication patterns in biological nervous systems yet have various differences from the structural and functional properties of biological brains, which make them incompatible with neuroscience evidences.



FEATURES
Which properties do you want to feed in?
5 HIDDEN LAYERS
OUTPUT
Test loss 0.000
Training loss 0.000
4 neurons
5 neurons
5 neurons
4 neurons
2 neurons
X_1
X_2
X_1^2
X_2^2
X_1X_2
$\sin(X_1)$
$\sin(X_2)$
This is the output from one neuron. Hover to see it larger.
The outputs are mixed with varying weights, shown by the thickness of the lines.
Colors shows data, neuron and weight values.
-1
0
1
Show test data
Discretize output

Fig 1.7 Structure of deep learning model

1.31 Types of Deep Learning Models

The following are the various types of Deep Learning models : Perceptron, Feed Forward, Radial Basis Network, Deep Feedforward, Recurrent Neural Network, Long/Short Term Memory, Gated Recurrent Unit, Auto Encoders, Variational AE, Denoising, Sparse AE, Markov Chain, Hopfield Network, Boltzmann Machine, Restricted BM, Deep Brief Network, Deep Convolutional Network, Deconvolutional Network, Deep Convolutional Inverse Graphics Network, Generative Adversarial Network, Liquid State Machine, Extreme Learning Machine, Echo State Network, Deep Residual Network, Kohonen Network, Support Vector Machine, Neural Turing Machine.

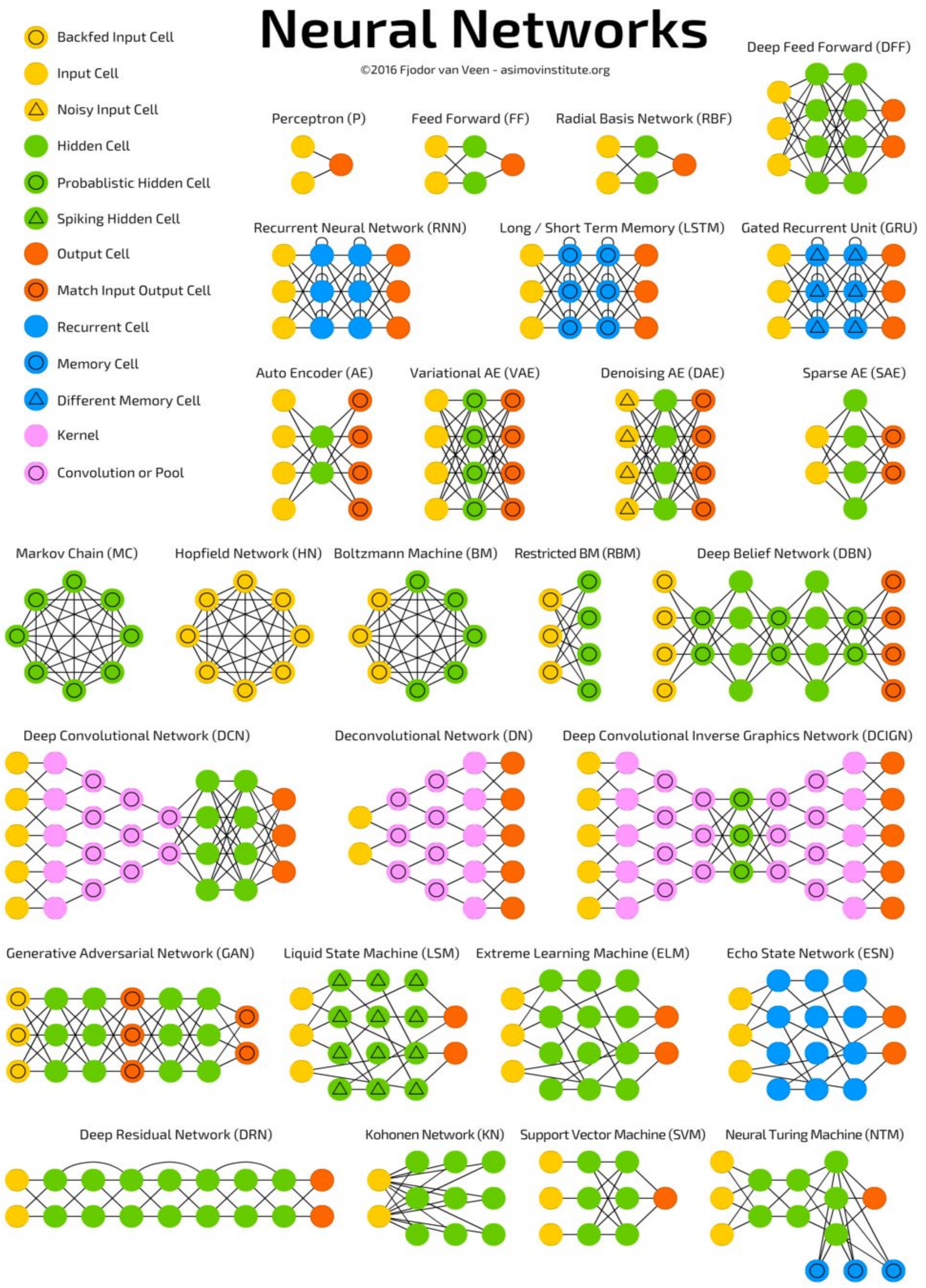

Neural Networks
©2016 Fjodor van Veen - asimovinstitute.org
Backfed Input Cell
Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Different Memory Cell
Kernel
Convolution or Pool
Perceptron (P)
Feed Forward (FF)
Radial Basis Network (RBF)
Deep Feed Forward (DFF)
Recurrent Neural Network (RNN)
Long / Short Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Auto Encoder (AE)
Variational AE (VAE)
Denoising AE (DAE)
Sparse AE (SAE)
Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)
Deep Convolutional Network (DCN)
Deconvolutional Network (DN)
Deep Convolutional Inverse Graphics Network (DCIGN)
Generative Adversarial Network (GAN)
Liquid State Machine (LSM)
Extreme Learning Machine (ELM)
Echo State Network (ESN)
Deep Residual Network (DRN)
Kohonen Network (KN)
Support Vector Machine (SVM)
Neural Turing Machine (NTM)

Fig. 1.8 Neural Networks

1.4 Convolutional Neural Network

In neural networks, Convolutional neural network (ConvNets or CNNs) is one of the main categories to do images recognition, images classifications. Objects detections, recognition faces etc., are some of the areas where CNNs are widely used. CNN image classifications takes an input image, process it and classify it under certain categories (Eg., Dog, Cat, Tiger, Lion). Computers sees an input image as array of pixels and it depends on the image resolution. Based on the image resolution, it will see h x w x d(h = Height, w = Width, d = Dimension). Eg., An image of 6 x 6 x 3 array of matrix of RGB (3 refers to RGB values) and an image of 4 x 4 x 1 array of matrix of grayscale image.

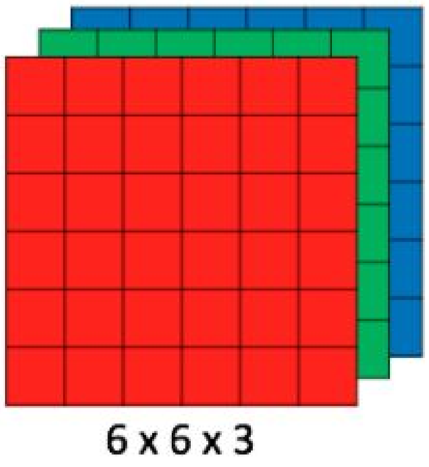


Fig. 1.9 Array of RGB Matrix

Technically, deep learning CNN models to train and test, each input image will pass it through a series of convolution layers with filters (Kernels), 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.

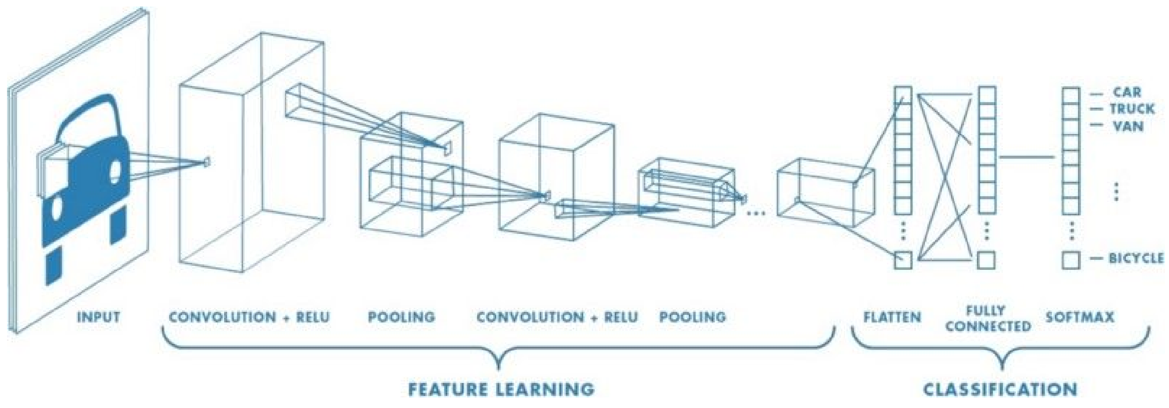


Fig. 1.10 Neural network with many convolutional layers

Convolution Layer is the first layer to extract features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data. It is a mathematical operation that takes two inputs such as image matrix and a filter or kernel.

- An image matrix (volume) of dimension **($h \times w \times d$)**
- A filter **($f_h \times f_w \times d$)**
- Outputs a volume dimension **($(h - f_h + 1) \times (w - f_w + 1) \times 1$)**

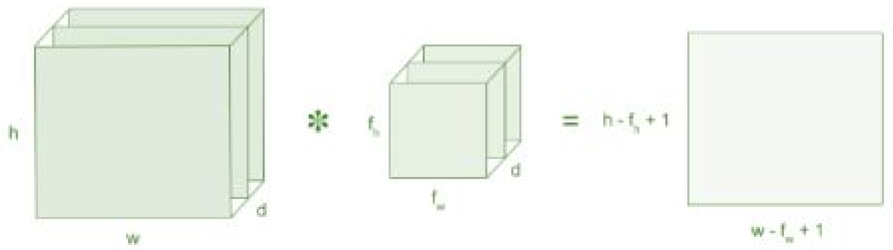


Fig. 1.11 Image matrix multiplies kernel or filter matrix

Consider a 5 x 5 whose image pixel values are 0, 1 and filter matrix 3 x 3 as shown in below.

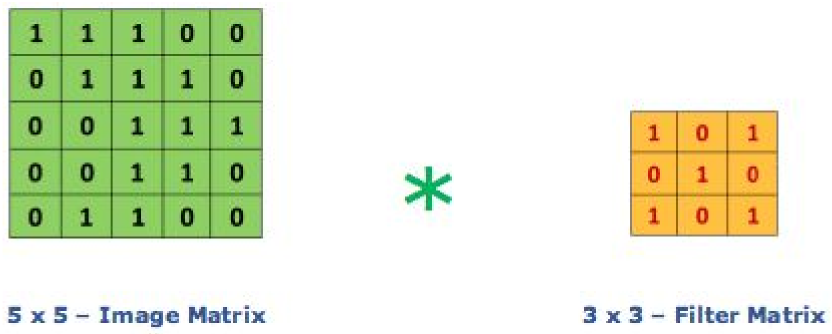


Fig. 1.12 Image matrix multiplies kernel or filter matrix

Then the convolution of 5 x 5 image matrix multiplied with 3 x 3 filter matrix which is called "Feature Map" as output shown in below.

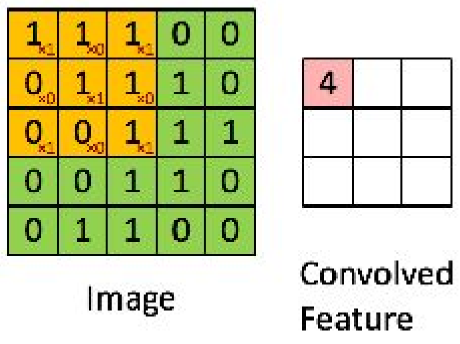


Fig. 1.13 3 x 3 Output matrix

Convolution of an image with different filters can perform operations such as edge detection, blur and sharpen by applying filters. The below example shows various convolution image after applying different types of filters (Kernels). Stride is the number of pixels shifts over the input matrix. When the stride is 1 then we move the filters to 1 pixel at a time. When the stride is 2 then we move the filters to 2 pixels at a time and so on. The below figure shows convolution would work with a stride of 2.

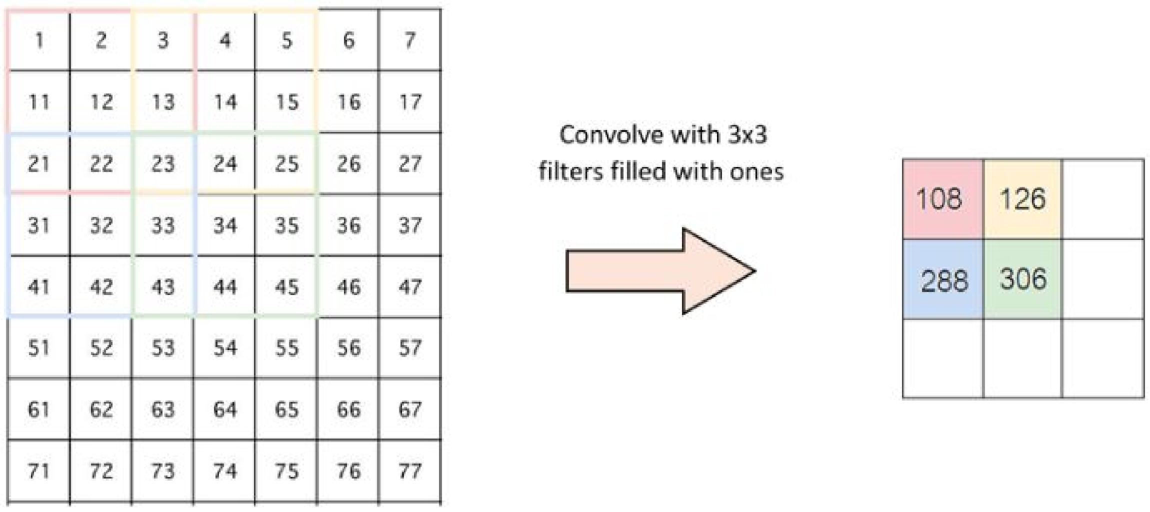


Fig. 1.14 Stride of 2 pixels

Sometimes filter does not fit perfectly fit the input image. We have two options : Pad the picture with zeros (zero-padding) so that it fits and drop the part of the image where the filter did not fit. This is called valid padding which keeps only valid part of the image. ReLU stands for Rectified Linear Unit for a non-linear operation. The output is $f(x) = \max(0,x)$. Why ReLU is important : Re LU's purpose is to introduce non-linearity in our ConvNet. Since, the real world data would want our ConvNet to learn would be non-negative linear values.

