Import and install required dependencies

In [ ]:
```
#install dependencies
#!pip install tensorflow==2.13.0rc1 opencv-python sklearn matplotlib
```

In [ ]:
```
#install mediapipe
#!pip install mediapipe
```

In [1]:
```
#import openCV
import cv2

#import numPy
import numpy as np

#import os
import os

#import matplotlib
from matplotlib import pyplot as plt

#import time
import time

#import mediaPipe
import mediapipe as mp
```

1. Keypoint Extraction using MediaPipe

Creating variables and functions for keypoint Extraction

In [2]:
```
#creating the variables and assigning functions
mediapipe_holistic = mp.solutions.holistic
mediapipe_drawing = mp.solutions.drawing_utils
```

In [3]:
```
# creating detection function
def detection_function(image, model):
    #convert BGR to RGB
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    #make image non-writeable
    image.flags.writeable = False
    #make prediction
    detected_landmarks = model.process(image)
    #make image writeable
    image.flags.writeable = True
    #convert RGB 2 BGR
    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
    return image, detected_landmarks
```

In [4]:
```
# function to visualize the landmarks using 'mediapipe_drawing'variable
def draw_styled_landmarks(image, detected_landmarks):
    # Draw left hand connections
    mediapipe_drawing.draw_landmarks(image,
                                     detected_landmarks.left_hand_landmarks,
                                     mediapipe_holistic.HAND_CONNECTIONS,
                                     mediapipe_drawing.DrawingSpec(color=(1,255,
                                     255), thickness=2, circle_radius=4),
                                     mediapipe_drawing.DrawingSpec(color=(255,
                                     15,10), thickness=2, circle_radius=2)
                                     )
    # Draw right hand connections
    mediapipe_drawing.draw_landmarks(image,
                                     detected_landmarks.right_hand_landmarks,
                                     mediapipe_holistic.HAND_CONNECTIONS,
                                     mediapipe_drawing.DrawingSpec(color=(5,
                                     255,3), thickness=2, circle_radius=4),
                                     mediapipe_drawing.DrawingSpec(color=(9,
                                     9,255), thickness=2, circle_radius=2)
                                     )
```

Extracting keypoints values from captured video frames

In [5]:
```
# access webcam (video capture device (1))
cam = cv2.VideoCapture(0)


# Set mediapipe model
with mediapipe_holistic.Holistic(min_detection_confidence=0.5,
                                 min_tracking_confidence=0.5)
                                 as holistic:

# begin while loop
    while cam.isOpened():

            # Read feed
            return_value, image_frame = cam.read()

            # Make detections
            # get the 'image' and 'detected_landmarks'
            image, detected_landmarks = detection_function(image_frame,
                                                           holistic)
            print(detected_landmarks)

            # Draw the landmarks
            draw_styled_landmarks(image, detected_landmarks)

            # Show o screen
            cv2.imshow('OpenCV window', image)

            # break statement
            if cv2.waitKey(10) & 0xFF == ord('q'):
                break
            #while loop end

    cam.release()
    cv2.destroyAllWindows()
```

```
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
```

```
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
```

```
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
```

```
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
```

In [6]:

```
#accessing the last frame to display landmark values (left-hand)
detected_landmarks.left_hand_landmarks
```

