**INTRODUCTION**

**1. INTRODUCTION**

As well stipulated by Nelson Mandela[1], “Talk to a man in a language he understands, that goes to his head. Talk to him in his own language, that goes to his heart”, language is undoubtedly essential to human interaction and has existed since human civilisation began. It is a medium humans use to communicate to express themselves and understand notions of the real world. Without it, no books, no cell phones and definitely not any word I am writing would have any meaning. It is so deeply embedded in our everyday routine that we often take it for granted and don’t realise its importance. Sadly, in the fast changing society we live in, people with hearing impairment are usually forgotten and left out. They have to struggle to bring up their ideas, voice out their opinions and express themselves to people who are different to them. Sign language, although being a medium of communication to deaf people, still have no meaning when conveyed to a non-sign language user. Hence, broadening the communication gap.To prevent this from happening, we are putting forward a sign language recognition system. It will be an ultimate tool for people with hearing disability to communicate their thoughts as well as a very good interpretation for non sign language user to understand what the latter is saying. Many countries have their own standard and interpretation of sign gestures. For instance, an alphabet in Korean sign language will not mean the same thing as in Indian sign language. While this highlights diversity, it also pinpoints the complexity of sign languages. Deep learning must be well versed with the gestures so that we can get a decent accuracy. In our proposed system, American Sign Language is used to create our datasets. Figure 1 shows the American Sign Language (ASL) alphabets.

Identification of sign gesture is performed with either of the two methods. First is a glove based method whereby the signer wears a pair of data gloves during the capture of hand movements. Second is a vision based method, further classified into static and dynamic recognition[2]. Static deals with the 2dimensional representation of gestures while dynamic is a real time live capture of the gestures. And despite having an accuracy of over 90%[3], wearing of gloves are uncomfortable and cannot be utilised in rainy weather. They are not easily carried around since their use require computer as well. In this case, we have decided to go with the static recognition of hand gestures because it increases accuracy as compared to when including dynamic hand gestures like for the alphabets J and Z. We are proposing this research so we can improve on accuracy using Convolution Neural Network(CNN).

**LITERATURE SURVEY**

**2. LITERATURE SURVEY**

# [1] Gradient-based learning applied to document recognition

Multilayer neural networks trained with the back-propagation algorithm constitute the best example of a successful gradient based learning technique. Given an appropriate network architecture, gradient-based learning algorithms can be used to synthesize a complex decision surface that can classify high-dimensional patterns, such as handwritten characters, with minimal preprocessing. This paper reviews various methods applied to handwritten character recognition and compares them on a standard handwritten digit recognition task. Convolutional neural networks, which are specifically designed to deal with the variability of 2D shapes, are shown to outperform all other techniques. Real-life document recognition systems are composed of multiple modules including field extraction, segmentation recognition, and language modeling. A new learning paradigm, called graph transformer networks (GTN), allows such multimodule systems to be trained globally using gradient-based methods so as to minimize an overall performance measure. Two systems for online handwriting recognition are described. Experiments demonstrate the advantage of global training, and the flexibility of graph transformer networks. A graph transformer network for reading a bank cheque is also described. It uses convolutional neural network character recognizers combined with global training techniques to provide record accuracy on business and personal cheques. It is deployed commercially and reads several million cheques per day.

# [2] 3D convolutional neural networks for human action recognition

We consider the automated recognition of human actions in surveillance videos. Most current methods build classifiers based on complex handcrafted features computed from the raw inputs. Convolutional neural networks (CNNs) are a type of deep model that can act directly on the raw inputs. However, such models are currently limited to handling 2D inputs. In this paper, we develop a novel 3D CNN model for action recognition. This model extracts features from both the spatial and the temporal dimensions by performing 3D convolutions, thereby capturing the motion information encoded in multiple adjacent frames. The developed model generates multiple channels of information from the input frames, and the final feature representation combines information from all channels. To further boost the performance, we propose regularizing the outputs with high-level features and combining the predictions of a variety of different models. We apply the developed models to recognize human actions in the real-world environment of airport surveillance videos, and they achieve superior performance in comparison to baseline methods.

# [3] Isolated sign language recognition using hidden markov models

# This paper is concerned with the video-based recognition of isolated signs. Concentrating on the manual parameters of sign language, the system aims for the signer dependent recognition of 262 different signs. For hidden Markov modelling a sign is considered a doubly stochastic process, represented by an unobservable state sequence. The observations emitted by the states are regarded as feature vectors, that are extracted from video frames. The system achieves recognition rates up to 94%.

# [4] Realtime american sign language recognition using desk and wearable computer based video

We present two real-time hidden Markov model-based systems for recognizing sentence-level continuous American sign language (ASL) using a single camera to track the user's unadorned hands. The first system observes the user from a desk mounted camera and achieves 92 percent word accuracy. The second system mounts the camera in a cap worn by the user and achieves 98 percent accuracy (97 percent with an unrestricted grammar). Both experiments use a 40-word lexicon.

**[5] Parallel hidden markov models for american sign language recognition**

The major challenge that faces American Sign Language (ASL) recognition now is to develop methods that will scale well with increasing vocabulary size. Unlike in spoken languages, phonemes can occur simultaneously in ASL. The number of possible combinations of phonemes after enforcing linguistic constraints is approximately 5.5/spl times/10/sup 8/. Gesture recognition, which is less constrained than ASL recognition, suffers from the same problem. Thus, it is not feasible to train conventional hidden Markov models (HMMs) for large-scab ASL applications. Factorial HMMs and coupled HMMs are two extensions to HMMs that explicitly attempt to model several processes occuring in parallel. Unfortunately, they still require consideration of the combinations at training time. In this paper we present a novel approach to ASL recognition that aspires to being a solution to the scalability problems. It is based on parallel HMMs (PaHMMs), which model the parallel processes independently. Thus, they can also be trained independently, and do not require consideration of the different combinations at training time. We develop the recognition algorithm for PaHMMs and show that it runs in time polynomial in the number of states, and in time linear in the number of parallel processes. We run several experiments with a 22 sign vocabulary and demonstrate that PaHMMs can improve the robustness of HMM-based recognition even on a small scale. Thus, PaHMMs are a very promising general recognition scheme with applications in both gesture and ASL recognition.

**METHODOLOGY**

**3. METHODOLOGY**

**3.1 EXISTING SYSTEM**

Sign language is learned by deaf and dumb, and usually it is not known to normal people, so it becomes a challenge for communication between a normal and hearing impaired person.

Understanding the exact context of symbolic expressions of deaf and dumb people is the challenging job in real life until unless it is properly specified.

**Drawbacks:**

Very difficult to understand

**3.2 PROPOSED SYSTEM**

Communication is always having a great impact in every domain and how it is considered the meaning of the thoughts and expressions that attract the researchers to bridge this gap for every living being.

This project is to identify the symbolic expression through images so that the communication gap between a normal and hearing impaired person can be easily bridged.

In this project we proposed idea for feasible communication between hearing impaired and normal person with the help of CNN

**Advantages:**

It is helpful for hearing impaired person.

**3.3 ARCHITECTURE OF SYSTEM**

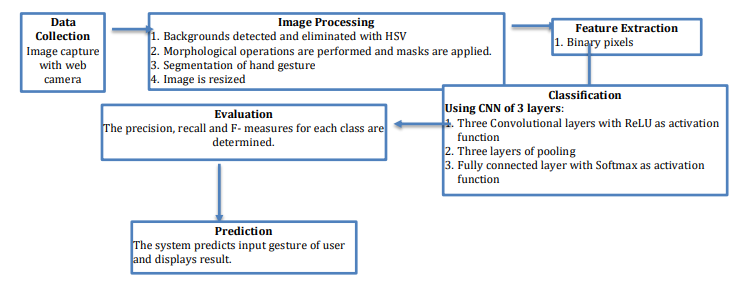


Fig 3.3 System Architecture

**3.4 DATA FLOW DIAGRAM:**

* The DFD is in like manner named as air pocket graph. It is a clear graphical representation that can be used to address a structure similarly as data information to the system, diverse taking care of finished on this information, and the yield information is delivered by this system.
* The information stream graph is a champion among the most noteworthy exhibiting gadgets. It is used to show the system parts. These parts are the structure method, the information used by the technique, an external component that works together with the system and the information streams in the system.
* DFD demonstrates how the information goes through the structure and how it is adjusted by a movement of changes. It is a graphical technique that depicts information stream and the progressions that are associated as information moves from commitment to yield.
* DFD is generally called air pocket diagram. Each DFD may be used to address a structure at any element of consideration. DFD may be partitioned into levels that address extending information stream and helpful detail.

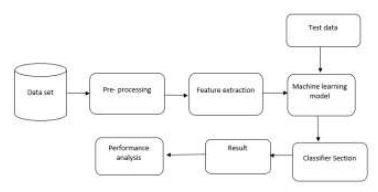


Fig 3.4 Data Flow Diagram

**3.5 UML DIAGRAMS**

* UML is refers to Unified Modeling Language. UML is a regulated comprehensively valuable showing language in the field of article orchestrated programming building.
* The major goal is for UML to transform into an average language for making models of article arranged PC programming. In its present structure UML is included two essential portions: a Meta-model and a documentation. Later on, some sort of system or method may similarly be added to; or related with, UML.
* The Unified Modeling Language is a standard language for deciding, Visualization, Constructing and chronicling the old rarities of programming system, similarly with respect to business exhibiting and other non-programming structures. The UML addresses a social event of best structuring practices that have exhibited productive in the showing of colossal and complex systems.
* The UML is a noteworthy bit of making articles masterminded programming and the item headway process. The UML uses commonly graphical documentations to deliver the arrangement of programming adventures.

**GOALS:**

The Primary goals in the arrangement of the UML are according to the accompanying:

* Give customers a readied to-use, expressive visual showing Language so they can make and exchange huge models.
* Give extendibility and specialization instruments to grow the middle thoughts.

* Be free of explicit programming vernaculars and progression process.
* Give a formal motivation to understanding the showing language.

**3.5.1 USECASE DIAGRAM:**

A usage case diagram as in their Unified Modeling Language (UML) is a fundamental kind of lead graph depicted by and made using a Use-case examination.

Its inspiration is intended to show a graphical layout of the value given by a structure to the degree on-screen characters, their goals (would in general as use cases), and any conditions between those utilization cases. The crucial explanation behind an utilization case graph is to show what structure cutoff focuses are performed for which on-screen character.. Occupations of the entertainers in the structure can be delineated.



Fig 3.5.1 Use Case Diagram

**3.5.2 CLASS DIAGRAM:**

In programming planning, a class diagram in the Unified Modeling Language (UML) is a sort of static structure plot that depicts the structure of a framework by showing the structure's classes, their properties, assignments (or systems), and the relationship among the classes. It explains which class contains information.



Fig 3.5.2 Class Diagram

**3.5.3 SEQUENCE DIAGRAM:**

A succession outline in UML is a sort of communication chart that presents how strategies work with one another and in what demand. It is a work of a Message Sequence Chart. Succession graphs are at times called event charts, event circumstances, and timing outlines.

****

Fig 3.5.3 Sequence Diagram

**3.5.4 ACTIVITY DIAGRAM**

Activity blueprints are graphical depictions of work methods of sequential exercises and activities by help for choice, cycle and simultaneousness. the Unified Modeling Language, activity charts can be utilized to depict the organization and operational all around mentioned work systems of parts in a structure. A development framework demonstrates the general development of control.