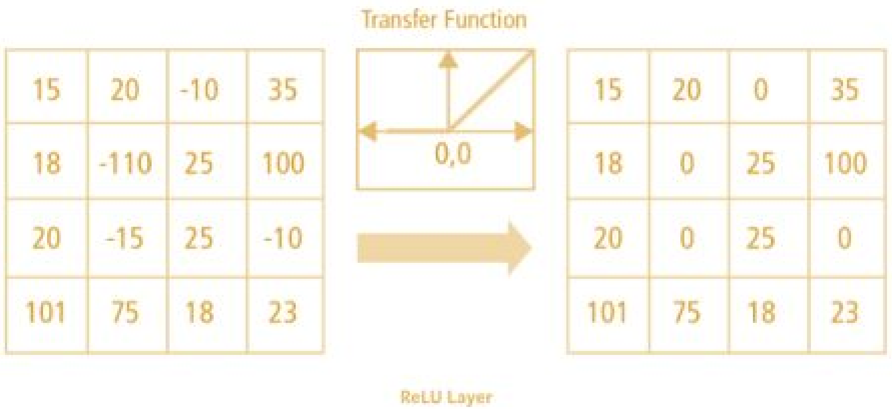


Fig. 1.15 ReLU operation

There are other nonlinear functions such as tanh or sigmoid can also be used instead of ReLU. Most of the data scientists uses ReLU since performance wise ReLU is better than other two. Pooling Layer section would reduce the number of parameters when the images are too large. Spatial pooling also called subsampling or downsampling which reduces the dimensionality of each map but retains the important information. Spatial pooling can be of different types: Max Pooling, Average Pooling and Sum Pooling. Max pooling take the largest element from the rectified feature map. Taking the largest element could also take the average pooling. Sum of all elements in the feature map call as sum pooling.

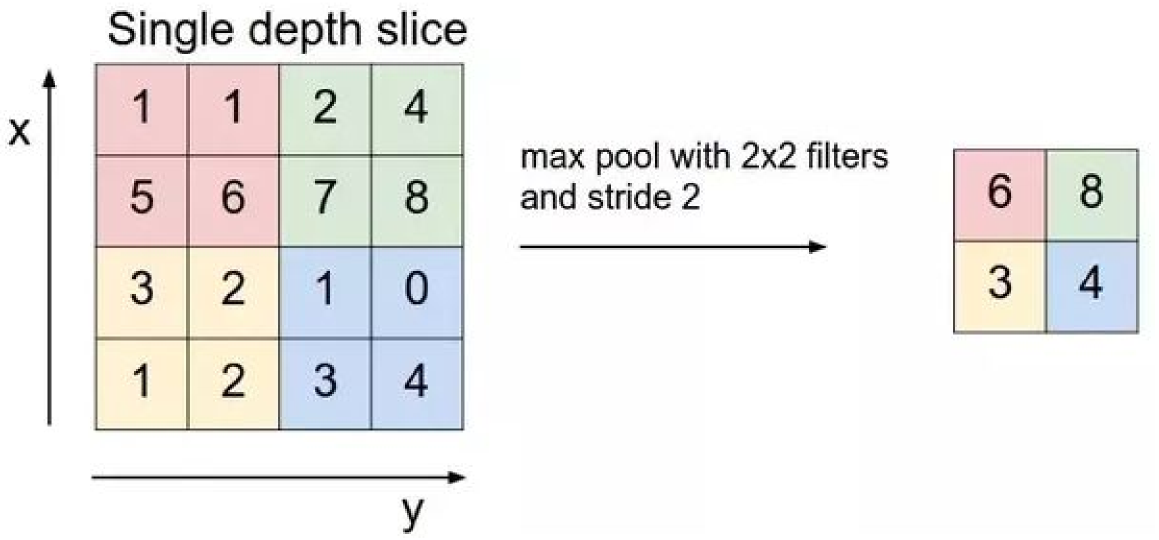


Fig. 1.16 Max Pooling

The layer we call as FC layer, we flattened our matrix into vector and feed it into a fully connected layer like neural network.

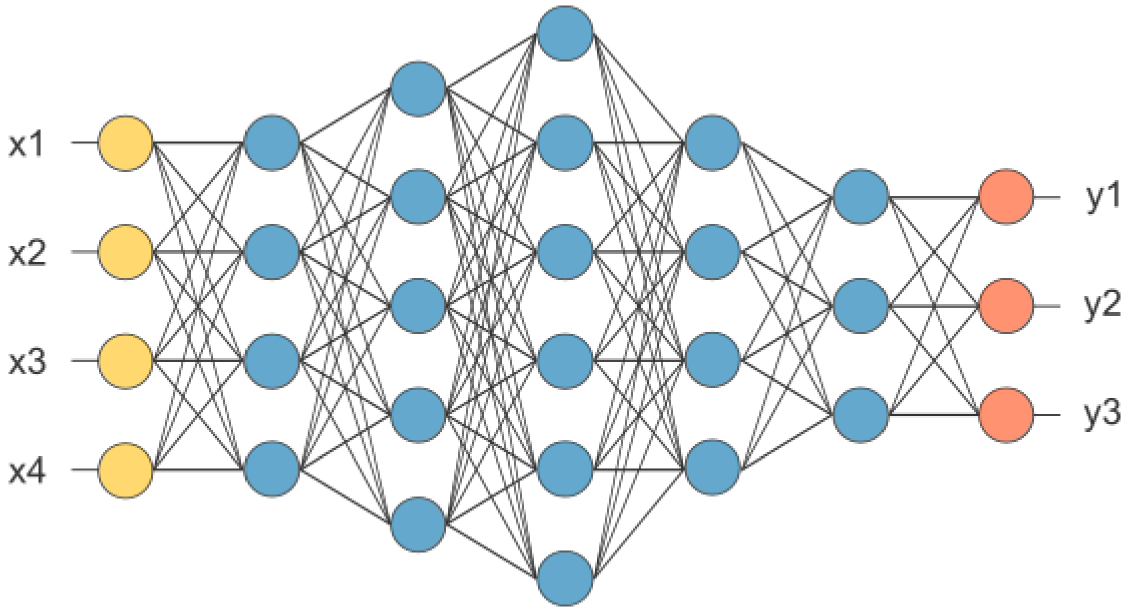


Fig. 1.17 After pooling layer, flattened as FC layer

In the above diagram, feature map matrix will be converted as vector (x1, x2, x3, …). With the fully connected layers, we combined these features together to create a model. Finally, we have an activation function such as softmax or sigmoid to classify the outputs as cat, dog, car, truck etc.,

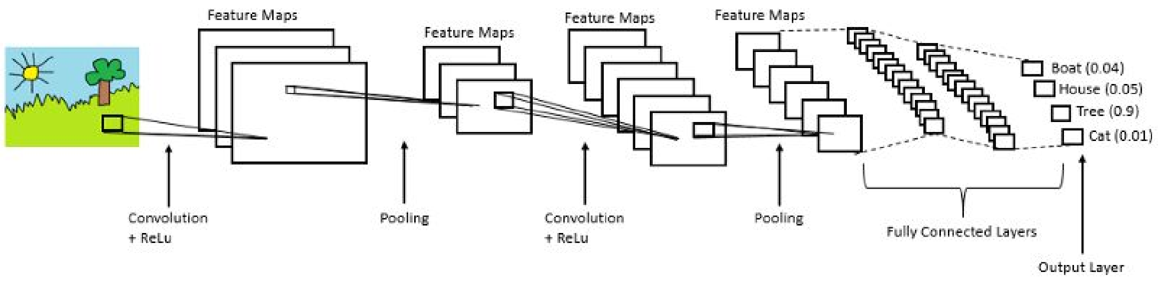


Fig. 1.18 Complete CNN architecture

In short, CNN provide input image into convolution layer, choose parameters, apply filters with strides, padding if requires. Perform convolution on the image and apply ReLU activation to the matrix, perform pooling to reduce dimensionality size, add as many convolutional layers until satisfied, flatten the output and feed into a fully connected layer (FC Layer), output the class using an activation function (Logistic Regression with cost functions) and classifies images.

1.5 Keras

Keras is an open source neural network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, or Theano. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), and its primary author and maintainer is François Chollet, a Google engineer. In 2017, Google's TensorFlow team decided to support Keras in TensorFlow core library. Chollet explained that Keras was conceived to be an interface rather than a standalone machine-learning framework. It offers a higher-level, more intuitive set of abstractions that make it easy to develop deep learning models regardless of the computational backend used. Microsoft added a CNTK backend to Keras as well, available as of CNTK v2.0. Keras contains numerous implementations of commonly used neural network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier. The code is hosted on GitHub, and community support forums include the GitHub issues page, and a Slack channel. Keras allows users to productize deep models on smartphones (iOS and Android), on the web, or on the Java Virtual Machine. It also allows use of distributed training of deep learning models on clusters of Graphics Processing Units (GPU) and Tensor processing units (TPU).

Fig. 1.19 Keras

1.6 Transfer Learning

Transfer Learning is the reuse of a pre-trained model on a new problem. It is currently very popular in the field of Deep Learning because it enables you to train Deep Neural Networks with comparatively little data. This is very useful since most real-world problems typically do not have millions of labeled data points to train such complex models. This blog post is intended to give you an overview of what Transfer Learning is, how it works, why you should use it and when you can use it. It will introduce you to the different approaches of Transfer Learning and provide you with some resources on already pre-trained models. In Transfer Learning, the knowledge of an already trained Machine Learning model is applied to a different but related problem. For example, if you trained a simple classifier to predict whether an image contains a backpack, you could use the knowledge that the model gained during its training to recognize other objects like sunglasses. With transfer learning, we basically try to exploit what has been learned in one task to improve generalization in another. We transfer the weights that a Network has learned at Task A to a new Task B. The general idea is to use knowledge, that a model has learned from a task where a lot of labeled training data is available, in a new task where we don't have a lot of data. Instead of starting the learning process from scratch, you start from patterns that have been learned from solving a related task. Transfer Learning is mostly used in Computer Vision and Natural Language Processing Tasks like Sentiment Analysis, because of the huge amount of computational power that is needed for them. It is not really a Machine Learning technique. Transfer Learning can be seen as a 'design methodology' within Machine Learning like for example, active learning. It is also not an exclusive part or study-area of Machine Learning. Nevertheless, it has become quite popular in the combination with Neural Networks, since they require huge amounts of data and computational power. For example, in computer vision, Neural Networks usually try to detect edges in their earlier layers, shapes in their middle layer and some task-specific features in the later layers. With transfer learning, you use the early and middle layers and only retrain the latter layers. It helps us to leverage the labeled data of the task it was initially trained on. Let's go back to the example of a model trained for recognizing a backpack on an Image, which will be used to identify Sunglasses. In the earlier layers, the model has learned to recognize objects and because of that, we will only re-train the latter layers, so that it will learn what separates sunglasses from other objects.

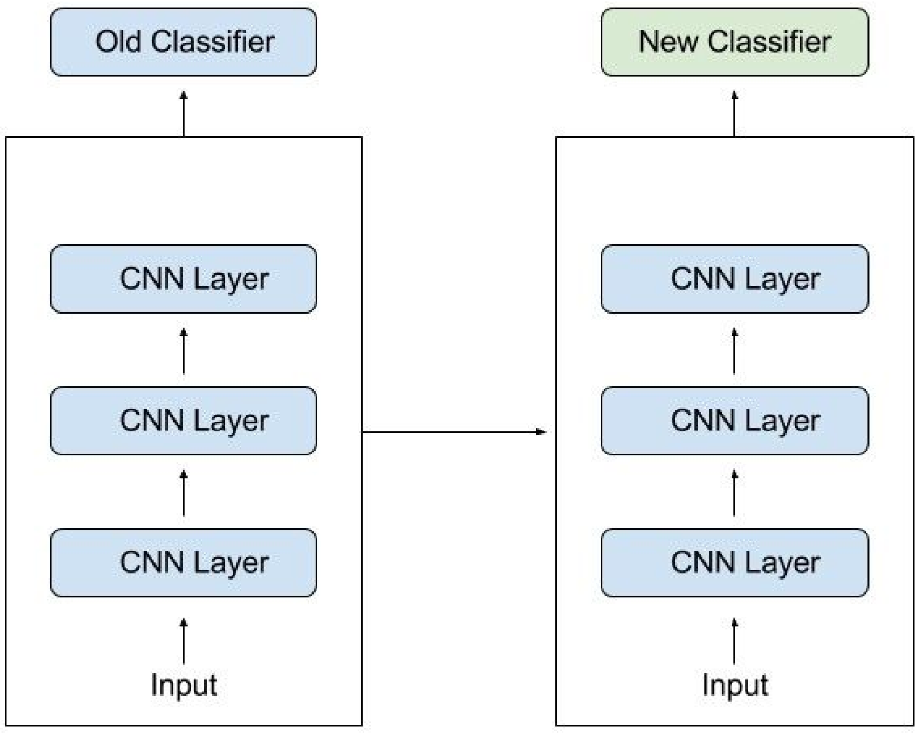


Fig. 1.20 Representation of Transfer Learning

In Transfer Learning, we try to transfer as much knowledge as possible from the previous task, the model was trained on, to the new task at hand. This knowledge can be in various forms depending on the problem and the data. For example, it could be how models are composed which would allow us to more easily identify novel objects. Using Transfer Learning has several benefits that we will discuss in this section. The main advantages are basically that you save training time, that your Neural Network performs better in most cases and that you don't need a lot of data. Usually, you need a lot of data to train a Neural Network from scratch but you don't always have access to enough data. That is where Transfer Learning comes into play because with it you can build a solid machine Learning model with comparatively little training data because the model is already pre-trained. This is especially valuable in Natural Language Processing (NLP) because there is mostly expert knowledge required to created large labeled datasets. Therefore you also save a lot of training time, because it can sometimes take days or even weeks to train a deep Neural Network from scratch on a complex task. As it is always the case in Machine Learning, it is hard to form rules that are generally applicable. But I will provide you with some guidelines. You would typically use Transfer Learning when (a) you don't have enough labeled training data to train your network from scratch and/or (b) there already exists a network that is pre-trained on a similar task, which is usually trained on massive amounts of data. Another case where its use would be appropriate is when Task-1 and Task-2 have the same input. If the original model was trained using TensorFlow, you can simply restore it and re-train some layers for your task. Note that Transfer Learning only works if the features learned from the first task are general, meaning that they can be useful for another related task as well. Also, the input of the model needs to have the same size as it was initially trained with. If you don't have that, you need to add a preprocessing step to resize your input to the needed size. Now we will discuss different approaches to Transfer Learning. Note that these have different names throughout literature but the overall concept is mostly the same. Imagine you want to solve Task A but don't have enough data to train a Deep Neural Network. One way around this issue would be to find a related Task B, where you have an abundance of data. Then you could train a Deep Neural Network on Task B and use this model as starting point to solve your initial Task A. If you have to use the whole model or only a few layers of it, depends heavily on the problem you are trying to solve. If you have the same input in both Tasks, you could maybe just reuse the model and make predictions for your new input. Alternatively, you could also just change and re-train different task-specific layers and the output layer. Approach 2 would be to use an already pre-trained model. There are a lot of these models out there, so you have to do a little bit of research. How many layers you reuse and how many you are training again, depends like I already said on your problem and it is therefore hard to form a general rule. Keras, for example, provides nine pre-trained models that you can use for Transfer Learning, Prediction, feature extraction and fine-tuning. There are also many research institutions that released models they have trained. This type of Transfer Learning is most commonly used throughout Deep Learning. Another approach is to use Deep Learning to discover the best representation of your problem, which means finding the most important features. This approach is also known as Representation Learning and can often result in a much better performance than can be obtained with hand-designed representation.