Out[6]:

```
landmark {
  x: 0.8672328
  y: 0.95512223
  z: 4.925319e-07
}
```

```
landmark {
  x: 0.78805244
  y: 0.91567147
  z: -0.032983683
}
landmark {
  x: 0.7260375
  y: 0.7999445
  z: -0.043909486
}
landmark {
  x: 0.68937373
  y: 0.6878332
  z: -0.05170609
}
landmark {
  x: 0.6507983
  y: 0.6169369
  z: -0.058083836
}
landmark {
  x: 0.76142716
  y: 0.6314674
  z: -0.013089947
}
landmark {
  x: 0.7426324
  y: 0.49407232
  z: -0.030419303
}
landmark {
  x: 0.7351809
  y: 0.41194654
  z: -0.046626087
}
landmark {
  x: 0.73148054
  y: 0.33647227
  z: -0.059442945
}
landmark {
  x: 0.80912966
  y: 0.60267764
  z: -0.013207907
}
landmark {
  x: 0.7948489
  y: 0.44424373
  z: -0.025664767
}
landmark {
  x: 0.78900963
  y: 0.3460162
  z: -0.039955854
}
landmark {
  x: 0.7848694
```

```
    y: 0.26286328
    z: -0.052027248
  }
  landmark {
    x: 0.85448897
    y: 0.6073079
    z: -0.019168675
  }
  landmark {
    x: 0.84751445
    y: 0.45606884
    z: -0.03567219
  }
  landmark {
    x: 0.8425583
    y: 0.36568794
    z: -0.047914352
  }
  landmark {
    x: 0.8365833
    y: 0.2869447
    z: -0.057682145
  }
  landmark {
    x: 0.9011621
    y: 0.6351237
    z: -0.028683169
  }
  landmark {
    x: 0.9065147
    y: 0.5230583
    z: -0.04525318
  }
  landmark {
    x: 0.9086623
    y: 0.44885045
    z: -0.05269893
  }
  landmark {
    x: 0.9084273
    y: 0.3791155
    z: -0.058459677
  }
```

In [7]:

```
# length of the detected landmarks (left hand)
len(detected_landmarks.left_hand_landmarks.landmark)
```

Out[7]:

```
21
```

In [8]:

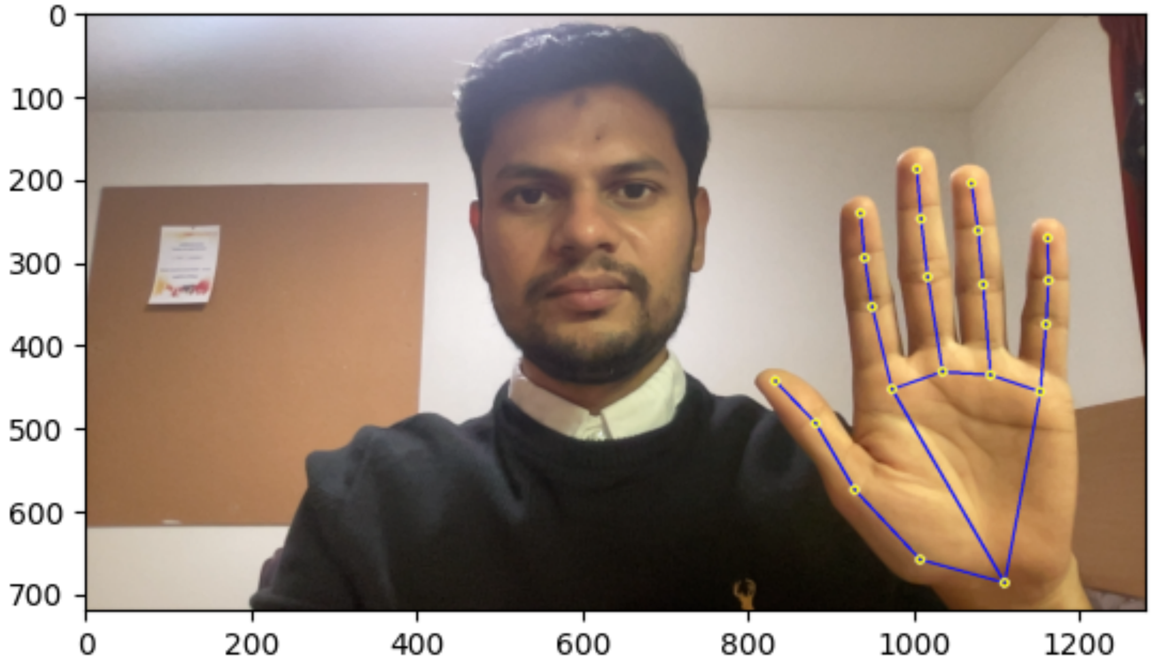
```
# latest video frame inform of array
image_frame
```

