National Telecommunication Institute

Digital Egypt Youth – Initiative

Artificial Intelligence (AI) and Internet of Things (IoT) Modules

REAL TIME SIGN LANGUAGE TRANSLATOR
A PROJECT REPORT

Submitted by: **Under the Supervision of:**

1- Ahmed Nabil Ali Elghamry
   Email: Dr.aelghamry@gmail.com
   Mobile: 01015309152

2- Ahmed Nader Goda
   Email: a.gouda8wo@gmail.com
   Mobile: 01128804804

3- Mohamed Reda Abd Elgawad
   Email: mohamedreda2271@gmail.com
   Mobile: 01023381292

4- Omar Mohamed Mahmoud
   Email: obadramer44@gmail.com
   Mobile: 01007372795

5- Raed Tarek Habib
   Email: raedtarek97@gmail.com
   Mobile: 01068631218

6- Shimaa Ahmed Daowd
   Email: Sh.daowd112@gmail.com
   Mobile: 01009404627

Dr. Mohamed Zorkany

(March - 2023)

DECLARATION

We, students of the Digital Egypt Youth – Initiative, Artificial Intelligence (AI) and Internet of Things (IoT) Modules, hereby declare that the project entitled "Real Time Sign Language Translator" is submitted by us as the final project of the modules in this initiative.

ACKNOWLEDGMENT

ABSTRACT

Inability to speak is considered to be true disability. People with this disability use different modes to communicate with others, there are number of methods available for their communication one such common method of communication is sign language.

Developing sign language application for deaf people can be very important, as they'll be able to communicate easily with even those who don't understand sign language. Our project aims at taking the basic step in bridging the communication gap between normal people, deaf and dumb people using sign language.

The main focus of this work is to create a vision-based system to identify sign language gestures from the video sequences in real time. The reason for choosing a system based on vision relates to the fact that it provides a simpler and more intuitive way of communication between a human and a computer. In this report, around 20 different gestures have been considered.

We used the following approach for the classification of sign language gestures:-

To train the model we have used VGG model which is a deep CNN (convolutional neural net). CNN was trained on the images obtained from the captured data set of train data. Trained CNN model was used to make predictions for individual frames to obtain a sequence of predictions or pool layer outputs for each gesture. The data set used consists of American Sign Language (ASL) Gestures, with 28,967 images belonging to about 30 gestures. Using the predictions by CNN accuracy of 100% was obtained.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

Motion of any body part like face, hand is a form of gesture. Here for gesture recognition, we are using image processing and computer vision. Gesture recognition enables computer to understand human actions and also acts as an interpreter between computer and human. This could provide potential to human to interact naturally with the computers without any physical contact of the mechanical devices. Gestures are performed by deaf and dumb community to perform sign language. This community used sign language for their communication when broadcasting audio is impossible, or typing and writing is difficult, but there is the vision possibility. At that time sign language is the only way for exchanging information between people. Normally sign language is used by everyone when they do not want to speak, but this is the only way of communication for deaf and dumb community. Sign language is also serving the same meaning as spoken language does. This is used by deaf and dumb community all over the world but in their regional form like ISL, ASL. Sign language can be performed by using Hand gesture either by one hand or two hands. It is of two type Isolated sign language and continuous sign language. Isolated sign language consists of single gesture having single word while continuous ISL or Continuous Sign language is a sequence of gestures that generate a meaningful sentence. In this report we performed isolated ASL gesture recognition technique.

1.1 Sign Language

Deaf people around the world communicate using sign language as distinct from spoken language in their everyday a visual language that uses a system of manual, facial and body movements as the means of communication. Sign language is not a universal language, and different sign languages are used in different countries, like the many spoken languages all over the world. Some countries such as Belgium, the UK, the USA or India may have more than one sign language. Hundreds of sign languages are in used around the world, for instance, Japanese Sign Language, British Sign Language (BSL), Spanish Sign Language, Turkish Sign Language.

Sign language is a visual language and consists of 3 major components:

Fingerspelling	Word level sign vocabulary	Non-manual features
Used to spell words letter by letter.	Used for the majority of communication.	Facial expressions and tongue, mouth and body position.
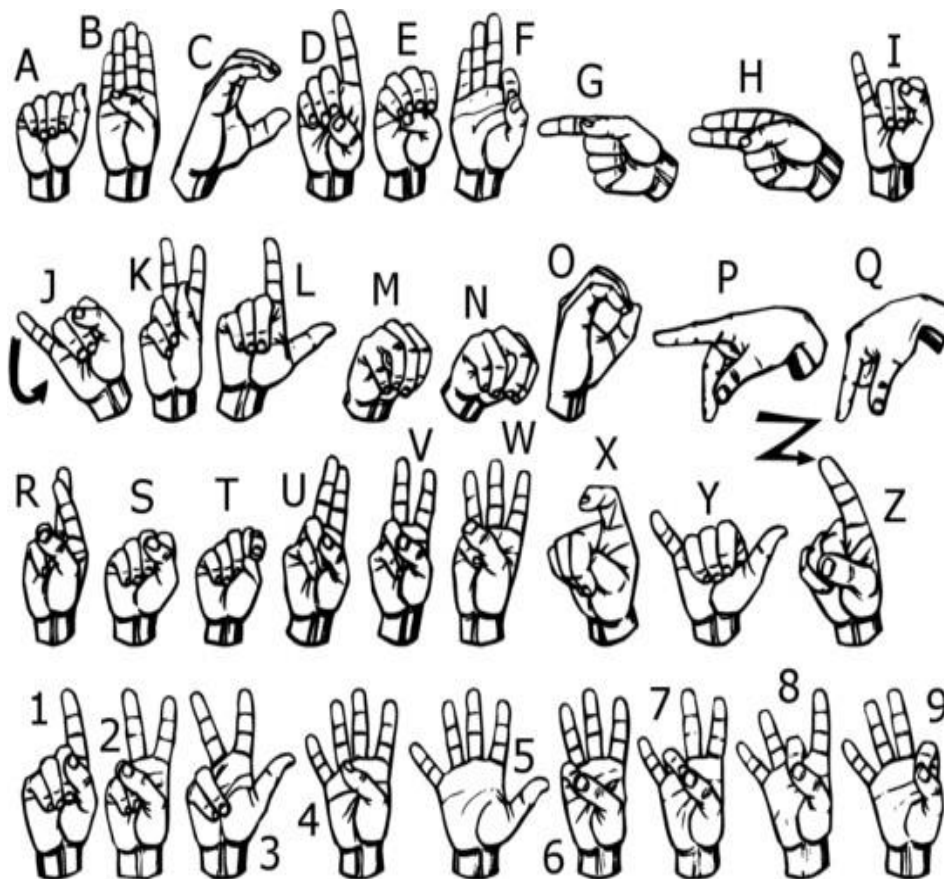
Fig 1: Finger Spelling American Sign Language

2. LITERATURE SURVEY

In the recent years, there has been tremendous research on the hand sign language gesture recognition. The technology for gesture recognition is given below.
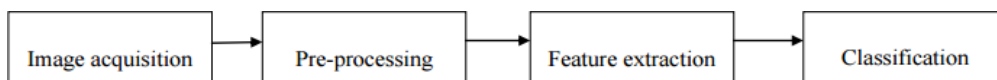
2.1 Vision-based

In vision-based methods computer camera is the input device for observing the information of hands or fingers. The Vision Based methods require only a camera, thus realizing a natural interaction between humans and computers without the use of any extra devices. These systems tend to complement biological vision by describing artificial vision systems that are implemented in software and/or hardware. This poses a challenging problem as these systems need to be background invariant, lighting insensitive, person and camera independent to achieve real time performance. Moreover, such systems must be optimized to meet the requirements, including accuracy and robustness.

The vision based hand gesture recognition system is shown in fig.--:

Fig 2: Block Diagram of vision based recognition system

Vision based analysis, is based on the way human beings perceive information about their surroundings, yet it is probably the most difficult to implement in a satisfactory way. Several different approaches have been tested so far.

1. One is to build a three-dimensional model of the human hand. The model is matched to images of the hand by one or more cameras, and parameters corresponding to palm orientation and joint angles are estimated. These parameters are then used to perform gesture classification.

2. Second one to capture the image using a camera then extract some feature and those features are used as input in a classification algorithm for classification.

2.1.1 Recognition of American Sign Language Gesture in Real Time

This paper demonstrates the statistical techniques for recognition of ASL gestures in real time which comprises both the hands. An image database was created by the authors and utilized which contained large set of images for each sign. Direction histogram is the feature used for classification due to its appeal for illumination and orientation invariance. Two different approaches utilized for recognition were Euclidean distance and K-nearest neighbor metrics.
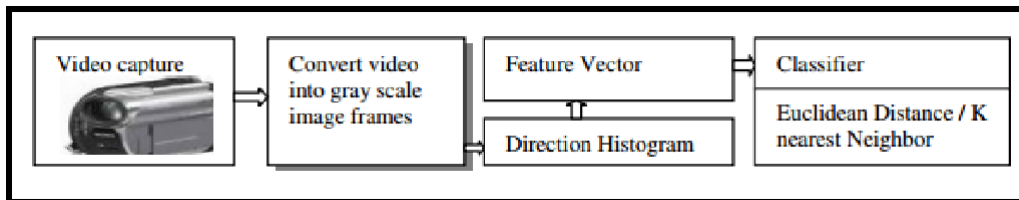
Fig 3: Methodology for real time ASL classification

3. ALGORITHMS

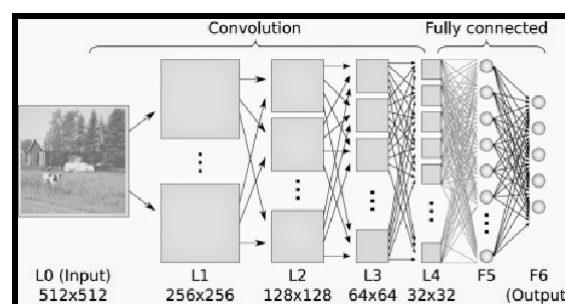
3.1 Convolutional Neural Network (CNN)

Neural networks, as its name suggests, is a machine learning technique which is modeled after the brain structure. It comprises of a network of learning units called neurons. These neurons learn how to convert **input signals** (e.g. picture of a cat) into corresponding **output signals** (e.g. the label "cat"), forming the basis of automated recognition.