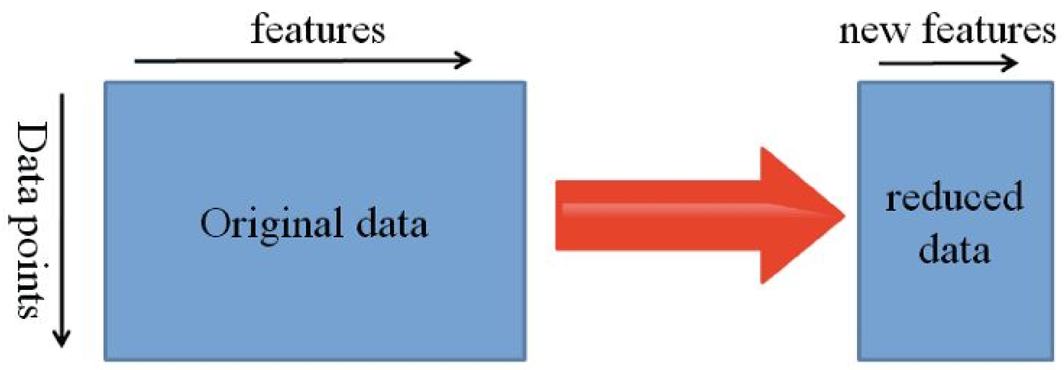


Fig. 1.21 Representation Learning

Most of the time in Machine Learning, features are manually hand-crafted by researchers and domain experts. Fortunately, Deep Learning can extract features automatically. Note that this does not mean that Feature Engineering and Domain knowledge isn’t important anymore because you still have to decide which features you put into your Network. But Neural Networks have the ability to learn which features, you have put into it, are really important and which ones aren’t. A representation learning algorithm can discover a good combination of features within a very short timeframe, even for complex tasks which would otherwise require a lot of human effort. The learned representation can then be used for other problems as well. You simply use the first layers to spot the right representation of features but you don’t use the output of the network because it is too task-specific. Simply feed data into your network and use one of the intermediate layers as the output layer. This layer can then be interpreted as a representation of the raw data. This approach is mostly used in Computer Vision because it can reduce the size of your dataset, which decreases computation time and makes it more suitable for traditional algorithms as well.

Chapter-2

RELATED WORK

2.1 American Sign Language Interpreter System for Deaf and Dumb Individuals

The proposed scheme recognizes and interprets American Sign Language symbols that have static gestures. The system provides an opportunity for deaf and dumb individuals to communicate and learn using computers. American Sign Language is a widely used and accepted standard for communication by people with hearing and speaking impairments. The proposed system recognizes and translates static hand gesture of alphabets in ASL into textual output. This text can further be converted into speech. The user of the system is free from data acquisition devices. The concepts of Principal Component Analysis (PCA) are used on the static gesture images of the ASL alphabet. The PCA features extracted from the image are used to classify the image into one of the ASL alphabet. The recognition of ASL gestures results in a textual output and is then can be converted into speech. Thus, the scheme helps the hearing and speech impaired to talk using computers. The aim of the work is to develop a scheme to recognize ASL gesture of alphabets for speech and hearing impaired. The system recognizes ASL alphabets without any data acquisition devices. The recognition of hand gestures of ASL results in a textual output and can be translated to speech. Thus, the developed gives the ability for deaf and dumb individuals to communicate and learn using computers.

2.2 American Sign Language Interpreter

There is currently an abundance of software in the market that is used to help teach people sign language. This software is very effective; however it is very difficult to know if you are making correct hand gestures. It is quite tedious to learn from the software hence making them less user friendly. In our paper we are discussing an effective use of glove for implementing an interactive sign language teaching programme. We believe that, with the use of glove, signs can be made with more accuracy and better consistency.

Having a glove also would enable us to practice sign language without having to be next to a computer. The sign language glove seems to be very useful to aid in communication with the deaf. We believe that such a translator could be an effective teacher to those who want to learn sign language. The inspiration behind this project came from the thought of helping to alleviate the language barrier which stands between the deaf and hearing communities. Attempting to translate fingerspelling to spoken English alphabet was just a minute step towards achieving this ultimate goal. The resulting translator program achieved this desired step.We have introduced the concept of a one-way sign to spoken translator as a suitable task for sign language recognition research. We have demonstrated a preliminary system for phrase- level sign recognition with a per sign accuracy of 94%, suggesting that sufficiently large vocabularies and high enough accuracies are possible for a mobile one way American Sign Language to English translator in a limited domain. Future research in this area would attempt to tackle the development of a wearable electronic translator that would recognize sign language and translate it to spoken English. A related research endeavour would focus on translating English into animated American Sign Language. This project was meant to be a prototype to check the feasibility of recognizing sign languages using sensor gloves. The completion of this prototype suggests that sensor gloves can be used for partial sign language recognition.

2.3 Sign Language Translator and Gesture Recognition

The mute/deaf individuals have a communication problem dealing with other people. It is hard for such individuals to express what they want to say since sign language is not understandable by everyone. This paper is to develop a Data Acquisition and Control (DAC) system that translates the sign language into text that can be read by anyone. This system is called Sign Language Translator and Gesture Recognition. We developed a smart glove that captures the gesture of the hand and interprets these gestures into readable text. This text can be sent wirelessly to a smart phone or shown in an embedded LCD display. It is evident from the experimental results that gestures can be captured by set of inexpensive sensors, which measure the positions and the orientation of the fingers. The current version of the system is able to interpret 20 out of 26 letters with a recognition accuracy of 96%. This paper presents an automatic hand-sign language translator – a critical system for mute/deaf individuals. Expected requirements and level of performances of such system are addressed here. The paper list system components and extends the explanations of hardware wire connections and components assembly. Furthermore, the software part of this system is extensively elaborated to include simple system initialization and recognition algorithms. The paper addresses the challenges of identifying ambiguous measurements and proposes respective technical solutions. It is evident from the experimental results that the system has the potential to help targeted individuals and communities. Especially that the system was able to recognize most of the letters (20 out of 26), and get to an average accuracy of 96%. We look forward to add the remaining letters to the system and better the system performance.

2.4 Sign Language Interpreter Using A Smart Glove

Sign language is the communication medium for the deaf and the mute people. It uses hand gestures along with the facial expressions and the body language to convey the intended message. This paper proposes a novel approach of interpreting the sign language using the portable smart glove. LED-LDR pair on each finger senses the signing gesture and couples the analog voltage to the microcontroller. The microcontroller MSP430G2553 converts these analog voltage values to digital samples and the ASCII code of the letter gestured is wirelessly transmitted using the ZigBee. Upon reception, the letter corresponding to the received ASCII code is displayed on the computer and the corresponding audio is played. In this paper, the smart glove approach proposed is meant to be a prototype to check the feasibility of recognizing sign languages using smart gloves. The completion of this prototype suggests that smart gloves can be used for partial sign language recognition. The system is composed of two main components. First is a sign language recognition part, which has the smart glove to recognize the signing gesture and create text representation of the signed gesture. Second is a translation part which converts text to audio output. With the ability to recognize sign languages especially natural forms, the deaf community will not be left behind as the industry moves away from the standard text inputs.

2.5 Machine Learning Model for Sign Language Interpretation using Webcam Images

Human beings interact with each other either using a natural language channel such as words, writing, or by body language (gestures) e.g. hand gestures, head gestures, facial expression, lip motion and so on. As understanding natural language is important, understanding sign language is also very important. The sign language is the basic communication method within hearing disabled people. People with hearing disabilities face problems in communicating with other hearing people without a translator. For this reason, the implementation of a system that recognize the sign language would have a significant benefit impact on deaf people social live. In this paper, we have proposed a marker-free, visual Indian Sign Language recognition system using image processing, computer vision and neural network methodologies, to identify the characteristics of the hand in images taken from a video through web camera. This approach will convert video of daily frequently used full sentences gesture into a text and then convert it into audio. Identification of hand shape from continuous frames will be done by using series of image processing operations. Interpretation of signs and corresponding meaning will be identified by using Haar Cascade Classifier. Finally displayed text will be converted into speech using speech synthesizer. The prediction using Haar Cascade Classifier for Indian Sign Language recognition is presented in this paper. The system interprets signs made in front of computer in textual and audio format. This method proves to be much faster than any other former methods like support vector machine, hidden markov model, backpropagation algorithm and k-nearest neighbor method. The convergence rate is also faster it improves the speed and accuracy rate of sign language interpretation within system model for real time system. The biggest benefit of this system is that the training setup can be done before the real use of system. Therefore it reduces processing power and increases efficiency by interpreting faster during testing time. According to the statistics system processes almost all frames correctly with the average accuracy speed of 92.68%.This system can be expanded further by adding whole dataset of ISL gestures.

2.6 A Conceptual Structure for Computer Vision

The research presented in this paper represents several novel conceptual contributions to the computer vision literature. In this position paper, our goal is to define the scope of computer vision analysis and discuss a new categorisation of the computer vision problem. We first provide a novel decomposition of computer vision into base components which we term the *axioms of vision*. These are used to define researcher-level and developer-level access to vision algorithms, in a way which does not require expert knowledge of computer vision. We discuss a new line of thought for computer vision by basing analyses on descriptions of the problem instead of in terms of algorithms. From this an abstraction can be developed to provide a layer above algorithmic details. This is extended to the idea of a formal description language which may be automatically interpreted thus allowing those not familiar with computer vision techniques to utilise sophisticated methods. Viewing the challenges of vision from the problem space will allow us to design systems which open up access to computer vision techniques to a much wider audience. In this paper we have presented the *axioms of vision*, a decomposition of the computer vision problem as a whole into small components which encapsulate core concepts. These axioms provide descriptions of their concept, and the implementations (units) provide functionality associated with the descriptions. Based on the axioms we can construct more sophisticated algorithms, maintain flexibility for research into new algorithms and promote re-use of existing code. We also presented novel paradigms of thought for the computer vision problem. If we think in terms of the problem itself and its associated conditions, we can highlight new areas of research which haven't been explored as well as providing a coherent model which can be used by those not specialised in vision. The formal description model could extend to any problem in vision. We have presented the idea of an interpreter which would take the formal description and translate it such that the correct algorithm for the problem may be chosen. We are currently working on all areas presented in this paper: we are implementing the units with the axiom descriptions to form the basis of an open source computer vision library; we are developing formal description models for many computer vision problems such as registration, tracking, correspondence and decomposition; we are also researching methods for automatic problem description extraction from images.

2.7 A Brief Introduction to OpenCV

The purpose of this paper is to introduce and quickly make a reader familiar with OpenCV (Open Source Computer Vision) basics without having to go through the lengthy reference manuals and books. OpenCV is an open source library for image and video analysis, originally introduced more than decade ago by Intel. Since then, a number of programmers have contributed to the most recent library developments. The latest major change took place in 2009 (OpenCV 2) which includes main changes to the C++ interface. Nowadays the library has >2500 optimized algorithms. It is extensively used around the world, having >2.5M downloads and >40K people in the user group. Regardless of whether one is a novice C++ programmer or a professional software developer, unaware of OpenCV, the main library content should be interesting for the graduate students and researchers in image processing and computer vision areas. To master every library element it is necessary to consult many books available on the topic of OpenCV. However, reading such more comprehensive material should be easier after comprehending some basics about OpenCV from this paper. The purpose of this paper is to quickly make a reader familiar with OpenCV basics without having to go through lengthy reference manuals and books. Given the total number of OpenCV implemented algorithms (~thousands) and possible challenges in Computer Vision in general, it was normally beyond the scope of this paper to go in depth about every possible OpenCV detail. Actually, some more advanced topics, such as the use of GPU accelerated codes, were not even mentioned. However, the paper did present many basic and popular Computer Vision algorithms, along with many key references for an interested reader to pursue further details. The shown content should raise the interest and strengthen the awareness about OpenCV among the graduate students and researchers in image processing and computer vision areas as a whole, who may not to be aware of it yet and/or its practical users. It is important to note that OpenCV is considered by many to be side by side with many commercial image processing packages, and yet it is an open source tool. Furthermore thanks to the fact that OpenCV keeps evolving is an additional guarantee that it will advance research in vision and promote the development of rich, vision-based CPU- intensive applications.

Chapter-3

SYSTEM ANALYSIS AND DESIGN

3.1 Software Requirement Specification

3.11 Requirements Gathering

The project needed several requirements to be gathered before proceeding to design. One among them includes gathering information about the project prerequisites that would be easy for a normal user to use and easy to understand the goal of the project. The functionality requirements of the project are simple and straightforward.

3.12 Requirements Specification

3.121 Software Requirements

Below are the software requirements for the project:-

Language	: Python
Tools	: Jupyter Notebook, Terminal
Technologies used	: Python, Machine Learning

3.122 Hardware Requirements

Processor	: Nvidia Class 10 GPU
Other Hardware Required	: Webcam

3.2 Use Case Diagram

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. Hence, when a system is analyzed to gather its functionalities, use cases are prepared and actors are identified.