Out[8]:
```
array([[[197, 190, 193],
         [198, 191, 194],
         [199, 192, 195],
         ...,
         [ 38,  40,  71],
         [ 43,  45,  76],
         [ 44,  46,  77]],

        [[197, 190, 193],
         [196, 189, 191],
         [199, 192, 195],
         ...,
         [ 39,  41,  73],
         [ 45,  47,  78],
         [ 41,  44,  75]],

        [[198, 191, 194],
         [200, 193, 196],
         [202, 195, 197],
         ...,
         [ 32,  34,  67],
         [ 46,  48,  81],
         [ 44,  45,  79]],

        ...,

        [[167, 185, 197],
         [166, 187, 196],
         [167, 188, 197],
         ...,
         [ 91, 102, 124],
         [ 89, 100, 121],
         [ 86,  98, 119]],

        [[166, 184, 196],
         [164, 185, 194],
         [163, 184, 193],
         ...,
         [ 93, 105, 126],
         [ 89, 100, 121],
         [ 89, 100, 121]],

        [[166, 184, 196],
         [164, 185, 194],
         [166, 187, 196],
         ...,
         [ 89, 100, 121],
         [ 89, 100, 121],
         [ 89, 100, 121]]], dtype=uint8)
```

In [9]:
```
# applying draw_styled_landmarks to current frame
draw_styled_landmarks(image_frame, detected_landmarks)
```

In [10]:
```
#visuaize the current captured frame in RGB format using matplotlib
plt.imshow(cv2.cvtColor(image_frame, cv2.COLOR_BGR2RGB))
```

Out[10]: <matplotlib.image.AxesImage at 0x2a02d5d50>



Store extracted Keypoints into numPy array

In [11]:
```
# creating variables to store extracted keypoints in a flattened array
left_hand = np.array([[res.x, res.y, res.z] for res in
            detected_landmarks.left_hand_landmarks.landmark]).flatten()
            if detected_landmarks.left_hand_landmarks
            else np.zeros(21*3)
right_hand = np.array([[res.x, res.y, res.z]
             for res
             in detected_landmarks.right_hand_landmarks.landmark])
             .flatten()
             if detected_landmarks.right_hand_landmarks
             else np.zeros(21*3)
```

In [12]:
```
# displaying keypoint values for left hand
left_hand
```

Out[12]:
```
array([ 8.67232800e-01,  9.55122232e-01,  4.92531910e-07,  7.88052440e-01
,
         9.15671468e-01, -3.29836830e-02,  7.26037502e-01,  7.99944520e-01
,
        -4.39094864e-02,  6.89373732e-01,  6.87833190e-01, -5.17060906e-02
,
         6.50798321e-01,  6.16936922e-01, -5.80838360e-02,  7.61427164e-01
,
         6.31467402e-01, -1.30899474e-02,  7.42632389e-01,  4.94072318e-01
,
        -3.04193031e-02,  7.35180914e-01,  4.11946535e-01, -4.66260873e-02
,
         7.31480539e-01,  3.36472273e-01, -5.94429448e-02,  8.09129655e-01
,
         6.02677643e-01, -1.32079069e-02,  7.94848919e-01,  4.44243729e-01
,
        -2.56647673e-02,  7.89009631e-01,  3.46016198e-01, -3.99558544e-02
,
         7.84869373e-01,  2.62863278e-01, -5.20272478e-02,  8.54488969e-01
,
         6.07307911e-01, -1.91686749e-02,  8.47514451e-01,  4.56068844e-01
,
        -3.56721915e-02,  8.42558324e-01,  3.65687937e-01, -4.79143523e-02
,
         8.36583316e-01,  2.86944687e-01, -5.76821454e-02,  9.01162088e-01
,
         6.35123730e-01, -2.86831688e-02,  9.06514704e-01,  5.23058295e-01
,
        -4.52531800e-02,  9.08662319e-01,  4.48850453e-01, -5.26989289e-02
,
         9.08427298e-01,  3.79115492e-01, -5.84596768e-02])
```

In [13]:
```
# shape of left_hand array # 21*3 = 63
left_hand.shape
```

Out[13]:
```
(63,)
```

In [14]:
```
# keypoint values for right hand
right_hand
```

Out[14]:
```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

In [15]:
```
# shape of right_hand array shape #np.zeros(21*3) = 63
right_hand.shape
```

Out[15]:
```
(63,)
```

Function to extract keypoints and concatenate into a single array

In [16]:
```
# function to extract keypoints and concatenate into single array
def mediapipe_keypoints(detected_landmarks):
    left_hand = np.array([[res.x, res.y, res.z] for res
                in detected_landmarks.left_hand_landmarks.landmark])
                .flatten() if detected_landmarks.left_hand_landmarks
                else np.zeros(21*3)
    right_hand = np.array([[res.x, res.y, res.z] for res
                 in detected_landmarks.right_hand_landmarks.landmark])
                 .flatten() if detected_landmarks.right_hand_landmarks
                 else np.zeros(21*3)
    return np.concatenate([ left_hand, right_hand])
```

In [17]:
```
#checking the final shape of the concatenated array
mediapipe_keypoints(detected_landmarks).shape

