REAL-TIME COMMUNICATION SYSTEM POWERED BY AI FOR SPECIALLY ABIED

NALAIYA THIRAN PROJECT REPORT

IBM-Project-9389-1658999257

Team ID : PNT2022TMID08443

Submitted by

M. PRADEEP	**(814319104037)**
R. RAVI KUMAR	**(814319104042)**
S. SUREN	**(814319104055)**
K. VIGNESH	**(814319104058)**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINNERING

IN

COMPUTER SCIENCE AND ENGINEERING

DHANALAKSHMI SRINIVASAN ENGINEERING COLLEGE

(AUTONOMOUS)

PERAMBALUR-621212

PROJECT REPORT

ABSTRACT

People with impaired speech and hearing uses Sign language as a form of communication. Disabled People use this sign language gestures as a tool of non-verbal communication to express their own emotions and thoughts to other common people. Conversing with people having a hearing disability is a major challenge. Deaf and Mute people use hand gesture sign language to communicate, hence normal people face problems in recognizing their language by signs made. Hence there is a need for systems that recognize the different signs and conveys the information to normal people. But these common people find it difficult to understand their expression, thus trained sign language expertise are needed during medical and legal appointment, educational and training session. Over the past few years, there has been an increase in demand for these services. Other form of services such as video remote human interpret using the high-speed Internet connection, has been introduced, thus these services provides an easy to use sign language interpret service, which can be used and benefited, yet have major limitations. To address this problem, we can implement artificial intelligence technology to analyse the user's hand with finger detection. In this proposed system we can design the vision based system in real time environments. And then using deep learning algorithm named as Convolutional neural network algorithm to classify the sign and provide the label about recognized sign.

1. INTRODUCTION

1.1 PROJECT OVERVIEW

Sign language recognition is the process of translating the user's gestures and signs into text. It aids those who are unable to interact with the general population in communication Using image processing methods and neural networks, the motion is mapped to pertinent text in the training data, transforming unprocessed photos and videos into text that can be read and understood. People who are dumb are typically prohibited from having regular conversations with other people in society. They sometimes struggle to communicate with regular people through gestures because the majority of people only recognise a small number of them. People who are deaf or have hearing loss are unable to communicate vocally, so they must frequently use some type of visual communication. The primary form of communication for the deaf and dumb community is sign language. Similar to other languages, it contains grammar and vocabulary, but it communicates primarily through images.

1.2 PURPOSE

The problem occurs when those who are stupid or deaf attempt to use these grammars of sign language to interact with others. This is because the majority of individuals are not familiar with these grammar standards. It has been noted that a foolish person can only communicate with members of his or her family or the deaf community. The popularity of international programmes and the financing they get highlight the value of sign language. In this age of technology, a computer-based solution is highly desired by the dumb community. Teaching a computer to recognise speech, facial expressions of emotion, and human gestures are some steps toward achieving this goal. Gestures are used to convey information nonverbally. Humans are capable of an endless amount of motions at any given moment. Since human motions are seen visually, computer vision researchers are particularly interested in them. The project's objective is to develop an HCI that can recognise human motions. These motions must be translated into machine language using a challenging programming process. In our paper, we concentrate on Image Processing and Template Matching for better output creation.

2. LITERATURE REVIEW

2.1 EXISTING PROBLEM

2.1.1 TITLE: A STUDY ON ARABIC SIGN LANGUAGE RECOGNITION FOR DIFERENTLY ABLED USING ADVANCED MACHINE LEARNING CLASSIFERS

AUTHOR: MOHAMMED MUSTAFA,2021.

Around 70 million people use sign language worldwide, and an automated method for translating it could significantly improve communication between sign language users and those who might not understand it. Nonverbal communication that includes the use of other bodily parts is called sign language. Face expressions, together with movements of the hands, eyes, and lips used in sign language communication to communicate information. People who have trouble hearing or speaking rely heavily on sign language as a form of communication in daily life. The inconsistent shape, size, and posture of the hands or fingers in an image was however shown by computer translation of sign language, which was highly complicated. SLR can be used in two main ways: based on picture or sensor. The main advantage of image-based frameworks is that people do not need to use complicated equipment. In any case, the preprocessing process necessitates large computations. Sensors frameworks use gloves fitted with sensors rather of relying just on cameras. Like spoken language, sign language does not confined to a certain location or region. It is trained differently over the world (Shin et al. 2019). It is sometimes referred to as Chinese Sign Language, American Sign Language, African Sign Language, and Arabic Sign Language (ArSL). India does not have a standardised sign language with important modifications, unlike sign languages in Europe and America. However, a dictionary of ISL was just created by Coimbatore's Vivekananda University for the Ramakrishna Missions. there are nearly there are currently 2037 signs available in Indian Sign Language (ISL). Similar to how SLR models are separated into sensor glove based and vision-based categories. Recent research on SLR can be divided into contact-based and vision-based methods. Physical interaction between sensing devices is a component of the contact-based technique and clients. It often employs an instrumented glove that uses electromyography, inertial estimation, or electromagnetic to capture information on the executed sign's position, extension, direction, and angle.

2.1.2 TITLE: SIGN LANGUAGE TRANSFORMERS: JOINT END-TO-END SIGN LANGUAGE RECOGNITION AND TRANSLATION

AUTHOR: NECATI CIHAN CAMG¨OZ, 2021.

The translation is improved by having a mid-level sign gloss representation, which efficiently recognises the various signs, according to earlier research on sign language translation. Performance significantly In fact, gloss level tokenization is necessary for the state-of-the-art in translation to function. We present a unique architecture based on transformers that simultaneously learns Continuous Sign Language Recognition and Translation while being end-to-end trainable. This is accomplished by combining the recognition and translation issues into a single, unified architecture employing a Connectionist Temporal Classification (CTC) loss. This collaborative approach achieves significant performance improvements while simultaneously resolving two related sequence-to-sequence learning problems without the need for ground-truth timing information. The primary form of communication for the Deaf community is sign language, which is their native tongue. They use a variety of complementing channels as visual languages to communicate ideas. This comprises both manual and non-manual characteristics, such as head, shoulder, and torso movement as well as manual characteristics like hand shape, movement, and stance. The purpose of sign language translation is to either extract an equivalent spoken language sentence from written text or translate written text into a video of signs. A clip of someone doing the continuous sign. However, a large portion of this latter work is done in the field of computer vision, where linguists refer to these channels as articulators. Word embedding with spatial embedding has concentrated on understanding the order of sign glosses rather than providing a complete translation into a spoken language counterpart (Sign Language Translation, or SLT). This distinction is crucial because spoken and sign languages have significantly different grammatical structures. Word order variations, the use of multiple channels to convey simultaneous information, and the use of direction and space to indicate the relationships between objects are just a few examples of these differences.

2.1.3 TITLE: SIGN LANGUAGE RECOGNITION SYSTEMS: A DECADE SYSTEMATIC LITERATURE REVIEW

AUTHOR: ANKITA WADHAWAN,2020.

As spoken languages are pronounced with the lips and heard with the ear, they utilise the "vocal-auditory" channel. Additionally, all writing systems come from, or are spoken languages' representations. Because they use the "corporal-visual" channel, which is created with the body and perceived with the eyes, sign languages (SLs) are unique. SLs are widely used by the deaf communities but are not internationally recognized. They are considered natural languages because deaf people can spontaneously gather and communicate with one another anywhere. SLs have independent vocabularies and grammatical structures and are not descended from spoken languages . The signs that the deaf use actually have the same internal structure as spoken words. The signs of SLs are produced using a small number of different sounds, just as hundreds of thousands of English words are. A fixed number of gestural characteristics. As a result, signs are not complete gestures but rather can be analysed as a collection of linguistically important characteristics. A gloss, the basic component of an SL and the closest representation of a sign's meaning, is made up of combinations of the aforementioned qualities. SLs, comparable to the spoken ones contain a list of grammatically flexible rules that apply to both manual and non-manual elements. Signers utilise both of them concurrently (and frequently with a flexible temporal structure) to create phrases in an SL. A particular feature may be the most important consideration when interpreting a gloss, depending on the context. It can change a verb's meaning, provide spatial and temporal context, and distinguish between things and people. A signer's glosses can be inferred from video recordings using a process known as sign language recognition (SLR). Despite the fact that there is a lot of labour, There is a severe paucity of comprehensive experimental research in the subject of SLR. Additionally, most articles don't release their code or present findings from all available datasets. As a result, experimental findings in the field of SL are rarely repeatable and interpretable.

2.1.4 TITLE: A COMPREHENSIVE STUDY ON SIGN LANGUAGE RECOGNITION METHODS

AUTHOR: NIKOLAS ADALOGLOU,2020

The sign language is used widely by people who are deaf-dumb these are used as a medium for communication. A sign language is nothing but composed of various gestures formed by different shapes of hand, its movements, orientations as well as the facial expressions. There are around 466 million people worldwide with hearing loss and 34 million of these are children. `Deaf' people have very little or no hearing ability. They use sign language for communication. People use different sign languages in different parts of the world. Compared to spoken languages they are very less in number. In existing system, lack of datasets along with variance in sign language with locality has resulted in restrained efforts in finger gesture detection. Existing project aims at taking the basic step in bridging the communication gap between normal people and deaf and dumb people using Indian sign language. Effective extension of this project to words and common expressions may not only make the deaf and dumb people communicate faster and easier with outer world, but also provide a boost in Developing autonomous systems for understanding and aiding them. The Indian Sign Language lags behind its American Counterpart as the research in this field is hampered by the lack of standard datasets. In addition to the intrinsic challenges of human motion analysis (such as variations in the participants' appearances, the characteristics of the human silhouette, and the execution of the repetition of operations, the presence of obstructions, etc.) A signer's glosses can be inferred from video recordings using a process known as sign language recognition (SLR). Despite the fact that there is a lot of labour, There is a severe paucity of comprehensive experimental research in the subject of SLR. Additionally, most articles don't release their code or present findings from all available datasets.

2.1.5 TITLE: TRANSFERRING CROSS-DOMAIN KNOWLEDGE FOR VIDEO SIGN LANGUAGE RECOGNITION

AUTHOR: DONGXU LI,2020

As a fundamental sign language interpretation task, word-level sign language recognition (WSLR) aims to help deaf people communicate. However, WSLR is very difficult because it requires quick body movements, facial expressions, and complex, fine-grained hand gestures. Isolated Sign Words Web News Sign Words Localizer has been demonstrated recently using deep learning approaches. Our model learns domain-invariant characteristics to transfer knowledge from web news signs to WSLR models. Our model recognises the example frames in the figure as the signature that best captures the gesture on the WSLR job, their advantages. Although the largest existing datasets have a limited number of instances, e.g., on average 10 to 50 instances per word, annotating WSLR datasets requires domain-specific knowledge. This is significantly less than typical video datasets on action learning and recognition, for example. The sign recognition task's inadequate training data may cause overfitting or in some other way hinder WSLR's performance.Models under realistic circumstances. On the other hand, there are many readily available news videos with subtitles available online that could be useful for WSLR. Despite the availability of sign news videos, it is quite difficult to translate this knowledge to WSLR. First, there are no annotations of temporal location or categories and just flimsy labels for the presence of signs in subtitles. Furthermore, these labels are loud. In this study, we provide a technique for transferring cross-domain knowledge from news signs to WSLR models to enhance their performance. More specifically, using a base WSLR model in a sliding window fashion, we first create a sign word localizer to extract sign words. Then, we suggest jointly coarse-aligning two domains. Employing isolated and news indicators to train a classifier. We compute and store the centroid of each class of the coarsely-aligned new words in an external memory termed prototype memory after getting the representations of the coarsely-aligned news words.

2.2 REFERENCES

[1] Mohammed Mustafa, " A study on arabic sign language recognition for diferently abled using advanced machine learning classifers",2020.

[2] Necati Cihan Camg¨oz, ": Sign language transformers: joint end-to-end sign language recognition and translation",2021.

[3]Ankita Wadhawan, "Sign language recognition systems: a decade systematic literature review",2020

[4] Nikolas Adaloglou " a comprehensive study on sign language recognition methods",2020.

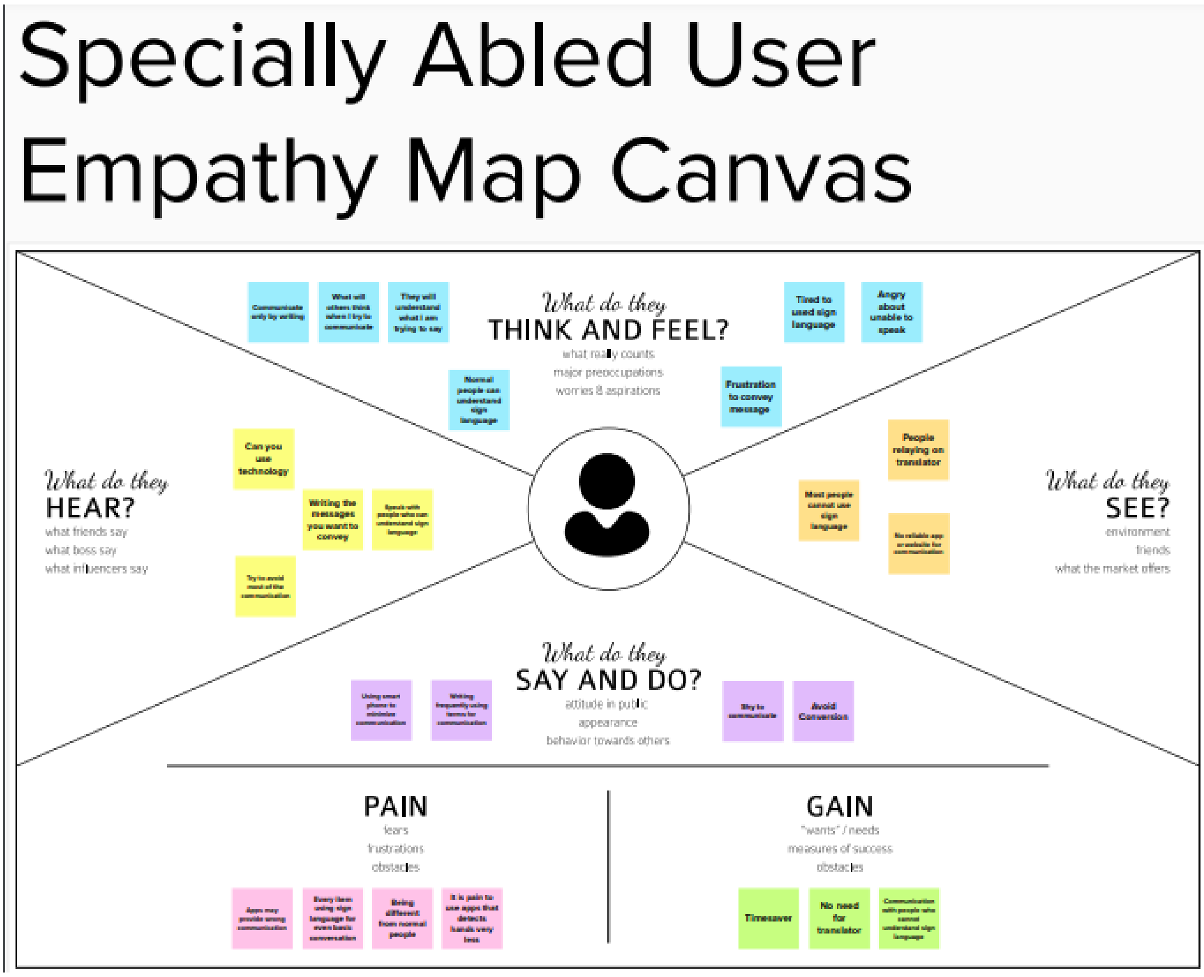
[5] Dongxu li," transferring cross-domain knowledge for video sign language recognition",2020