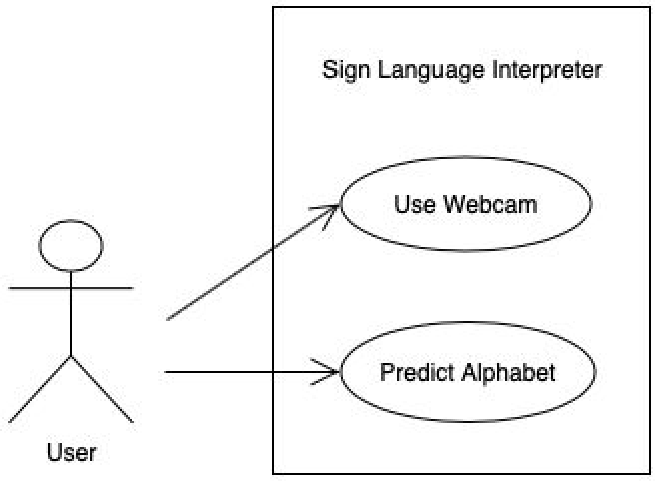


Fig. 3.1 Use Case Diagram

3.3 ER Diagram

ER-modeling is a data modeling technique used in software engineering to produce a conceptual data model of an information system. Diagrams created using this ER-modeling technique are called Entity-Relationship Diagrams, or ER diagrams or ERDs.

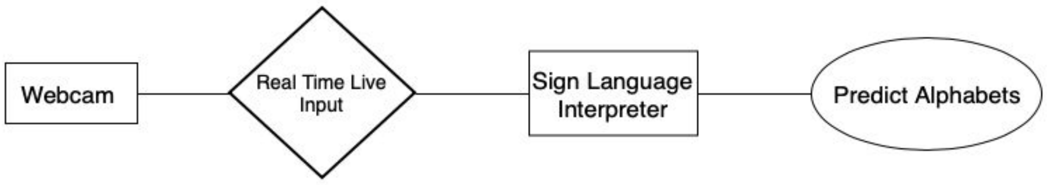


Fig. 3.2 E-R Diagram

3.4 Data Flow Diagram (DFD)

3.41 DFD - 0 LEVEL

Highest abstraction level DFD is known as Level 0 DFD, which depicts the entire information system as one diagram concealing all the underlying details. Level 0 DFDs are also known as context level DFDs.

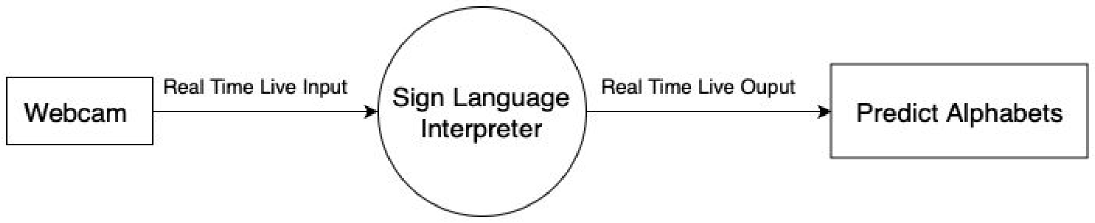


Fig. 3.3 0-Level DFD

3.42 DFD -1 LEVEL

Level 1 DFD depicts basic modules in the system and flow of data among various modules. Level 1 DFD also mentions basic processes and sources of information.

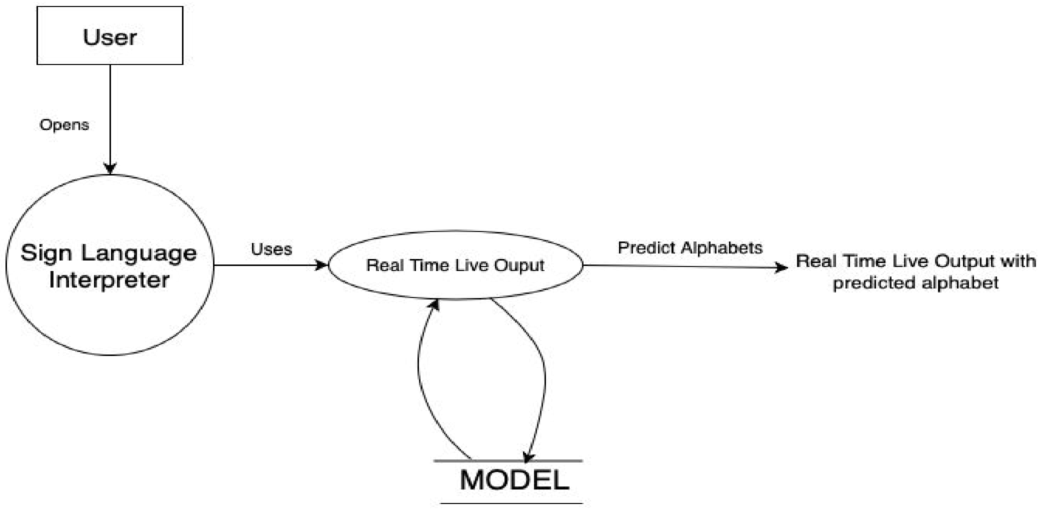

User
Opens
Sign Language Interpreter
Uses
Real Time Live Ouput
Predict Alphabets
Real Time Live Output with predicted alphabet
MODEL

Fig. 3.4 1-Level DFD

Chapter - 4

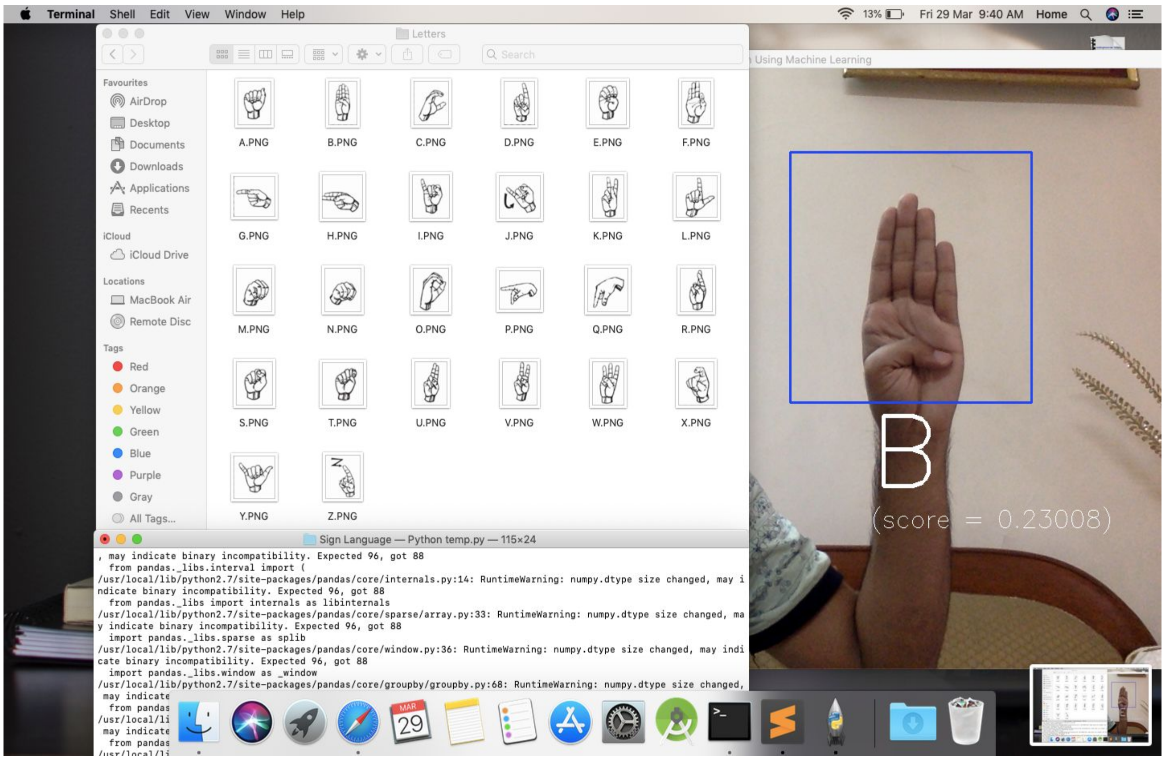
PROPOSED WORK

After doing research and study, we came up with an idea of building a Sign Language Interpreter, a smart tool, that is capable of predicting the alphabets by analysing the sign made by the user. Sign language is a visual language that is used by deaf and dumb people as their mother tongue. Unlike acoustically conveyed sound patterns, sign language uses body language and manual communication to fluidly convey the thoughts of a person. It is achieved by simultaneously combining hand shapes, orientation and movement of the hands, arms or body, and facial expressions. Hence, this project is needed to fathom the feelings of the hearing/speaking disabled persons. It is of utmost importance to comprehend whatever they want to convey because they belong to that section of the society which is most of the times left unheard. There are very few people in this world who are accustomed to sign language and henceforth can understand the deaf-dumbs' emotions. It is our duty to reach out to them, listen to them, befriend them, fulfill their needs and most importantly help them out if they are in any trouble. At the outset we must be acquainted with the sign language, which is accomplished by this project. The project will be capable of elucidating the user's gestures into all the 26 alphabets of English language. These gestures can then be processed by the system to obtain some meaning out of it.

Chapter – 5

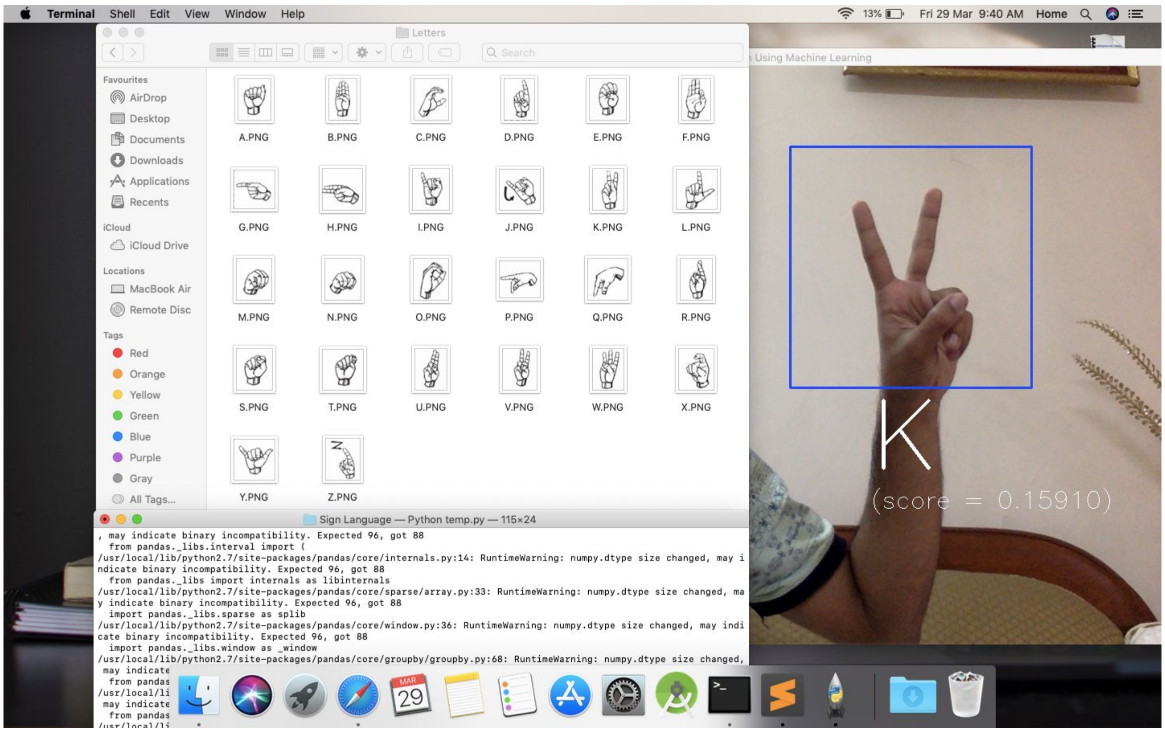
IMPLEMENTATION AND RESULTS

This section highlights the implemented work and the results of the project. Below are the snapshots of the outputs generated by the project.



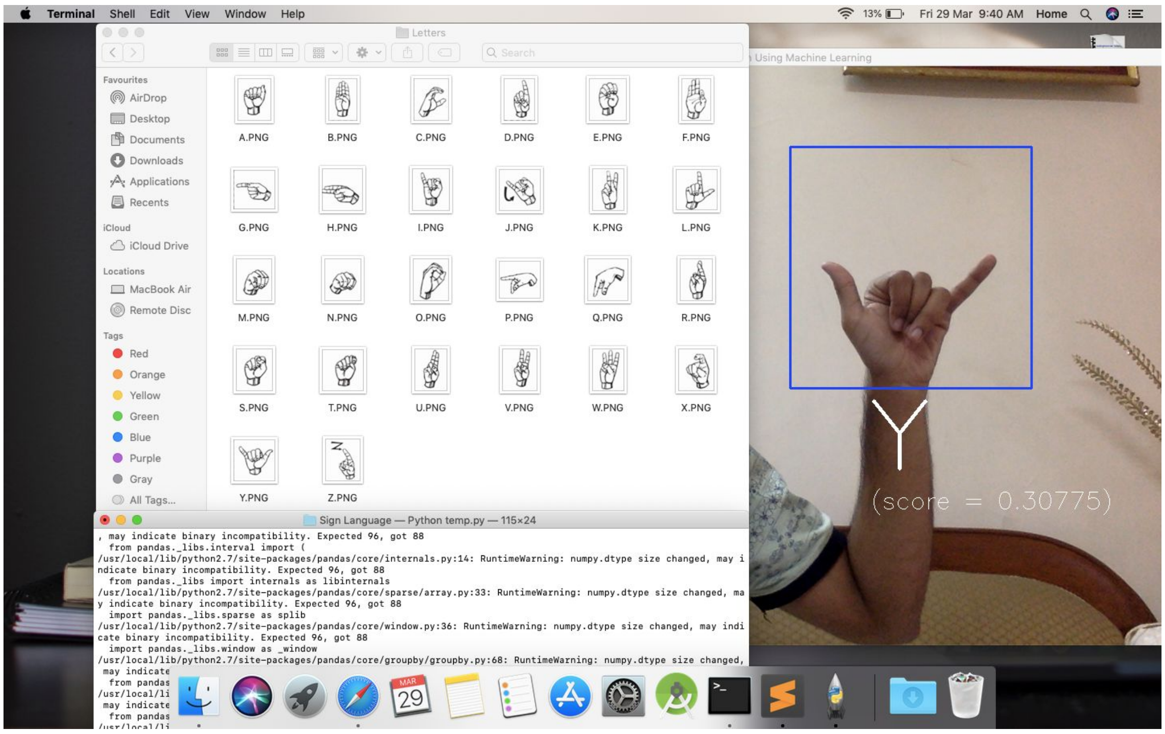
```
, may indicate binary incompatibility. Expected 96, got 88
  from pandas._libs.interval import (
/usr/local/lib/python2.7/site-packages/pandas/core/internals.py:14: RuntimeWarning: numpy.dtype size changed, may i
ndicate binary incompatibility. Expected 96, got 88
  from pandas._libs import internals as libinternals
/usr/local/lib/python2.7/site-packages/pandas/core/sparse/array.py:33: RuntimeWarning: numpy.dtype size changed, ma
y indicate binary incompatibility. Expected 96, got 88
  import pandas._libs.sparse as splib
/usr/local/lib/python2.7/site-packages/pandas/core/window.py:36: RuntimeWarning: numpy.dtype size changed, may indi
cate binary incompatibility. Expected 96, got 88
  import pandas._libs.window as _window
/usr/local/lib/python2.7/site-packages/pandas/core/groupby/groupby.py:68: RuntimeWarning: numpy.dtype size changed,
 may indicate
  from pandas
/usr/local/li
 may indicate
  from pandas
```