#expected result:
# (left-hand keypoints * (x,y,z co-ordinates))
# + (right-hand keypoints * (x,y,z co-ordinates))
# 21*3 + 21*3 = 126
```

Out[17]:
```
(126,)
```

In [18]:
```
# storing the resultant array in a variable
total_keypoints = mediapipe_keypoints(detected_landmarks)
```

In [19]:
```
# Displaing the concatenated array
total_keypoints
```

Out[19]:
```
array([ 8.67232800e-01,  9.55122232e-01,  4.92531910e-07,  7.88052440e-01
,
        9.15671468e-01, -3.29836830e-02,  7.26037502e-01,  7.99944520e-01
,
       -4.39094864e-02,  6.89373732e-01,  6.87833190e-01, -5.17060906e-02
,
        6.50798321e-01,  6.16936922e-01, -5.80838360e-02,  7.61427164e-01
,
        6.31467402e-01, -1.30899474e-02,  7.42632389e-01,  4.94072318e-01
,
       -3.04193031e-02,  7.35180914e-01,  4.11946535e-01, -4.66260873e-02
,
        7.31480539e-01,  3.36472273e-01, -5.94429448e-02,  8.09129655e-01
,
        6.02677643e-01, -1.32079069e-02,  7.94848919e-01,  4.44243729e-01
,
       -2.56647673e-02,  7.89009631e-01,  3.46016198e-01, -3.99558544e-02
,
        7.84869373e-01,  2.62863278e-01, -5.20272478e-02,  8.54488969e-01
,
        6.07307911e-01, -1.91686749e-02,  8.47514451e-01,  4.56068844e-01
,
       -3.56721915e-02,  8.42558324e-01,  3.65687937e-01, -4.79143523e-02
,
        8.36583316e-01,  2.86944687e-01, -5.76821454e-02,  9.01162088e-01
,
        6.35123730e-01, -2.86831688e-02,  9.06514704e-01,  5.23058295e-01
```

```
,
       -4.52531800e-02,  9.08662319e-01,  4.48850453e-01, -5.26989289e-02
,
        9.08427298e-01,  3.79115492e-01, -5.84596768e-02,  0.00000000e+00
,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00
,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00
,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00
,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00
,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00
,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00
,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00
,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00
,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00
,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00
,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00
,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00
,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00
,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00
,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00
,
        0.00000000e+00,  0.00000000e+00])
```

2. Create Datasets for BSL fingerspelling

Set variables for Dataset creation

In [20]:
```
# Path for exported data, numpy arrays
DATA_PATH = os.path.join('MP_Data')

# BSL fingerspelling alphabets
alphabets = np.array(['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'])

# 30 video seqences per each alphapbet
no_sequences = 30

# 20 frames per each video sequence
sequence_length = 20
```

Create Folders for Datasets

In [21]:
```
# create folders
for alphabet in alphabets:
    for sequence in range(no_sequences):
        try:
            os.makedirs(os.path.join(DATA_PATH, alphabet,
                                     str(sequence)))
        except:
            pass
```

Collecting datasets using openCV and mediaPipe

In [ ]:
```
# access webcam (video capture device (1)
cam = cv2.VideoCapture(1)

# Set mediapipe model
with mediapipe_holistic.Holistic(min_detection_confidence=0.5,
                    min_tracking_confidence=0.5) as holistic:

    # Loop through each alphabet
    for alphabet in alphabets:
        # Loop through each video sequence
        for sequence in range(no_sequences):
            # Loop through sequence length of each video
            for frame_number in range(sequence_length):

                # Read feed
                return_value, image_frame = cam.read()

                # Make detections
                image, detected_landmarks = detection_function
                (image_frame, holistic)

                # Draw landmarks
                draw_styled_landmarks(image, detected_landmarks)

                # creating of datasets
                if frame_number == 0:
                    cv2.putText(image, 'STARTING COLLECTION',
```