2.3 PROBLEM STATEMENT DEFINITION

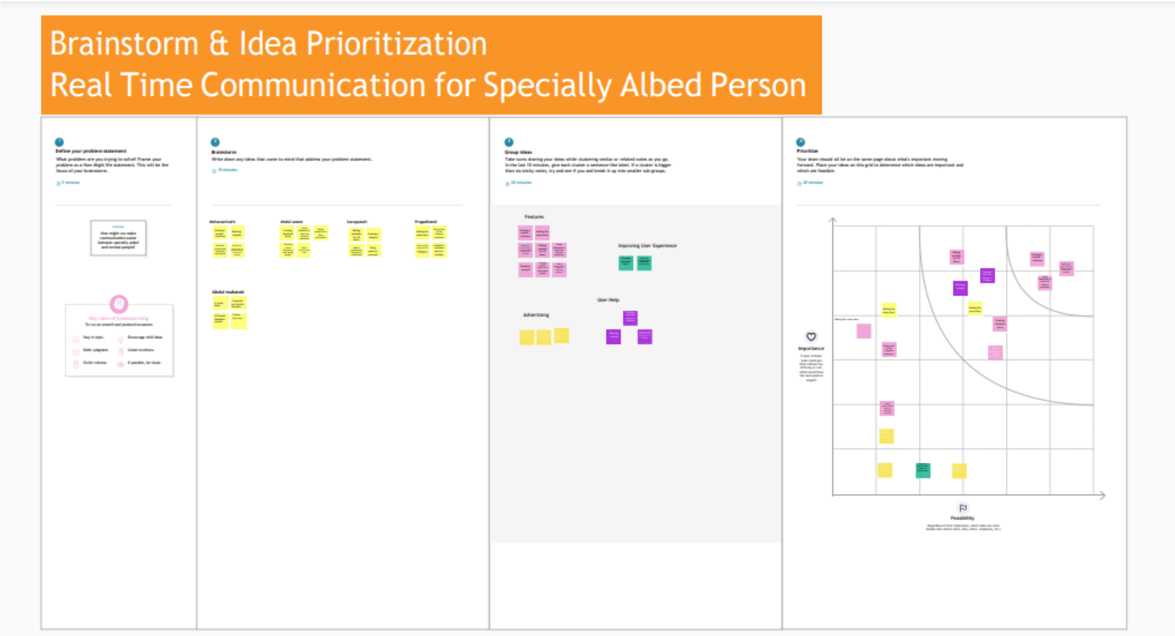
The sign language is used widely by people who are deaf-dumb these are used as a medium for communication. A sign language is nothing but composed of various gestures formed by different shapes of hand, its movements, orientations as well as the facial expressions. There are around 466 million people worldwide with hearing loss and 34 million of these are children. `Deaf' people have very little or no hearing ability. They use sign language for communication. People use different sign languages in different parts of the world. Compared to spoken languages they are very less in number. In existing system, lack of datasets along with variance in sign language with locality has resulted in restrained efforts in finger gesture detection. Existing project aims at taking the basic step in bridging the communication gap between normal people and deaf and dumb people using Indian sign language. Effective extension of this project to words and common expressions may not only make the deaf and dumb people communicate faster and easier with outer world, but also provide a boost in Developing autonomous systems for understanding and aiding them. The Indian Sign Language lags behind its American Counterpart as the research in this field is hampered by the lack of standard datasets

3. IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS



3.2 IDEATION & BRAINSTORMING



3.3 PROPOSED SOLUTION

In computer vision-based gesture recognition, the camera is used as input, and image processing is done before recognition. The techniques utilised to recognise the processed motions after that include Neural Network approaches and the region of interest algorithm. A vision-based sign language identification system has a fundamental drawback in that the process of gathering images is sensitive to a variety of environmental factors, including camera positioning, background circumstances, and lightning sensitivity. However, it is more practical and economical than using a camera and tracker to gather information. However, neural network techniques like the Hidden Markov Model are integrated with camera data for increased accuracy.

3.4 PROBLEM SOLUTION FIT

People who are deaf-dumb frequently employ sign language as a means of communicating. A sign language is nothing more than a collection of varied hand gestures created by varying hand shapes, movements, and orientations, as well as face expressions. 34 million of the 466 million people with hearing loss in the world's population are children. People who identify as "deaf" have very little or no hearing. They communicate using sign language. Around the world, many sign languages are used by people. They are quite few in number when compared to spoken languages. In computer vision-based gesture recognition, the camera is used as input, and image processing is done before recognition. The techniques utilised to recognise the processed motions after that include Neural Network approaches and the region of interest algorithm. A vision-based sign language identification system has a fundamental drawback in that the process of gathering images is sensitive to a variety of environmental factors, including camera positioning, background circumstances, and lightning sensitivity. However, it is more practical and economical than using a camera and tracker to gather information. However, neural network techniques like the Hidden Markov Model are integrated with camera data for increased accept.

4.REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENT

HAND IMAGE ACQUISITION:

The hand gesture, during daily life, is a natural communication method mostly used only among people who have some difficulty in speaking or hearing. However, a human computer interaction system based on gestures has various application scenarios. In this module, we can input the hand images from real time camera. The inbuilt camera can be connected to the system. Gesture recognition has become a hot topic for decades. Nowadays two methods are used primarily to perform gesture recognition. One is based on professional, wearable electromagnetic devices, like special gloves. The other one utilizes computer vision. The former one is mainly used in the film industry. It performs well but is costly and unusable in some environment. The latter one involves image processing. However, the performance of gesture recognition directly based on the features extracted by image processing is relatively limited. Hand image captured from web camera. The purpose of Web camera is to capture the human generated hand gesture and store its image in memory. The package called python framework is used for storing image in memory.

BINARIZATION

Background subtraction is one of the major tasks in the field of computer vision and image processing whose aim is to detect changes in image sequences. Background subtraction is any technique which allows an image's foreground to be extracted for further processing (object recognition etc.). Many applications do not need to know everything about the evolution of movement in a video sequence, but only require the information of changes in the scene, because an image's regions of interest are objects (humans, cars, text etc.) in its foreground. After the stage of image preprocessing (which may include image denoising, post processing like morphology etc.) object localization is required which may make use of this technique. Detecting foreground to separate these changes taking place in the foreground of the background. It is a set of techniques that typically analyze the video sequences in real time and are recorded with a stationary camera. All detection techniques are based on modeling the background of the image i.e. set the background and detect which changes occur. Defining the background can be very difficult when it contains shapes, shadows, and moving objects. In defining the background it is assumed that the stationary objects could vary in color and intensity over time. Scenarios where these techniques apply tend to be very diverse. There can be highly variable sequences, such as images with very different lighting, interiors, exteriors, quality, and noise. In addition to processing in real time, systems need to be able to adapt to these changes. The implement the techniques to extract the foreground from background image. Using Binarization approach to assign the values to background and foreground. Foreground pixels are identified in real time environments.

REGION OF FINGER DETECTION

Segmentation refers to the process of partitioning a digital image into multiple segments. In other words, grouping of pixels into different groups is known as Segmentation. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain visual characteristics The division of an image into meaningful structures, image segmentation, is often an essential step in image analysis, object representation, visualization, and many other image processing tasks. But segmentation of a satellite image into differently textured regions (groups) is a difficult problem. One does not know a priori what types of textures exist in a satellite image, how many textures there are, and what regions have certain textures. The monitoring task can be performed by unsupervised segmentation and supervised segmentation techniques. A region of interest (ROI) is a subset of an image or a dataset identified for a particular purpose. In other words, region of interest (ROI) can be defined as a portion of an image which is needed to be filtered or to be performed some other operation on.

CLASSIFICATION OF FINGER GESTURES

Artificial Neural Networks (ANN) can learn and therefore can be trained to recognize patterns, find solutions, forecast future events and classify data. CNN is well documented to be used for traffic related tasks. Neural Networks learning and behavior is dependent on the way its individual computing elements are connected and by the strengths of these connections or weights. These weights can be adjusted automatically by training the network according to a specified learning rule until it performs the desired task correctly. CNN is a supervised learning method i.e. a machine learning algorithm that uses known dataset also known as training dataset. These known parameters help CNN to make predictions. Input data along with their response values are the fundamental components of a training dataset. In order to have higher predictive power and the ability to generalize for several new datasets, the best way is to use larger training datasets. The fingers can be classified by using convolutional neural network algorithm. CNN is a common method of training artificial neural networks so as to minimize the objective function. It is a supervised learning method, and is a generalization of the delta rule. It requires a dataset of the desired output for many inputs, making up the training set. It is most useful for feed-forward networks (networks that have no feedback, or simply, that have no connections that loop).

SIGN RECOGNITION

Sign Language is a well-structured code gesture, every gesture has meaning assigned to it. Sign Language is the only means of communication for deaf people. With the advancement of science and technology many techniques have been developed not only to minimize the problem of deaf people but also to implement it in different fields. From the classification of sign features, label the signs with improved accuracy rate.

4.2 NON FUNCTIONAL REQUIREMENTS

Usability

The system shall allow the users to access the system with pc using web application. The system uses a web application as an interface. The system is user friendly which makes the system easy

Availability

The system is available 100% for the user and is used 24 hrs a day and 365 days a year. The system shall be operational 24 hours a day and 7 days a week.

Scalability

Scalability is the measure of a system's ability to increase or decrease in performance and cost in response to changes in application and system processing demands.

Security

A security requirement is a statement of needed security functionality that ensures one of many different security properties of software is being satisfied.

Performance

The information is refreshed depending upon whether some updates have occurred or not in the application. The system shall respond to the member in not less than two seconds from the time of the request submittal. The system shall be allowed to take more time when doing large processing jobs. Responses to view information shall take no longer than 5 seconds to appear on the screen.

Reliability

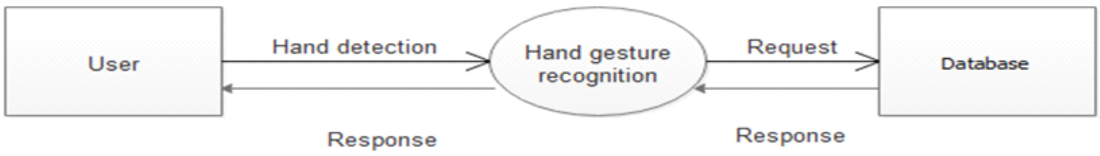
The system has to be 100% reliable due to the importance of data and the damages that can be caused by incorrect or incomplete data. The system will run 7 days a week. 24 hours a day.

5. PROJECT DESIGN

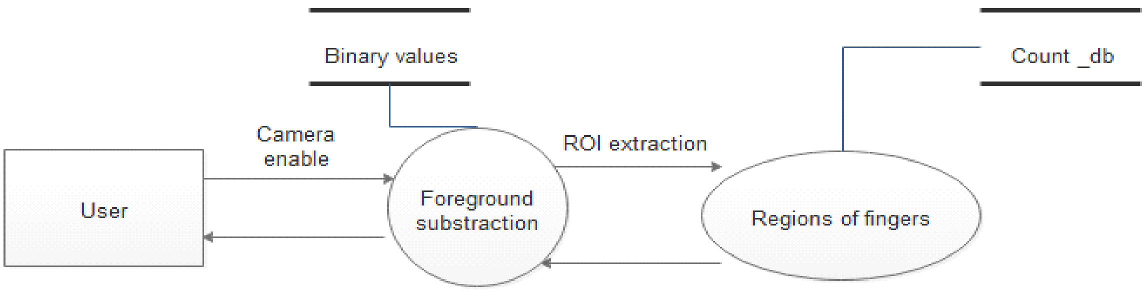
5.1 DATA FLOW DIAGRAMS

A data flow diagram is a two-dimensional diagram that explains how data is processed and transferred in a system. The graphical depiction identifies each source of data and how it interacts with other data sources to reach a common output. Individuals seeking to draft a data flow diagram must identify external inputs and outputs, determine how the inputs and outputs relate to each other, and explain with graphics how these connections relate and what they result in.

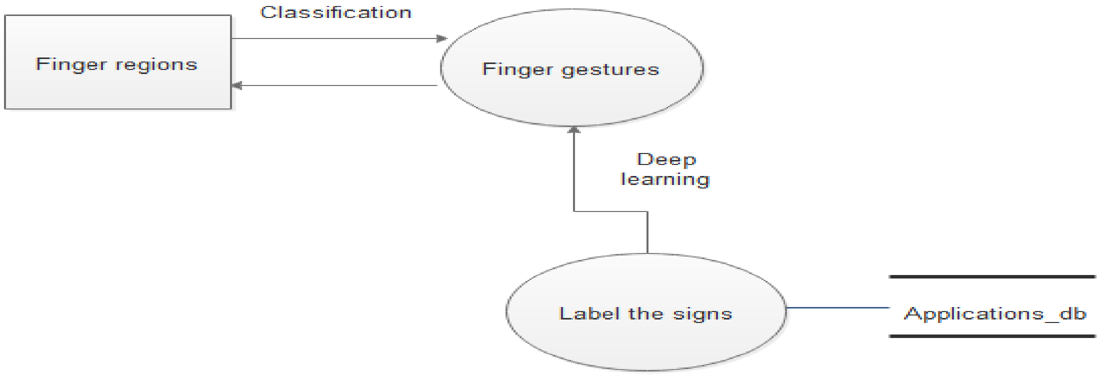
LEVEL 0:



Level 1:

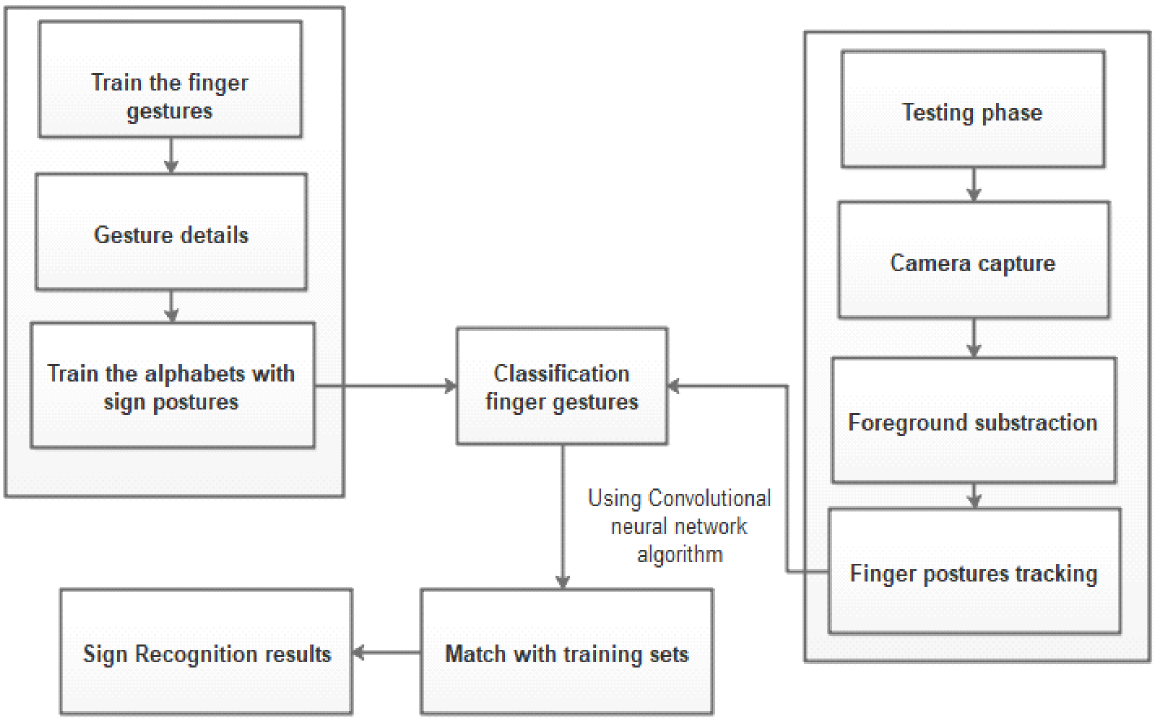


Level 2:



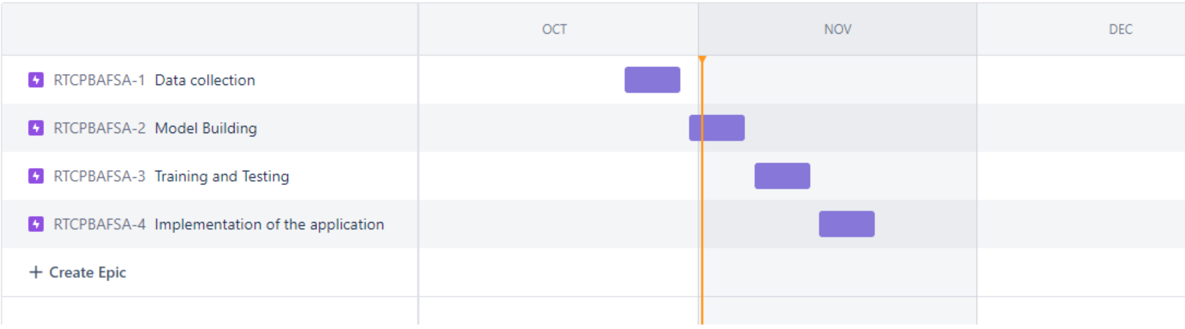
5.2 SOLUTION & TECHNICAL ARCHITECTURE

Software architecture involves the high level structure of software system abstraction, by using decomposition and composition, with architectural style and quality attributes. A software architecture design must conform to the major functionality and performance requirements of the system, as well as satisfy the non-functional requirements such as reliability, scalability, portability, and availability. Software architecture must describe its group of components, their connections, interactions among them and deployment configuration of all components.



6. PROJECT PLANNING & SCHEDULING

6.1SPRINT PLANNING & ESTIMATION



6.2 SPRINT DELIVERY SCHEDULE

Project Tracker, Velocity & Burndown Chart: (4 Marks)

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	10	6 Days	24 Oct 2022	29 Oct 2022	8	29 Oct 2022
Sprint-2	10	6 Days	31 Oct 2022	04 Nov 2022	5	04 Nov 2022
Sprint-3	10	6 Days	07 Nov 2022	11 Nov 2022	7	11 Nov 2022
Sprint-4	10	6 Days	14 Nov 2022	18 Nov 2022	5	18 Nov 2022

6.3 REPORTS FROM JIRA

JIRA has categorized reports in four levels, which are

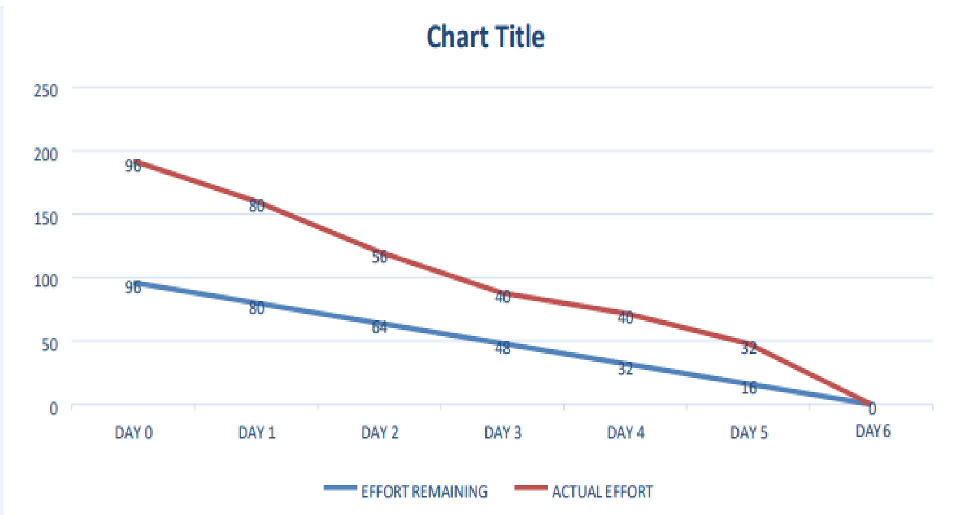
- Agile
- Issue Analysis
- Forecast & Management
- Others

Velocity:

$$AV = \frac{\text{sprint duration}}{\text{velocity}}$$

$$AV \quad = 6/10 = 0.6$$

Burndown chart:



SPRINT BURNDOWN

7. CODING & SOLUTION

7.1 FEATURE 1

```
import csv

import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split

RANDOM_SEED = 42

dataset = 'model/keypoint_classifier/keypoint.csv'
model_save_path = 'keypoint_classifier_new.h5'

NUM_CLASSES = 26

X_dataset = np.loadtxt(dataset, delimiter=',', dtype='float32',
usecols=list(range(1, (21 * 2) + 1)))

y_dataset = np.loadtxt(dataset, delimiter=',', dtype='int32', usecols=(0))

print(len(X_dataset))
print(len(y_dataset))

print(y_dataset)
print(X_dataset.shape)

train_ratio = 0.80
test_ratio = 0.20

X_train, X_test, y_train, y_test = train_test_split(X_dataset, y_dataset,
test_size=1-train_ratio, random_state=RANDOM_SEED)

model = tf.keras.models.Sequential([
    tf.keras.layers.Input((21 * 2, )),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(20, activation='relu'),
    tf.keras.layers.Dropout(0.4),
    tf.keras.layers.Dense(10, activation='relu'),
    tf.keras.layers.Dense(NUM_CLASSES, activation='softmax')
])

model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics=['ac
curacy'])

cp_callback = tf.keras.callbacks.ModelCheckpoint(model_save_path, verbose=1,
save_weights_only=False)
es_callback = tf.keras.callbacks.EarlyStopping(patience=20, verbose=1)

model.summary()

hist=model.fit(X_train,y_train,epochs=500,batch_size=128,validation_data=(X_test,
y_test),callbacks=[cp_callback, es_callback])
```

```python
import matplotlib.pyplot as plt
# val_loss, val_acc = model.evaluate(X_test, y_test, batch_size=128)
scores = model.evaluate(X_test,y_test, verbose=0)
#print("CNN Error: %.2f%%" % (100 - scores[1] * 100))
model.save(model_save_path,include_optimizer=False)
plt.plot(hist.history['accuracy'])
plt.plot(hist.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
# summarize history for loss

plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

from keras.models import load_model
from sklearn.metrics import classification_report, confusion_matrix
import numpy as np
import time
import matplotlib.pyplot as plt


def plot_confusion_matrix(cm,
                          target_names,
                          title='Confusion matrix',
                          cmap=None,
                          normalize=True):
    import itertools

    accuracy = np.trace(cm) / float(np.sum(cm))
    misclass = 1 - accuracy

    if cmap is None:
        cmap = plt.get_cmap('Blues')

    plt.figure(figsize=(20, 20))
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()

    if target_names is not None:
        tick_marks = np.arange(len(target_names))
        plt.xticks(tick_marks, target_names, rotation=45)
        plt.yticks(tick_marks, target_names)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    thresh = cm.max() / 1.5 if normalize else cm.max() / 2
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        if normalize:
            plt.text(j, i, "{:0.4f}".format(cm[i, j]),
```

```
                     horizontalalignment="center",
                     color="white" if cm[i, j] > thresh else "black")
        else:
            plt.text(j, i, "{:,}".format(cm[i, j]),
                     horizontalalignment="center",
                     color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label\naccuracy={:0.4f};
misclass={:0.4f}'.format(accuracy, misclass))
    plt.savefig('model/keypoint_classifier/confusion_matrix.png')


model = load_model('model/keypoint_classifier/keypoint_classifier_new.h5')
pred_labels=[]
start_time = time.time()
pred_probabs = model.predict(X_test)
end_time = time.time()
pred_time = end_time - start_time
avg_pred_time = pred_time / X_test.shape[0]
print('Average prediction time: %fs' % (avg_pred_time))

for pred_probab in pred_probabs:
    pred_labels.append(list(pred_probab).index(max(pred_probab)))

cm = confusion_matrix(y_test, np.array(pred_labels))

classification_report = classification_report(y_test, np.array(pred_labels))
print('\n\nClassification Report')
print('---------------------------')
print(classification_report)
plot_confusion_matrix(cm, range(26), normalize=False)
```

7.2 FEATURE 2

```
from flask import Flask, render_template, flash, request, session
from flask import render_template, redirect, url_for, request



import smtplib



app = Flask(__name__)
app.config.from_object(__name__)
app.config['SECRET_KEY'] = '7d441f27d441f27567d441f2b6176a'

app.config['DEBUG']


@app.route("/")
def homepage():
    return render_template('index.html')
@app.route("/UserLogin")
def UserLogin():
    return render_template('UserLogin.html')


@app.route("/start", methods=['GET', 'POST'])
def start():
    error = None
    if request.method == 'POST':
        import csv
        import copy
        import cv2 as cv
        import mediapipe as mp
        from model import KeyPointClassifier
        from app_files import calc_landmark_list, draw_info_text, draw_landmarks,
get_args, pre_process_landmark
        from PIL import Image, ImageDraw, ImageFont
        import numpy as np


        args = get_args()
        cap_device = args.device
        cap_width = args.width
        cap_height = args.height

        use_static_image_mode = args.use_static_image_mode
        min_detection_confidence = args.min_detection_confidence
        min_tracking_confidence = args.min_tracking_confidence

        cap = cv.VideoCapture(cap_device)
        cap.set(cv.CAP_PROP_FRAME_WIDTH, cap_width)
        cap.set(cv.CAP_PROP_FRAME_HEIGHT, cap_height)

        mp_hands = mp.solutions.hands
        hands = mp_hands.Hands(
            static_image_mode=use_static_image_mode,
```

```
            max_num_hands=1,
            min_detection_confidence=min_detection_confidence,
            min_tracking_confidence=min_tracking_confidence,
        )

        keypoint_classifier = KeyPointClassifier()

        with open('model/keypoint_classifier/keypoint_classifier_label.csv',
encoding='utf-8-sig') as f:
            keypoint_classifier_labels = csv.reader(f)
            keypoint_classifier_labels = [
                row[0] for row in keypoint_classifier_labels
            ]

        flag = 0

        import win32com.client as wincl
        speak = wincl.Dispatch("SAPI.SpVoice")

        while True:
            key = cv.waitKey(10)
            if key == 27:  # ESC
                break

            ret, image = cap.read()
            if not ret:
                break
            image = cv.flip(image, 1)
            debug_image = copy.deepcopy(image)
            # print(debug_image.shape)
            # cv.imshow("debug_image",debug_image)
            image = cv.cvtColor(image, cv.COLOR_BGR2RGB)

            image.flags.writeable = False
            results = hands.process(image)
            image.flags.writeable = True

            if results.multi_hand_landmarks is not None:
                for hand_landmarks, handedness in
zip(results.multi_hand_landmarks, results.multi_handedness):
                    landmark_list = calc_landmark_list(debug_image,
hand_landmarks)

                    # print(hand_landmarks)
                    pre_processed_landmark_list =
pre_process_landmark(landmark_list)

                    hand_sign_id =
keypoint_classifier(pre_processed_landmark_list)

                    debug_image = draw_landmarks(debug_image, landmark_list)
                    flag += 1
                    print(flag)
                    if (flag == 100):
                        flag = 0

                        speak.Speak(keypoint_classifier_labels[hand_sign_id])

                    debug_image = draw_info_text(
```

```python
                        debug_image,
                        handedness,
                        keypoint_classifier_labels[hand_sign_id])

            cv.imshow('Hand Gesture Recognition', debug_image)

        cap.release()
        cv.destroyAllWindows()



    return render_template('UserLogin.html')









if __name__ == '__main__':
    app.run(debug=True, use_reloader=True)
```

8. TESTING

8.1 TEST CASES

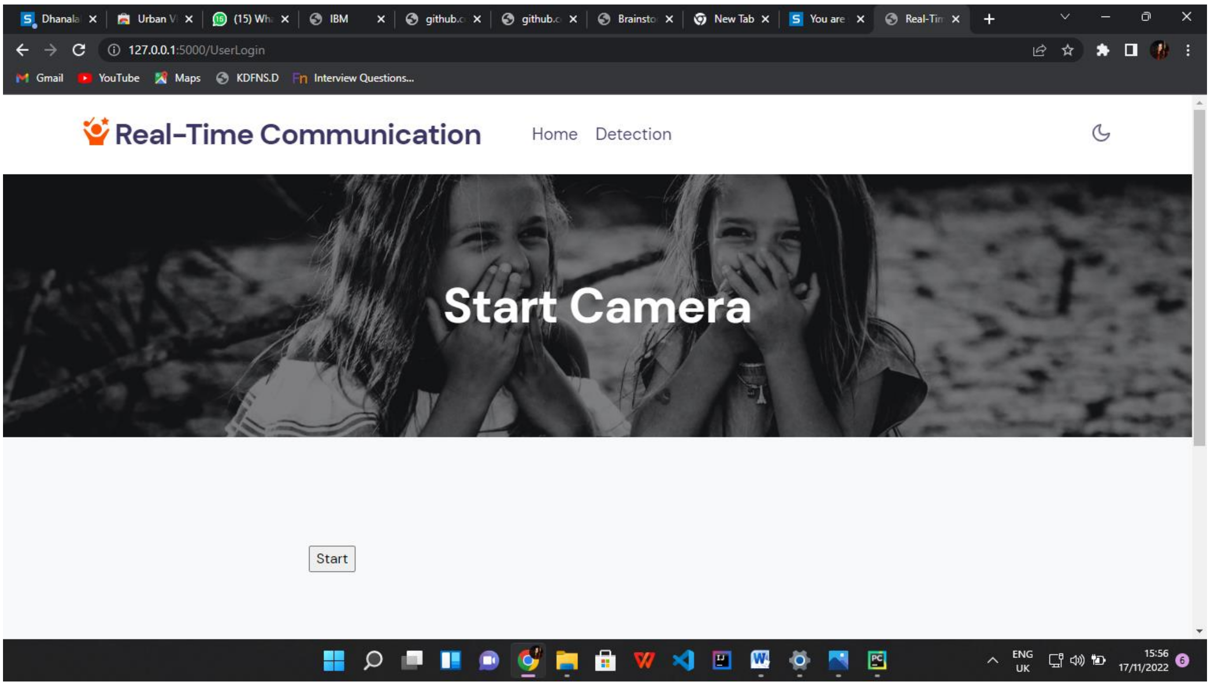
A test case has components that describe input, action and an expected response, in order to determine if a feature of an application is working correctly. A test case is a set of instructions on “HOW” to validate a particular test objective/target, which when followed will tell us if the expected behavior of the system is satisfied or not.
Characteristics of a good test case:

- Accurate: Exacts the purpose.
- Economical: No unnecessary steps or words.
- Traceable: Capable of being traced to requirements.
- Repeatable: Can be used to perform the test over and over.
- Reusable: Can be reused if necessary

S.NO	**FUNCTION**	**DESCRIPTION**	**EXPECTED OUTPUT**	**ACTUAL OUTPUT**	**STATUS**
1	Framework construction	Generate the GUI for admin and user	Individual page for admin and user	Individual page for admin and user	Success
2	Read the comments	Comments Analysis	Comments in text format	Comments in text format	Success
3	Classification	Classify the Datasets	Finger Gestures	Finger Gestures	Success
4	Rules implementation	Block the comments and friends	Block the users	Block the users	Success

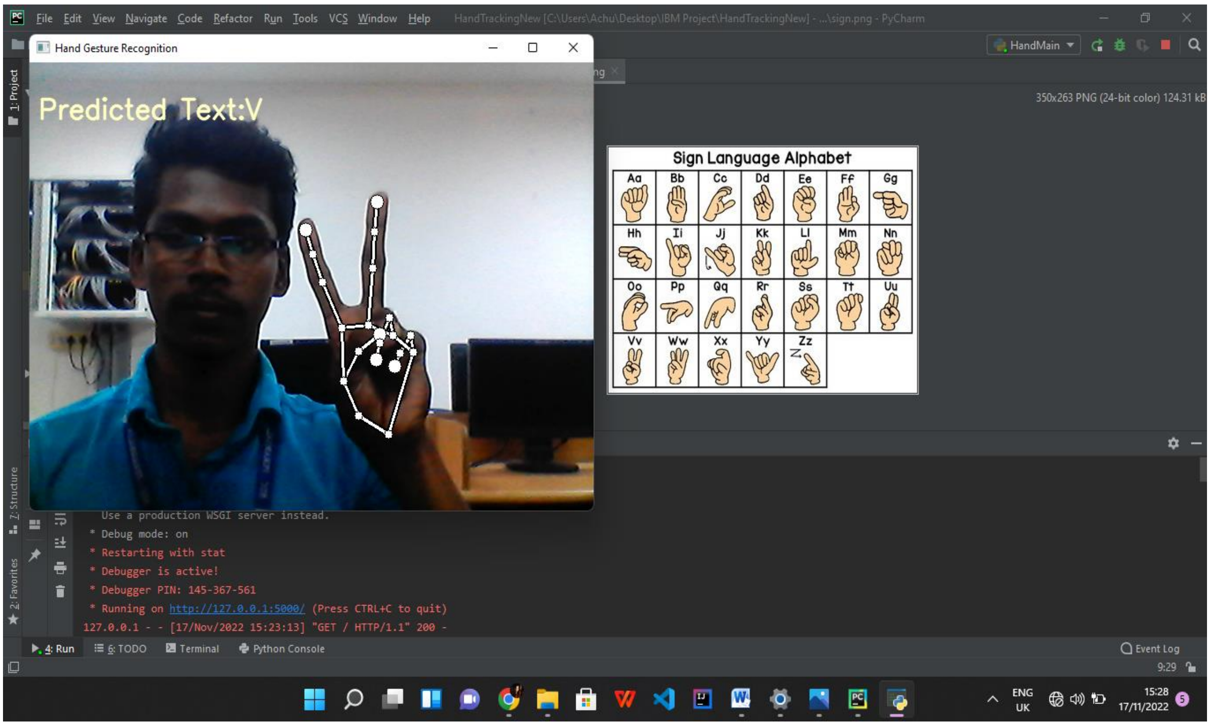
8.2 USER ACCEPTANCE TESTING

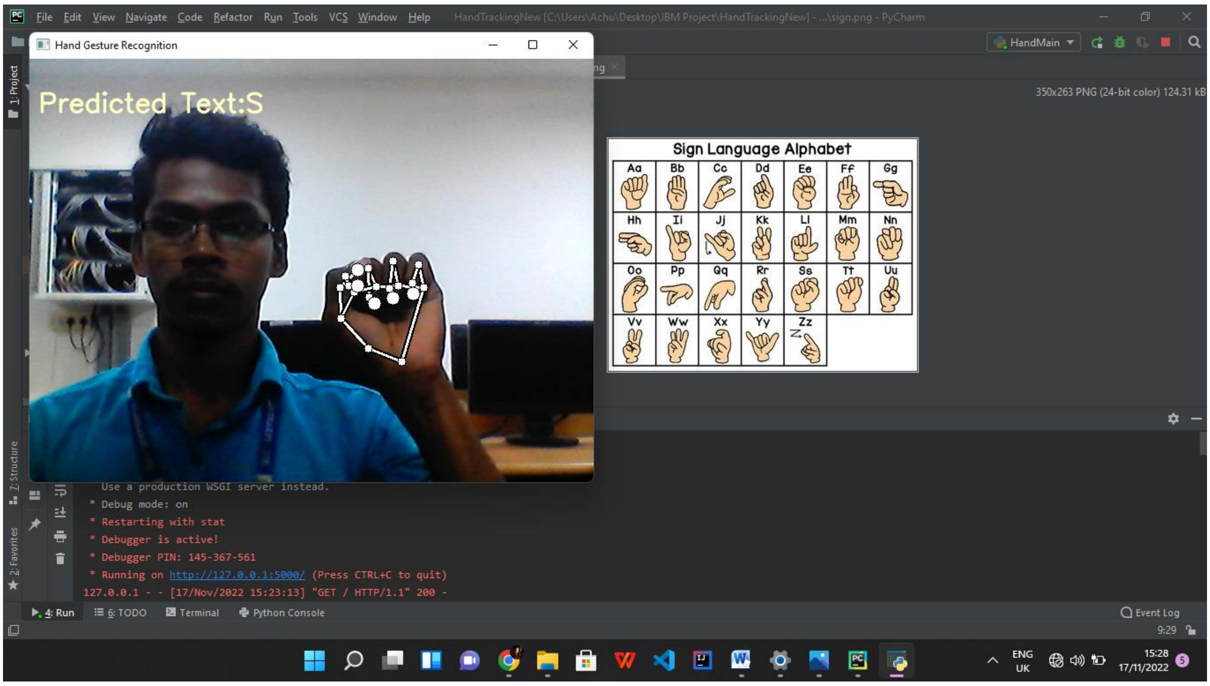
Acceptance testing can be defined in many ways, but a simple definition is the succeeds when the software functions in a manner that can be reasonable expected by the customer. After the acceptance test has been conducted, one of the two possible conditions exists. This is to fine whether the inputs are accepted by the database or other validations. For example accept only numbers in the numeric field, date format data in the date field. Also the null check for the not null fields. If any error occurs then show the error messages. The function of performance characteristics to specification and is accepted. A deviation from specification is uncovered and a deficiency list is created. User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.



9. RESULTS

9.1 PERFORMANCE METRICS





10. ADVANTAGES & DISADVANTAGES

DISADVANTAGES

- Need hardware control to detect the hands
- Hand segmentation become complex of various backgrounds
- Segmentation accuracy is less in hand tracking

ADVANTAGES

- Segmentation accuracy is high
- Easy to detect the finger postures
- Track fingers and sign recognition with less computational steps
- No need for additional hardware system

11. CONCLUSION

The ability to look, listen, talk, and respond appropriately to events is one of the most valuable gifts a human being can have. However, some unfortunate people are denied this opportunity. People get to know one another through sharing their ideas, thoughts, and experiences with others around them. There are several ways to accomplish this, the best of which is the gift of "Speech." Everyone can very persuasively transfer their thoughts and comprehend each other through speech. Our initiative intends to close the gap by including a low-cost computer into the communication chain, allowing sign language to be captured, recognised, and translated into speech for the benefit of blind individuals. An image processing technique is employed in this paper to recognise the handmade movements. This application is used to present a modern integrated planned system for hear impaired people. The camera-based zone of interest can aid in the user's data collection. Each action will be significant in its own right.

12. FUTURE SCOPE

Despite it having average accuracy, our system is still well-matched with the existing systems, given that it can perform recognition at the given accuracy with larger vocabularies and without an aid such as gloves or hand markings. In future, we can extend the framework to implement various deep learning algorithms to recognize the signs and implement in real time applications.

13. APPENDIX

SOURCE CODE

```
from flask import Flask, render_template, flash, request, session
from flask import render_template, redirect, url_for, request


import smtplib
 app = Flask(__name__)
app.config.from_object(__name__)
app.config['SECRET_KEY'] = '7d441f27d441f27567d441f2b6176a'
app.config['DEBUG']


@app.route("/")
def homepage():
    return render_template('index.html')
@app.route("/UserLogin")
def UserLogin():
    return render_template('UserLogin.html')


@app.route("/start", methods=['GET', 'POST'])
def start():
    error = None
    if request.method == 'POST':
        import csv
        import copy
        import cv2 as cv
        import mediapipe as mp
```

```
from model import KeyPointClassifier
from app_files import calc_landmark_list, draw_info_text, draw_landmarks, get_args, pre_process_landmark
from PIL import Image, ImageDraw, ImageFont
import numpy as np

args = get_args()
cap_device = args.device
cap_width = args.width
cap_height = args.height

use_static_image_mode = args.use_static_image_mode
min_detection_confidence = args.min_detection_confidence
min_tracking_confidence = args.min_tracking_confidence

cap = cv.VideoCapture(cap_device)
cap.set(cv.CAP_PROP_FRAME_WIDTH, cap_width)
cap.set(cv.CAP_PROP_FRAME_HEIGHT, cap_height)

mp_hands = mp.solutions.hands
hands = mp_hands.Hands(
    static_image_mode=use_static_image_mode,
    max_num_hands=1,
    min_detection_confidence=min_detection_confidence,
    min_tracking_confidence=min_tracking_confidence,
)
```

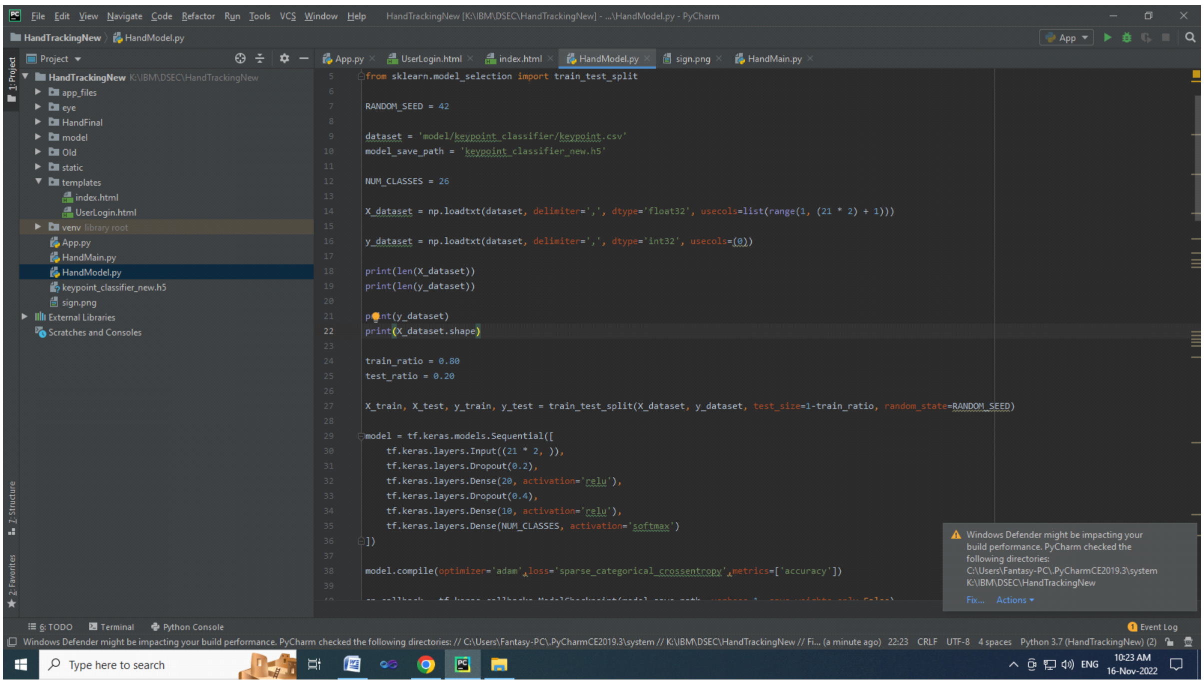
```
    keypoint_classifier = KeyPointClassifier()

    with
open('model/keypoint_classifier/keypoint_classifier_label.csv',
encoding='utf-8-sig') as f:
        keypoint_classifier_labels = csv.reader(f)
        keypoint_classifier_labels = [
            row[0] for row in keypoint_classifier_labels
        ]

    flag = 0

    import win32com.client as wincl
    speak = wincl.Dispatch("SAPI.SpVoice")

    while True:
        key = cv.waitKey(10)
        if key == 27:  # ESC
            break

        ret, image = cap.read()
        if not ret:
            break
        image = cv.flip(image, 1)
        debug_image = copy.deepcopy(image)
        # print(debug_image.shape)
        # cv.imshow("debug_image",debug_image)
```

```
        image = cv.cvtColor(image, cv.COLOR_BGR2RGB)

        image.flags.writeable = False
        results = hands.process(image)
        image.flags.writeable = True

        if results.multi_hand_landmarks is not None:
            for hand_landmarks, handedness in
zip(results.multi_hand_landmarks, results.multi_handedness):
                landmark_list = calc_landmark_list(debug_image,
hand_landmarks)

                # print(hand_landmarks)
                pre_processed_landmark_list =
pre_process_landmark(landmark_list)

                hand_sign_id =
keypoint_classifier(pre_processed_landmark_list)

                debug_image = draw_landmarks(debug_image,
landmark_list)
                flag += 1
                print(flag)
                if (flag == 100):
                    flag = 0

                    speak.Speak(keypoint_classifier_labels[hand_sign_id])

                debug_image = draw_info_text(
```

```
                debug_image,
                handedness,
                keypoint_classifier_labels[hand_sign_id])

          cv.imshow('Hand Gesture Recognition', debug_image)

      cap.release()
      cv.destroyAllWindows
        return render_template('UserLogin.html')

if __name__ == '__main__':
app.run(debug=True, use_reloader=True)

import mediapipe as mp
import cv2
import numpy as np
import uuid
import os

'''import subprocess as sp
programName = "notepad.exe"
#fileName = "sms.txt"
#sp.Popen([programName, fileName])
sp.Popen([programName])'''
mp_drawing = mp.solutions.drawing_utils
mp_hands = mp.solutions.hands
cap = cv2.VideoCapture(0)
```

```
 with mp_hands.Hands(min_detection_confidence=0.8,
min_tracking_confidence=0.5) as hands:
    while cap.isOpened():
        ret, frame = cap.read()

        # BGR 2 RGB
        image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

        # Flip on horizontal
        image = cv2.flip(image, 1)

        # Set flag
        image.flags.writeable = False

        # Detections
        results = hands.process(image)

        # Set flag to true
        image.flags.writeable = True

        # RGB 2 BGR
        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
       # print(results)

        # Rendering results
        if results.multi_hand_landmarks:
            for num, hand in enumerate(results.multi_hand_landmarks):
```

```
                mp_drawing.draw_landmarks(image, hand,
mp_hands.HAND_CONNECTIONS,
                                          mp_drawing.DrawingSpec(color=(14, 22,
76), thickness=2, circle_radius=4),
                                          mp_drawing.DrawingSpec(color=(24, 44,
250), thickness=2, circle_radius=2),
                                          )

        cv2.imshow('Hand Tracking', image)

        if cv2.waitKey(10) & 0xFF == ord('q'):
            break

cap.release()
cv2.destroyAllWindows()
```



K:\IBM\DSEC\HandTrackingNew\venv\Scripts\python.exe K:/IBM/DSEC/HandTrackingNew/HandModel.py

2022-11-16 10:23:27.058620: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlerror: cudart64_110.dll not found

2022-11-16 10:23:27.059995: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.