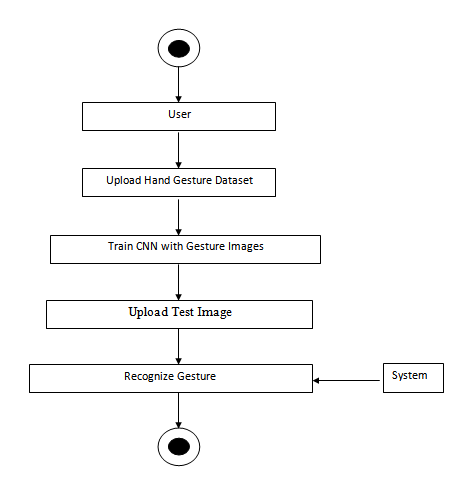


Fig 3.5.4 Activity Diagram

### 3.6 IMPLEMENTATION

**DESCSRIPTION:**

**DATA COLLECTION**

Data collection is indelibly an essential part in this research as our result highly depends on it. We have therefore created our own dataset of ASL having 2000 images of 10 static alphabet signs. We have 10 classes of static alphabets which are A,B,C,D,K,N,O,T and Y. Two datasets have been made by 2 different signers. Each of them has performed one alphabetical gesture 200 times in alternate lighting conditions. The dataset folder of alphabetic sign gestures is further split into 2 more folders, one for training and the other for testing. Out of the 2000 images captured, 1600 images are used for training and the rest for testing. To get higher consistency, we have captured the photos in the same background with a webcam each time a command is given. The images obtained are saved in the png format .It is to be pinpointed that there is no loss in quality whenever an image in png format is opened ,closed and stored again.PNG is also good in handling high contrast and detailed image. The webcam will capture the images in the RGB colourspace.

**Segmentation**

The first image is then transformed to grayscale. As much as this process will result in the loss of colour in the region of the skin gesture, it will also enhance the robustness of our system to changes in lighting or illumination. Non-black pixels in the transformed image are binarised while the others remain unchanged, therefore black.The hand gesture is segmented firstly by taking out all the joined components in the image and secondly by letting only the part which is immensely connected, in our case is the hand gesture. The frame is resized to a size of 64 by 64 pixel. At the end of the segmentation process, binary images of size 64 by 64 are obtained where the area in white represents the hand gesture, and the black coloured area is the rest.

**Feature Extraction**

One of the most crucial part in image processing is to select and extract important features from an image. Images when captured and stored as a dataset usually take up a whole lot of space as they are comprised of a huge amount of data. Feature extraction helps us solve this problem by reducing the data after having extracted the important features automatically.It also contributes in maintaining the accuracy of the classifier and simplifies its complexity. In our case, the features found to be crucial are the binary pixels of the images. Scaling the images to 64 pixels has led us to get sufficient features to effectively classify the American Sign Language gestures . In total, we have 4096 number of features, obtained after multiplying 64 by 64 pixels.

**Classification**

In our proposed system, we apply a 2D CNN model with a tensor flow library. The convolution layers scan the images with a filter of size 3 by 3. The dot product between the frame pixel and the weights of the filter are calculated. This particular step extracts important features from the input image to pass on further. The pooling layers are then applied after each convolution layer. One pooling layer decrements the activation map of the previous layer. It merges all the features that were learned in the previous layers’ activation maps. This helps to reduce overfitting of the training data and generalises the features represented by the network. In our case, the input layer of the convolutional neural network has 32 feature maps of size 3 by 3, and the activation function is a Rectified Linear Unit. The max pool layer has a size of 2×2. The dropout is set to 50 percent and the layer is flattened. The last layer of the network is a fully connected output layer with ten units, and the activation function is Softmax. Then we compile the model by using category cross-entropy as the loss function and Adam as the optimiser.

**SYSTEM STUDY**

**4. SYSTEM STUDY**

**4.1 FEASIBILITY STUDY**

The feasibility of the undertaking is broke down in this stage and strategic agreement is advanced with a general arrangement for the task and some cost appraisals. During framework investigation the feasibility investigation of the proposed framework is to be done. This is to guarantee that the proposed framework isn't a weight to the organization. For feasibility investigation, some comprehension of the significant necessities for the framework is fundamental.

Three key contemplations associated with the feasibility examination are

* + ECONOMICAL FEASIBILITY
  + TECHNICAL FEASIBILITY
  + SOCIAL FEASIBILITY

**4.1.1 ECONOMICAL FEASIBILITY**

This examination is completed to check the financial effect that the framework will have on the association. The measure of store that the organization can fill the innovative work of the framework is constrained. The consumptions must be legitimized. Hence the created framework also inside the financial limit and this was accomplished on the grounds that the majority of the innovations utilized are uninhibitedly accessible. Just the altered items must be acquired.

**4.1.2 TECHNICAL FEASIBILITY**

This examination is completed to check the specialized feasibility, that is, the specialized prerequisites of the framework. Any framework created must not have an intense interest on the accessible specialized assets. This will prompt levels of popularity on the accessible specialized assets. This will prompt levels of popularity being set on the customer. The created framework must have a humble necessity, as just negligible or invalid changes are required for actualizing this framework.

**4.1.3 SOCIAL FEASIBILITY**

The part of study is to check the dimension of acknowledgment of the framework by the client. This incorporates the way toward preparing the client to utilize the framework proficiently. The client must not feel compromised by the framework, rather should acknowledge it as a need. The dimension of acknowledgment by the clients exclusively relies upon the techniques that are utilized to instruct the client about the framework and to make him acquainted with it. His dimension of certainty must be raised with the goal that he is likewise ready to make some useful analysis, which is invited, as he is the last client of the framework.

**OPERATING ENVIRONMENT**

**5. OPERATING ENVIRONMENT**

**5.1 SOFTWARE-REQUIREMENTS**

* OPERATING SYSTEM : WINDOWS 7/8.
* CODING LANGUAGE : PYTHON

**5.2 HARDWARE-REQUIREMENTS**

* PROCESSOR : i3/i5.
* HARD DISK : 500 GB.
* RAM : 2 GB.

**SYSTEM DEVOLOPMENT**

**ENVIRONMENT**

**6. SYSTEM DEVOLOPMENT ENVIRONMENT**

**6.1 Python Introduction**

**Python** is a general purpose, dynamic, high level and interpreted programming language. It supports Object Oriented programming approach to develop applications. It is simple and easy to learn and provides lots of high-level data structures.

Python is easy to learn yet powerful and versatile scripting language which makes it attractive for Application Development.

Python's syntax and dynamic typing with its interpreted nature, makes it an ideal language for scripting and rapid application development.

Python supports multiple programming pattern, including object oriented, imperative and functional or procedural programming styles.

Python is not intended to work on special area such as web programming. That is why it is known as multipurpose because it can be used with web, enterprise, 3D CAD etc.

We don't need to use data types to declare variable because it is dynamically typed so we can write a=10 to assign an integer value in an integer variable.

Python makes the development and debugging fast because there is no compilation step included in python development and edit-test-debug cycle is very fast.

# Python History

* Python laid its foundation in the late 1980s.
* The implementation of Python was started in the December 1989 by **Guido Van Rossum** at CWI in Netherland.
* In February 1991, van Rossum published the code (labeled version 0.9.0) to alt.sources.
* In 1994, Python 1.0 was released with new features like: lambda, map, filter, and reduce.
* Python 2.0 added new features like: list comprehensions, garbage collection system.
* On December 3, 2008, Python 3.0 (also called "Py3K") was released. It was designed to rectify fundamental flaw of the language.
* ABC programming language is said to be the predecessor of Python language which was capable of Exception Handling and interfacing with Amoeba Operating System.
* Python is influenced by following programming languages:
  + ABC language.
  + Modula-3

# Python Features

Python provides lots of features that are listed below.

#### 1) Easy to Learn and Use

Python is easy to learn and use. It is developer-friendly and high level programming language.

#### 2) Expressive Language

Python language is more expressive means that it is more understandable and readable.

#### 3) Interpreted Language

Python is an interpreted language i.e. interpreter executes the code line by line at a time. This makes debugging easy and thus suitable for beginners.

#### 4) Cross-platform Language

Python can run equally on different platforms such as Windows, Linux, Unix and Macintosh etc. So, we can say that Python is a portable language.

#### 5) Free and Open Source

Python language is freely available at [offical web address](https://www.python.org/" \t "blank). The source-code is also available. Therefore it is open source.

#### 6) Object-Oriented Language

Python supports object oriented language and concepts of classes and objects come into existence.

#### 7) Extensible

It implies that other languages such as C/C++ can be used to compile the code and thus it can be used further in our python code.

#### 8) Large Standard Library

Python has a large and broad library and prvides rich set of module and functions for rapid application development.

#### 9) GUI Programming Support

Graphical user interfaces can be developed using Python.

#### 10) Integrated

It can be easily integrated with languages like C, C++, JAVA etc.

# Python Applications

Python is known for its general purpose nature that makes it applicable in almost each domain of software development. Python as a whole can be used in any sphere of development.

Here, we are specifing applications areas where python can be applied.

#### 1) Web Applications

We can use Python to develop web applications. It provides libraries to handle internet protocols such as HTML and XML, JSON, Email processing, request, beautiful Soup, Feed parser etc. It also provides Frameworks such as Django, Pyramid, Flask etc to design and delelop web based applications. Some important developments are: Python WikiEngines, Pocoo, Python Blog Software etc.

#### 2) Desktop GUI Applications

Python provides Tk GUI library to develop user interface in python based application. Some other useful toolkits wxWidgets, Kivy, pyqt that are useable on several platforms. The Kivy is popular for writing multitouch applications.

#### 3) Software Development

Python is helpful for software development process. It works as a support language and can be used for build control and management, testing etc.

#### 4) Scientific and Numeric

Python is popular and widely used in scientific and numeric computing. Some useful library and package are SciPy, Pandas, IPython etc. SciPy is group of packages of engineering, science and mathematics.

#### 5) Business Applications

Python is used to build Bussiness applications like ERP and e-commerce systems. Tryton is a high level application platform.

#### 6) Console Based Application

We can use Python to develop console based applications. For example: **IPython**.

#### 7) Audio or Video based Applications

Python is awesome to perform multiple tasks and can be used to develop multimedia applications. Some of real applications are: TimPlayer, cplay etc.

#### 8) 3D CAD Applications

To create CAD application Fandango is a real application which provides full features of CAD.

#### 9) Enterprise Applications

Python can be used to create applications which can be used within an Enterprise or an Organization. Some real time applications are: OpenErp, Tryton, Picalo etc.

#### 10) Applications for Images

Using Python several application can be developed for image. Applications developed are: VPython, Gogh, imgSeek etc.

There are several such applications which can be developed using Python

# How to Install Python (Environment Set-up)

In this section of the tutorial, we will discuss the installation of python on various operating systems.

### Why Python

* Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
* Python has a simple syntax similar to the English language.
* Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
* Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
* Python can be treated in a procedural way, an object-orientated way or a functional way.