A convolutional neural network (CNN, or ConvNet) is a type of feed-forward artificial neural network in which the connectivity pattern between its neurons is inspired by the organization of the animal visual cortex.

CNNs have repetitive blocks of neurons that are applied across space (for images) or time (for audio signals etc). For images, these blocks of neurons can be interpreted as 2D convolutional kernels, repeatedly applied over each patch of the image. For speech, they can be seen as the 1D convolutional kernels applied across time-windows. At training time, the weights for these repeated blocks are 'shared', i.e. the weight gradients learned over various image patches are averaged.

3.1.1 CNN Summarized in 4 Steps

There are four main steps in CNN: convolution, subsampling, activation and full connectedness.

Fig 4: Convolutional neural network
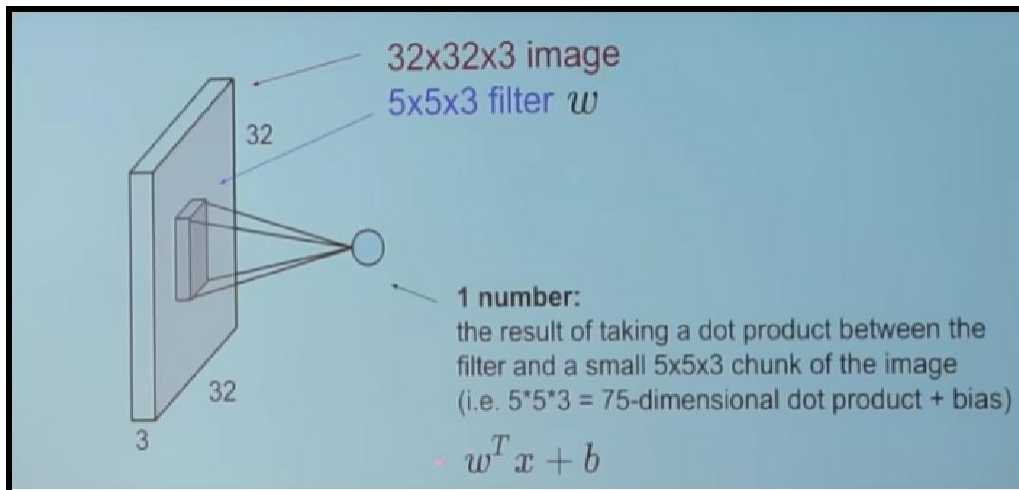
3.1.1.1 Convolution

The first layers that receive an input signal are called convolution filters. Convolution is a process where the network tries to label the input signal by referring to what it has learned in the past. If the input signal looks like previous cat images it has seen before, the "cat" reference signal will be mixed into, or convolved with, the input signal. The resulting output signal is then passed on to the next layer.
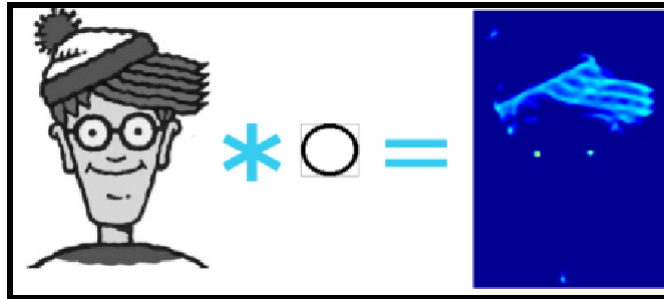
Fig 5: Convolving Wally with a circle filter. The circle filter responds strongly to the eyes.

Convolution has the nice property of being **translational invariant**. Intuitively, this means that each convolution filter represents a feature of interest (e.g whiskers, fur), and the CNN algorithm learns which features comprise the resulting reference (i.e. cat). The output signal strength is not dependent on where the features are located, but simply whether the features are present. Hence, a cat could be sitting in different positions, and the CNN algorithm would still be able to recognize it.

For e.g suppose we convolve a 32x32x3 (32x32 image with 3 channels R,G and B respectively) with a 5x5x3 filter. We take the 5*5*3 filter and slide it over the complete image and along the way take the dot product between the filter and chunks of the input image.