18256

18256

[0 0 0 ... 25 25 25]

(18256, 42)

2022-11-16 10:23:45.396541: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'nvcuda.dll'; dlerror: nvcuda.dll not found

2022-11-16 10:23:45.405263: W tensorflow/stream_executor/cuda/cuda_driver.cc:269] failed call to cuInit: UNKNOWN ERROR (303)

2022-11-16 10:23:45.425596: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:169] retrieving CUDA diagnostic information for host: DESKTOP-9BF8NUN

2022-11-16 10:23:45.426393: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176] hostname: DESKTOP-9BF8NUN

2022-11-16 10:23:45.472047: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX AVX2

To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.

Model: "sequential"

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
```

```
 dropout (Dropout)          (None, 42)                0

 dense (Dense)              (None, 20)                860

 dropout_1 (Dropout)        (None, 20)                0

 dense_1 (Dense)            (None, 10)                210

 dense_2 (Dense)            (None, 26)                286

=================================================================
Total params: 1,356
Trainable params: 1,356
Non-trainable params: 0
_________________________________________________________________
Epoch 1/500
110/115 [===========================>..] - ETA: 0s - loss: 3.1533 - accuracy: 0.0749
Epoch 00001: saving model to keypoint_classifier_new.h5
115/115 [==============================] - 2s 7ms/step - loss: 3.1485 - accuracy: 0.0762 - val_loss: 2.8981 - val_accuracy: 0.1780
Epoch 2/500
 92/115 [=======================>......] - ETA: 0s - loss: 2.8557 - accuracy: 0.1536
Epoch 00002: saving model to keypoint_classifier_new.h5
115/115 [==============================] - 0s 2ms/step - loss: 2.8336 - accuracy: 0.1576 - val_loss: 2.5658 - val_accuracy: 0.2393
```

Epoch 3/500

 87/115 [=====================>........] - ETA: 0s - loss: 2.5864 - accuracy: 0.1987

Epoch 00003: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 2.5600 - accuracy: 0.2045 - val_loss: 2.1658 - val_accuracy: 0.4291

Epoch 4/500

 87/115 [=====================>........] - ETA: 0s - loss: 2.3257 - accuracy: 0.2660

Epoch 00004: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 2.2942 - accuracy: 0.2716 - val_loss: 1.8626 - val_accuracy: 0.5112

Epoch 5/500

 88/115 [=====================>........] - ETA: 0s - loss: 2.1345 - accuracy: 0.3141

Epoch 00005: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 2.1152 - accuracy: 0.3181 - val_loss: 1.6628 - val_accuracy: 0.5271

Epoch 6/500

112/115 [============================>.] - ETA: 0s - loss: 1.9940 - accuracy: 0.3447

Epoch 00006: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.9914 - accuracy: 0.3452 - val_loss: 1.5173 - val_accuracy: 0.5999

Epoch 7/500

 96/115 [========================>.....] - ETA: 0s - loss: 1.9116 - accuracy: 0.3737

Epoch 00007: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.9015 - accuracy: 0.3768 - val_loss: 1.3988 - val_accuracy: 0.6465

Epoch 8/500

 90/115 [======================>.......] - ETA: 0s - loss: 1.8375 - accuracy: 0.3854

Epoch 00008: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.8250 - accuracy: 0.3900 - val_loss: 1.3196 - val_accuracy: 0.6525

Epoch 9/500

 93/115 [=======================>......] - ETA: 0s - loss: 1.7643 - accuracy: 0.4152

Epoch 00009: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.7564 - accuracy: 0.4185 - val_loss: 1.2383 - val_accuracy: 0.6585

Epoch 10/500

 92/115 [=======================>......] - ETA: 0s - loss: 1.7183 - accuracy: 0.4237

Epoch 00010: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.7084 - accuracy: 0.4285 - val_loss: 1.1719 - val_accuracy: 0.7065

Epoch 11/500

 90/115 [======================>.......] - ETA: 0s - loss: 1.6613 - accuracy: 0.4456

Epoch 00011: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.6624 - accuracy: 0.4440 - val_loss: 1.1348 - val_accuracy: 0.7347

Epoch 12/500

 86/115 [=====================>........] - ETA: 0s - loss: 1.6192 - accuracy: 0.4604

Epoch 00012: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.6145 - accuracy: 0.4614 - val_loss: 1.0654 - val_accuracy: 0.7226

Epoch 13/500

 92/115 [=======================>......] - ETA: 0s - loss: 1.6070 - accuracy: 0.4662

Epoch 00013: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.5992 - accuracy: 0.4672 - val_loss: 1.0411 - val_accuracy: 0.7511

Epoch 14/500

 95/115 [=======================>......] - ETA: 0s - loss: 1.5811 - accuracy: 0.4733

Epoch 00014: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.5833 - accuracy: 0.4755 - val_loss: 1.0114 - val_accuracy: 0.7757

Epoch 15/500

 99/115 [========================>.....] - ETA: 0s - loss: 1.5575 - accuracy: 0.4846

Epoch 00015: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.5573 - accuracy: 0.4850 - val_loss: 0.9931 - val_accuracy: 0.7933

Epoch 16/500

 88/115 [=====================>........] - ETA: 0s - loss: 1.5166 - accuracy: 0.4942

Epoch 00016: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.5181 - accuracy: 0.4914 - val_loss: 0.9478 - val_accuracy: 0.8108

Epoch 17/500

 86/115 [=====================>........] - ETA: 0s - loss: 1.4846 - accuracy: 0.5118

Epoch 00017: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.4872 - accuracy: 0.5097 - val_loss: 0.9215 - val_accuracy: 0.8379

Epoch 18/500

 89/115 [======================>.......] - ETA: 0s - loss: 1.4850 - accuracy: 0.5140

Epoch 00018: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.4802 - accuracy: 0.5134 - val_loss: 0.8900 - val_accuracy: 0.8401

Epoch 19/500

 98/115 [========================>.....] - ETA: 0s - loss: 1.4635 - accuracy: 0.5140

Epoch 00019: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.4632 - accuracy: 0.5136 - val_loss: 0.8831 - val_accuracy: 0.8442

Epoch 20/500

 98/115 [========================>.....] - ETA: 0s - loss: 1.4318 - accuracy: 0.5307

Epoch 00020: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.4315 - accuracy: 0.5293 - val_loss: 0.8566 - val_accuracy: 0.8488

Epoch 21/500

 87/115 [=====================>........] - ETA: 0s - loss: 1.4193 - accuracy: 0.5280

Epoch 00021: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.4121 - accuracy: 0.5301 - val_loss: 0.8331 - val_accuracy: 0.8535

Epoch 22/500

 88/115 [=====================>........] - ETA: 0s - loss: 1.3989 - accuracy: 0.5400

Epoch 00022: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.3945 - accuracy: 0.5430 - val_loss: 0.8175 - val_accuracy: 0.8442

Epoch 23/500

 91/115 [======================>.......] - ETA: 0s - loss: 1.3799 - accuracy: 0.5458

Epoch 00023: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.3813 - accuracy: 0.5473 - val_loss: 0.8004 - val_accuracy: 0.8502

Epoch 24/500

 98/115 [========================>.....] - ETA: 0s - loss: 1.3902 - accuracy: 0.5403

Epoch 00024: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.3866 - accuracy: 0.5383 - val_loss: 0.8011 - val_accuracy: 0.8590

Epoch 25/500

 85/115 [=====================>........] - ETA: 0s - loss: 1.3595 - accuracy: 0.5489

Epoch 00025: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.3556 - accuracy: 0.5495 - val_loss: 0.7652 - val_accuracy: 0.8647

Epoch 26/500

 85/115 [=====================>........] - ETA: 0s - loss: 1.3708 - accuracy: 0.5460

Epoch 00026: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.3578 - accuracy: 0.5529 - val_loss: 0.7676 - val_accuracy: 0.8656

Epoch 27/500

 97/115 [========================>.....] - ETA: 0s - loss: 1.3515 - accuracy: 0.5498

Epoch 00027: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.3498 - accuracy: 0.5503 - val_loss: 0.7649 - val_accuracy: 0.8683

Epoch 28/500

 95/115 [=======================>......] - ETA: 0s - loss: 1.3376 - accuracy: 0.5545

Epoch 00028: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.3368 - accuracy: 0.5542 - val_loss: 0.7624 - val_accuracy: 0.8598

Epoch 29/500

 93/115 [=======================>......] - ETA: 0s - loss: 1.3110 - accuracy: 0.5687

Epoch 00029: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.3222 - accuracy: 0.5650 - val_loss: 0.7400 - val_accuracy: 0.8850

Epoch 30/500

 87/115 [=====================>........] - ETA: 0s - loss: 1.3270 - accuracy: 0.5628

Epoch 00030: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.3157 - accuracy: 0.5653 - val_loss: 0.7223 - val_accuracy: 0.8798

Epoch 31/500

 86/115 [=====================>........] - ETA: 0s - loss: 1.2841 - accuracy: 0.5714

Epoch 00031: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.2907 - accuracy: 0.5679 - val_loss: 0.7276 - val_accuracy: 0.8639

Epoch 32/500

 91/115 [======================>.......] - ETA: 0s - loss: 1.3095 - accuracy: 0.5664

Epoch 00032: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.3095 - accuracy: 0.5679 - val_loss: 0.7148 - val_accuracy: 0.8727

Epoch 33/500

 84/115 [====================>.........] - ETA: 0s - loss: 1.2773 - accuracy: 0.5738

Epoch 00033: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.2880 - accuracy: 0.5731 - val_loss: 0.7160 - val_accuracy: 0.8946

Epoch 34/500

 85/115 [=====================>........] - ETA: 0s - loss: 1.2770 - accuracy: 0.5785

Epoch 00034: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.2772 - accuracy: 0.5768 - val_loss: 0.7071 - val_accuracy: 0.8864

Epoch 35/500

 86/115 [=====================>........] - ETA: 0s - loss: 1.2649 - accuracy: 0.5804

Epoch 00035: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.2640 - accuracy: 0.5800 - val_loss: 0.6843 - val_accuracy: 0.8951

Epoch 36/500

 82/115 [====================>.........] - ETA: 0s - loss: 1.2555 - accuracy: 0.5781

Epoch 00036: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.2605 - accuracy: 0.5782 - val_loss: 0.6959 - val_accuracy: 0.8866

Epoch 37/500

 90/115 [======================>.......] - ETA: 0s - loss: 1.2769 - accuracy: 0.5791

Epoch 00037: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.2700 - accuracy: 0.5788 - val_loss: 0.6757 - val_accuracy: 0.8836

Epoch 38/500

 84/115 [====================>.........] - ETA: 0s - loss: 1.2672 - accuracy: 0.5785

Epoch 00038: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.2661 - accuracy: 0.5798 - val_loss: 0.6870 - val_accuracy: 0.8861

Epoch 39/500

 83/115 [====================>.........] - ETA: 0s - loss: 1.2696 - accuracy: 0.5865

Epoch 00039: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.2666 - accuracy: 0.5863 - val_loss: 0.6776 - val_accuracy: 0.8886

Epoch 40/500

 83/115 [====================>.........] - ETA: 0s - loss: 1.2613 - accuracy: 0.5819

Epoch 00040: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.2475 - accuracy: 0.5865 - val_loss: 0.6579 - val_accuracy: 0.8877

Epoch 41/500

 87/115 [=====================>........] - ETA: 0s - loss: 1.2465 - accuracy: 0.5890

Epoch 00041: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.2407 - accuracy: 0.5900 - val_loss: 0.6638 - val_accuracy: 0.8907

Epoch 42/500

 85/115 [=====================>........] - ETA: 0s - loss: 1.2380 - accuracy: 0.5897

Epoch 00042: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.2345 - accuracy: 0.5923 - val_loss: 0.6583 - val_accuracy: 0.8976

Epoch 43/500

 86/115 [=====================>........] - ETA: 0s - loss: 1.2393 - accuracy: 0.5898

Epoch 00043: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.2389 - accuracy: 0.5894 - val_loss: 0.6443 - val_accuracy: 0.9017

Epoch 44/500

 83/115 [====================>.........] - ETA: 0s - loss: 1.2468 - accuracy: 0.5895

Epoch 00044: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.2284 - accuracy: 0.5957 - val_loss: 0.6446 - val_accuracy: 0.8814

Epoch 45/500

 86/115 [=====================>........] - ETA: 0s - loss: 1.2395 - accuracy: 0.5898

Epoch 00045: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.2342 - accuracy: 0.5920 - val_loss: 0.6420 - val_accuracy: 0.8894

Epoch 46/500

 86/115 [=====================>........] - ETA: 0s - loss: 1.2108 - accuracy: 0.6007

Epoch 00046: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.2136 - accuracy: 0.5967 - val_loss: 0.6500 - val_accuracy: 0.8916

Epoch 47/500

 88/115 [=====================>........] - ETA: 0s - loss: 1.2140 - accuracy: 0.5977

Epoch 00047: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.2161 - accuracy: 0.5987 - val_loss: 0.6331 - val_accuracy: 0.8990

Epoch 48/500

 89/115 [======================>.......] - ETA: 0s - loss: 1.2116 - accuracy: 0.5997

Epoch 00048: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.2125 - accuracy: 0.5965 - val_loss: 0.6354 - val_accuracy: 0.8913

Epoch 49/500

 87/115 [=====================>........] - ETA: 0s - loss: 1.2093 - accuracy: 0.5950

Epoch 00049: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.2094 - accuracy: 0.5958 - val_loss: 0.6305 - val_accuracy: 0.8896

Epoch 50/500

 88/115 [=====================>........] - ETA: 0s - loss: 1.2005 - accuracy: 0.6036

Epoch 00050: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.2069 - accuracy: 0.6020 - val_loss: 0.6356 - val_accuracy: 0.8781

Epoch 51/500

110/115 [===========================>..] - ETA: 0s - loss: 1.1913 - accuracy: 0.6006

Epoch 00051: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1921 - accuracy: 0.6004 - val_loss: 0.6250 - val_accuracy: 0.8905

Epoch 52/500

 76/115 [==================>...........] - ETA: 0s - loss: 1.1923 - accuracy: 0.6040

Epoch 00052: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1970 - accuracy: 0.6018 - val_loss: 0.6193 - val_accuracy: 0.8973

Epoch 53/500

 77/115 [===================>..........] - ETA: 0s - loss: 1.1859 - accuracy: 0.6006

Epoch 00053: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1867 - accuracy: 0.6040 - val_loss: 0.6195 - val_accuracy: 0.8905

Epoch 54/500

 92/115 [=======================>......] - ETA: 0s - loss: 1.2054 - accuracy: 0.5999

Epoch 00054: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1982 - accuracy: 0.6034 - val_loss: 0.6126 - val_accuracy: 0.9022

Epoch 55/500

 82/115 [====================>.........] - ETA: 0s - loss: 1.1838 - accuracy: 0.6052

Epoch 00055: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1891 - accuracy: 0.6052 - val_loss: 0.6119 - val_accuracy: 0.9025

Epoch 56/500

 85/115 [=====================>........] - ETA: 0s - loss: 1.1912 - accuracy: 0.6090

Epoch 00056: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1923 - accuracy: 0.6111 - val_loss: 0.6000 - val_accuracy: 0.8932

Epoch 57/500

 78/115 [===================>..........] - ETA: 0s - loss: 1.1785 - accuracy: 0.6075

Epoch 00057: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1847 - accuracy: 0.6083 - val_loss: 0.6094 - val_accuracy: 0.8847

Epoch 58/500

 88/115 [=====================>........] - ETA: 0s - loss: 1.1870 - accuracy: 0.6047

Epoch 00058: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1793 - accuracy: 0.6088 - val_loss: 0.5927 - val_accuracy: 0.9055

Epoch 59/500

 82/115 [====================>.........] - ETA: 0s - loss: 1.1707 - accuracy: 0.6108

Epoch 00059: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1726 - accuracy: 0.6128 - val_loss: 0.5974 - val_accuracy: 0.8883

Epoch 60/500

 84/115 [====================>.........] - ETA: 0s - loss: 1.1768 - accuracy: 0.6074

Epoch 00060: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1812 - accuracy: 0.6064 - val_loss: 0.5945 - val_accuracy: 0.8959

Epoch 61/500

 87/115 [=====================>........] - ETA: 0s - loss: 1.1686 - accuracy: 0.6187

Epoch 00061: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1763 - accuracy: 0.6180 - val_loss: 0.5959 - val_accuracy: 0.8954

Epoch 62/500

112/115 [============================>.] - ETA: 0s - loss: 1.1684 - accuracy: 0.6126

Epoch 00062: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1689 - accuracy: 0.6122 - val_loss: 0.6045 - val_accuracy: 0.8886

Epoch 63/500

 82/115 [====================>.........] - ETA: 0s - loss: 1.1658 - accuracy: 0.6137

Epoch 00063: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1605 - accuracy: 0.6163 - val_loss: 0.5836 - val_accuracy: 0.8938

Epoch 64/500