### Good to know

* The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
* In this tutorial Python will be written in a text editor. It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

### Python Syntax compared to other programming languages

* Python was designed for readability, and has some similarities to the English language with influence from mathematics.
* Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
* Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

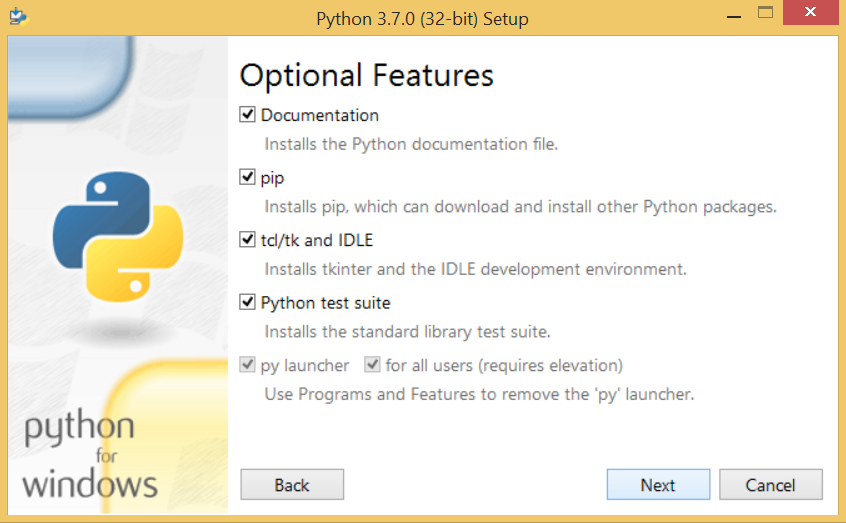
### Installation on Windows

Visit the link <https://www.python.org/downloads/> to download the latest release of Python. In this process, we will install Python 3.6.7 on our Windows operating system.

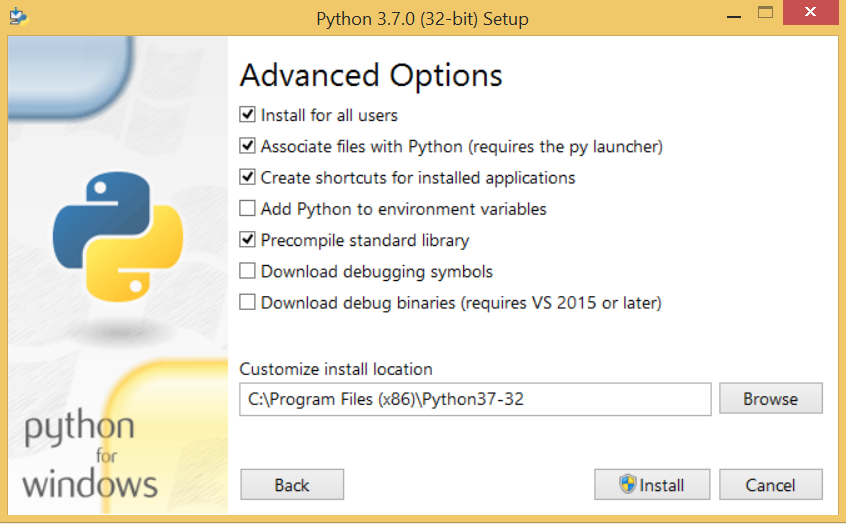
Double-click the executable file which is downloaded; the following window will open. Select Customize installation and proceed.

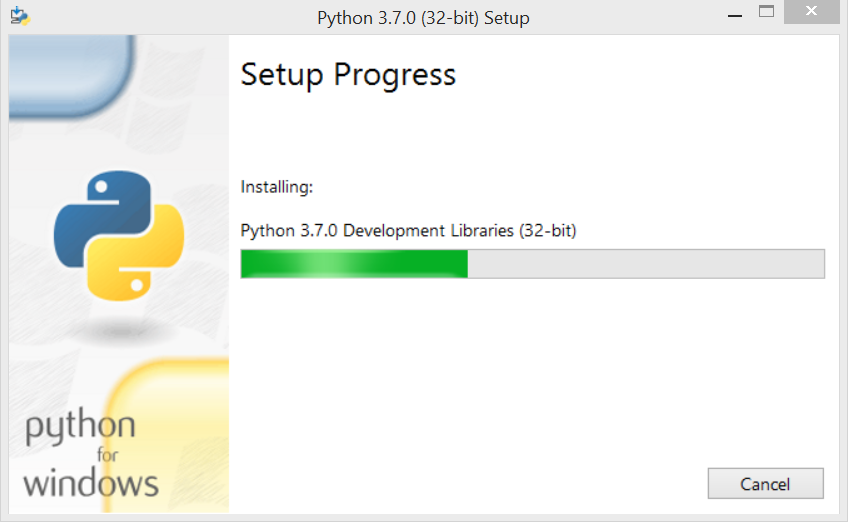
The following window shows all the optional features. All the features need to be installed and are checked by default; we need to click next to continue.

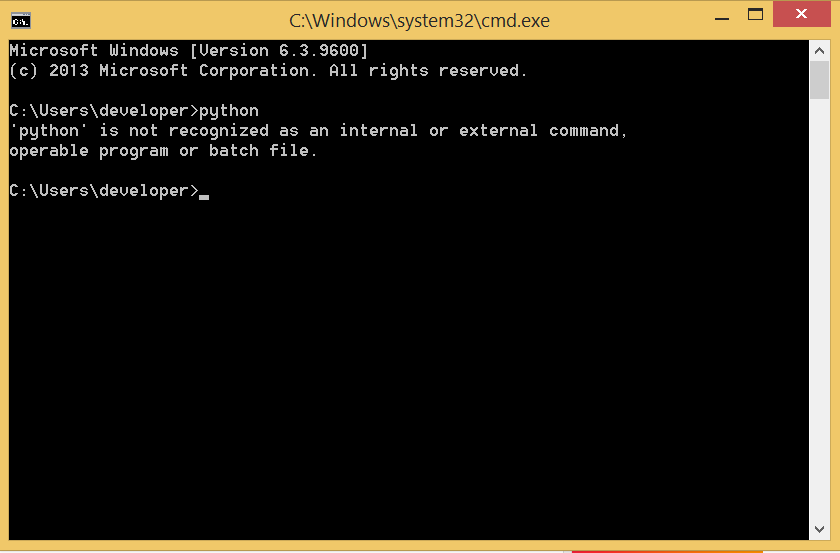
The following window shows a list of advanced options. Check all the options which you want to install and click next. Here, we must notice that the first check-box (install for all users) must be checked.



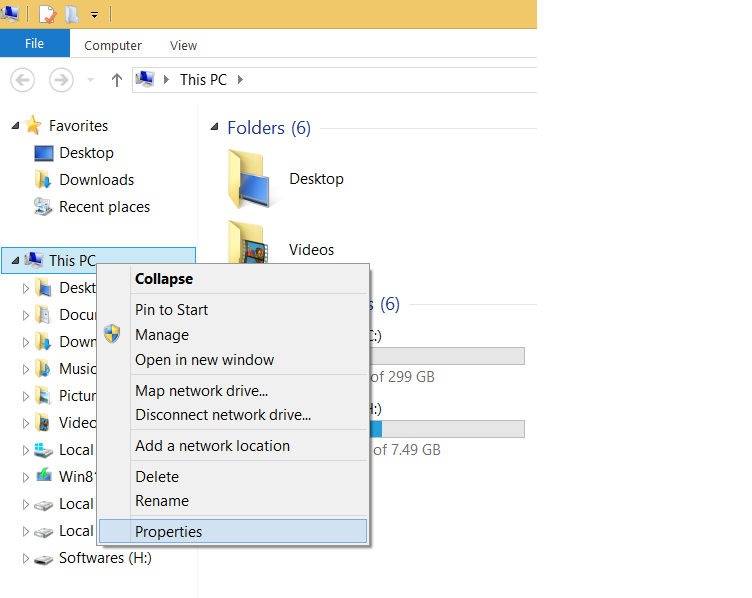
Now, we are ready to install python-3.6.6. Lets install it.

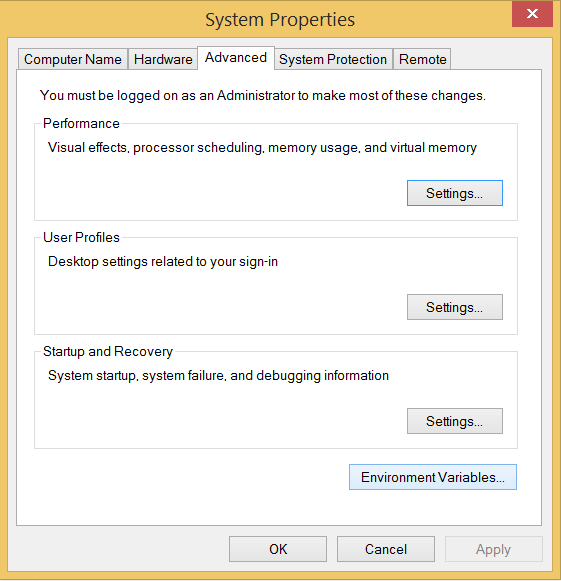


Now, try to run python on the command prompt. Type the command python in case of python2 or python3 in case of python3. It will show an error as given in the below image. It is because we haven't set the path.

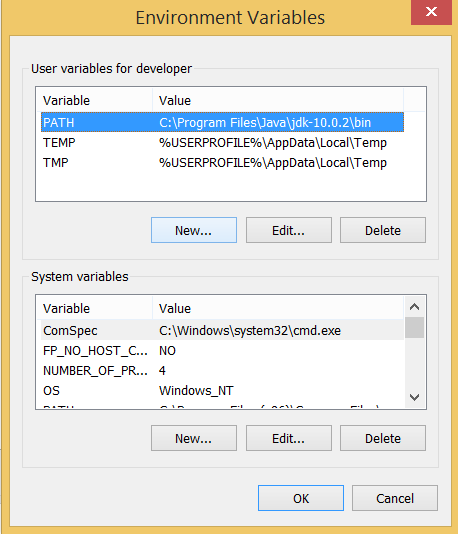


To set the path of python, we need to the right click on "my computer" and go to Properties → Advanced → Environment Variables.

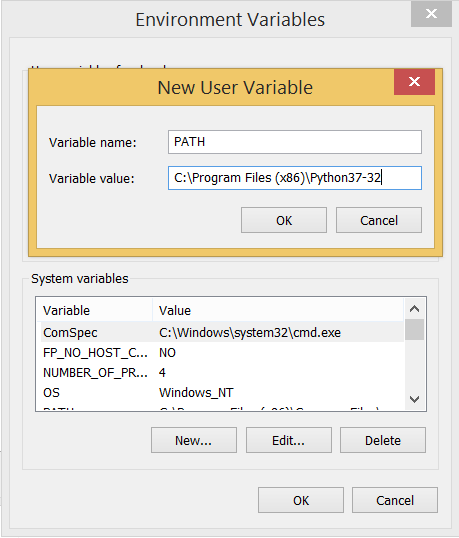




Add the new path variable in the user variable section.



Type PATH as the variable name and set the path to the installation directory of the python shown in the below image.



Now, the path is set, we are ready to run python on our local system. Restart CMD, and type python again. It will open the python interpreter shell where we can execute the python statements.