Fig 6: Dot Product of Filter with single chunk of Input Image[12]
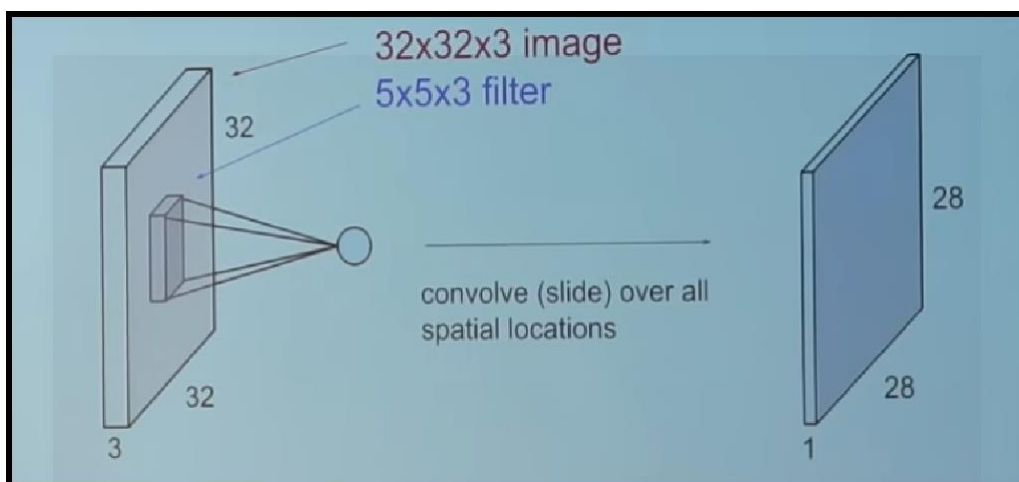
Fig 7: Dot Product or Convolve over all possible 5x5 spatial location in Input Image[12]

The convolution layer is the main building block of a convolutional neural network. The convolution layer comprises of a set of independent filters (6 in the example shown). Each filter is independently convolved with the image and we end up with 6 feature maps of shape 28*28*1.
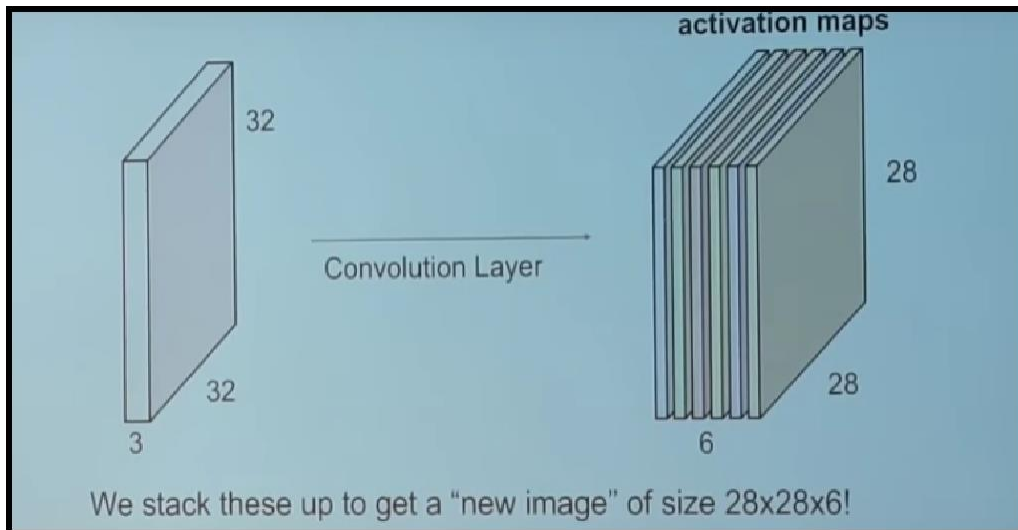
Fig 8: Input Image Convolving with a Convolutional layer of 6 independent filters[12]

The CNN may consists of several Convolutional layers each of which can have similar or different number of independent filters. For example the following diagram shows the effect of two Convolutional layers having 6 and 10 filters respectively.
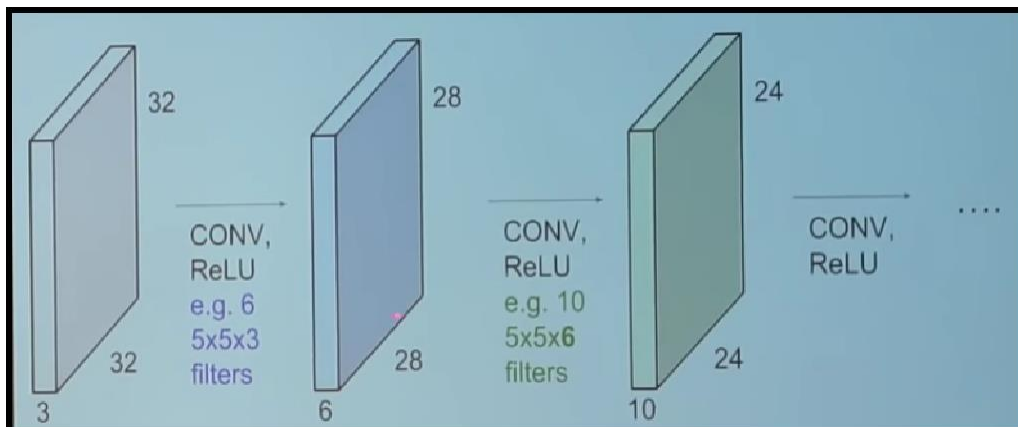
Fig 9: Input Image Convolving with two Convolutional layers having 6 and 10 filters respectively[12]

All these filters are initialized randomly and become our parameters which will be learned by the network subsequently.