 86/115 [=====================>........] - ETA: 0s - loss: 1.1694 - accuracy: 0.6141

Epoch 00064: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1614 - accuracy: 0.6173 - val_loss: 0.5813 - val_accuracy: 0.8872

Epoch 65/500

 86/115 [=====================>........] - ETA: 0s - loss: 1.1595 - accuracy: 0.6175

Epoch 00065: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1624 - accuracy: 0.6152 - val_loss: 0.5924 - val_accuracy: 0.8927

Epoch 66/500

 85/115 [=====================>........] - ETA: 0s - loss: 1.1624 - accuracy: 0.6192

Epoch 00066: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1645 - accuracy: 0.6194 - val_loss: 0.5849 - val_accuracy: 0.8844

Epoch 67/500

 84/115 [====================>.........] - ETA: 0s - loss: 1.1378 - accuracy: 0.6226

Epoch 00067: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1431 - accuracy: 0.6238 - val_loss: 0.5791 - val_accuracy: 0.8823

Epoch 68/500

 85/115 [=====================>........] - ETA: 0s - loss: 1.1529 - accuracy: 0.6123

Epoch 00068: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1443 - accuracy: 0.6172 - val_loss: 0.5880 - val_accuracy: 0.8875

Epoch 69/500

 89/115 [======================>.......] - ETA: 0s - loss: 1.1339 - accuracy: 0.6295

Epoch 00069: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1328 - accuracy: 0.6270 - val_loss: 0.5684 - val_accuracy: 0.8951

Epoch 70/500

 88/115 [=====================>........] - ETA: 0s - loss: 1.1530 - accuracy: 0.6196

Epoch 00070: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1536 - accuracy: 0.6200 - val_loss: 0.5748 - val_accuracy: 0.8823

Epoch 71/500

 82/115 [====================>.........] - ETA: 0s - loss: 1.1461 - accuracy: 0.6275

Epoch 00071: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1411 - accuracy: 0.6290 - val_loss: 0.5631 - val_accuracy: 0.8916

Epoch 72/500

 84/115 [====================>.........] - ETA: 0s - loss: 1.1460 - accuracy: 0.6242

Epoch 00072: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1378 - accuracy: 0.6267 - val_loss: 0.5608 - val_accuracy: 0.8957

Epoch 73/500

 85/115 [=====================>........] - ETA: 0s - loss: 1.1282 - accuracy: 0.6301

Epoch 00073: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1246 - accuracy: 0.6284 - val_loss: 0.5640 - val_accuracy: 0.8861

Epoch 74/500

 86/115 [=====================>........] - ETA: 0s - loss: 1.1320 - accuracy: 0.6317

Epoch 00074: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1265 - accuracy: 0.6310 - val_loss: 0.5636 - val_accuracy: 0.8847

Epoch 75/500

 86/115 [=====================>........] - ETA: 0s - loss: 1.1317 - accuracy: 0.6295

Epoch 00075: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1357 - accuracy: 0.6277 - val_loss: 0.5488 - val_accuracy: 0.8992

Epoch 76/500

 81/115 [====================>.........] - ETA: 0s - loss: 1.1255 - accuracy: 0.6338

Epoch 00076: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1209 - accuracy: 0.6354 - val_loss: 0.5549 - val_accuracy: 0.8984

Epoch 77/500

 84/115 [====================>.........] - ETA: 0s - loss: 1.1272 - accuracy: 0.6261

Epoch 00077: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1364 - accuracy: 0.6211 - val_loss: 0.5514 - val_accuracy: 0.8968

Epoch 78/500

 83/115 [====================>.........] - ETA: 0s - loss: 1.1256 - accuracy: 0.6229

Epoch 00078: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1218 - accuracy: 0.6243 - val_loss: 0.5397 - val_accuracy: 0.8872

Epoch 79/500

 86/115 [=====================>........] - ETA: 0s - loss: 1.1111 - accuracy: 0.6320

Epoch 00079: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1108 - accuracy: 0.6339 - val_loss: 0.5436 - val_accuracy: 0.8938

Epoch 80/500

 87/115 [=====================>........] - ETA: 0s - loss: 1.1038 - accuracy: 0.6325

Epoch 00080: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1056 - accuracy: 0.6310 - val_loss: 0.5308 - val_accuracy: 0.8995

Epoch 81/500

 88/115 [=====================>........] - ETA: 0s - loss: 1.1211 - accuracy: 0.6307

Epoch 00081: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1193 - accuracy: 0.6317 - val_loss: 0.5384 - val_accuracy: 0.8861

Epoch 82/500

 87/115 [=====================>........] - ETA: 0s - loss: 1.1175 - accuracy: 0.6309

Epoch 00082: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1157 - accuracy: 0.6311 - val_loss: 0.5318 - val_accuracy: 0.9014

Epoch 83/500

 89/115 [======================>.......] - ETA: 0s - loss: 1.1114 - accuracy: 0.6326

Epoch 00083: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1131 - accuracy: 0.6312 - val_loss: 0.5483 - val_accuracy: 0.8883

Epoch 84/500

 89/115 [======================>.......] - ETA: 0s - loss: 1.1074 - accuracy: 0.6311

Epoch 00084: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1039 - accuracy: 0.6326 - val_loss: 0.5479 - val_accuracy: 0.8921

Epoch 85/500

 76/115 [==================>...........] - ETA: 0s - loss: 1.1027 - accuracy: 0.6359

Epoch 00085: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1086 - accuracy: 0.6360 - val_loss: 0.5398 - val_accuracy: 0.8962

Epoch 86/500

 86/115 [=====================>........] - ETA: 0s - loss: 1.1136 - accuracy: 0.6366

Epoch 00086: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1118 - accuracy: 0.6356 - val_loss: 0.5381 - val_accuracy: 0.9001

Epoch 87/500

 85/115 [=====================>........] - ETA: 0s - loss: 1.1113 - accuracy: 0.6316

Epoch 00087: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1143 - accuracy: 0.6326 - val_loss: 0.5444 - val_accuracy: 0.8855

Epoch 88/500

 84/115 [====================>.........] - ETA: 0s - loss: 1.1025 - accuracy: 0.6334

Epoch 00088: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0979 - accuracy: 0.6354 - val_loss: 0.5263 - val_accuracy: 0.8886

Epoch 89/500

 83/115 [====================>.........] - ETA: 0s - loss: 1.1007 - accuracy: 0.6365

Epoch 00089: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1047 - accuracy: 0.6340 - val_loss: 0.5320 - val_accuracy: 0.9033

Epoch 90/500

 84/115 [====================>.........] - ETA: 0s - loss: 1.0950 - accuracy: 0.6390

Epoch 00090: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1010 - accuracy: 0.6378 - val_loss: 0.5295 - val_accuracy: 0.9003

Epoch 91/500

 85/115 [=====================>........] - ETA: 0s - loss: 1.1161 - accuracy: 0.6404

Epoch 00091: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1099 - accuracy: 0.6424 - val_loss: 0.5207 - val_accuracy: 0.8976

Epoch 92/500

 87/115 [=====================>........] - ETA: 0s - loss: 1.0962 - accuracy: 0.6378

Epoch 00092: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0881 - accuracy: 0.6413 - val_loss: 0.5105 - val_accuracy: 0.9066

Epoch 93/500

86/115 [=====================>........] - ETA: 0s - loss: 1.0969 - accuracy: 0.6424

Epoch 00093: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1004 - accuracy: 0.6419 - val_loss: 0.5191 - val_accuracy: 0.9014

Epoch 94/500

87/115 [=====================>........] - ETA: 0s - loss: 1.1075 - accuracy: 0.6330

Epoch 00094: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1051 - accuracy: 0.6359 - val_loss: 0.5264 - val_accuracy: 0.8957

Epoch 95/500

97/115 [========================>.....] - ETA: 0s - loss: 1.0957 - accuracy: 0.6401

Epoch 00095: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 4ms/step - loss: 1.0942 - accuracy: 0.6407 - val_loss: 0.5197 - val_accuracy: 0.9020

Epoch 96/500

97/115 [========================>.....] - ETA: 0s - loss: 1.0883 - accuracy: 0.6367

Epoch 00096: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0891 - accuracy: 0.6346 - val_loss: 0.5178 - val_accuracy: 0.8853

Epoch 97/500

88/115 [=====================>........] - ETA: 0s - loss: 1.1078 - accuracy: 0.6326

Epoch 00097: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1017 - accuracy: 0.6326 - val_loss: 0.5128 - val_accuracy: 0.8940

Epoch 98/500

 88/115 [=====================>........] - ETA: 0s - loss: 1.1070 - accuracy: 0.6401

Epoch 00098: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0983 - accuracy: 0.6396 - val_loss: 0.5221 - val_accuracy: 0.8970

Epoch 99/500

 90/115 [======================>.......] - ETA: 0s - loss: 1.0795 - accuracy: 0.6468

Epoch 00099: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0821 - accuracy: 0.6443 - val_loss: 0.5118 - val_accuracy: 0.8976

Epoch 100/500

 86/115 [=====================>........] - ETA: 0s - loss: 1.0948 - accuracy: 0.6351

Epoch 00100: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0998 - accuracy: 0.6352 - val_loss: 0.5243 - val_accuracy: 0.8847

Epoch 101/500

 87/115 [=====================>........] - ETA: 0s - loss: 1.0989 - accuracy: 0.6377

Epoch 00101: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.1043 - accuracy: 0.6361 - val_loss: 0.5123 - val_accuracy: 0.9020

Epoch 102/500

 87/115 [=====================>........] - ETA: 0s - loss: 1.0889 - accuracy: 0.6439

Epoch 00102: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0914 - accuracy: 0.6441 - val_loss: 0.5127 - val_accuracy: 0.9055

Epoch 103/500

 88/115 [=====================>........] - ETA: 0s - loss: 1.0995 - accuracy: 0.6405

Epoch 00103: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0884 - accuracy: 0.6426 - val_loss: 0.5163 - val_accuracy: 0.8820

Epoch 104/500

 88/115 [=====================>........] - ETA: 0s - loss: 1.0872 - accuracy: 0.6377

Epoch 00104: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0768 - accuracy: 0.6417 - val_loss: 0.5036 - val_accuracy: 0.8981

Epoch 105/500

 87/115 [=====================>........] - ETA: 0s - loss: 1.0916 - accuracy: 0.6427

Epoch 00105: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0856 - accuracy: 0.6439 - val_loss: 0.5156 - val_accuracy: 0.9001

Epoch 106/500

 86/115 [=====================>........] - ETA: 0s - loss: 1.0824 - accuracy: 0.6441

Epoch 00106: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0818 - accuracy: 0.6446 - val_loss: 0.4993 - val_accuracy: 0.9064

Epoch 107/500

 89/115 [======================>.......] - ETA: 0s - loss: 1.0978 - accuracy: 0.6333

Epoch 00107: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0891 - accuracy: 0.6350 - val_loss: 0.5161 - val_accuracy: 0.8913

Epoch 108/500

 89/115 [======================>.......] - ETA: 0s - loss: 1.0779 - accuracy: 0.6409

Epoch 00108: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0764 - accuracy: 0.6437 - val_loss: 0.5144 - val_accuracy: 0.8932

Epoch 109/500

 89/115 [======================>.......] - ETA: 0s - loss: 1.0836 - accuracy: 0.6473

Epoch 00109: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0833 - accuracy: 0.6443 - val_loss: 0.5123 - val_accuracy: 0.8954

Epoch 110/500

 84/115 [====================>.........] - ETA: 0s - loss: 1.0841 - accuracy: 0.6399

Epoch 00110: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0875 - accuracy: 0.6399 - val_loss: 0.5085 - val_accuracy: 0.9058

Epoch 111/500

 89/115 [======================>.......] - ETA: 0s - loss: 1.0623 - accuracy: 0.6391

Epoch 00111: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0625 - accuracy: 0.6419 - val_loss: 0.4986 - val_accuracy: 0.9055

Epoch 112/500

 84/115 [====================>.........] - ETA: 0s - loss: 1.0824 - accuracy: 0.6395

Epoch 00112: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0797 - accuracy: 0.6414 - val_loss: 0.5085 - val_accuracy: 0.9091

Epoch 113/500

 80/115 [===================>..........] - ETA: 0s - loss: 1.0779 - accuracy: 0.6373

Epoch 00113: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0815 - accuracy: 0.6354 - val_loss: 0.5082 - val_accuracy: 0.9033

Epoch 114/500

 87/115 [=====================>........] - ETA: 0s - loss: 1.0677 - accuracy: 0.6493

Epoch 00114: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0693 - accuracy: 0.6491 - val_loss: 0.4973 - val_accuracy: 0.9151

Epoch 115/500

 83/115 [====================>.........] - ETA: 0s - loss: 1.0683 - accuracy: 0.6427

Epoch 00115: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0646 - accuracy: 0.6456 - val_loss: 0.4903 - val_accuracy: 0.8932

Epoch 116/500

 85/115 [=====================>........] - ETA: 0s - loss: 1.0639 - accuracy: 0.6464

Epoch 00116: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0680 - accuracy: 0.6446 - val_loss: 0.5056 - val_accuracy: 0.9014

Epoch 117/500

 84/115 [====================>.........] - ETA: 0s - loss: 1.0538 - accuracy: 0.6539

Epoch 00117: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0569 - accuracy: 0.6522 - val_loss: 0.5016 - val_accuracy: 0.9099

Epoch 118/500

 89/115 [======================>.......] - ETA: 0s - loss: 1.0748 - accuracy: 0.6482

Epoch 00118: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0767 - accuracy: 0.6469 - val_loss: 0.4993 - val_accuracy: 0.9091

Epoch 119/500

 89/115 [======================>.......] - ETA: 0s - loss: 1.0664 - accuracy: 0.6488

Epoch 00119: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0682 - accuracy: 0.6461 - val_loss: 0.4982 - val_accuracy: 0.9072

Epoch 120/500

 91/115 [======================>.......] - ETA: 0s - loss: 1.0699 - accuracy: 0.6453

Epoch 00120: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0727 - accuracy: 0.6435 - val_loss: 0.4987 - val_accuracy: 0.8842

Epoch 121/500

 88/115 [=====================>........] - ETA: 0s - loss: 1.0503 - accuracy: 0.6524

Epoch 00121: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0491 - accuracy: 0.6521 - val_loss: 0.5007 - val_accuracy: 0.9042

Epoch 122/500

 83/115 [====================>.........] - ETA: 0s - loss: 1.0695 - accuracy: 0.6478

Epoch 00122: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0773 - accuracy: 0.6452 - val_loss: 0.4941 - val_accuracy: 0.8987

Epoch 123/500

 86/115 [=====================>........] - ETA: 0s - loss: 1.0646 - accuracy: 0.6472

Epoch 00123: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0651 - accuracy: 0.6478 - val_loss: 0.4970 - val_accuracy: 0.9025

Epoch 124/500

 86/115 [=====================>........] - ETA: 0s - loss: 1.0717 - accuracy: 0.6430

Epoch 00124: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0640 - accuracy: 0.6463 - val_loss: 0.4948 - val_accuracy: 0.9022

Epoch 125/500

 85/115 [=====================>........] - ETA: 0s - loss: 1.0738 - accuracy: 0.6445

Epoch 00125: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0659 - accuracy: 0.6461 - val_loss: 0.4795 - val_accuracy: 0.9058

Epoch 126/500

 79/115 [===================>..........] - ETA: 0s - loss: 1.0534 - accuracy: 0.6547

Epoch 00126: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0561 - accuracy: 0.6575 - val_loss: 0.4909 - val_accuracy: 0.8984

Epoch 127/500

 85/115 [=====================>........] - ETA: 0s - loss: 1.0647 - accuracy: 0.6482

Epoch 00127: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0621 - accuracy: 0.6483 - val_loss: 0.5136 - val_accuracy: 0.9061

Epoch 128/500

 81/115 [====================>.........] - ETA: 0s - loss: 1.0596 - accuracy: 0.6426

Epoch 00128: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0534 - accuracy: 0.6477 - val_loss: 0.4854 - val_accuracy: 0.8902

Epoch 129/500

 88/115 [=====================>........] - ETA: 0s - loss: 1.0649 - accuracy: 0.6465

Epoch 00129: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0596 - accuracy: 0.6482 - val_loss: 0.4965 - val_accuracy: 0.9022

Epoch 130/500

 88/115 [=====================>........] - ETA: 0s - loss: 1.0637 - accuracy: 0.6499

Epoch 00130: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0591 - accuracy: 0.6501 - val_loss: 0.4972 - val_accuracy: 0.8929

Epoch 131/500

 89/115 [======================>.......] - ETA: 0s - loss: 1.0543 - accuracy: 0.6499

Epoch 00131: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0499 - accuracy: 0.6515 - val_loss: 0.4995 - val_accuracy: 0.9077

Epoch 132/500

 88/115 [=====================>........] - ETA: 0s - loss: 1.0667 - accuracy: 0.6483

Epoch 00132: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0664 - accuracy: 0.6498 - val_loss: 0.4786 - val_accuracy: 0.9129

Epoch 133/500

 88/115 [=====================>........] - ETA: 0s - loss: 1.0645 - accuracy: 0.6463