**Virtual Environments and Packages**

**Introduction**

Python applications will often use packages and modules that don’t come as part of the standard library. Applications will sometimes need a specific version of a library, because the application may require that a particular bug has been fixed or the application may be written using an obsolete version of the library’s interface.

This means it may not be possible for one Python installation to meet the requirements of every application. If application A needs version 1.0 of a particular module but application B needs version 2.0, then the requirements are in conflict and installing either version 1.0 or 2.0 will leave one application unable to run.

The solution for this problem is to create a virtual environment, a self-contained directory tree that contains a Python installation for a particular version of Python, plus a number of additional packages.

Different applications can then use different virtual environments. To resolve the earlier example of conflicting requirements, application A can have its own virtual environment with version 1.0 installed while application B has another virtual environment with version 2.0. If application B requires a library be upgraded to version 3.0, this will not affect application A’s environment.

**Creating Virtual Environments**

The module used to create and manage virtual environments is called venv. venv will usually install the most recent version of Python that you have available. If you have multiple versions of Python on your system, you can select a specific Python version by running python3 or whichever version you want.

To create a virtual environment, decide upon a directory where you want to place it, and run the venv module as a script with the directory path:

python3 -m venv tutorial-env

This will create the tutorial-env directory if it doesn’t exist, and also create directories inside it containing a copy of the Python interpreter, the standard library, and various supporting files.

A common directory location for a virtual environment is .venv. This name keeps the directory typically hidden in your shell and thus out of the way while giving it a name that explains why the directory exists. It also prevents clashing with .env environment variable definition files that some tooling supports.

Once you’ve created a virtual environment, you may activate it.

On Windows, run:

tutorial-env\Scripts\activate.bat

On Unix or MacOS, run:

source tutorial-env/bin/activate

(This script is written for the bash shell. If you use the csh or fish shells, there are alternate activate.csh and activate.fish scripts you should use instead.)

Activating the virtual environment will change your shell’s prompt to show what virtual environment you’re using, and modify the environment so that running python will get you that particular version and installation of Python. For example:

$ source ~/envs/tutorial-env/bin/activate

(tutorial-env) $ python

Python 3.5.1 (default, May 6 2016, 10:59:36)

...

>>> import sys

>>>sys.path

['', '/usr/local/lib/python35.zip', ...,

'~/envs/tutorial-env/lib/python3.5/site-packages']

>>>

**Managing Packages with pip**

You can install, upgrade, and remove packages using a program called pip. By default pip will install packages from the Python Package Index, <https://pypi.org>. You can browse the Python Package Index by going to it in your web browser, or you can use pip’s limited search feature:

(tutorial-env) $ pip search astronomy

skyfield - Elegant astronomy for Python

gary - Galactic astronomy and gravitational dynamics.

novas - The United States Naval Observatory NOVAS astronomy library

astroobs - Provides astronomy ephemeris to plan telescope observations

PyAstronomy - A collection of astronomy related tools for Python.

...

pip has a number of subcommands: “search”, “install”, “uninstall”, “freeze”, etc. (Consult the Installing Python Modules guide for complete documentation for pip.)

You can install the latest version of a package by specifying a package’s name:

(tutorial-env) $ pip install novas

Collecting novas

Downloading novas-3.1.1.3.tar.gz (136kB)

Installing collected packages: novas

Running setup.py install for novas

Successfully installed novas-3.1.1.3

You can also install a specific version of a package by giving the package name followed by == and the version number:

(tutorial-env) $ pip install requests==2.6.0

Collecting requests==2.6.0

Using cached requests-2.6.0-py2.py3-none-any.whl

Installing collected packages: requests

Successfully installed requests-2.6.0

If you re-run this command, pip will notice that the requested version is already installed and do nothing. You can supply a different version number to get that version, or you can run pip install --upgrade to upgrade the package to the latest version:

(tutorial-env) $ pip install --upgrade requests

Collecting requests

Installing collected packages: requests

Found existing installation: requests 2.6.0

Uninstalling requests-2.6.0:

Successfully uninstalled requests-2.6.0

Successfully installed requests-2.7.0

pip uninstall followed by one or more package names will remove the packages from the virtual environment.

pip show will display information about a particular package:

(tutorial-env) $ pip show requests

---

Metadata-Version: 2.0

Name: requests

Version: 2.7.0

Summary: Python HTTP for Humans.

Home-page: http://python-requests.org

Author: Kenneth Reitz

Author-email: me@kennethreitz.com

License: Apache 2.0

Location: /Users/akuchling/envs/tutorial-env/lib/python3.4/site-packages

Requires:

pip list will display all of the packages installed in the virtual environment:

(tutorial-env) $ pip list

novas (3.1.1.3)

numpy (1.9.2)

pip (7.0.3)

requests (2.7.0)

setuptools (16.0)

pip freeze will produce a similar list of the installed packages, but the output uses the format that pip install expects. A common convention is to put this list in a requirements.txt file:

(tutorial-env) $ pip freeze > requirements.txt

(tutorial-env) $ cat requirements.txt

novas==3.1.1.3

numpy==1.9.2

requests==2.7.0

The requirements.txt can then be committed to version control and shipped as part of an application. Users can then install all the necessary packages with install -r:

(tutorial-env) $ pip install -r requirements.txt

Collecting novas==3.1.1.3 (from -r requirements.txt (line 1))

...

Collecting numpy==1.9.2 (from -r requirements.txt (line 2))

...

Collecting requests==2.7.0 (from -r requirements.txt (line 3))

...

Installing collected packages: novas, numpy, requests

Running setup.py install for novas

Successfully installed novas-3.1.1.3 numpy-1.9.2 requests-2.7.0

pip has many more options. Consult the Installing Python Modules guide for complete documentation for pip. When you’ve written a package and want to make it available on the Python Package Index, consult the Distributing Python Modules guide.

**SYSTEM TESTING**

**7. SYSTEM TESTING**

* 1. **SYSTEM TESTING**

Testing is the procedure where the deformities are Identified, detached, oppressed for correction and guarantee that the item is sans imperfection so as to give quality to it and thus consumer loyalty of the testing we must know the following things.

* Recognition of defects: defects must be identified first in the product.
* Detaching the defects: After identification defects must be listed. Isolation means separation. Physical separation is done by the developer.
* Submitted for rectification: This is the responsibility of the TE to send the list of defects for rectification.
  1. **TYPES OF TESTING**

**Unit Testing**

This is a basic testing, that tons of its improvement and is conspicuous. Unit tests perform essential tests at part level and test a specific business method, application, or possibly structure plan. Unit tests ensure that each striking method for a business strategy performs correctly to the documented conclusions and contains obviously portrayed sources of info and foreseen results.

**Integration Testing**

Integration tests are intended to test incorporated programming parts to decide whether they really keep running as one program. Testing is occasion driven and is increasingly worried about the fundamental result of screens or fields. Integration tests exhibit that in spite of the fact that the parts were separately fulfillment, as appeared by effectively unit testing, the mix of segments is right and predictable. Integration testing is explicitly gone for uncovering the issues that emerge from the mix of segments.

**Functional Testing:**

Practical tests give conscious efficient shows that cutoff points endeavored are open as constrained by the business and specific necessities, structure documentation, and client manuals. Useful testing is centered on the going with things:

Affiliation and plan of practical tests is fixated on requirements, key limits, or uncommon examinations. Besides, deliberate consideration identifying with recognizes Business method streams; data fields, predefined strategies, and dynamic systems must be considered for testing. Before utilitarian testing is done, additional tests are perceived and the feasible estimation of current tests is settled.

**System Testing**

Framework testing ensures that the entire consolidated programming structure meets necessities. It tests a course of action to ensure known and obvious results. An instance of system testing is the course of action arranged structure mix test. System testing relies upon methodology delineations and streams, focusing on pre-driven technique associations and mix centers.

**7.3 White Box Testing**

White Box Testing is an endeavoring wherein in which the thing analyzer contemplates the inward limits, structure and language of the thing, or maybe its motivation. It is reason. It is utilized to test zones that can't be come to from a revelation level.

**7.4 Black Box Testing**

Disclosure Testing will endeavor the thing with no learning of the inner activities, structure or language of the module being endeavored. Dim tests, as most different sorts of tests, must be shaped from a definitive source record, for example, explicit or necessities report, for example, confirmation or prerequisites account. It is an endeavoring where the thing under test is overseen, as a divulgence .you can't "see" into it. The test gives wellsprings of information and reacts to yields without considering how the thing limits.

**SAMPLE CODE**

**8. SAMPLE CODE**

**HandGestureRecognize.py**

from tkinter import messagebox

from tkinter import \*

from tkinter import simpledialog

import tkinter

from tkinter import filedialog

from tkinter.filedialog import askopenfilename

import cv2

import random

import numpy as np

from keras.utils.np\_utils import to\_categorical

from keras.layers import MaxPooling2D

from keras.layers import Dense, Dropout, Activation, Flatten

from keras.layers import Convolution2D

from keras.models import Sequential

from keras.models import model\_from\_json

import pickle

import os

import imutils

from gtts import gTTS

from playsound import playsound

import os

from threading import Thread

main = tkinter.Tk()

main.title("Sign Language Recognition to Text & Voice using CNN Advance")

main.geometry("1300x1200")

global filename

global classifier

bg = None

playcount = 0

#names = ['Palm','I','Fist','Fist Moved','Thumbs up','Index','OK','Palm Moved','C','Down']

names = ['C','Thumbs Down','Fist','I','Ok','Palm','Thumbs up']

def getID(name):

index = 0

for i in range(len(names)):

if names[i] == name:

index = i

break

return index

bgModel = cv2.createBackgroundSubtractorMOG2(0, 50)

def deleteDirectory():

filelist = [ f for f in os.listdir('play') if f.endswith(".mp3") ]

for f in filelist:

os.remove(os.path.join('play', f))

def play(playcount,gesture):

class PlayThread(Thread):

def \_\_init\_\_(self,playcount,gesture):

Thread.\_\_init\_\_(self)

self.gesture = gesture

self.playcount = playcount

def run(self):

t1 = gTTS(text=self.gesture, lang='en', slow=False)

t1.save("play/"+str(self.playcount)+".mp3")

playsound("play/"+str(self.playcount)+".mp3")

newthread = PlayThread(playcount,gesture)

newthread.start()

def remove\_background(frame):

fgmask = bgModel.apply(frame, learningRate=0)

kernel = np.ones((3, 3), np.uint8)

fgmask = cv2.erode(fgmask, kernel, iterations=1)

res = cv2.bitwise\_and(frame, frame, mask=fgmask)

return res

def uploadDataset():

global filename

global labels

labels = []

filename = filedialog.askdirectory(initialdir=".")

pathlabel.config(text=filename)

text.delete('1.0', END)

text.insert(END,filename+" loaded\n\n");

def trainCNN():

global classifier

text.delete('1.0', END)

X\_train = np.load('model1/X.txt.npy')

Y\_train = np.load('model1/Y.txt.npy')

text.insert(END,"CNN is training on total images : "+str(len(X\_train))+"\n")

if os.path.exists('model1/model.json'):

with open('model1/model.json', "r") as json\_file:

loaded\_model\_json = json\_file.read()

classifier = model\_from\_json(loaded\_model\_json)

classifier.load\_weights("model1/model\_weights.h5")

classifier.\_make\_predict\_function()

print(classifier.summary())

f = open('model1/history.pckl', 'rb')

data = pickle.load(f)

f.close()

acc = data['accuracy']

accuracy = acc[9] \* 100

text.insert(END,"CNN Hand Gesture Training Model Prediction Accuracy = "+str(accuracy))

else:

classifier = Sequential()

classifier.add(Convolution2D(32, 3, 3, input\_shape = (64, 64, 3), activation = 'relu'))

