*Alphabet Recognition Through Gestures-Deep Learning*

Amol Bhilare

Department of Computer Science

Vishwakarma Institute of Technology

Pune,India

amol.bhilare@vit.edu

Harjas Singh Gandhi

Department of Computer Science

Vishwakarma Institute of Technology

Pune,India

harjas.gandhi18@vit.edu

Krish Agarwal

Department of Computer Science

Vishwakarma Institute of Technology

Pune,India

krish.agarwal18@vit.edu

Ganesh Tarone

Department of Computer Science

Vishwakarma Institute of Technology

Pune,India

ganesh.tarone18@vit.edu

Atharv Nirmal

Department of Computer Science

Vishwakarma Institute of Technology

Pune,India

Atharv.nirmal18@vit.edu

Pranjal Rane

Department of Computer Science

Vishwakarma Institute of Technology

Pune,India

rane.pranjal18@vit.edu

***Abstract-****A popular demonstration of the capability of deep learning techniques is object recognition in image data. This deep learning application in python recognizes alphabet through gestures captured real-time on a webcam. The user is allowed to write the alphabet on the screen using an object-of-interest (a water bottle cap in this case or any other blue object),* *we train the model using a model checkpointer, which will help us save the best model*

***Keywords —****Convolutional Neural Network, EMNIST Dataset , Multilayer Perceptron Model, Sequential Model*

**1.INTRODUCTION**

This deep learning application in python recognizes alphabet through gestures captured real time on a webcam or can be implemented on camera. The user is allowed to write the alphabet on the screen using an object-of-interest.

Our project presents an overview of the challenging field of static hand gesture recognition, which mainly consists of the recognition of well-defined signs based on a posture of the hand. Since human beings tend to differ in terms of size and shape, the most challenging problem consists of the segmentation

and the correct classification of the information’s gathered from the input data, captured by one or more cameras.

The aim of our task is to show which techniques have successfully been tested and used in order to

solve the problems mentioned above yielding a robust and reliable static hand gesture recognition system. With the development of information technology in our society, we can expect that computer systems to a larger extent will be embedded into our environment. These environments will impose needs for new types of human computer-interaction, with interfaces that are natural and easy to use.

The user interface (UI) of the personal computer has evolved from a text-based command line to a graphical interface with keyboard and mouse inputs. However, they are inconvenient and unnatural. The use of hand gestures provides an attractive alternative to these cumbersome interface devices for human-computer interaction (HCI)

# Literature Review

**Paper1**:Hand Gesture Recognition By Deepali Khada:

Vision based real time gesture recognition system received a great attention in recent years because of its manifoldness applications and the ability to interact with system efficiently through human computer interaction.

**Paper 2**:Gesture Recognition -IOSR Journal:

Gesture Recognition means identification and recognition of gestures originates from any type of body motion but commonly originate from face or hand. Current focuses in the field include emotion recognition from the face and hand gesture recognition .Gesture recognition enables humans to interface with the machine (HMI) and interact naturally without any mechanical devices

**Paper3**:RF Based Gesture recognition-Computer vision

These were the thee basic papers that were referred and based on there respective findings or conclusion ides for our own project was formulated.

All of these are different types of gesture recognition so were found to be very useful.

# RecognitionMethodology/Experimental

## Materials/Components/Flowchart/Block Diagram/Theory

## https://miro.medium.com/max/658/1*RctvQPFWV881-QDuVh1UDw.png https://miro.medium.com/max/1073/1*OVWotVv5nxRFEtRzwhsMUA.png

# working

## Step 1: Train A Multilayer Perceptron Model

* 1. **Load Data**

We use Python’s mnist library to load the data.

Get the data ready to be fed to the model. Splitting the data into train and test sets, standardizing the images and other preliminary stuff.

**1.2 Define Model**

In Keras, models are defined as sequence of layers. We first initialize a ‘Sequential Model’ and then we add the layers with respective neurons in them.

The model, as expected, takes 28 x 28 pixels (we flatten out the image and pass each of the pixel in a 1-D vector) as an input. The output of model has to be a decision on one of the letters, so we set the output layer with 26 neurons (the decision is made in probabilities).

**1.3 Compile Model**

Now that the model is defined, we can compile it. Compiling the model uses the efficient numerical libraries under the covers (the so-called backend) such as Theano or TensorFlow.

Here, we specify some properties needed to train the network. By training, we are trying find the best set of weights to make the decision on the input. We must specify the loss function to use to evaluate a set of weights, the optimizer used to search through different weights for the network and any optional metrics we would like to collect and report during training.

**1.4 Fit Model**:

Here, we train the model using a model check pointer, which will help us save the best model (best in terms of the metric we defined in the previous step).

**1.5 Evaluate Model**

Test accuracy of the model on the EMNIST dataset was **91.1%**.

## Step 2: Train A Convolutional Neural Network Model

**2.1 and 2.2 — Load Data and Define Model**

These two steps are exactly the same as the steps we implemented in building the MLP model.

**2.3 Define Model**

We have defined the above CNN architecture to solve the problem at hand.

**2.3 Compile Model**

Unlike the MLP model, this time We are using the ADADELTA optimizer

**2.4 Fit Model**

To know how the model variables batch\_size and epochs affect out model performance.

**2.5 Evaluate Model**

Test accuracy for the model on the EMNIST dataset was **93.1%**.

**2.6 Put It All Together**

Putting it all together, we get the complete code needed to build a decent CNN model trained on the EMNIST data.