Epoch 00133: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0595 - accuracy: 0.6478 - val_loss: 0.4903 - val_accuracy: 0.9132

Epoch 134/500

 88/115 [=====================>........] - ETA: 0s - loss: 1.0675 - accuracy: 0.6480

Epoch 00134: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0671 - accuracy: 0.6480 - val_loss: 0.4963 - val_accuracy: 0.9047

Epoch 135/500

 86/115 [=====================>........] - ETA: 0s - loss: 1.0608 - accuracy: 0.6545

Epoch 00135: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0588 - accuracy: 0.6540 - val_loss: 0.4954 - val_accuracy: 0.9031

Epoch 136/500

 82/115 [====================>.........] - ETA: 0s - loss: 1.0325 - accuracy: 0.6581

Epoch 00136: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0425 - accuracy: 0.6549 - val_loss: 0.4899 - val_accuracy: 0.9099

Epoch 137/500

 87/115 [=====================>........] - ETA: 0s - loss: 1.0534 - accuracy: 0.6483

Epoch 00137: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0497 - accuracy: 0.6502 - val_loss: 0.4888 - val_accuracy: 0.9110

Epoch 138/500

 85/115 [=====================>........] - ETA: 0s - loss: 1.0592 - accuracy: 0.6473

Epoch 00138: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0603 - accuracy: 0.6498 - val_loss: 0.4996 - val_accuracy: 0.9096

Epoch 139/500

 87/115 [=====================>........] - ETA: 0s - loss: 1.0554 - accuracy: 0.6501

Epoch 00139: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0613 - accuracy: 0.6491 - val_loss: 0.4880 - val_accuracy: 0.9083

Epoch 140/500

 87/115 [=====================>........] - ETA: 0s - loss: 1.0595 - accuracy: 0.6466

Epoch 00140: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0643 - accuracy: 0.6441 - val_loss: 0.4880 - val_accuracy: 0.8918

Epoch 141/500

 86/115 [=====================>........] - ETA: 0s - loss: 1.0611 - accuracy: 0.6488

Epoch 00141: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0591 - accuracy: 0.6493 - val_loss: 0.4965 - val_accuracy: 0.9080

Epoch 142/500

 76/115 [==================>...........] - ETA: 0s - loss: 1.0581 - accuracy: 0.6532

Epoch 00142: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0529 - accuracy: 0.6540 - val_loss: 0.4757 - val_accuracy: 0.9118

Epoch 143/500

85/115 [=====================>........] - ETA: 0s - loss: 1.0605 - accuracy: 0.6540

Epoch 00143: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0619 - accuracy: 0.6506 - val_loss: 0.4675 - val_accuracy: 0.9124

Epoch 144/500

85/115 [=====================>........] - ETA: 0s - loss: 1.0626 - accuracy: 0.6514

Epoch 00144: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0549 - accuracy: 0.6535 - val_loss: 0.4843 - val_accuracy: 0.9025

Epoch 145/500

87/115 [=====================>........] - ETA: 0s - loss: 1.0556 - accuracy: 0.6511

Epoch 00145: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0568 - accuracy: 0.6527 - val_loss: 0.5036 - val_accuracy: 0.9140

Epoch 146/500

86/115 [=====================>........] - ETA: 0s - loss: 1.0619 - accuracy: 0.6510

Epoch 00146: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0570 - accuracy: 0.6515 - val_loss: 0.4790 - val_accuracy: 0.9066

Epoch 147/500

83/115 [====================>.........] - ETA: 0s - loss: 1.0485 - accuracy: 0.6524

Epoch 00147: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0411 - accuracy: 0.6535 - val_loss: 0.4871 - val_accuracy: 0.8954

Epoch 148/500

 85/115 [=====================>........] - ETA: 0s - loss: 1.0457 - accuracy: 0.6551

Epoch 00148: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0494 - accuracy: 0.6520 - val_loss: 0.4791 - val_accuracy: 0.9102

Epoch 149/500

 87/115 [=====================>........] - ETA: 0s - loss: 1.0365 - accuracy: 0.6530

Epoch 00149: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0344 - accuracy: 0.6558 - val_loss: 0.4750 - val_accuracy: 0.9127

Epoch 150/500

 84/115 [====================>.........] - ETA: 0s - loss: 1.0519 - accuracy: 0.6497

Epoch 00150: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0462 - accuracy: 0.6551 - val_loss: 0.4725 - val_accuracy: 0.9009

Epoch 151/500

 78/115 [===================>..........] - ETA: 0s - loss: 1.0504 - accuracy: 0.6511

Epoch 00151: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0560 - accuracy: 0.6501 - val_loss: 0.4833 - val_accuracy: 0.9105

Epoch 152/500

 85/115 [=====================>........] - ETA: 0s - loss: 1.0304 - accuracy: 0.6551

Epoch 00152: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0294 - accuracy: 0.6577 - val_loss: 0.4714 - val_accuracy: 0.9157

Epoch 153/500

 81/115 [====================>.........] - ETA: 0s - loss: 1.0425 - accuracy: 0.6573

Epoch 00153: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0395 - accuracy: 0.6603 - val_loss: 0.4699 - val_accuracy: 0.9072

Epoch 154/500

 86/115 [=====================>........] - ETA: 0s - loss: 1.0287 - accuracy: 0.6615

Epoch 00154: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0294 - accuracy: 0.6613 - val_loss: 0.4755 - val_accuracy: 0.9110

Epoch 155/500

 85/115 [=====================>........] - ETA: 0s - loss: 1.0528 - accuracy: 0.6520

Epoch 00155: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0510 - accuracy: 0.6510 - val_loss: 0.4895 - val_accuracy: 0.9096

Epoch 156/500

 86/115 [=====================>........] - ETA: 0s - loss: 1.0446 - accuracy: 0.6575

Epoch 00156: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0415 - accuracy: 0.6571 - val_loss: 0.4716 - val_accuracy: 0.9091

Epoch 157/500

 83/115 [====================>.........] - ETA: 0s - loss: 1.0435 - accuracy: 0.6542

Epoch 00157: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0504 - accuracy: 0.6519 - val_loss: 0.4848 - val_accuracy: 0.9127

Epoch 158/500

 89/115 [======================>.......] - ETA: 0s - loss: 1.0296 - accuracy: 0.6559

Epoch 00158: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0343 - accuracy: 0.6558 - val_loss: 0.4737 - val_accuracy: 0.9127

Epoch 159/500

 87/115 [=====================>........] - ETA: 0s - loss: 1.0433 - accuracy: 0.6553

Epoch 00159: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0430 - accuracy: 0.6558 - val_loss: 0.4780 - val_accuracy: 0.9170

Epoch 160/500

 88/115 [=====================>........] - ETA: 0s - loss: 1.0350 - accuracy: 0.6573

Epoch 00160: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0398 - accuracy: 0.6558 - val_loss: 0.4715 - val_accuracy: 0.9077

Epoch 161/500

 82/115 [====================>.........] - ETA: 0s - loss: 1.0351 - accuracy: 0.6572

Epoch 00161: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0361 - accuracy: 0.6570 - val_loss: 0.4758 - val_accuracy: 0.9162

Epoch 162/500

 85/115 [=====================>........] - ETA: 0s - loss: 1.0359 - accuracy: 0.6579

Epoch 00162: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0412 - accuracy: 0.6564 - val_loss: 0.4887 - val_accuracy: 0.9170

Epoch 163/500

 83/115 [====================>.........] - ETA: 0s - loss: 1.0461 - accuracy: 0.6536

Epoch 00163: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0410 - accuracy: 0.6552 - val_loss: 0.4637 - val_accuracy: 0.9179

Epoch 164/500

 85/115 [=====================>........] - ETA: 0s - loss: 1.0379 - accuracy: 0.6561

Epoch 00164: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0386 - accuracy: 0.6572 - val_loss: 0.4703 - val_accuracy: 0.9143

Epoch 165/500

 86/115 [=====================>........] - ETA: 0s - loss: 1.0332 - accuracy: 0.6632

Epoch 00165: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0406 - accuracy: 0.6606 - val_loss: 0.4772 - val_accuracy: 0.9107

Epoch 166/500

 82/115 [====================>.........] - ETA: 0s - loss: 1.0376 - accuracy: 0.6583

Epoch 00166: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0406 - accuracy: 0.6556 - val_loss: 0.4666 - val_accuracy: 0.9168

Epoch 167/500

 89/115 [======================>.......] - ETA: 0s - loss: 1.0362 - accuracy: 0.6574

Epoch 00167: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0302 - accuracy: 0.6598 - val_loss: 0.4742 - val_accuracy: 0.9116

Epoch 168/500

 81/115 [====================>.........] - ETA: 0s - loss: 1.0291 - accuracy: 0.6601

Epoch 00168: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0291 - accuracy: 0.6590 - val_loss: 0.4696 - val_accuracy: 0.9148

Epoch 169/500

 88/115 [=====================>........] - ETA: 0s - loss: 1.0343 - accuracy: 0.6549

Epoch 00169: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0288 - accuracy: 0.6591 - val_loss: 0.4856 - val_accuracy: 0.8929

Epoch 170/500

115/115 [==============================] - ETA: 0s - loss: 1.0504 - accuracy: 0.6526

Epoch 00170: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 3ms/step - loss: 1.0504 - accuracy: 0.6526 - val_loss: 0.4796 - val_accuracy: 0.9135

Epoch 171/500

 84/115 [====================>.........] - ETA: 0s - loss: 1.0306 - accuracy: 0.6580

Epoch 00171: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0274 - accuracy: 0.6587 - val_loss: 0.4709 - val_accuracy: 0.9157

Epoch 172/500

 84/115 [====================>.........] - ETA: 0s - loss: 1.0434 - accuracy: 0.6589

Epoch 00172: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0379 - accuracy: 0.6595 - val_loss: 0.4662 - val_accuracy: 0.9165

Epoch 173/500

 86/115 [=====================>........] - ETA: 0s - loss: 1.0303 - accuracy: 0.6523

Epoch 00173: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0256 - accuracy: 0.6550 - val_loss: 0.4654 - val_accuracy: 0.9127

Epoch 174/500

 85/115 [=====================>........] - ETA: 0s - loss: 1.0175 - accuracy: 0.6613

Epoch 00174: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0211 - accuracy: 0.6611 - val_loss: 0.4793 - val_accuracy: 0.9181

Epoch 175/500

 85/115 [=====================>........] - ETA: 0s - loss: 1.0401 - accuracy: 0.6518

Epoch 00175: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0399 - accuracy: 0.6542 - val_loss: 0.4721 - val_accuracy: 0.9066

Epoch 176/500

 81/115 [====================>.........] - ETA: 0s - loss: 1.0191 - accuracy: 0.6624

Epoch 00176: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0354 - accuracy: 0.6571 - val_loss: 0.4941 - val_accuracy: 0.9083

Epoch 177/500

 85/115 [=====================>........] - ETA: 0s - loss: 1.0306 - accuracy: 0.6611

Epoch 00177: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0370 - accuracy: 0.6586 - val_loss: 0.4794 - val_accuracy: 0.9094

Epoch 178/500

 85/115 [=====================>........] - ETA: 0s - loss: 1.0235 - accuracy: 0.6624

Epoch 00178: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0192 - accuracy: 0.6632 - val_loss: 0.4660 - val_accuracy: 0.9140

Epoch 179/500

 79/115 [===================>..........] - ETA: 0s - loss: 1.0099 - accuracy: 0.6667

Epoch 00179: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0111 - accuracy: 0.6673 - val_loss: 0.4686 - val_accuracy: 0.9083

Epoch 180/500

 84/115 [====================>.........] - ETA: 0s - loss: 1.0117 - accuracy: 0.6616

Epoch 00180: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0189 - accuracy: 0.6592 - val_loss: 0.4619 - val_accuracy: 0.9165

Epoch 181/500

 86/115 [=====================>........] - ETA: 0s - loss: 1.0361 - accuracy: 0.6607

Epoch 00181: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0323 - accuracy: 0.6587 - val_loss: 0.4828 - val_accuracy: 0.9135

Epoch 182/500

 84/115 [====================>.........] - ETA: 0s - loss: 1.0382 - accuracy: 0.6528

Epoch 00182: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0396 - accuracy: 0.6548 - val_loss: 0.4881 - val_accuracy: 0.9001

Epoch 183/500

84/115 [====================>.........] - ETA: 0s - loss: 1.0379 - accuracy: 0.6603

Epoch 00183: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0259 - accuracy: 0.6636 - val_loss: 0.4754 - val_accuracy: 0.9137

Epoch 184/500

79/115 [===================>..........] - ETA: 0s - loss: 1.0287 - accuracy: 0.6548

Epoch 00184: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0320 - accuracy: 0.6572 - val_loss: 0.4697 - val_accuracy: 0.9184

Epoch 185/500

84/115 [====================>.........] - ETA: 0s - loss: 1.0205 - accuracy: 0.6618

Epoch 00185: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0203 - accuracy: 0.6611 - val_loss: 0.4683 - val_accuracy: 0.9148

Epoch 186/500

83/115 [====================>.........] - ETA: 0s - loss: 1.0151 - accuracy: 0.6627

Epoch 00186: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0230 - accuracy: 0.6600 - val_loss: 0.4802 - val_accuracy: 0.9143

Epoch 187/500

86/115 [=====================>........] - ETA: 0s - loss: 1.0267 - accuracy: 0.6615

Epoch 00187: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0256 - accuracy: 0.6608 - val_loss: 0.4767 - val_accuracy: 0.9170

Epoch 188/500

 83/115 [====================>.........] - ETA: 0s - loss: 1.0219 - accuracy: 0.6580

Epoch 00188: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0294 - accuracy: 0.6603 - val_loss: 0.4765 - val_accuracy: 0.9102

Epoch 189/500

 86/115 [=====================>........] - ETA: 0s - loss: 1.0367 - accuracy: 0.6547

Epoch 00189: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0342 - accuracy: 0.6567 - val_loss: 0.4770 - val_accuracy: 0.9094

Epoch 190/500

 87/115 [=====================>........] - ETA: 0s - loss: 1.0140 - accuracy: 0.6656

Epoch 00190: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0239 - accuracy: 0.6625 - val_loss: 0.4886 - val_accuracy: 0.9066

Epoch 191/500

 85/115 [=====================>........] - ETA: 0s - loss: 1.0206 - accuracy: 0.6581

Epoch 00191: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0256 - accuracy: 0.6584 - val_loss: 0.4849 - val_accuracy: 0.8938

Epoch 192/500

 85/115 [=====================>........] - ETA: 0s - loss: 1.0121 - accuracy: 0.6642

Epoch 00192: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0147 - accuracy: 0.6635 - val_loss: 0.4825 - val_accuracy: 0.9162

Epoch 193/500

 88/115 [=====================>........] - ETA: 0s - loss: 1.0216 - accuracy: 0.6644

Epoch 00193: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0154 - accuracy: 0.6660 - val_loss: 0.4630 - val_accuracy: 0.9154

Epoch 194/500

 88/115 [=====================>........] - ETA: 0s - loss: 1.0312 - accuracy: 0.6586

Epoch 00194: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0320 - accuracy: 0.6581 - val_loss: 0.4751 - val_accuracy: 0.9113

Epoch 195/500

 83/115 [====================>.........] - ETA: 0s - loss: 1.0221 - accuracy: 0.6584

Epoch 00195: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0252 - accuracy: 0.6580 - val_loss: 0.4821 - val_accuracy: 0.9083

Epoch 196/500

 82/115 [====================>.........] - ETA: 0s - loss: 1.0358 - accuracy: 0.6574

Epoch 00196: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0279 - accuracy: 0.6602 - val_loss: 0.4754 - val_accuracy: 0.9159

Epoch 197/500

114/115 [============================>.] - ETA: 0s - loss: 1.0193 - accuracy: 0.6580

Epoch 00197: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0190 - accuracy: 0.6582 - val_loss: 0.4712 - val_accuracy: 0.9083

Epoch 198/500

 86/115 [=====================>........] - ETA: 0s - loss: 1.0141 - accuracy: 0.6662

Epoch 00198: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0147 - accuracy: 0.6632 - val_loss: 0.4684 - val_accuracy: 0.9124

Epoch 199/500

 86/115 [=====================>........] - ETA: 0s - loss: 1.0179 - accuracy: 0.6606

Epoch 00199: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0172 - accuracy: 0.6600 - val_loss: 0.4815 - val_accuracy: 0.8981

Epoch 200/500

 88/115 [=====================>........] - ETA: 0s - loss: 1.0254 - accuracy: 0.6589

Epoch 00200: saving model to keypoint_classifier_new.h5

115/115 [==============================] - 0s 2ms/step - loss: 1.0281 - accuracy: 0.6602 - val_loss: 0.4783 - val_accuracy: 0.9107

Epoch 00200: early stopping

WARNING:tensorflow:No training configuration found in the save file, so the model was *not* compiled. Compile it manually.

Average prediction time: 0.000052s

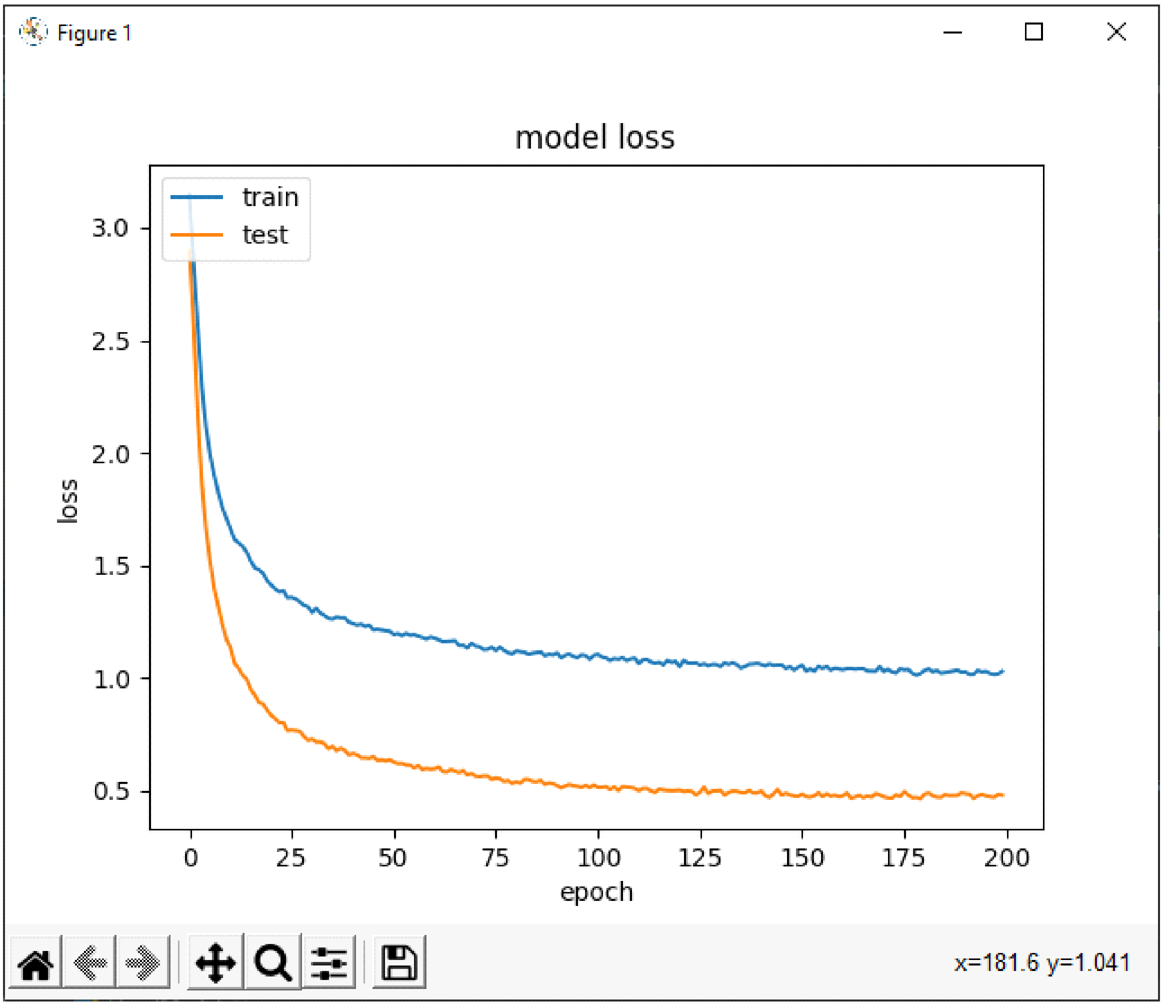
Classification Report