Fig. 5.1 Sign Language Interpreter Results



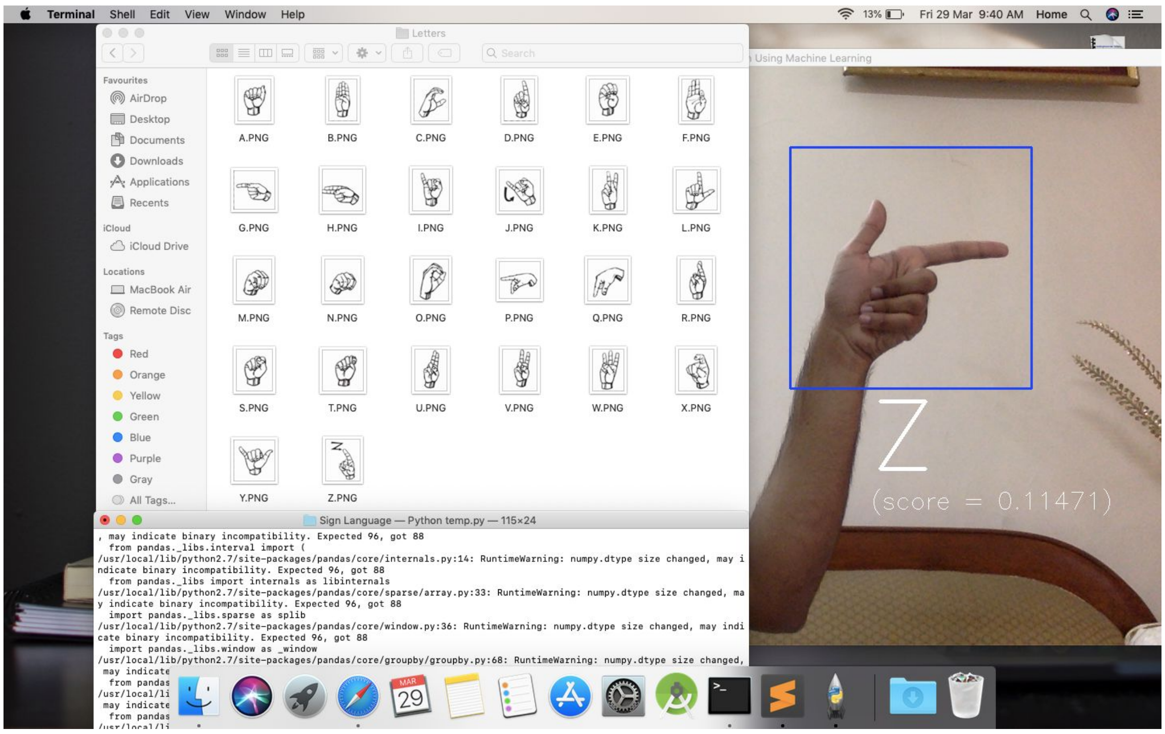
```
Sign Language — Python temp.py — 115×24
, may indicate binary incompatibility. Expected 96, got 88
  from pandas._libs.interval import (
/usr/local/lib/python2.7/site-packages/pandas/core/internals.py:14: RuntimeWarning: numpy.dtype size changed, may i
ndicate binary incompatibility. Expected 96, got 88
  from pandas._libs import internals as libinternals
/usr/local/lib/python2.7/site-packages/pandas/core/sparse/array.py:33: RuntimeWarning: numpy.dtype size changed, ma
y indicate binary incompatibility. Expected 96, got 88
  import pandas._libs.sparse as splib
/usr/local/lib/python2.7/site-packages/pandas/core/window.py:36: RuntimeWarning: numpy.dtype size changed, may indi
cate binary incompatibility. Expected 96, got 88
  import pandas._libs.window as _window
/usr/local/lib/python2.7/site-packages/pandas/core/groupby/groupby.py:68: RuntimeWarning: numpy.dtype size changed,
 may indicate
  from pandas
/usr/local/li
 may indicate
  from pandas
```

Fig. 5.2 Sign Language Interpreter Results



```
Sign Language — Python temp.py — 115×24
, may indicate binary incompatibility. Expected 96, got 88
  from pandas._libs.interval import (
/usr/local/lib/python2.7/site-packages/pandas/core/internals.py:14: RuntimeWarning: numpy.dtype size changed, may i
ndicate binary incompatibility. Expected 96, got 88
  from pandas._libs import internals as libinternals
/usr/local/lib/python2.7/site-packages/pandas/core/sparse/array.py:33: RuntimeWarning: numpy.dtype size changed, ma
y indicate binary incompatibility. Expected 96, got 88
  import pandas._libs.sparse as splib
/usr/local/lib/python2.7/site-packages/pandas/core/window.py:36: RuntimeWarning: numpy.dtype size changed, may indi
cate binary incompatibility. Expected 96, got 88
  import pandas._libs.window as _window
/usr/local/lib/python2.7/site-packages/pandas/core/groupby/groupby.py:68: RuntimeWarning: numpy.dtype size changed,
 may indicate
  from pandas
/usr/local/li
 may indicate
  from pandas
```

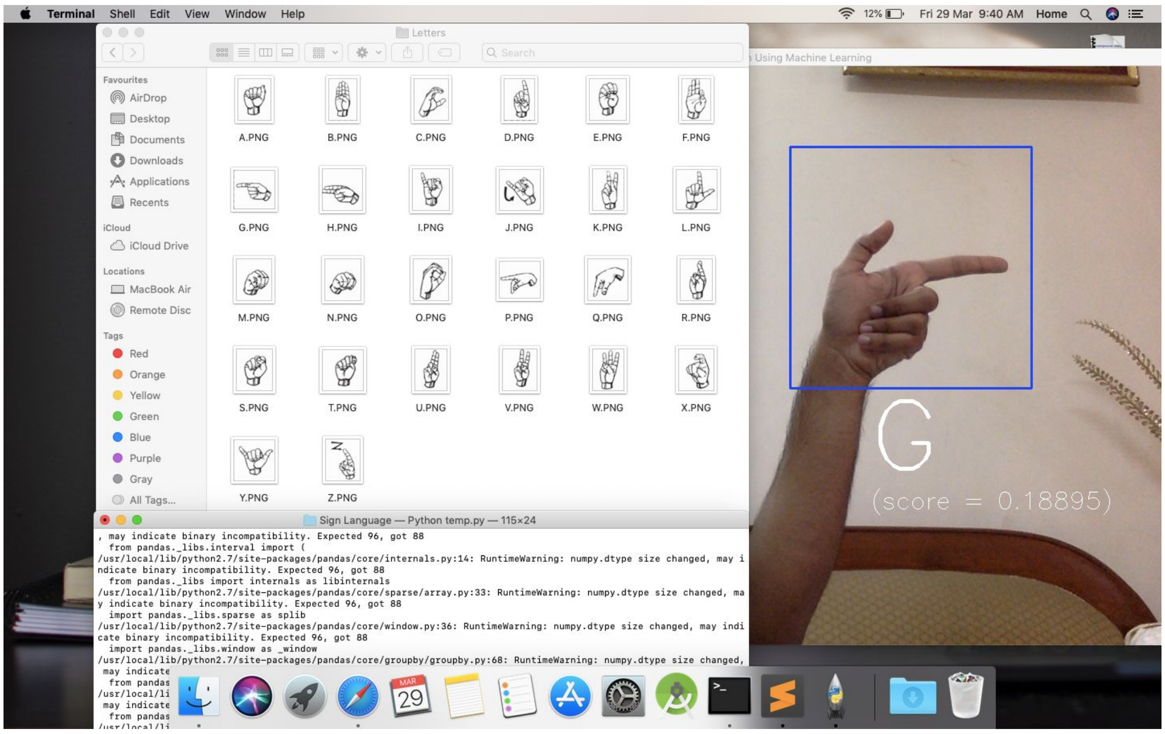
Fig. 5.3 Sign Language Interpreter Results


Terminal Shell Edit View Window Help
Fri 29 Mar 9:40 AM
Letters
A.PNG B.PNG C.PNG D.PNG E.PNG F.PNG
G.PNG H.PNG I.PNG J.PNG K.PNG L.PNG
M.PNG N.PNG O.PNG P.PNG Q.PNG R.PNG
S.PNG T.PNG U.PNG V.PNG W.PNG X.PNG
Y.PNG Z.PNG
Z
(score = 0.11471)

Sign Language — Python temp.py — 115×24

```
, may indicate binary incompatibility. Expected 96, got 88
  from pandas._libs.interval import (
/usr/local/lib/python2.7/site-packages/pandas/core/internals.py:14: RuntimeWarning: numpy.dtype size changed, may i
ndicate binary incompatibility. Expected 96, got 88
  from pandas._libs import internals as libinternals
/usr/local/lib/python2.7/site-packages/pandas/core/sparse/array.py:33: RuntimeWarning: numpy.dtype size changed, ma
y indicate binary incompatibility. Expected 96, got 88
  import pandas._libs.sparse as splib
/usr/local/lib/python2.7/site-packages/pandas/core/window.py:36: RuntimeWarning: numpy.dtype size changed, may indi
cate binary incompatibility. Expected 96, got 88
  import pandas._libs.window as _window
/usr/local/lib/python2.7/site-packages/pandas/core/groupby/groupby.py:68: RuntimeWarning: numpy.dtype size changed,
 may indicate
  from pandas
/usr/local/li
 may indicate
  from pandas
```

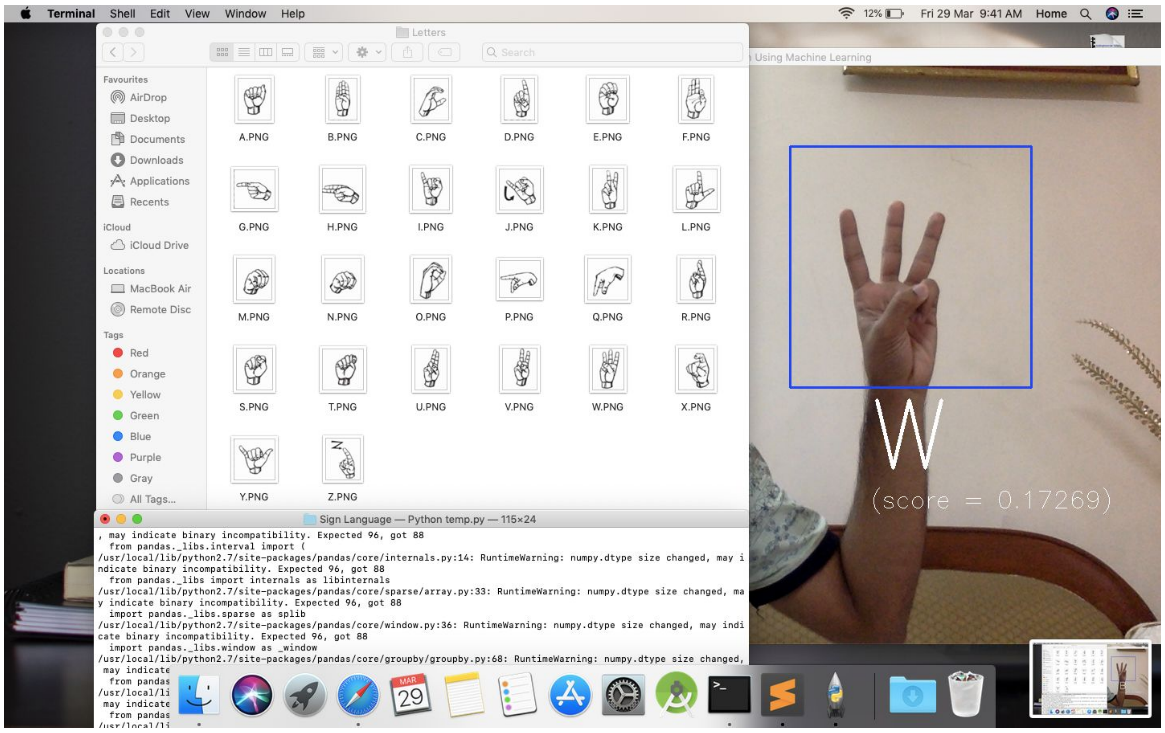
Fig. 5.4 Sign Language Interpreter Results


Terminal Shell Edit View Window Help
Fri 29 Mar 9:40 AM
Letters
A.PNG B.PNG C.PNG D.PNG E.PNG F.PNG
G.PNG H.PNG I.PNG J.PNG K.PNG L.PNG
M.PNG N.PNG O.PNG P.PNG Q.PNG R.PNG
S.PNG T.PNG U.PNG V.PNG W.PNG X.PNG
Y.PNG Z.PNG
G
(score = 0.18895)

Sign Language — Python temp.py — 115×24

```
, may indicate binary incompatibility. Expected 96, got 88
  from pandas._libs.interval import (
/usr/local/lib/python2.7/site-packages/pandas/core/internals.py:14: RuntimeWarning: numpy.dtype size changed, may i
ndicate binary incompatibility. Expected 96, got 88
  from pandas._libs import internals as libinternals
/usr/local/lib/python2.7/site-packages/pandas/core/sparse/array.py:33: RuntimeWarning: numpy.dtype size changed, ma
y indicate binary incompatibility. Expected 96, got 88
  import pandas._libs.sparse as splib
/usr/local/lib/python2.7/site-packages/pandas/core/window.py:36: RuntimeWarning: numpy.dtype size changed, may indi
cate binary incompatibility. Expected 96, got 88
  import pandas._libs.window as _window
/usr/local/lib/python2.7/site-packages/pandas/core/groupby/groupby.py:68: RuntimeWarning: numpy.dtype size changed,
 may indicate
  from pandas
/usr/local/li
 may indicate
  from pandas
```