## Step 3: Initializing Stuff

Before we look into the recognition code, lets initialize stuff.

First, we load the models built in the previous steps. We then create a letters dictionary, blueLowe**r**and blueUpperboundaries to detect the blue bottle cap, a kernal to smooth things along the way, an empty blackboard to store the writings in white (just like the alphabet in the EMNIST dataset), a deque to store all the **points** generated by the pen (blue bottle cap), and a couple of default value variables.

## Step 4: Capturing The Writings

Once we start reading the input video frame by frame, we try to find the blue bottle cap and use it as a pen. We use the OpenCV’s **cv2.VideoCapture()**method to read the video, frame by frame (using a while loop), either from a video file or from a webcam in real time. In this case, we pass 0 to the method to read from a webcam. The following code demonstrates the same.

Once we start reading the webcam feed, we constantly look for a blue color object in the frames with the help of **cv2.inRange()** method and use the blueUpper and blueLower variables initialized beforehand. Once we find the contour, we do a series of image operations and make it smooth. Smoothing just makes our lives easier

Once we find the contour (the **if** condition passes when a contour is found), we use the center of the contour (blue cap) to draw on the screen as it moves,**cv2.minEnclosingCircle()** and **cv2.circle()** the **cv2.minEnclosingCircle()** and **cv2.circle()** methods, gets the center of the contour found with the help of **cv2.moments()** method. In the end, the center is stored in a deque called **points**so that we can join them all to form a full writing.

We display the drawing on both **frame** and **blackboard**. One for external display and the other to pass it to the model.

## Step 5: Scraping The Writing And Passing It To The Model

Once the user finishes writing, we take the points we stored earlier, join them up, put them on a black board and pass it to the models.

The control enters this **elif** block when we stop writing (because there were no contours detected). Once we verify that the **points** deque is not empty, we are now sure that the writing is done. Now we take the **blackboard** image, do a quick contour search again (to scrape the writing out). Once found, we cut it appropriately, resize it meet the input dimension requirements of the models we built i.e., 28 x 28 pixels. And pass it to both of the models!

## Step 6: Showing Model Predictions

We then show the predictions made by our models on the **frame** window. And then we display it using the **cv2.imshow()** method. After falling out of the **while** loop we entered to read data from the webcam, we release the camera and destroy all the windows.

# Advantages

a) Low cost.

b) The given system is handy andportable, and thus can be easily carried from one place to another

. c) There is no circuitry thus can be easily troubleshooted.

d) There are no moving parts, so device wear is not an issue. e) There is no direct contact between the user and the device, so there is no hygiene problem.

f) Less cost for making a gesture control computer.

g) This system would be designed to be those that seem natural to users, thereby decreasing the learning time required.

h) They will reduce our need for devices like mouse, keys, remote control or keys for interaction with the electronic devices.

# Limitations

1. The sensitivity of the device sometimes becomes a problem. While sensing the blue colour any object with bluish shade is taken as the input
2. The accuracy also is device and platform dependant
3. We have set the colour to be detected as blue any other object wont be detected for example if a person doesnot find blue coloured object ,the code is useless

# Future Scope

**Medical Applications –**Advanced robotics systems with gesture recognition can be placed in hospitals or homes to recognize and treat life threatening conditions like heart attacks or strokes.

**Alternative computer interfaces –** Gesture recognition, along with voice recognition, facial recognition, lip movement recognition and eye tracking combined can be used to create something called a perceptual user interface (PUI), a completely different way to interact with computer systems which will improve usability and creativity by leaps and bounds.

**Entertainment applications –** Most videogames today are played either on game consoles, arcade units or PCs, and all require a combination of input devices. Gesture recognition can be used to truly immerse a players in the game world like never before.

**Automation systems –** In homes, offices, transport vehicles and more, gesture recognition can be incorporated to greatly increase usability and reduce the resources necessary to create primary or secondary input systems like remote controls, car entertainment systems with buttons or similar.

**An easier life for the disabled –** One of the biggest challenges faced today is providing separate and equally non cumbersome services to the differently abled and handicapped. While there are special provisions around the world, there’s still huge room for improvement to bring all lives on equal footing. Gesture recognition technology can eliminate a lot of manual labor and make life much easier for those who aren’t as fortunate as most of us are.

# Conclusion

We built two deep learning models, an MLP model and a CNN model, trained on the famous EMNIST data. And used those models to predict alphabet written by an object of our interest in real-time. I encourage you to tweak the architectures of both the models and see how they affect your predictions.

Acknowledgment

We are very grateful to our Director, Mr. Rajesh Jalnekar, Head of the Department Prof.Vivek Deshpande and our guide Prof.Amol Bhilare because of who we managed to complete our research. Furthermore, we would like to thank everyone who knowingly or unknowingly help our project towards completion.

References

1. Kaustubh Kalgaonkar, Bhiksha Raj, One-handed gesture recognition using opencv, IEEE, 2009
2. S. P. Tarzia, R. P. Dick, P. A. Dinda, and G. Memik, “ measurement of user presence and attention,” in Proceedings of the 11th international conference on Ubiquitous computing. ACM, 2009
3. <https://in.pycon.org/cfp/2018/proposals/handwritten-digit-and-character-recognition-using-python~bkV6a/>
4. C. E. Shannon, "A mathematical theory of communication", Bell Syst. Tech. J., vol. 27, pp. 379-423, 1948.
5. M. Abadi et al., "TensorFlow: Large-scale machine learning on heterogeneous distributed systems", 2016