```
              precision    recall  f1-score   support

           0       0.99      0.90      0.95       105
```

1	0.91	1.00	0.95	99
2	0.92	0.66	0.77	146
3	0.73	0.58	0.65	185
4	0.63	0.95	0.76	168
5	1.00	0.85	0.92	88
6	1.00	0.92	0.96	39
7	0.98	0.75	0.85	73
8	0.91	1.00	0.95	86
9	0.96	1.00	0.98	54
10	0.93	0.93	0.93	123
11	0.95	0.83	0.89	196
12	1.00	0.97	0.98	230
13	0.97	1.00	0.98	143
14	0.95	0.97	0.96	125
15	0.99	1.00	0.99	228
16	0.99	0.96	0.97	193
17	0.87	1.00	0.93	256
18	0.95	0.99	0.97	117
19	0.64	1.00	0.78	140
20	0.92	0.87	0.89	165
21	1.00	0.98	0.99	134
22	0.99	1.00	1.00	139
23	0.99	0.54	0.70	156
24	0.99	1.00	0.99	96
25	0.94	0.95	0.95	168
accuracy			0.90	3652

macro avg	0.93	0.91	0.91	3652
weighted avg	0.92	0.90	0.90	3652

Process finished with exit code 0



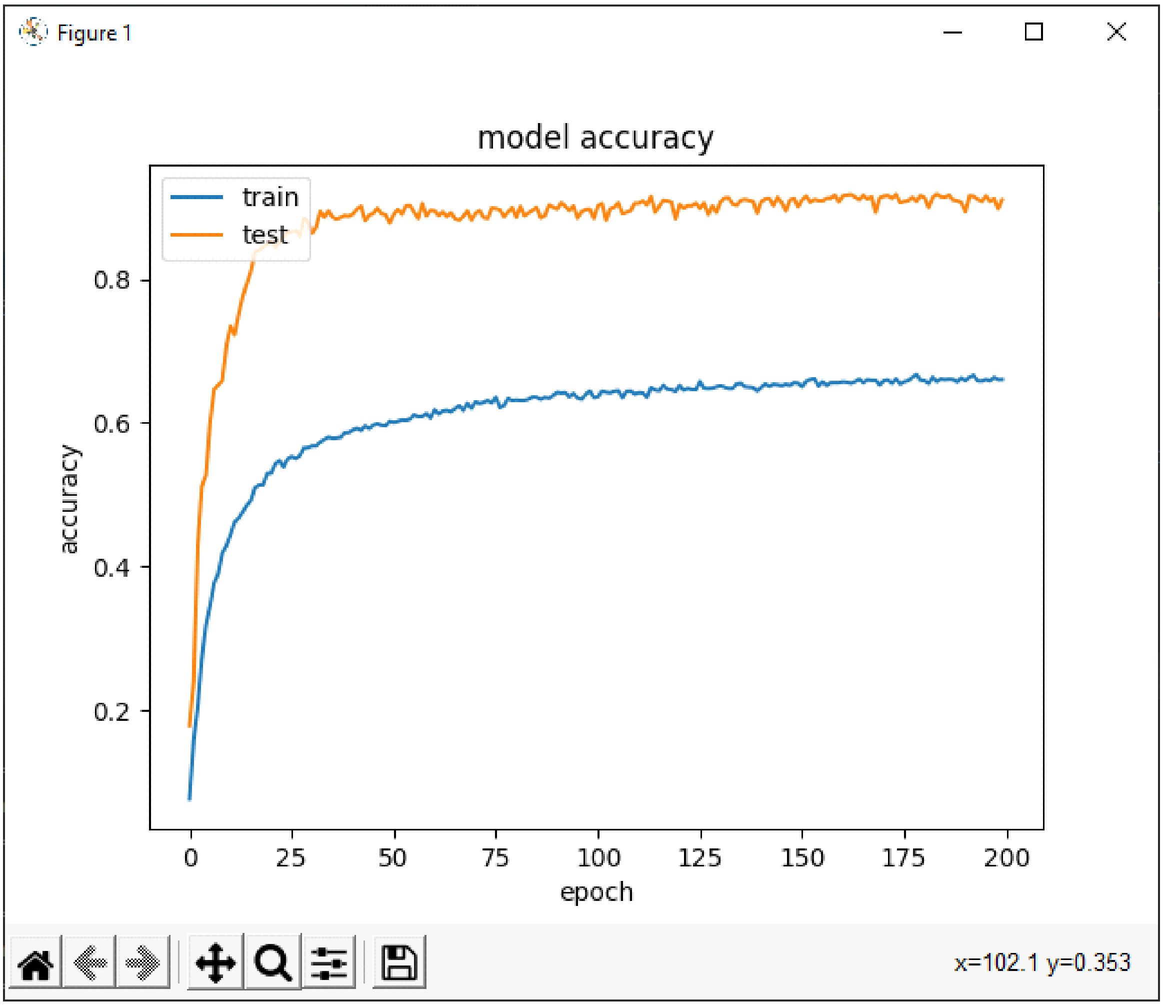


Figure 1
model accuracy
train
test
0.8
0.6
0.4
0.2
accuracy
0
25
50
75
100
125
150
175
200
epoch
x=102.1 y=0.353

GITHUB ACCOUNT LINK:

https://github.com/IBM-EPBL/IBM-Project-9389-1658999257

PROJECT DEMO LINK:

https://drive.google.com/file/d/1H57cyENYduuslVfAioLJ2Ki0m6mN6eo1/view?usp=share_link