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

classifier.add(Convolution2D(32, 3, 3, activation = 'relu'))

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

classifier.add(Flatten())

classifier.add(Dense(output\_dim = 256, activation = 'relu'))

classifier.add(Dense(output\_dim = 5, activation = 'softmax'))

print(classifier.summary())

classifier.compile(optimizer = 'adam', loss = 'categorical\_crossentropy', metrics = ['accuracy'])

hist = classifier.fit(X\_train, Y\_train, batch\_size=16, epochs=10, shuffle=True, verbose=2)

classifier.save\_weights('model1/model\_weights.h5')

model\_json = classifier.to\_json()

with open("model1/model.json", "w") as json\_file:

json\_file.write(model\_json)

f = open('model1/history.pckl', 'wb')

pickle.dump(hist.history, f)

f.close()

f = open('model1/history.pckl', 'rb')

data = pickle.load(f)

f.close()

acc = data['accuracy']

accuracy = acc[9] \* 100

text.insert(END,"CNN Hand Gesture Training Model Prediction Accuracy = "+str(accuracy))

def run\_avg(image, aWeight):

global bg

if bg is None:

bg = image.copy().astype("float")

return

cv2.accumulateWeighted(image, bg, aWeight)

def segment(image, threshold=25):

global bg

diff = cv2.absdiff(bg.astype("uint8"), image)

thresholded = cv2.threshold(diff, threshold, 255, cv2.THRESH\_BINARY)[1]

( cnts, \_) = cv2.findContours(thresholded.copy(), cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

if len(cnts) == 0:

return

else:

segmented = max(cnts, key=cv2.contourArea)

return (thresholded, segmented)

def webcamPredict():

global playcount

oldresult = 'none'

count = 0

fgbg2 = cv2.createBackgroundSubtractorKNN();

aWeight = 0.5

camera = cv2.VideoCapture(0)

top, right, bottom, left = 10, 350, 325, 690

num\_frames = 0

while(True):

(grabbed, frame) = camera.read()

frame = imutils.resize(frame, width=700)

frame = cv2.flip(frame, 1)

clone = frame.copy()

(height, width) = frame.shape[:2]

roi = frame[top:bottom, right:left]

gray = cv2.cvtColor(roi, cv2.COLOR\_BGR2GRAY)

gray = cv2.GaussianBlur(gray, (41, 41), 0)

if num\_frames < 30:

run\_avg(gray, aWeight)

else:

temp = gray

hand = segment(gray)

if hand is not None:

(thresholded, segmented) = hand

cv2.drawContours(clone, [segmented + (right, top)], -1, (0, 0, 255))

#cv2.imwrite("test.jpg",temp)

#cv2.imshow("Thesholded", temp)

#ret, thresh = cv2.threshold(temp, 150, 255, cv2.THRESH\_BINARY + cv2.THRESH\_OTSU)

#thresh = cv2.resize(thresh, (64, 64))

#thresh = np.array(thresh)

#img = np.stack((thresh,)\*3, axis=-1)

roi = frame[top:bottom, right:left]

roi = fgbg2.apply(roi);

cv2.imwrite("test.jpg",roi)

#cv2.imwrite("newDataset/Fist/"+str(count)+".png",roi)

#count = count + 1

#print(count)

img = cv2.imread("test.jpg")

img = cv2.resize(img, (64, 64))

img = img.reshape(1, 64, 64, 3)

img = np.array(img, dtype='float32')

img /= 255

predict = classifier.predict(img)

value = np.amax(predict)

cl = np.argmax(predict)

result = names[np.argmax(predict)]

if value >= 0.99:

print(str(value)+" "+str(result))

cv2.putText(clone, 'Gesture Recognize as : '+str(result), (10, 25), cv2.FONT\_HERSHEY\_SIMPLEX,0.5, (0, 255, 255), 2)

if oldresult != result:

play(playcount,result)

oldresult = result

playcount = playcount + 1

else:

cv2.putText(clone, '', (10, 25), cv2.FONT\_HERSHEY\_SIMPLEX,0.5, (0, 255, 255), 2)

cv2.imshow("video frame", roi)

cv2.rectangle(clone, (left, top), (right, bottom), (0,255,0), 2)

num\_frames += 1

cv2.imshow("Video Feed", clone)

keypress = cv2.waitKey(1) & 0xFF

if keypress == ord("q"):

break

camera.release()

cv2.destroyAllWindows()

font = ('times', 16, 'bold')

title = Label(main, text='Sign Language Recognition to Text & Voice using CNN Advance',anchor=W, justify=CENTER)

title.config(bg='yellow4', fg='white')

title.config(font=font)

title.config(height=3, width=120)

title.place(x=0,y=5)

font1 = ('times', 13, 'bold')

upload = Button(main, text="Upload Hand Gesture Dataset", command=uploadDataset)

upload.place(x=50,y=100)

upload.config(font=font1)

pathlabel = Label(main)

pathlabel.config(bg='yellow4', fg='white')

pathlabel.config(font=font1)

pathlabel.place(x=50,y=150)

markovButton = Button(main, text="Train CNN with Gesture Images", command=trainCNN)

markovButton.place(x=50,y=200)

markovButton.config(font=font1)

predictButton = Button(main, text="Sign Language Recognition from Webcam", command=webcamPredict)

predictButton.place(x=50,y=250)

predictButton.config(font=font1)

font1 = ('times', 12, 'bold')

text=Text(main,height=15,width=78)

scroll=Scrollbar(text)

text.configure(yscrollcommand=scroll.set)

text.place(x=450,y=100)

text.config(font=font1)

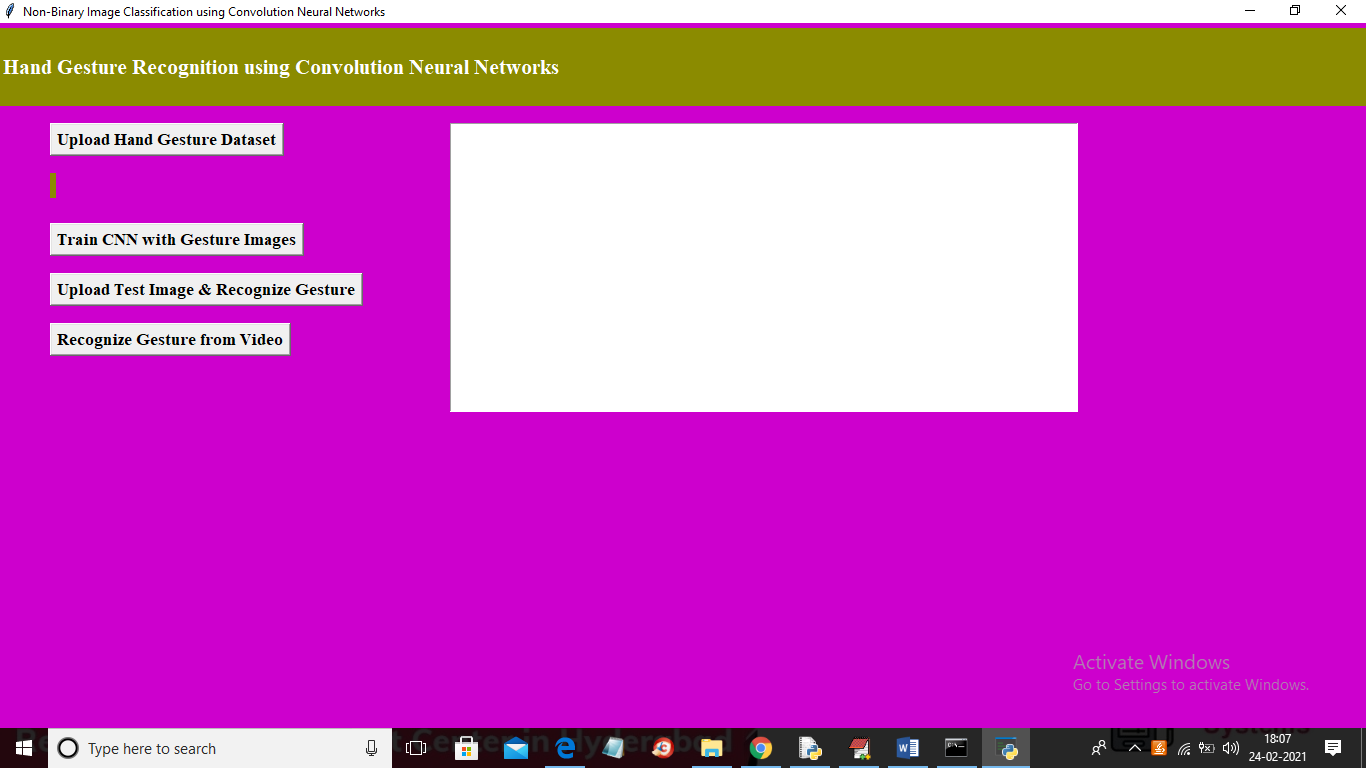
deleteDirectory()

main.config(bg='magenta3')

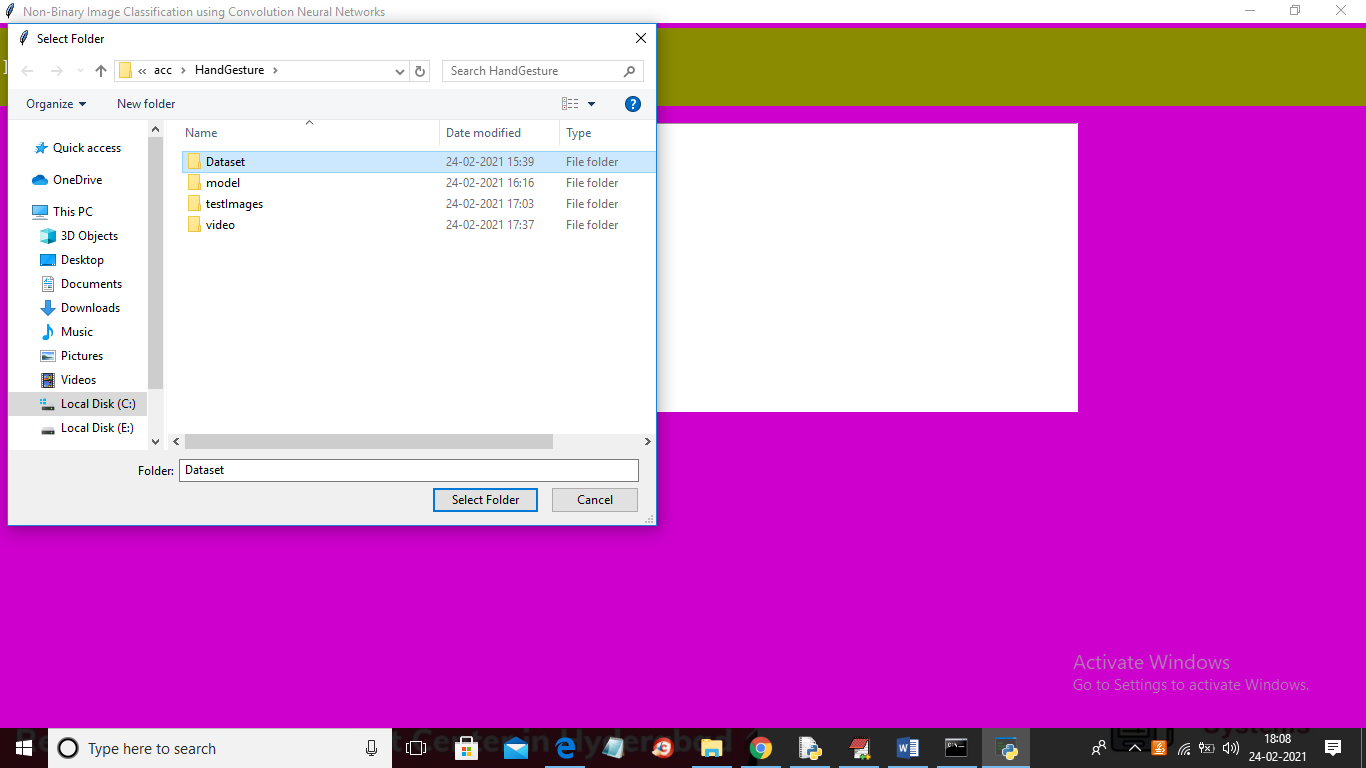
main.mainloop()

