CODE:

```
from tkinter import messagebox

from tkinter import *

from tkinter import simpledialog

import tkinter

from tkinter import filedialog

from tkinter.filedialog import askopenfilename

import cv2

import random

import numpy as np

from keras.utils.np_utils import to_categorical

from keras.layers import  MaxPooling2D

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

from keras.layers import Convolution2D

from keras.models import Sequential

from keras.models import model_from_json

import pickle

import os

import imutils

from gtts import gTTS

from playsound import playsound

import os
```

```
from threading import Thread

from keras.preprocessing import image


main = tkinter.Tk()

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

main.geometry("1300x1200")


global filename

global classifier

global digit_model


bg = None

playcount = 0


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

names = ['C', 'Down', 'Fist', 'Five', 'I', 'LeftForward', 'Ok', 'Palm', 'Palmmoved',
'Peace', 'Rad', 'RightForward', 'RightReverse', 'Straight', 'Thumb']


def getID(name):

   index = 0

   for i in range(len(names)):
       if names[i] == name:
           index = i
           break
   return index


def loadDigitModel():
   global digit_model
   with open('model/cnn_model.json', "r") as json_file:
       loaded_model_json = json_file.read()
       digit_model = model_from_json(loaded_model_json)
   json_file.close()
   digit_model.load_weights("model/cnn_weights.h5")
   digit_model._make_predict_function()



bgModel = cv2.createBackgroundSubtractorMOG2(0, 50)


def deleteDirectory():
   filelist = [ f for f in os.listdir('play') if f.endswith(".mp3") ]
   for f in filelist:
       os.remove(os.path.join('play', f))
```

```
def play(playcount,gesture):
    class PlayThread(Thread):
        def __init__(self,playcount,gesture):
            Thread.__init__(self)
            self.gesture = gesture
            self.playcount = playcount

        def run(self):
            t1 = gTTS(text=self.gesture, lang='en', slow=False)
            t1.save("play/"+str(self.playcount)+".mp3")
            playsound("play/"+str(self.playcount)+".mp3")



    newthread = PlayThread(playcount,gesture)
    newthread.start()


def remove_background(frame):
    fgmask = bgModel.apply(frame, learningRate=0)
    kernel = np.ones((3, 3), np.uint8)
    fgmask = cv2.erode(fgmask, kernel, iterations=1)
    res = cv2.bitwise_and(frame, frame, mask=fgmask)
    return res
```

```
def uploadDataset():
    global filename
    global labels
    labels = []
    filename = filedialog.askdirectory(initialdir=".")
    pathlabel.config(text=filename)
    text.delete('1.0', END)
    text.insert(END,filename+" loaded\n\n");



def trainCNN():
    global classifier
    text.delete('1.0', END)
    X_train = np.load('model/X.txt.npy')
    Y_train = np.load('model/Y.txt.npy')
    print(Y_train.shape)
    text.insert(END,"CNN is training on total images : "+str(len(X_train))+"\n")

    if os.path.exists('model/model.json'):
        with open('model/model.json', "r") as json_file:
            loaded_model_json = json_file.read()

            classifier = model_from_json(loaded_model_json)

        classifier.load_weights("model/model_weights.h5")

        classifier._make_predict_function()

        print(classifier.summary())

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

        data = pickle.load(f)

        f.close()

        acc = data['accuracy']

        accuracy = acc[9] * 100

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

    else:

        classifier = Sequential()

        classifier.add(Convolution2D(32, 3, 3, input_shape = (64, 64, 3), activation
= 'relu'))

        classifier.add(MaxPooling2D(pool_size = (2, 2)))

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

        classifier.add(MaxPooling2D(pool_size = (2, 2)))

        classifier.add(Flatten())

        classifier.add(Dense(output_dim = 256, activation = 'relu'))

        classifier.add(Dense(output_dim = 5, activation = 'softmax'))
```

```
print(classifier.summary())
classifier.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
hist = classifier.fit(X_train, Y_train, batch_size=16, epochs=10, shuffle=True, verbose=2)
classifier.save_weights('model/model_weights.h5')
model_json = classifier.to_json()
with open("model/model.json", "w") as json_file:
    json_file.write(model_json)
f = open('model/history.pckl', 'wb')
pickle.dump(hist.history, f)
f.close()
f = open('model/history.pckl', 'rb')
data = pickle.load(f)
f.close()
acc = data['accuracy']
accuracy = acc[9] * 100
text.insert(END,"CNN Hand Gesture Training Model Prediction Accuracy = "+str(accuracy))
```

```
def run_avg(image, aWeight):
    global bg
    if bg is None:
        bg = image.copy().astype("float")
        return
    cv2.accumulateWeighted(image, bg, aWeight)


def segment(image, threshold=25):
    global bg
    diff = cv2.absdiff(bg.astype("uint8"), image)
    thresholded = cv2.threshold(diff, threshold, 255, cv2.THRESH_BINARY)[1]
    ( cnts, _) = cv2.findContours(thresholded.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
    if len(cnts) == 0:
        return