3.1.1.2 Subsampling

Inputs from the convolution layer can be "smoothened" to reduce the sensitivity of the filters to noise and variations. This smoothing process is called **subsampling**, and can be achieved by taking averages or taking the maximum over a sample of the signal. Examples of subsampling methods (for image signals) include reducing the size of the image, or reducing the color contrast across red, green, blue (RGB) channels.
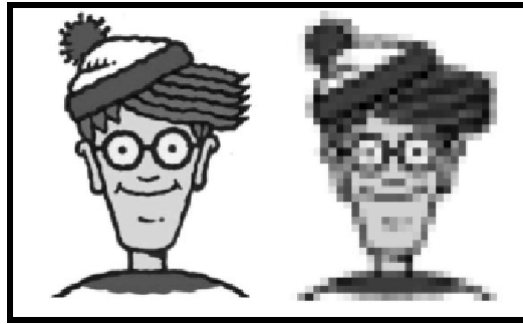
Fig 10: Sub sampling Wally by 10 times. This creates a lower resolution image.

3.1.1.3 Pooling

A pooling layer is another building block of a CNN.
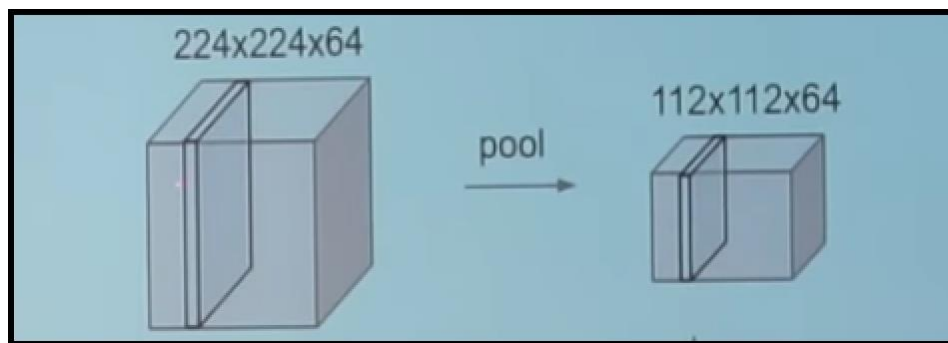
Fig 11: Pooling to reduce size from 224x224 to 112x112 [12]

Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network. Pooling layer operates on each feature map independently.

The most common approach used in pooling is max pooling in which maximum of a region taken as its representative. For example in the following diagram a 2x2 region is replaced by the maximum value in it.
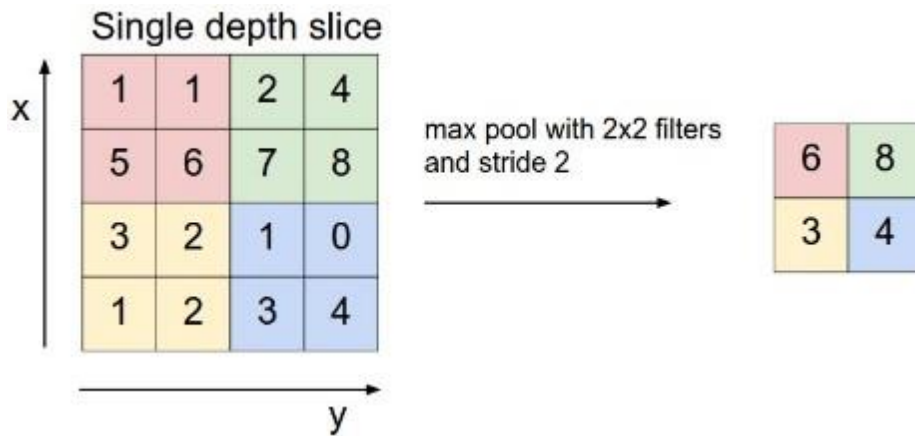
Fig 12: Max Pooling [12]

3.1.1.4 Activation

The activation layer controls how the signal flows from one layer to the next, emulating how neurons are fired in our brain. Output signals which are strongly associated with past references would activate more neurons, enabling signals to be propagated more efficiently for identification.

CNN is compatible with a wide variety of complex activation functions to model signal propagation, the most common function being the Rectified Linear Unit (ReLU), which is favored for its faster training speed.

3.1.1.5 Fully Connected

The last layers in the network are fully connected, meaning that neurons of preceding layers are connected to every neuron in subsequent layers. This mimics high level reasoning where all possible pathways from the input to output are considered.

3.1.1.6 (During Training) Loss

When training the neural network, there is additional layer called the loss layer. This layer provides feedback to the neural network on whether it identified inputs correctly, and if not, how far off its guesses were. This helps to guide the neural network to reinforce the right concepts as it trains. This is always the last layer during training.