**RESULTS & OUTPUTS**

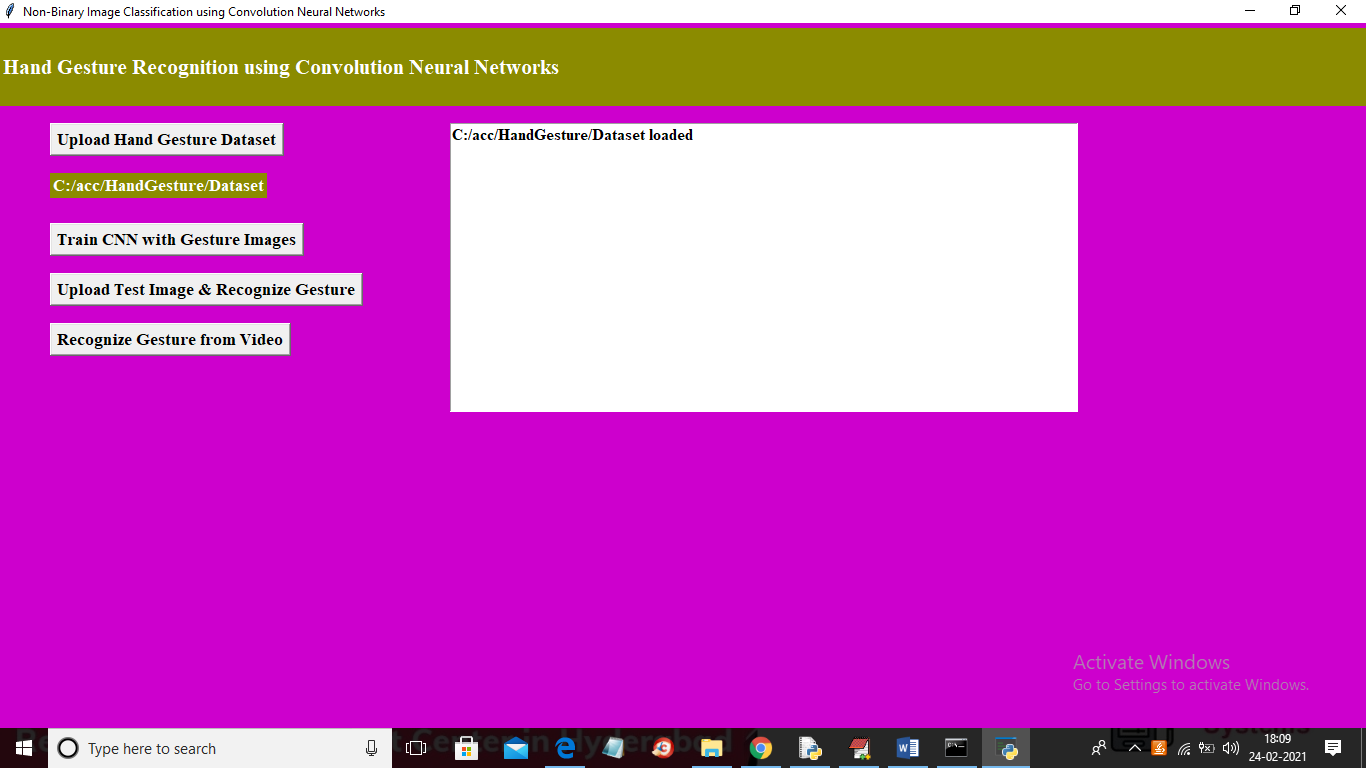
**9. OUTPUT & RESULTS**



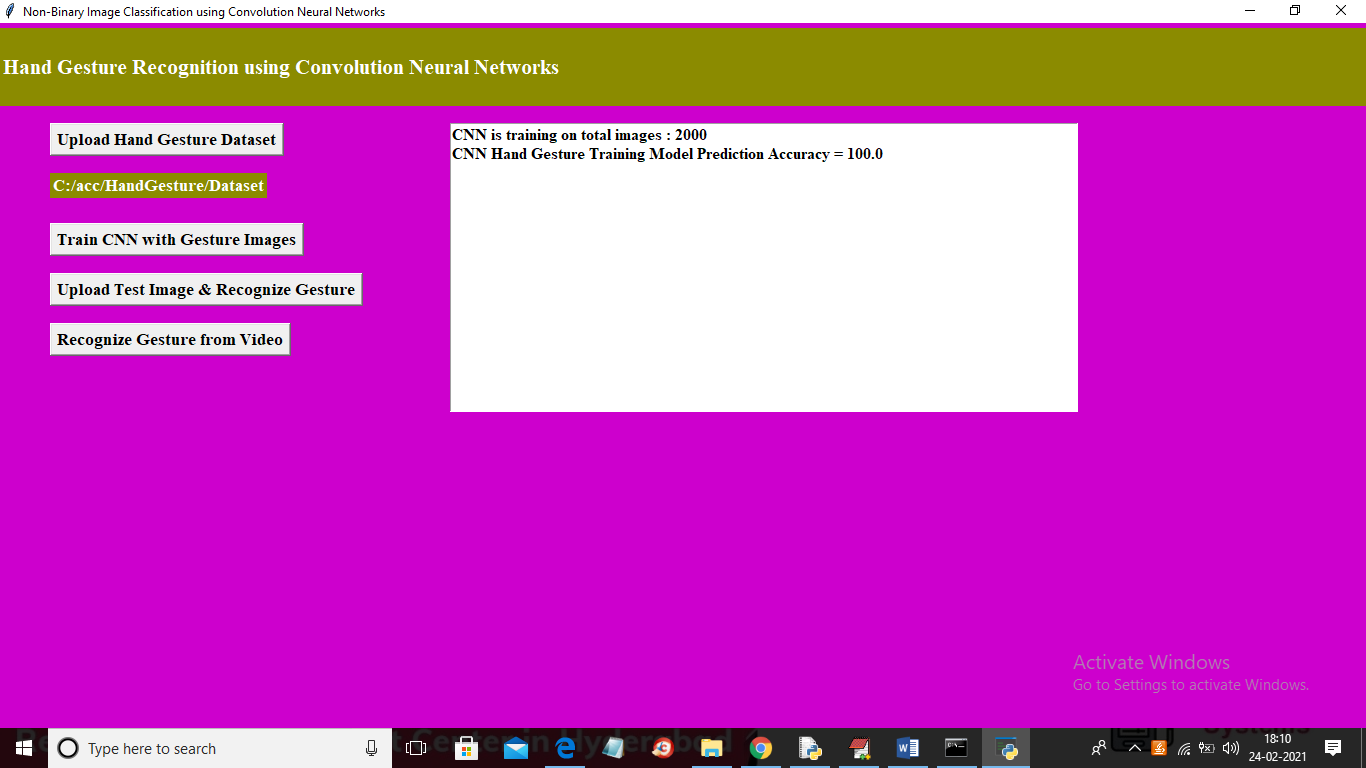
In above screen click on ‘Upload Hand Gesture Dataset’ button to upload dataset and to get below screen



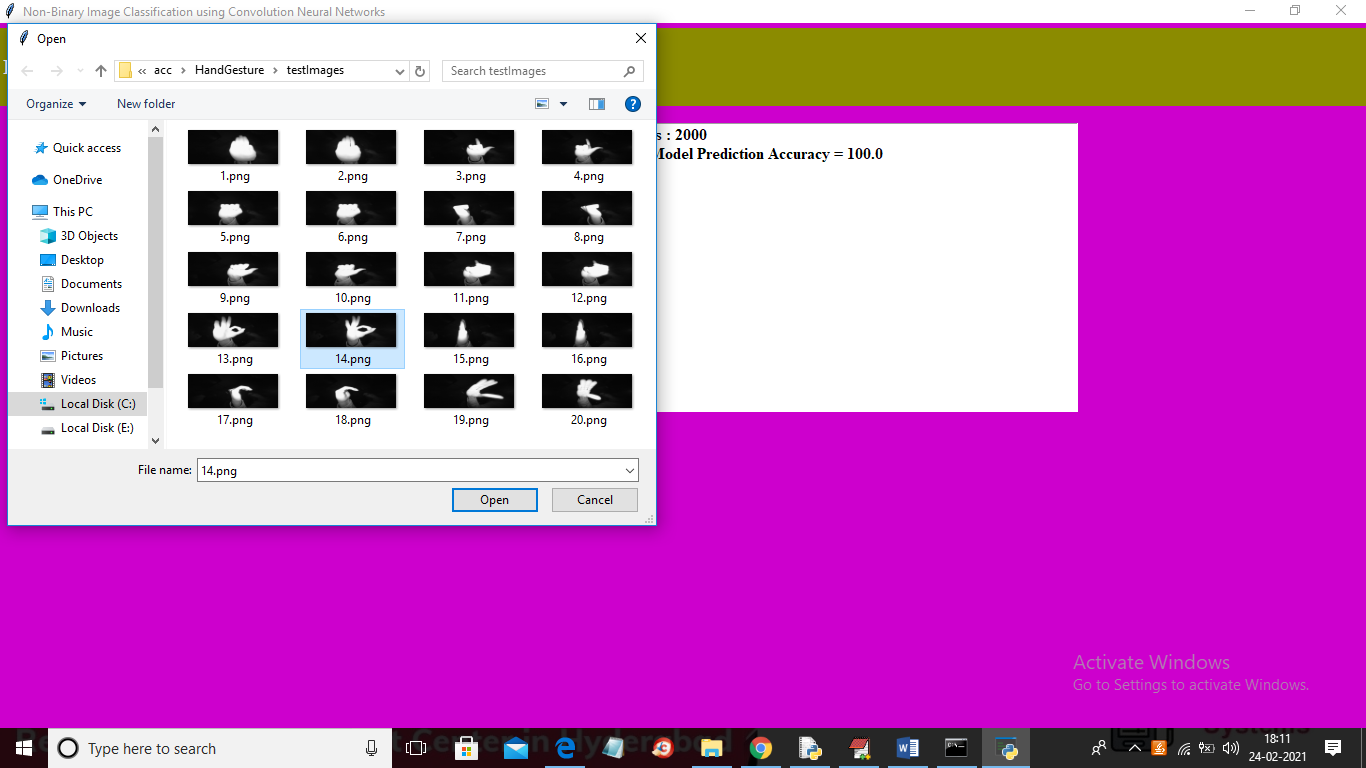
In above screen selecting and uploading ‘Dataset’ folder and then click on ‘Select Folder’ button to load dataset and to get below screen