    else:
        segmented = max(cnts, key=cv2.contourArea)
        return (thresholded, segmented)



def webcamPredict():
    global playcount
```

```
oldresult = 'none'

count = 0

fgbg2 = cv2.createBackgroundSubtractorKNN();

aWeight = 0.5

camera = cv2.VideoCapture(0)

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

num_frames = 0

while(True):

    (grabbed, frame) = camera.read()

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

    frame = cv2.flip(frame, 1)

    clone = frame.copy()

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

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

    gray = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)

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

    if num_frames < 30:

        run_avg(gray, aWeight)

    else:

        temp = gray

        hand = segment(gray)

        if hand is not None:
            (thresholded, segmented) = hand
            cv2.drawContours(clone, [segmented + (right, top)], -1, (0, 0, 255))
            #cv2.imwrite("test.jpg",temp)
            #cv2.imshow("Thesholded", temp)
            #ret, thresh = cv2.threshold(temp, 150, 255, cv2.THRESH_BINARY +
cv2.THRESH_OTSU)
            #thresh = cv2.resize(thresh, (64, 64))
            #thresh = np.array(thresh)
            #img = np.stack((thresh,)*3, axis=-1)
            roi = frame[top:bottom, right:left]
            roi = fgbg2.apply(roi);
            cv2.imwrite("test.jpg",roi)
            '''
            cv2.imwrite("newDataset/Straight/"+str(count)+".png",roi)
            count = count + 1
            print(count)
            if count > 400:
               break
            '''
            img = cv2.imread("test.jpg")
            img = cv2.resize(img, (32, 32))
            img = img.reshape(1, 32, 32, 3)
            img = np.array(img, dtype='float32')
            img /= 255
            predict = classifier.predict(img)
            value = np.amax(predict)
            cl = np.argmax(predict)
            result = names[np.argmax(predict)]
            if value >= 0.99:
               print(str(value)+" "+str(result))
               cv2.putText(clone, 'Gesture Recognize as : '+str(result), (10, 25),
cv2.FONT_HERSHEY_SIMPLEX,0.5, (0, 255, 255), 2)
               '''
               if oldresult != result:
                  play(playcount,result)
                  oldresult = result
                  playcount = playcount + 1
               '''
            else:
               cv2.putText(clone,                  '',                  (10,                  25),
cv2.FONT_HERSHEY_SIMPLEX,0.5, (0, 255, 255), 2)
            cv2.imshow("video frame", roi)
      cv2.rectangle(clone, (left, top), (right, bottom), (0,255,0), 2)
      num_frames += 1
```

```
        cv2.imshow("Video Feed", clone)

        keypress = cv2.waitKey(1) & 0xFF

        if keypress == ord("q"):

            break

    camera.release()

    cv2.destroyAllWindows()


def digitPredict():

    global digit_model

    labels = ['1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J',
'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T',

              'U', 'V', 'W', 'X', 'Y', 'Z']

    filename = filedialog.askopenfilename(initialdir="testDigitImages")

    image = cv2.imread(filename)

    img = cv2.resize(image, (32,32))

    im2arr = np.array(img)

    im2arr = im2arr.reshape(1,32,32,3)

    img = np.asarray(im2arr)

    img = img.astype('float32')

    img = img/255

    preds = digit_model.predict(img)

    predict = np.argmax(preds)
    img = cv2.imread(filename)
    img = cv2.resize(img, (600,400))
    cv2.putText(img, 'Image Contains Digit/Alphabet as : '+labels[predict], (10, 25),  cv2.FONT_HERSHEY_SIMPLEX,0.7, (255, 0, 0), 2)
    cv2.imshow('Image Contains Digit/Alphabet as : '+labels[predict], img)
    cv2.waitKey(0)


font = ('times', 16, 'bold')
title = Label(main, text='Sign Language Recognition to Text & Voice using CNN Advance',anchor=W, justify=CENTER)
title.config(bg='yellow4', fg='white')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=0,y=5)



font1 = ('times', 13, 'bold')
upload = Button(main, text="Upload Hand Gesture Dataset", command=uploadDataset)
upload.place(x=50,y=100)
upload.config(font=font1)
```

```
pathlabel = Label(main)

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

pathlabel.config(font=font1)

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


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

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

markovButton.config(font=font1)


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

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

predictButton.config(font=font1)


DigitButton = Button(main, text="Digit Recognition from Image", command=digitPredict)

DigitButton.place(x=50,y=300)

DigitButton.config(font=font1)
```

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

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

scroll=Scrollbar(text)

text.configure(yscrollcommand=scroll.set)

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

text.config(font=font1)


deleteDirectory()

loadDigitModel()

main.config(bg='magenta3')

main.mainloop()
```