3.1.2 Implementation

Algorithms used in training CNN are analogous to studying for exams using flash cards. First, you draw several flashcards and check if you have mastered the concepts on each card. For cards with concepts that you already know,
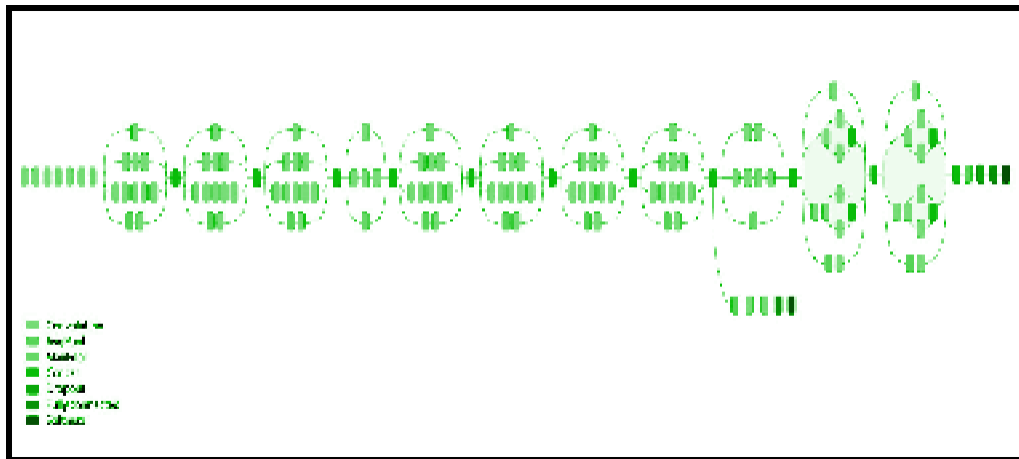
discard them. For those cards with concepts that you are unsure of, put them back into the pile. Repeat this process until you are fairly certain that you know enough concepts to do well in the exam. This method allows you to focus on less familiar concepts by revisiting them often. Formally, these algorithms are called gradient descent algorithms for forward pass learning. Modern deep learning algorithm uses a variation called stochastic gradient descent, where instead of drawing the flashcards sequentially, you draw them at random. If similar topics are drawn in sequence, the learners might over-estimate how well they know the topic. The random approach helps to minimize any form of bias in the learning of topics.

Learning algorithms require feedback. This is done using a **validation set** where the CNN would make predictions and compare them with the true labels or ground truth. The predictions which errors are made are then fed backwards to the CNN to refine the weights learned, in a so called backwards pass. Formally, this algorithm is called **backpropagation of errors**, and it requires functions in the CNN to be differentiable (almost).

CNNs are too complex to implement from scratch. Today, machine learning practitioners often utilize toolboxes developed such as Caffe, Torch, MatConvNet and Tensor flow for their work.

3.1.3 Inception [14]

We've used the Inception v3 model of the Tensor Flow library. Inception is a huge image classification model with millions of parameters that can differentiate a large number of kinds of images. We only trained the final layer of that network, so training will end in a reasonable amount of time. Inception-v3 is trained for the ImageNet Large Visual Recognition Challenge using the data from 2012 where it reached a top-5 error rate of as low as 3.46%.

Fig 13: Inception v3 model Architecture

We performed transfer learning on Inception model that is we downloaded the pre-trained Inception v3 model (trained on ImageNet Dataset consisting of 1000 classes) , added a new final layer corresponding to the number of categories and then trained the final layer on the dataset.

The kinds of information that make it possible for the model to differentiate among 1,000 classes are also useful for distinguishing other objects. By using this pre-trained network, we are using that information as input to the final classification layer that distinguishes our dataset.

3.2 Our VGG Model

VGG- Network is a convolutional neural network model proposed by K. Simonyan and A. Zisserman in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition" [1]. This architecture achieved top-5 test accuracy of 92.7% in ImageNet, which has over 14 million images belonging to 1000 classes.

It is one of the famous architectures in the deep learning field. Replacing large kernel-sized filters with 11 and 5 in the first and second layer respectively showed the improvement over AlexNet architecture, with multiple 3×3 kernel-sized filters one after another. It was trained for weeks and was using NVIDIA Titan Black GPU's.
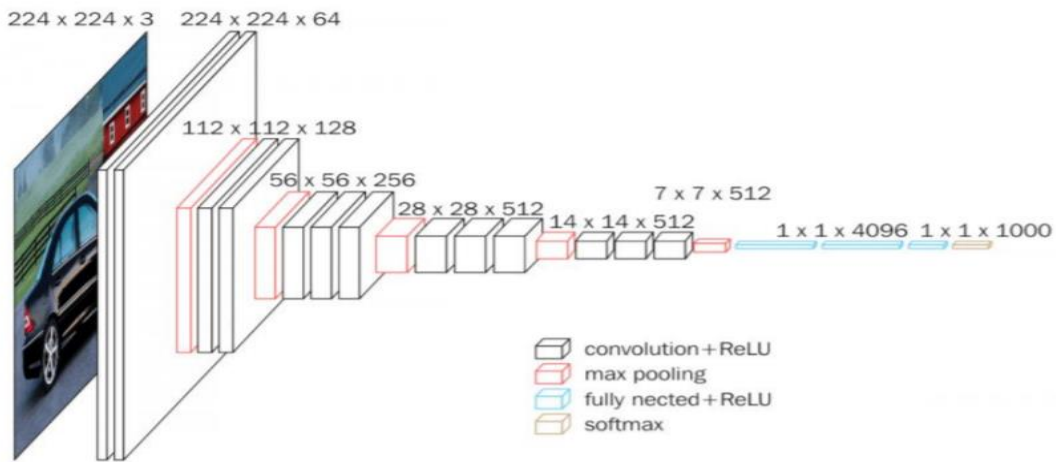
Fig 14: Our VGG Model