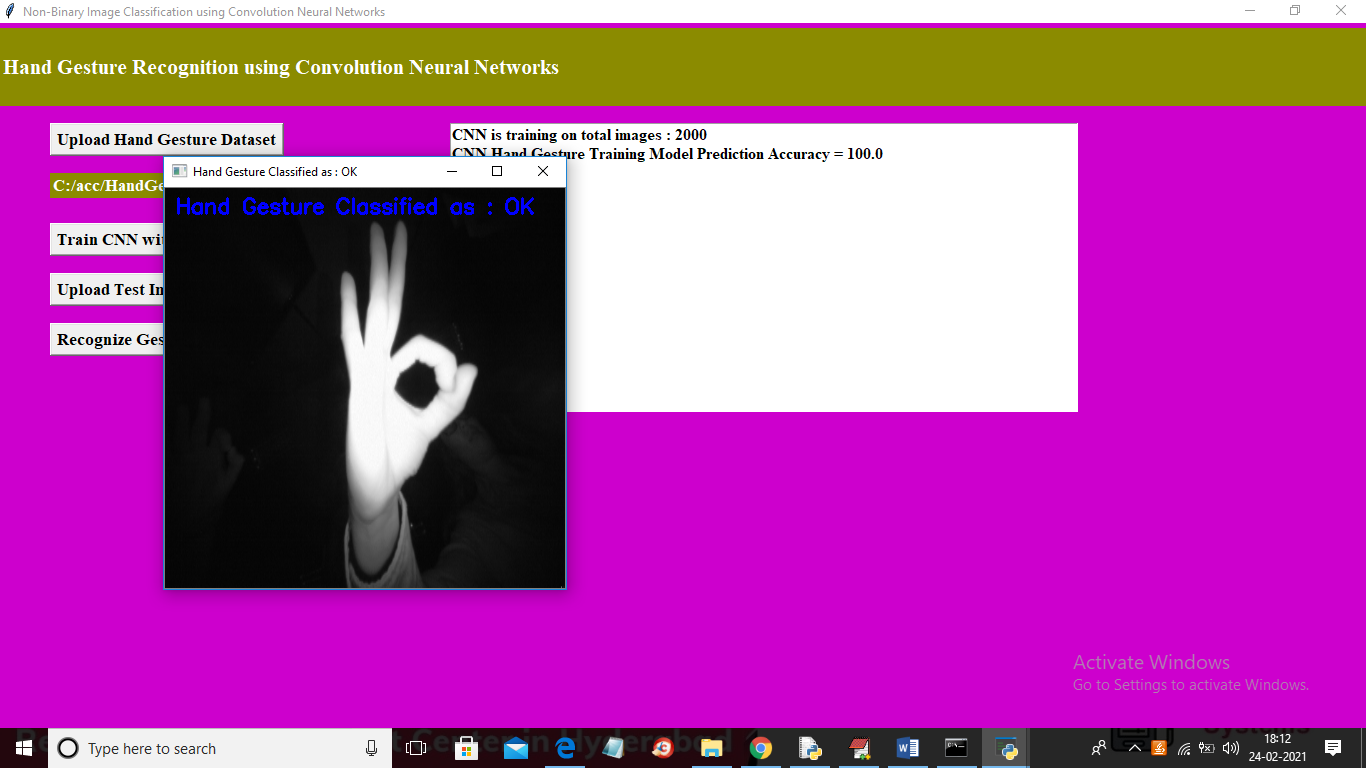
In above screen dataset loaded and now click on ‘Train CNN with Gesture Images’ button to trained CNN model and to get below screen



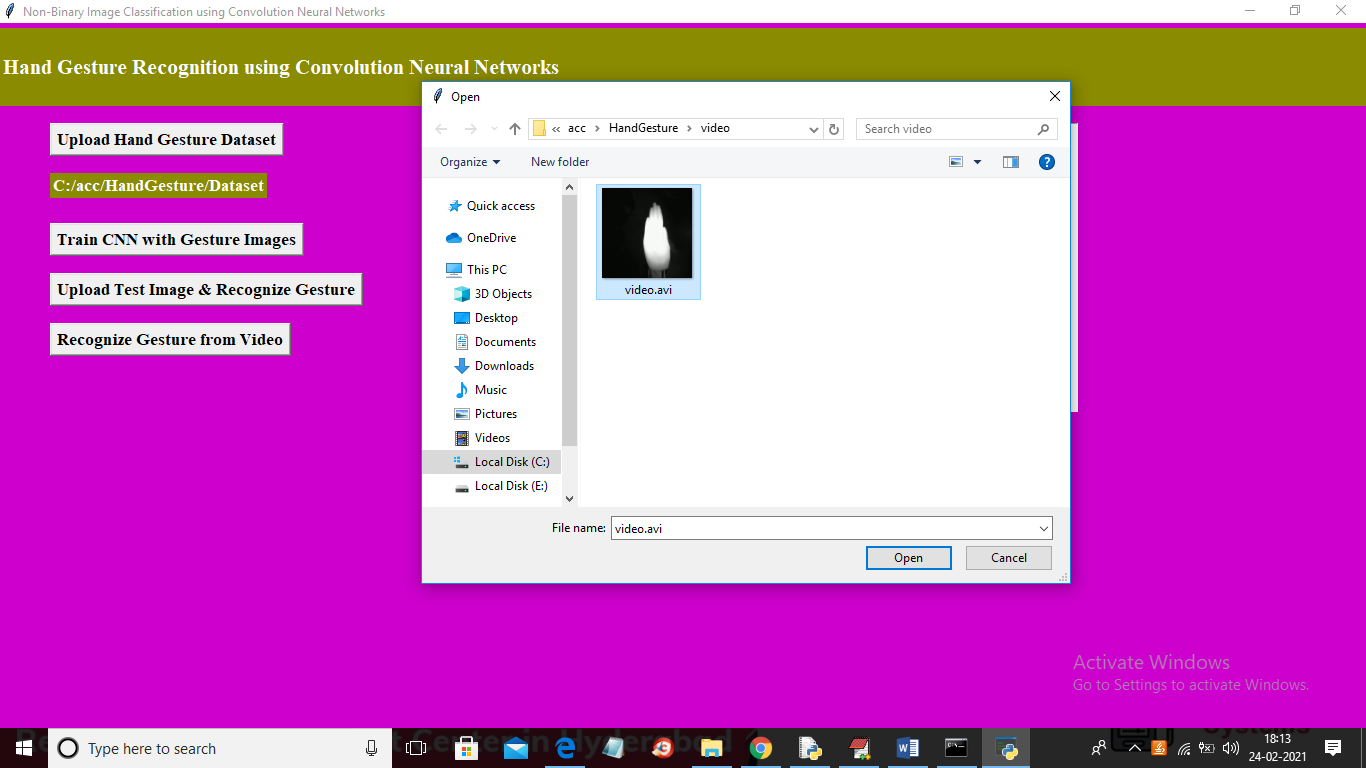
In above screen CNN model trained on 2000 images and its prediction accuracy we got as 100% and now model is ready and now click on ‘Upload Test Image & Recognize Gesture’ button to upload image and to gesture recognition



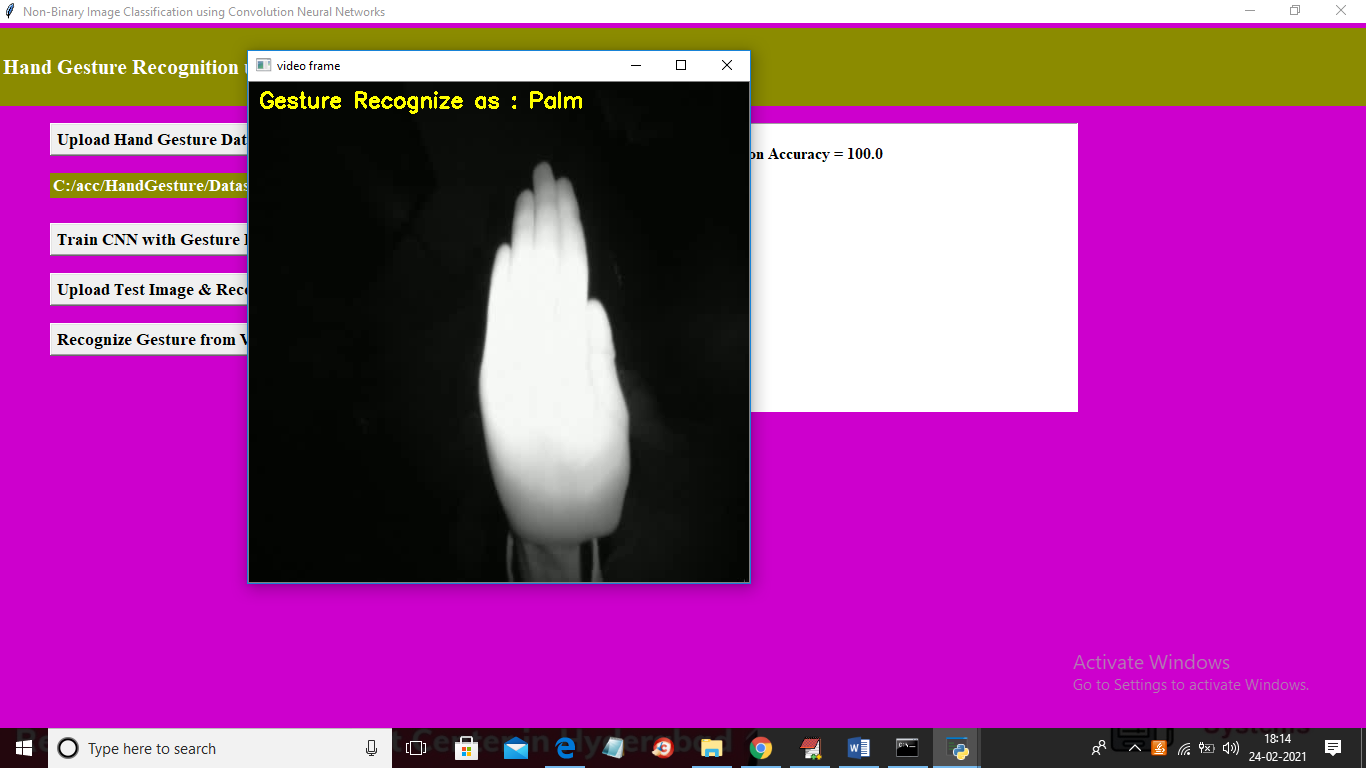
In above screen selecting and uploading ’14.png’ file and then click Open button to get below result