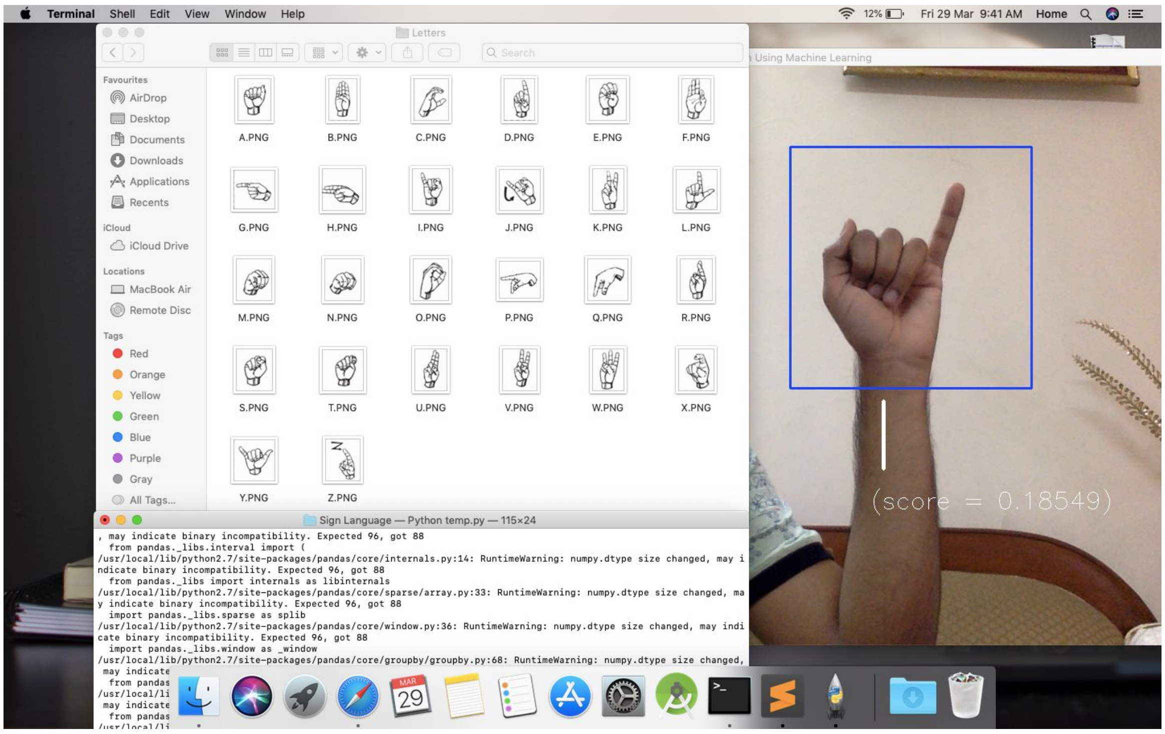
Fig. 5.5 Sign Language Interpreter Results


W
(score = 0.17269)
Letters
A.PNG B.PNG C.PNG D.PNG E.PNG F.PNG
G.PNG H.PNG I.PNG J.PNG K.PNG L.PNG
M.PNG N.PNG O.PNG P.PNG Q.PNG R.PNG
S.PNG T.PNG U.PNG V.PNG W.PNG X.PNG
Y.PNG Z.PNG

Sign Language — Python temp.py — 115×24

```
, may indicate binary incompatibility. Expected 96, got 88
  from pandas._libs.interval import (
/usr/local/lib/python2.7/site-packages/pandas/core/internals.py:14: RuntimeWarning: numpy.dtype size changed, may i
ndicate binary incompatibility. Expected 96, got 88
  from pandas._libs import internals as libinternals
/usr/local/lib/python2.7/site-packages/pandas/core/sparse/array.py:33: RuntimeWarning: numpy.dtype size changed, ma
y indicate binary incompatibility. Expected 96, got 88
  import pandas._libs.sparse as splib
/usr/local/lib/python2.7/site-packages/pandas/core/window.py:36: RuntimeWarning: numpy.dtype size changed, may indi
cate binary incompatibility. Expected 96, got 88
  import pandas._libs.window as _window
/usr/local/lib/python2.7/site-packages/pandas/core/groupby/groupby.py:68: RuntimeWarning: numpy.dtype size changed,
 may indicate
  from pandas
/usr/local/li
 may indicate
  from pandas
```

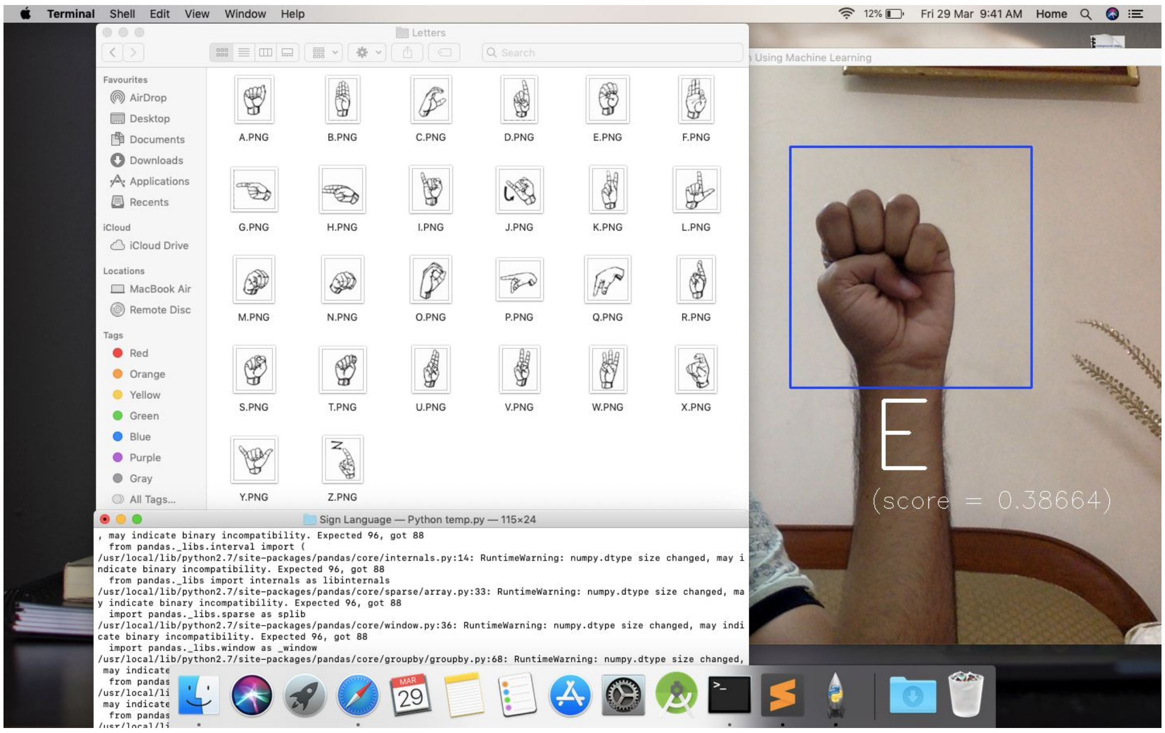
Fig. 5.6 Sign Language Interpreter Results


I
(score = 0.18549)
Letters
A.PNG B.PNG C.PNG D.PNG E.PNG F.PNG
G.PNG H.PNG I.PNG J.PNG K.PNG L.PNG
M.PNG N.PNG O.PNG P.PNG Q.PNG R.PNG
S.PNG T.PNG U.PNG V.PNG W.PNG X.PNG
Y.PNG Z.PNG

Sign Language — Python temp.py — 115×24

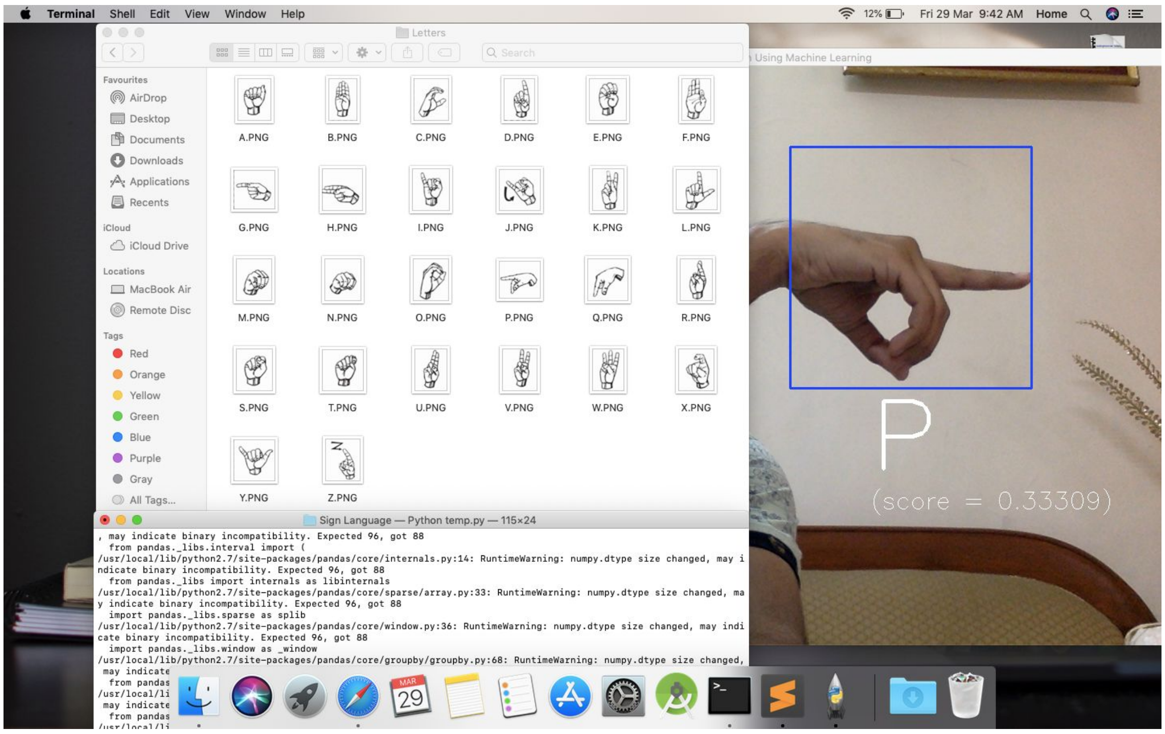
```
, may indicate binary incompatibility. Expected 96, got 88
  from pandas._libs.interval import (
/usr/local/lib/python2.7/site-packages/pandas/core/internals.py:14: RuntimeWarning: numpy.dtype size changed, may i
ndicate binary incompatibility. Expected 96, got 88
  from pandas._libs import internals as libinternals
/usr/local/lib/python2.7/site-packages/pandas/core/sparse/array.py:33: RuntimeWarning: numpy.dtype size changed, ma
y indicate binary incompatibility. Expected 96, got 88
  import pandas._libs.sparse as splib
/usr/local/lib/python2.7/site-packages/pandas/core/window.py:36: RuntimeWarning: numpy.dtype size changed, may indi
cate binary incompatibility. Expected 96, got 88
  import pandas._libs.window as _window
/usr/local/lib/python2.7/site-packages/pandas/core/groupby/groupby.py:68: RuntimeWarning: numpy.dtype size changed,
 may indicate
  from pandas
/usr/local/li
 may indicate
  from pandas
```

Fig. 5.7 Sign Language Interpreter Results


Terminal Shell Edit View Window Help
Fri 29 Mar 9:41 AM
Letters
A.PNG B.PNG C.PNG D.PNG E.PNG F.PNG
G.PNG H.PNG I.PNG J.PNG K.PNG L.PNG
M.PNG N.PNG O.PNG P.PNG Q.PNG R.PNG
S.PNG T.PNG U.PNG V.PNG W.PNG X.PNG
Y.PNG Z.PNG
Sign Language — Python temp.py — 115×24
E
(score = 0.38664)

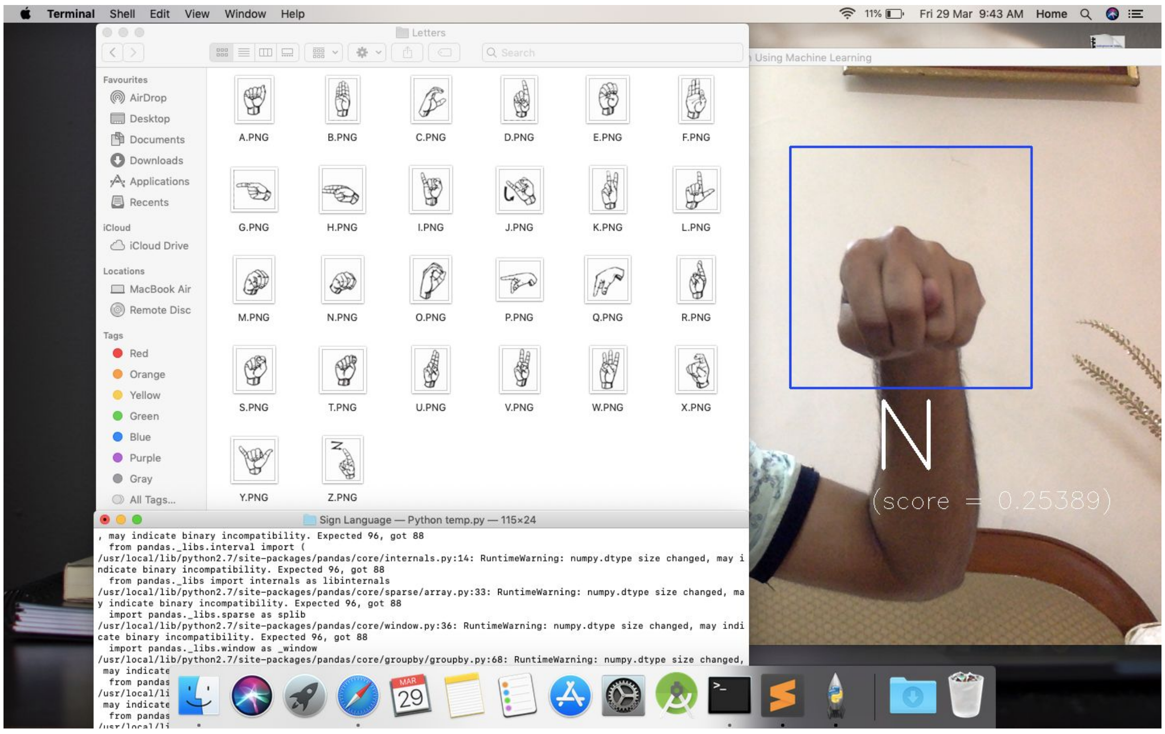
```
, may indicate binary incompatibility. Expected 96, got 88
  from pandas._libs.interval import (
/usr/local/lib/python2.7/site-packages/pandas/core/internals.py:14: RuntimeWarning: numpy.dtype size changed, may i
ndicate binary incompatibility. Expected 96, got 88
  from pandas._libs import internals as libinternals
/usr/local/lib/python2.7/site-packages/pandas/core/sparse/array.py:33: RuntimeWarning: numpy.dtype size changed, ma
y indicate binary incompatibility. Expected 96, got 88
  import pandas._libs.sparse as splib
/usr/local/lib/python2.7/site-packages/pandas/core/window.py:36: RuntimeWarning: numpy.dtype size changed, may indi
cate binary incompatibility. Expected 96, got 88
  import pandas._libs.window as _window
/usr/local/lib/python2.7/site-packages/pandas/core/groupby/groupby.py:68: RuntimeWarning: numpy.dtype size changed,
 may indicate
  from pandas
/usr/local/li
 may indicate
  from pandas
```