The input to the convolution neural network is a fixed-size 224×224 RGB image. The only preprocessing it does is subtracting the mean RGB values, which are computed on the training dataset, from each pixel.

Then the image is running through a stack of convolutional (Conv.) layers, where there are filters with a very small receptive field that is 3×3, which is the smallest size to capture the notion of left/right, up/down, and center part.

In one of the configurations, it also utilizes 1×1 convolution filters, which can be observed as a linear transformation of the input channels followed by non-linearity. The convolutional strides are fixed to 1 pixel; the spatial padding of convolutional layer input is such that the spatial resolution is maintained after convolution, that is the padding is 1 pixel for 3×3 Conv. Layers. Then the Spatial pooling is carried out by five max-pooling layers, 16 which follow some of the Conv. layers but not all the Conv. layers are followed by max-pooling. This Max-pooling is performed over a 2×2-pixel window, with stride 2.

4. EXPERIMENTAL DESIGN

4.1 Data Set Used

The data set used consists of American Sign Language (ASL) Gestures, with 28,967 images belonging to 30 gestures. 4 non-expert subjects executed about 1000 repetitions of every gesture thereby producing about 1000 images per class or gesture.

Id	Name	Id	Name	Id	Name
1	Bathroom	13	Mom	25	Monday
2	Dad	14	Need	26	Saturday
3	Deaf	15	No	27	Sunday
4	Drink	16	Open	28	Thursday
5	Eat	17	Please	29	Tuesday
6	Goodbye	18	Sorry	30	Wednesday
7	Hello!	19	Space		
8	I Love You	20	Telephone		
9	Like	21	Think		
10	Love	22	YES		
11	Me	23	You		
12	Mine	24	Friday		

Out of the 1,000 images per gesture or class, 75% i.e. 750 were used for training and 25% i.e. 250 were used for testing.

4.2 Our Approach

In our Approach we created our data set by capturing the gestures from a webcam using Open CV and the MediaPipe API to capture only the hands, then, we split our data set into train set and test set, to train the model to be able to make predictions.

4.2.1 Methodology

- First, we captured the frames from the real time webcam.
- After the first step, only the hands were detected.
- Then, we started taking photos and that were stored into a preset directory for our data set.
- The images were cropped to the hands only.
- Then we split our data set into train and test sets
- Trained the model then evaluated the test results
- Finally, make predictions by recognizing American Sign Language gestures in real time

In further subsections of this section, each step of the methodology has been shown diagrammatically for better understanding of that step.
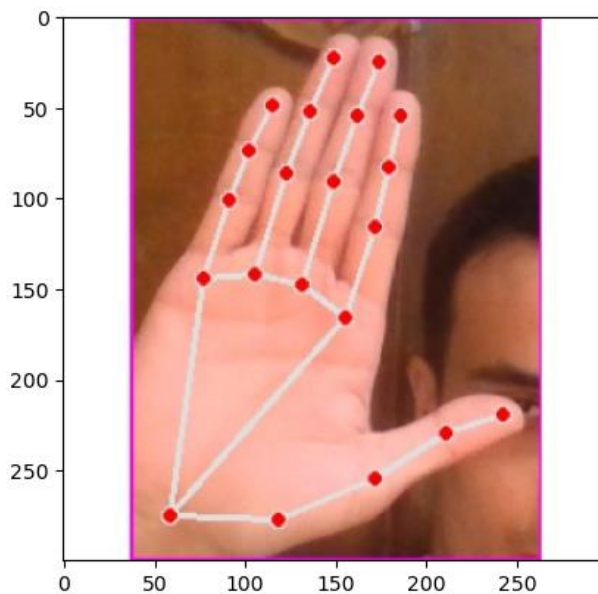
4.2.1.1 Webcam Capture and Hands Detection (Creating Our Data set)

Taking a webcam captures for different hand gestures using Open CV with both hands being detected using MediaPipe API.