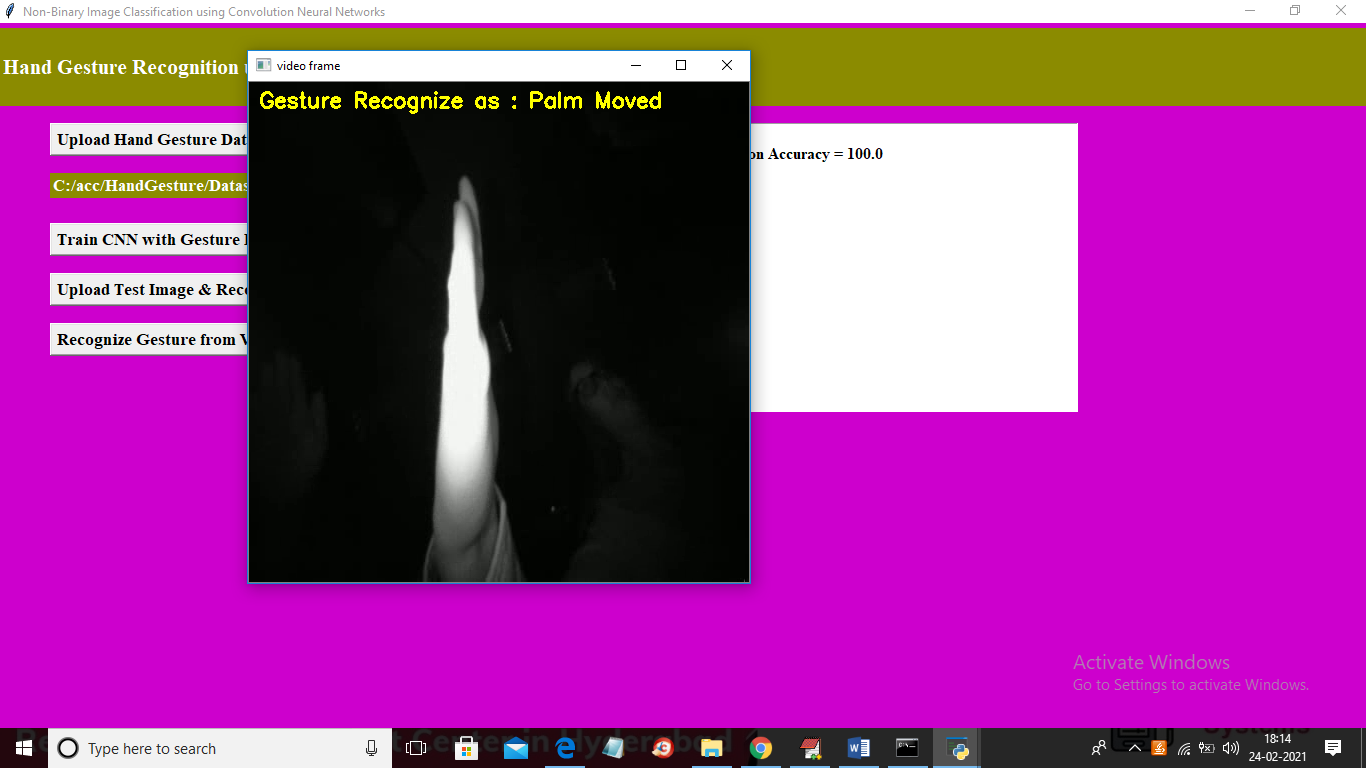


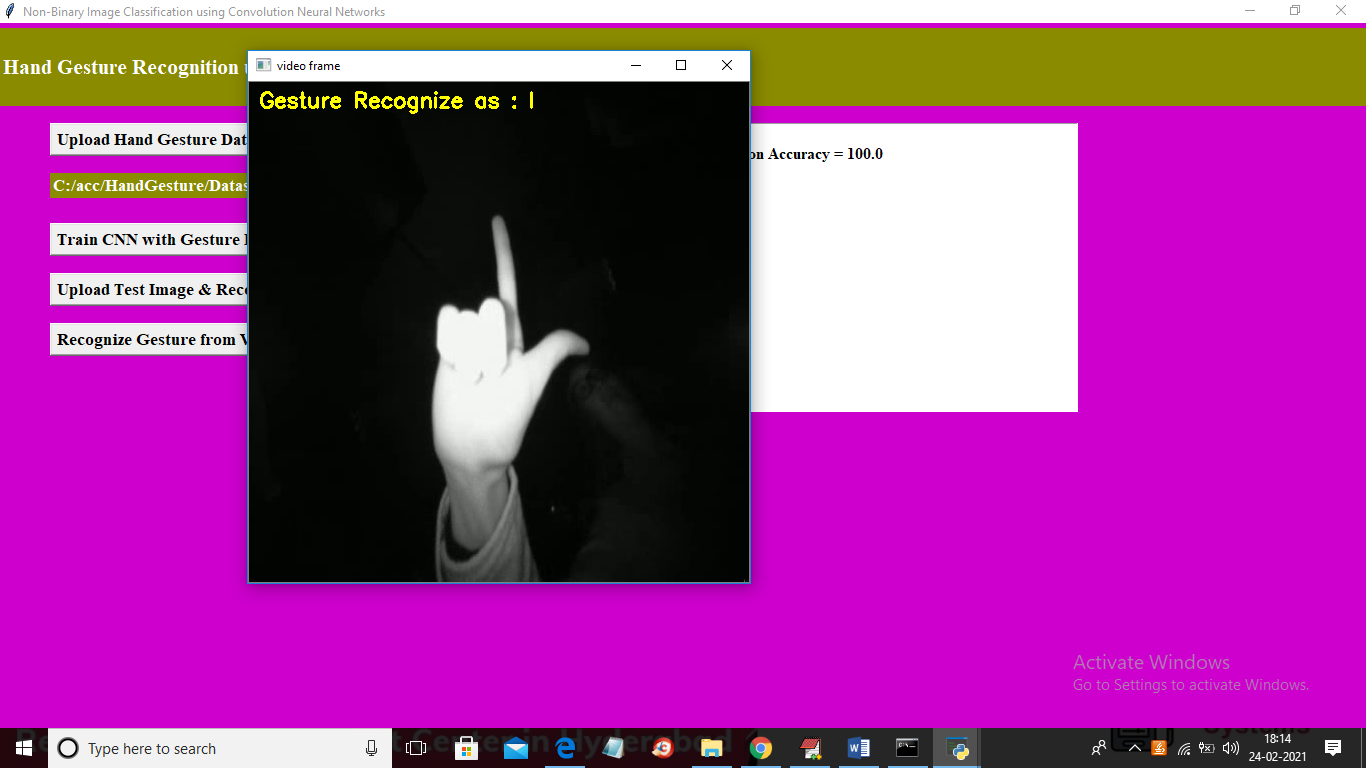
In above screen gesture recognize as OK and similarly you can upload any image and get result and now click on ‘Recognize Gesture from Video’ button to upload video and get result



In above screen selecting and uploading ‘video.avi’ file and then click on ‘Open’ button to get below result







In above screen as video play then will get recognition result

**CONCLUSION**

**10. CONCLUSION**

Many breakthroughs have been made in the field of artificial intelligence, machine learning and computer vision. They have immensely contributed in how we perceive things around us and improve the way in which we apply their techniques in our everyday lives. Many researches have been conducted on sign gesture recognition using different techniques like ANN, LSTM and 3D CNN. However, most of them require extra computing power . On the other hand, our research paper requires low computing power and gives a remarkable accuracy of above 90%. In our research, we proposed to normalise and rescale our images to 64 pixels in order to extract features (binary pixels) and make the system more robust. We use CNN to classify the 10 alphabetical American sign gestures and successfully achieve an accuracy of 98% which is better than other related work stated in this paper.

**BIBILIOGRAPHY**

**11. BIBILIOGRAPHY**

**REFERENCES:**

**TEXT BOOK REFERENCES:**

1. **Python Phrasebook**
2. **The Python Language Reference Manual**
3. **Python Essential Reference**.

**WEB REFERENCES:**

1. Wikipedia,URL:http://www.wikipedia.org
2. Google,URL:http://www.google.co.in

[1] <https://peda.net/id/08f8c4a8511>

[2] K. Bantupalli and Y. Xie, "American Sign Language Recognition using Deep Learning and Computer Vision," 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 2018, pp. 4896-4899, doi: 10.1109/BigData.2018.8622141.

[3] CABRERA, MARIA & BOGADO, JUAN & Fermín, Leonardo & Acuña, Raul & RALEV, DIMITAR. (2012). GLOVE-BASED GESTURE RECOGNITION SYSTEM. 10.1142/9789814415958\_0095.

[4] He, Siming. (2019). Research of a Sign Language Translation System Based on Deep Learning. 392-396. 10.1109/AIAM48774.2019.00083.

[5] International Conference on Trendz in Information Sciences and Computing (TISC). : 30-35, 2012.

[6] Herath, H.C.M. & W.A.L.V.Kumari, & Senevirathne, W.A.P.B & Dissanayake, Maheshi. (2013). IMAGE BASED SIGN LANGUAGE RECOGNITION SYSTEM FOR SINHALA SIGN LANGUAGE

[7] M. Geetha and U. C. Manjusha, , “A Vision Based Recognition of Indian Sign Language Alphabets and Numerals Using B-Spline Approximation”, International Journal on Computer Science and Engineering (IJCSE), vol. 4, no. 3, pp. 406-415. 2012.

[8] Pigou L., Dieleman S., Kindermans PJ., Schrauwen B. (2015) Sign Language Recognition Using Convolutional Neural Networks. In: Agapito L., Bronstein M., Rother C. (eds) Computer Vision - ECCV 2014 Workshops. ECCV 2014. Lecture Notes in Computer Science, vol 8925. Springer, Cham. <https://doi.org/10.1007/978-3-319-16178-5_40>

[9] Escalera, S., Baró, X., Gonzàlez, J., Bautista, M., Madadi, M., Reyes, M., . . . Guyon, I. (2014). ChaLearn Looking at People Challenge 2014: Dataset and Results. Workshop at the European Conference on Computer Vision (pp. 459-473). Springer, . Cham.

[10] Huang, J., Zhou, W., & Li, H. (2015). Sign Language Recognition using 3D convolutional neural networks. IEEE International Conference on Multimedia and Expo (ICME) (pp. 1-6). Turin: IEEE.

[11] Jaoa Carriera, A. Z. (2018). Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on (pp. 4724-4733). IEEE. Honolulu.

[12] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on (pp. 248-255). IEEE. Miami, FL, USA .

[13] Soomro, K., Zamir , A. R., & Shah, M. (2012). UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild. Computer Vision and Pattern Recognition, arXiv:1212.0402v1, 1-7.

[14] Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., & Serre, T. (2011). HMDB: a large video database for human motion recognition. Computer Vision (ICCV), 2011 IEEE International Conference on (pp. 2556-2563). IEEE

[15] Zhao, Ming & Bu, Jiajun & Chen, C.. (2002). Robust background subtraction in HSV color space. Proceedings of SPIE MSAV, vol. 1. 4861. 10.1117/12.456333.

[16] Chowdhury, A., Sang-jin Cho, & Ui-Pil Chong. (2011). A background subtraction method using color information in the frame averaging process. Proceedings of 2011 6th International Forum on Strategic Technology. doi:10.1109/ifost.2011.6021252