Fig. 5.8 Sign Language Interpreter Results


Terminal Shell Edit View Window Help
Fri 29 Mar 9:42 AM
Letters
A.PNG B.PNG C.PNG D.PNG E.PNG F.PNG
G.PNG H.PNG I.PNG J.PNG K.PNG L.PNG
M.PNG N.PNG O.PNG P.PNG Q.PNG R.PNG
S.PNG T.PNG U.PNG V.PNG W.PNG X.PNG
Y.PNG Z.PNG
Sign Language — Python temp.py — 115×24
P
(score = 0.33309)

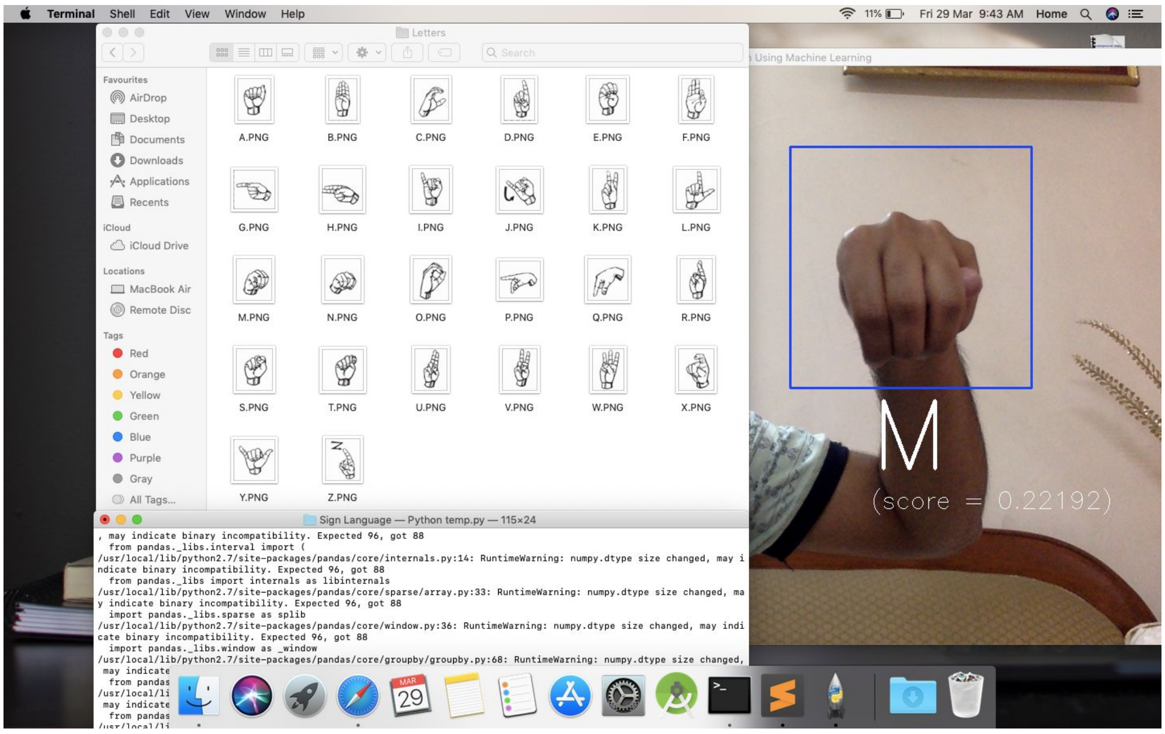
```
, may indicate binary incompatibility. Expected 96, got 88
  from pandas._libs.interval import (
/usr/local/lib/python2.7/site-packages/pandas/core/internals.py:14: RuntimeWarning: numpy.dtype size changed, may i
ndicate binary incompatibility. Expected 96, got 88
  from pandas._libs import internals as libinternals
/usr/local/lib/python2.7/site-packages/pandas/core/sparse/array.py:33: RuntimeWarning: numpy.dtype size changed, ma
y indicate binary incompatibility. Expected 96, got 88
  import pandas._libs.sparse as splib
/usr/local/lib/python2.7/site-packages/pandas/core/window.py:36: RuntimeWarning: numpy.dtype size changed, may indi
cate binary incompatibility. Expected 96, got 88
  import pandas._libs.window as _window
/usr/local/lib/python2.7/site-packages/pandas/core/groupby/groupby.py:68: RuntimeWarning: numpy.dtype size changed,
 may indicate
  from pandas
/usr/local/li
 may indicate
  from pandas
```

Fig. 5.9 Sign Language Interpreter Results



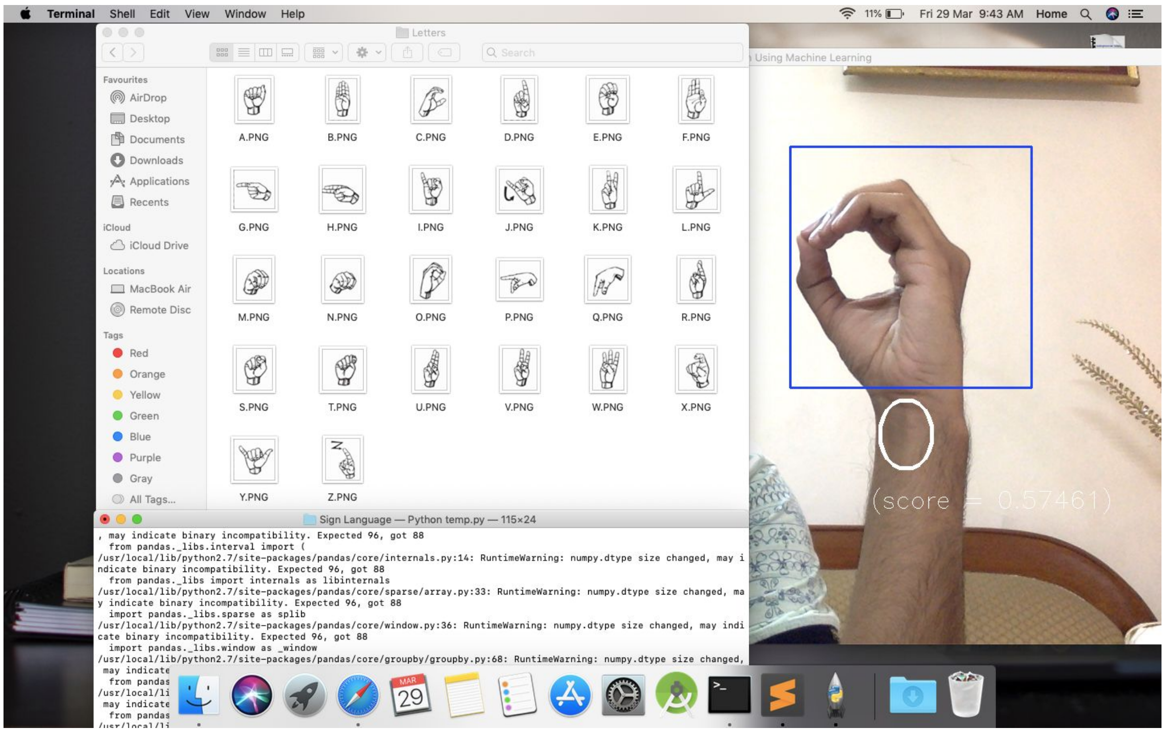
```
, may indicate binary incompatibility. Expected 96, got 88
  from pandas._libs.interval import (
/usr/local/lib/python2.7/site-packages/pandas/core/internals.py:14: RuntimeWarning: numpy.dtype size changed, may i
ndicate binary incompatibility. Expected 96, got 88
  from pandas._libs import internals as libinternals
/usr/local/lib/python2.7/site-packages/pandas/core/sparse/array.py:33: RuntimeWarning: numpy.dtype size changed, ma
y indicate binary incompatibility. Expected 96, got 88
  import pandas._libs.sparse as splib
/usr/local/lib/python2.7/site-packages/pandas/core/window.py:36: RuntimeWarning: numpy.dtype size changed, may indi
cate binary incompatibility. Expected 96, got 88
  import pandas._libs.window as _window
/usr/local/lib/python2.7/site-packages/pandas/core/groupby/groupby.py:68: RuntimeWarning: numpy.dtype size changed,
 may indicate
  from pandas
/usr/local/li
 may indicate
  from pandas
```

Fig. 5.10 Sign Language Interpreter Results



```
, may indicate binary incompatibility. Expected 96, got 88
  from pandas._libs.interval import (
/usr/local/lib/python2.7/site-packages/pandas/core/internals.py:14: RuntimeWarning: numpy.dtype size changed, may i
ndicate binary incompatibility. Expected 96, got 88
  from pandas._libs import internals as libinternals
/usr/local/lib/python2.7/site-packages/pandas/core/sparse/array.py:33: RuntimeWarning: numpy.dtype size changed, ma
y indicate binary incompatibility. Expected 96, got 88
  import pandas._libs.sparse as splib
/usr/local/lib/python2.7/site-packages/pandas/core/window.py:36: RuntimeWarning: numpy.dtype size changed, may indi
cate binary incompatibility. Expected 96, got 88
  import pandas._libs.window as _window
/usr/local/lib/python2.7/site-packages/pandas/core/groupby/groupby.py:68: RuntimeWarning: numpy.dtype size changed,
 may indicate
  from pandas
/usr/local/li
 may indicate
  from pandas
```

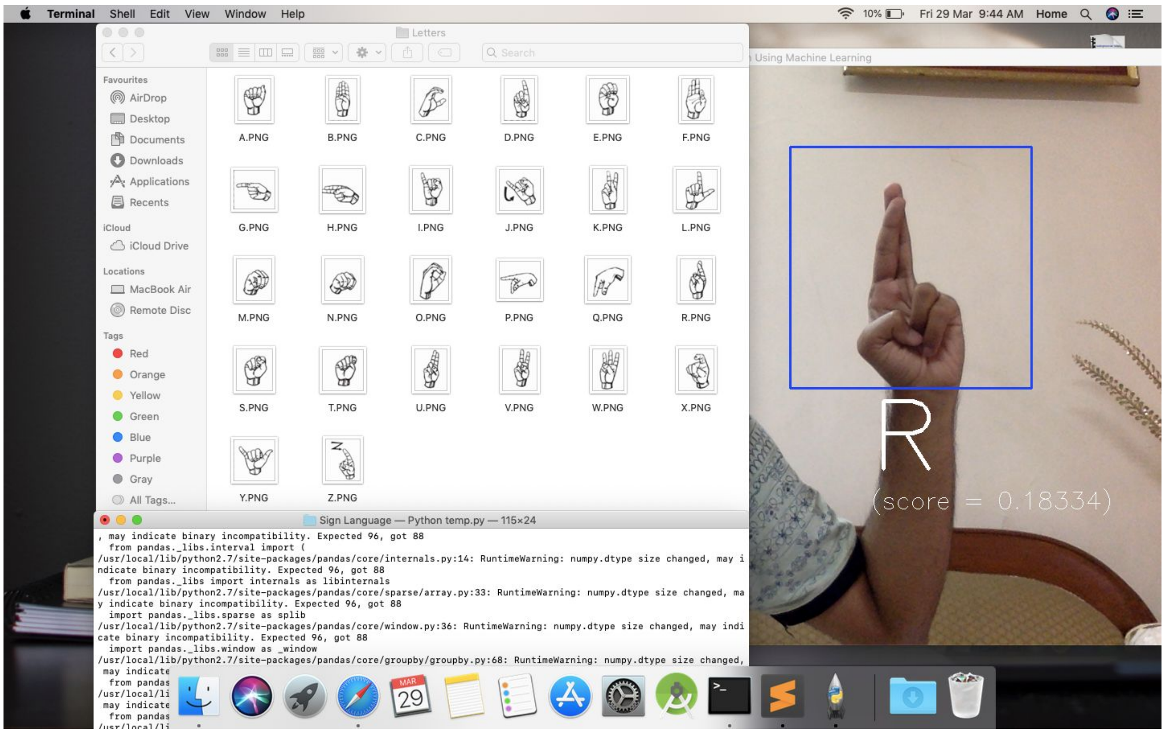
Fig. 5.11 Sign Language Interpreter Results



Sign Language — Python temp.py — 115×24

```
, may indicate binary incompatibility. Expected 96, got 88
  from pandas._libs.interval import (
/usr/local/lib/python2.7/site-packages/pandas/core/internals.py:14: RuntimeWarning: numpy.dtype size changed, may i
ndicate binary incompatibility. Expected 96, got 88
  from pandas._libs import internals as libinternals
/usr/local/lib/python2.7/site-packages/pandas/core/sparse/array.py:33: RuntimeWarning: numpy.dtype size changed, ma
y indicate binary incompatibility. Expected 96, got 88
  import pandas._libs.sparse as splib
/usr/local/lib/python2.7/site-packages/pandas/core/window.py:36: RuntimeWarning: numpy.dtype size changed, may indi
cate binary incompatibility. Expected 96, got 88
  import pandas._libs.window as _window
/usr/local/lib/python2.7/site-packages/pandas/core/groupby/groupby.py:68: RuntimeWarning: numpy.dtype size changed,
 may indicate
  from pandas
/usr/local/li
 may indicate
  from pandas
```