The final image consists of a cropped image only around the hand.
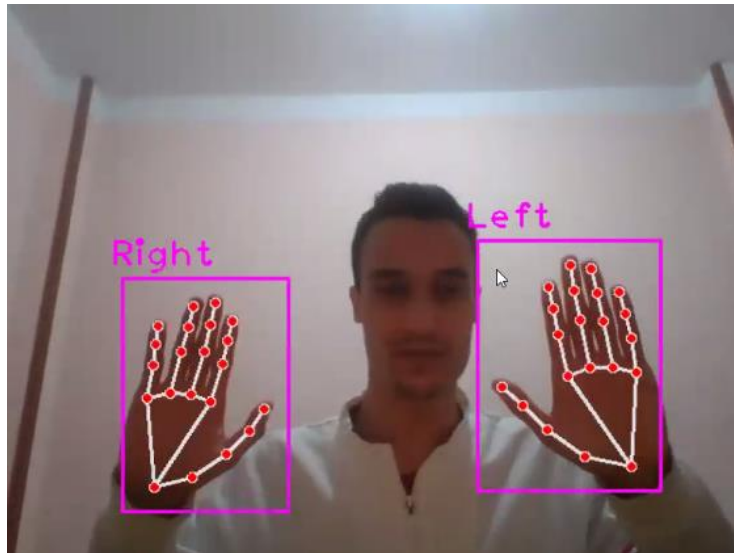
Fig 15: Capturing and Detecting Hands

Fig 16: Final Image after Capturing Sign Language Gesture

4.2.1.2 Train CNN and Prediction

```python
classifier1 = classifier_vgg16.output # Head mode
classifier1 = Flatten()(classifier1) # Adding layer of flatten
classifier1 = Dense(units=256, activation='relu')(classifier1)
classifier1 = Dropout(0.6)(classifier1)
classifier1 = Dense(units=2, activation='softmax')(classifier1)


model = Model(inputs = classifier_vgg16.input , outputs = classifier1)
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

Fig 17 Our Model

```
history = model.fit(X_train, y_train_cat, epochs =5, batch_size = 64,validation_data=(X_val,y_val_cat))

Epoch 1/5
12/12 [==============================] - 15s 343ms/step - loss: 0.7889 - accuracy: 0.8516 - val_loss: 0.0150 - val_accuracy: 0.9948
Epoch 2/5
12/12 [==============================] - 3s 256ms/step - loss: 0.0237 - accuracy: 0.9935 - val_loss: 9.0177e-07 - val_accuracy: 1.0000
Epoch 3/5
12/12 [==============================] - 3s 256ms/step - loss: 0.0547 - accuracy: 0.9896 - val_loss: 0.0089 - val_accuracy: 0.9948
Epoch 4/5
12/12 [==============================] - 3s 255ms/step - loss: 0.0224 - accuracy: 0.9961 - val_loss: 1.9792e-05 - val_accuracy: 1.0000
Epoch 5/5
12/12 [==============================] - 3s 255ms/step - loss: 1.8782e-05 - accuracy: 1.0000 - val_loss: 9.8826e-09 - val_accuracy: 1.0000

score = model.evaluate(x = X_test, y =y_test_cat, verbose = 0)
print('Accuracy for test images:', round(score[1]*100, 3), '%')

Accuracy for test images: 100.0 %
```
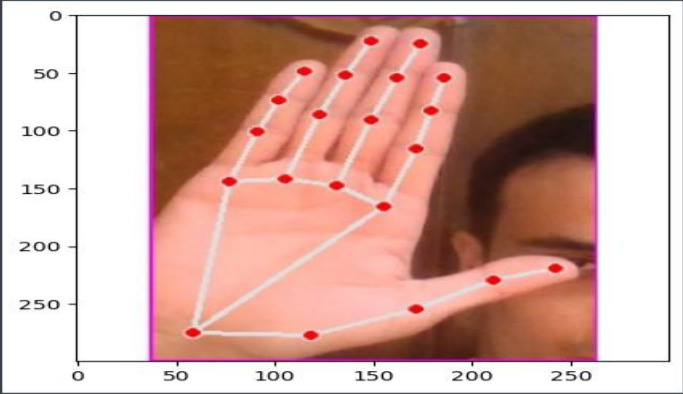
Fig 18 Evaluating our Model (Accuracy = 100%)

Fig 19 Making Predictions

17

6. RESULTS

5.1 Result of First Gesture (Hello)

Fig 20 (Hello)

5.2 Result of Second Gesture (Telephone)

Fig 21 (Telephone)

5.3 Result of Third Gesture (Please)

Fig 22 (Please)

5.4 Result of Fourth Gesture (Sorry)

Fig 23 (Sorry)

7. CONCLUSION AND FUTURE WORK

Hand gestures are a powerful way for human communication, with lots of potential applications in the area of human computer interaction. Vision-based hand gesture recognition techniques have many proven advantages compared with traditional devices. However, hand gesture recognition is a difficult problem and the current work is only a small contribution towards achieving the results needed in the field of sign language gesture recognition. This report presented a vision-based system able to interpret isolated hand gestures from the American Sign Language (ASL).

This project for individual gestures can also be extended recognizing continuous sign language gestures with both hands and the face using the Single Shot Multi-Box Detector.

We wish to extend our work further by expanding our data set to include almost every single word in the American Sign Language (ASL), and then deploy this project into a mobile application to be used in real life.