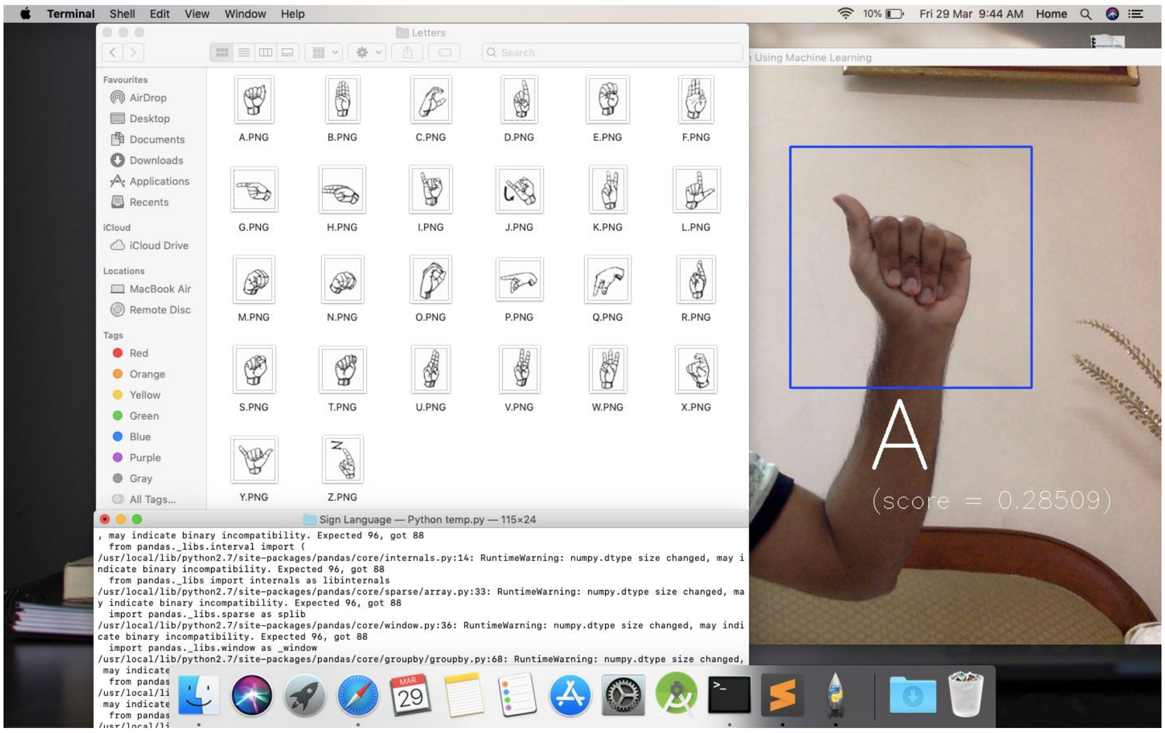
Fig. 5.12 Sign Language Interpreter Results



Sign Language — Python temp.py — 115×24

```
, may indicate binary incompatibility. Expected 96, got 88
  from pandas._libs.interval import (
/usr/local/lib/python2.7/site-packages/pandas/core/internals.py:14: RuntimeWarning: numpy.dtype size changed, may i
ndicate binary incompatibility. Expected 96, got 88
  from pandas._libs import internals as libinternals
/usr/local/lib/python2.7/site-packages/pandas/core/sparse/array.py:33: RuntimeWarning: numpy.dtype size changed, ma
y indicate binary incompatibility. Expected 96, got 88
  import pandas._libs.sparse as splib
/usr/local/lib/python2.7/site-packages/pandas/core/window.py:36: RuntimeWarning: numpy.dtype size changed, may indi
cate binary incompatibility. Expected 96, got 88
  import pandas._libs.window as _window
/usr/local/lib/python2.7/site-packages/pandas/core/groupby/groupby.py:68: RuntimeWarning: numpy.dtype size changed,
 may indicate
  from pandas
/usr/local/li
 may indicate
  from pandas
```

Fig. 5.13 Sign Language Interpreter Results



Sign Language — Python temp.py — 115×24

```
, may indicate binary incompatibility. Expected 96, got 88
  from pandas._libs.interval import (
/usr/local/lib/python2.7/site-packages/pandas/core/internals.py:14: RuntimeWarning: numpy.dtype size changed, may i
ndicate binary incompatibility. Expected 96, got 88
  from pandas._libs import internals as libinternals
/usr/local/lib/python2.7/site-packages/pandas/core/sparse/array.py:33: RuntimeWarning: numpy.dtype size changed, ma
y indicate binary incompatibility. Expected 96, got 88
  import pandas._libs.sparse as splib
/usr/local/lib/python2.7/site-packages/pandas/core/window.py:36: RuntimeWarning: numpy.dtype size changed, may indi
cate binary incompatibility. Expected 96, got 88
  import pandas._libs.window as _window
/usr/local/lib/python2.7/site-packages/pandas/core/groupby/groupby.py:68: RuntimeWarning: numpy.dtype size changed,
 may indicate
  from pandas
/usr/local/li
 may indicate
  from pandas
```

Fig. 5.14 Sign Language Interpreter Results

Chapter-6

CONCLUSION

The aim of the project was to classify the alphabet by predicting the sign made by the user in real time. By now we have successfully implemented the complete project that is recognising hand from a real time live input and predicting the alphabet made by the user . The project is currently working on real time live input without any flaw or discrepancy, that too in a quite fast speed. A lot of effort was put to make it work efficiently. The project has been designed, implemented and tested with real devices by users successfully. The project helped in understanding the challenges involved in developing a machine learning project, the ways to overcome them and in better understanding the intricacies of project development process. The project also helped in understanding the value of designing the components of overall project before implementing them. The project has also taught me programming skills and refining the design and implementation logic of the software at every phase of the development life cycle to improve the overall performance of the project.

Chapter-7

FUTURE SCOPE

This project can be extended further in many possible ways and some of them are listed below:

1. The project could be optimized so that it require less space, time and low specification hardware to run.
2. It could be trained on a large dataset to improve accuracy.
3. Current shortcomings would be fixed in the later updates.

REFERENCES

[1] https://medium.com/datadriveninvestor/what-is-machine-learning-55028d8bdd53

[2] https://medium.com/deep-math-machine-learning-ai/different-types-of-machine-learning-and-their-types-34760b9128a2

[3] https://www.quora.com/What-is-the-difference-between-regression-classification-and-clustering-in-machine-learning

[4] https://medium.com/cracking-the-data-science-interview/a-tour-of-the-top-10-algorithms-for-machine-learning-newbies-7228aa8ef541

[5] https://medium.com/intro-to-artificial-intelligence/deep-learning-series-1-intro-to-deep-learning-abb1780ee20

[6] https://playground.tensorflow.org/

[7] https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

[8] https://keras.io

[9] https://towardsdatascience.com/transfer-learning-946518f95666

[10] Understanding of a Convolutional Neural Network : Saad ALBAWI , Tareq Abed MOHAMMED, Saad AL-ZAWI; **ICET 2017, Antalya, Turkey**

[11] American Sign Language Interpreter : Kunal Kadam , Rucha Ganu , Ankita Bhosekar , Prof.S.D.Joshi : **2012 IEEE Fourth International Conference on Technology for Education**

[12] American Sign Language Interpreter System for using Deaf and Dumb Individuals : Sruthi Upendran, Thamizharasi. A : **2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)**

[13] A brief introduction to OpenCV : Ivan Culjak, David Abram, Tomislav Pribanic, Hrvoje Dzapo, Mario Cifrek : **MIPRO 2012, May 21-25,2012, Opatija, Croatia**

[14] A Conceptual Structure for Computer Vision: Gregor Miller, Sidney Fels and Steve Oldridge : **2011 Canadian Conference on Computer and Robot Vision**

[15] Machine Learning Model for Sign Language Interpretation using Webcam Images : Kanchan Dabre, Surekha Dholay : **2014 International Conference on Circuits, Systems, Communication and Information Technology Applications (CSCITA)**

[16] Sign Language Interpreter Using A Smart Glove : Nikhita Praveen, Naveen Karanth, Megha M S : **2014 International Conference on Circuits, Systems, Communication and Information Technology Applications (CSCITA)**

[17] Sign Language Translator and Gesture Recognition : Mohammed Elmahgiubi, Mohamed Ennajar Nabil Drawil, and Mohamed Samir Elbuni : **2014 International Conference on Circuits, Systems, Communication and Information Technology Applications (CSCITA)**

[18] American Sign Language Interpreter System for Deaf and Dumb Individuals : Sruthi Upendran, Thamizharasi. A **: 2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)**

[19] American Sign Language Interpreter : Kunal Kadam, Rucha Ganu, Ankita Bhosekar, Prof.S.D.Joshi : **2012 IEEE Fourth International Conference on Technology for Education**

[20] American Sign Language Interpreter System for Deaf and Dumb Individuals : Sruthi Upendran, Thamizharasi. A : **2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)**

APPENDIX

Python Libraries

This category encompasses those Python-based libraries that are used in this project development. A summary of some key core Python libraries available to the Python developer is as follows:

1. **Cv2** - OpenCV releases two types of Python interfaces, cv and cv2. The latest one is cv2. In this, everything is returned as NumPy objects like ndarray and native Python objects like lists,tuples,dictionary, etc. So due to this NumPy support, you can do any numpy operation here. NumPy is a highly stable and fast array processing library. For example, if you load an image, an ndarray is returned.In short, with cv2 everything is simplified and pretty fast.
2. **NumPy** is the foundational library for scientific computing in Python, and many of the libraries on this list use NumPy arrays as their basic inputs and outputs. In short, NumPy introduces objects for multidimensional arrays and matrices, as well as routines that allow developers to perform advanced mathematical and statistical functions on those arrays with as little code as possible.
3. **SciPy** builds on NumPy by adding a collection of algorithms and high-level commands for manipulating and visualizing data. This package includes functions for computing integrals numerically, solving differential equations, optimization, and more.
4. **Pandas** adds data structures and tools that are designed for practical data analysis in finance, statistics, social sciences, and engineering. Pandas works well with incomplete, messy, and unlabeled data (i.e., the kind of data you're likely to encounter in the real world), and provides tools for shaping, merging, reshaping, and slicing datasets.
5. **Matplotlib** is the standard Python library for creating 2D plots and graphs. It's pretty low-level, meaning it requires more commands to generate nice-looking

graphs and figures than with some more advanced libraries. However, the flip side of that is flexibility. With enough commands, you can make just about any kind of graph you want with matplotlib.

6. **Scikit-learn** builds on NumPy and SciPy by adding a set of algorithms for common machine learning and data mining tasks, including clustering, regression, and classification. As a library, scikit-learn has a lot going for it. Its tools are well-documented and its contributors include many machine learning experts. What's more, it's a very curated library, meaning developers won't have to choose between different versions of the same algorithm. Its power and ease of use make it popular with a lot of data-heavy startups, including Evernote, OKCupid, Spotify, and Birchbox.
7. **Theano** uses NumPy-like syntax to optimize and evaluate mathematical expressions. What sets Theano apart is that it takes advantage of the computer's GPU in order to make data-intensive calculations up to 100x faster than the CPU alone. Theano's speed makes it especially valuable for deep learning and other computationally complex tasks.
8. **TensorFlow** is another high-profile entrant into machine learning, developed by Google as an open-source successor to DistBelief, their previous framework for training neural networks. TensorFlow uses a system of multi-layered nodes that allow you to quickly set up, train, and deploy artificial neural networks with large datasets. It's what allows Google to identify objects in photos or understand spoken words in its voice-recognition app.

Code

```
import sys
import os

import matplotlib
import numpy as np
import matplotlib.pyplot as plt
import copy
import cv2

# Disable tensorflow compilation warnings
os.environ['TF_CPP_MIN_LOG_LEVEL']='2'
import tensorflow as tf

face_cascade = cv2.CascadeClassifier("haarcascade_frontalface.xml")



def showSeq(seq, originalImg, grayImg):

    faces = face_cascade.detectMultiScale(grayImg, 1.3, 5)
    for(x,y,w,h) in faces:
        font = cv2.FONT_HERSHEY_SIMPLEX
         cv2.putText(originalImg, seq, (x+w, y+h), font, 0.5, (0,255,255),
2, cv2.LINE_AA)
        cv2.rectangle(originalImg, (x,y), (x+w, y+h), (255,0,0), 2)

    return originalImg



def predict(image_data):

    predictions = sess.run(softmax_tensor, \
             {'DecodeJpeg/contents:0': image_data})

    # Sort to show labels of first prediction in order of confidence
    top_k = predictions[0].argsort()[-len(predictions[0]):][::-1]
    max_score = 0.0
    res = ''
    for node_id in top_k:
        human_string = label_lines[node_id]
        score = predictions[0][node_id]
        if score > max_score:
            max_score = score
            res = human_string
    return res, max_score

# Loads label file, strips off carriage return
label_lines = [line.rstrip() for line
                   in tf.gfile.GFile("logs/trained_labels.txt")]

# Unpersists graph from file
with tf.gfile.FastGFile("logs/trained_graph.pb", 'rb') as f:
    graph_def = tf.GraphDef()
    graph_def.ParseFromString(f.read())
    _ = tf.import_graph_def(graph_def, name='')

with tf.Session() as sess:
    # Feed the image_data as input to the graph and get first prediction
    softmax_tensor = sess.graph.get_tensor_by_name('final_result:0')

    c = 0

    cap = cv2.VideoCapture(0)

    res, score = '', 0.0
    i = 0
    mem = ''
    consecutive = 0
    sequence = ''
    while True:
        ret, img = cap.read()
        img = cv2.flip(img, 1)
        if ret:
            #x1, y1, x2, y2 = 100, 100, 300, 300
            x1, y1, x2, y2 = 700, 100, 1000, 400
            img_cropped = img[y1:y2, x1:x2]

            c += 1
            image_data = cv2.imencode('.jpg', img_cropped)[1].tostring()
            a = cv2.waitKey(33)
            if i == 4:
                res_tmp, score = predict(image_data)
                res = res_tmp
                i = 0
                if mem == res:
                    consecutive += 1
                else:
                    consecutive = 0
                if consecutive == 2 and res not in ['nothing']:
                    if res == 'space':
                        sequence += ' '
                    elif res == 'del':
                        sequence = sequence[:-1]
                    else:
                        sequence += res
                    consecutive = 0
            i += 1
                  cv2.putText(img,  '%s'  %  (res.upper()),  (100,400),
cv2.FONT_HERSHEY_SIMPLEX, 4, (255,255,255), 4)
            cv2.putText(img, '(score = %.5f)' % (float(score)), (100,450),
cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,255))
            mem = res
            cv2.rectangle(img, (x1, y1), (x2, y2), (255,0,0), 2)
            cv2.imshow("img", img)
            img_sequence = np.zeros((200,1200,3), np.uint8)
             cv2.putText(img_sequence, '%s' % (sequence.upper()), (30,30),
cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,255), 2)
            if cv2.waitKey(1) & 0xFF == ord("q"):
                seq = sequence.upper()
                break
```

```
        else:
            break

cv2.destroyAllWindows()

while True:
    ret, img = cap.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    img = showSeq(seq, img, gray)
    cv2.imshow("img", img)

    if cv2.waitKey(1) & 0xFF == ord("q"):
        break




cv2.destroyAllWindows()
cv2.VideoCapture(0).release()
```