```
                                          (120,200),
                                         cv2.FONT_HERSHEY_SIMPLEX, 1, (0,255, 0),
                                          4, cv2.LINE_AA)
        cv2.putText(image, 'Collecting frames for {} Video Number {}'
                                          .format(alphabet, sequence), (15,12),
                                         cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0,
                                                      255), 1, cv2.LINE_AA)

                    cv2.imshow('OpenCV Data Collection', image)
                    cv2.waitKey(2000)
                else:
        cv2.putText(image, 'Collecting frames for {} Video Number {}'
                                         .format(alphabet, sequence), (15,12),
                                         cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0,
                                                      255), 1, cv2.LINE_AA)
                    # Show to screen
                    cv2.imshow('OpenCV Data Collection', image)

                # Export keypoints
                keypoints = mediapipe_keypoints(detected_landmarks)
                npy_path = os.path.join(DATA_PATH, alphabet,
                                        str(sequence), str(frame_number))
                np.save(npy_path, keypoints)

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

    cam.release()
    cv2.destroyAllWindows()
```

In [ ]:
```
cam.release()
cv2.destroyAllWindows()
```

labeling datasets

In [22]:
```
# create a lable dictionary to represent the alphabet index and
#their labels
alphabet_labels = {label:num for num, label in enumerate(alphabets)}
```

In [23]:
```
#display labels
alphabet_labels
```

Out[23]:
```
{'A': 0,
 'B': 1,
 'C': 2,
 'D': 3,
 'E': 4,
 'F': 5,
 'G': 6,
 'H': 7,
 'I': 8,
 'J': 9,
 'K': 10,
 'L': 11,
 'M': 12,
 'N': 13,
 'O': 14,
 'P': 15,
 'Q': 16,
 'R': 17,
 'S': 18,
 'T': 19,
 'U': 20,
 'V': 21,
 'W': 22,
 'X': 23,
 'Y': 24,
 'Z': 25}
```

Combine all data together

In [24]:
```
# bringing all data together and structuring it into a single array

# initializing empty arrays
sequences, labels = [], []
for alphabet in alphabets:
    for sequence in range(no_sequences):
        window = []
        for frame_number in range(sequence_length):
            res = np.load(os.path.join(DATA_PATH, alphabet,
                    str(sequence), "{}.npy".format(frame_number)))
            window.append(res)
        sequences.append(window)
        labels.append(alphabet_labels[alphabet])
```

In [25]:
```
# checking shape of final array
np.array(sequences).shape
```

Out[25]:
```
(780, 20, 126)
```

In [26]:
```
# checking shape of the labels
np.array(labels).shape
```

Out[26]:
```
(780,)
```

preprocess data for training

In [27]:
```
# import dependencies for splitting dataset and convert
#data using one-hot encoding
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.utils import to_categorical
```

In [28]:
```
# storing the sequences in 'X'
X = np.array(sequences)
```

In [29]:
```
# checking shape of 'X'
X.shape
```

Out[29]:
```
(780, 20, 126)
```

In [30]:
```
# converting the labels into binary flat using one-hot encoding
Y = to_categorical(labels).astype(int)
```

In [31]:
```
Y
```

Out[31]:
```
array([[1, 0, 0, ..., 0, 0, 0],
       [1, 0, 0, ..., 0, 0, 0],
       [1, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 1],
       [0, 0, 0, ..., 0, 0, 1],
       [0, 0, 0, ..., 0, 0, 1]])
```

In [32]:
```
Y.shape
```

Out[32]:
```
(780, 26)
```

Split dataset into Train and Test categories

In [33]:
```
# splitting the dataset into training and testing
#(training data = 90%, testing data = 10%)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.1)
```

In [34]:
```
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train.reshape(-1,
                    X_train.shape[-1])).reshape(X_train.shape)
X_test_scaled = scaler.transform(X_test.reshape(-1,
                    X_test.shape[-1])).reshape(X_test.shape)
```

In [35]:
```
# checking the shapes of training and testing data after
# the splitting of datasets
X_train.shape
```

Out[35]:
```
(702, 20, 126)
```

In [36]:
```
X_test.shape
```

Out[36]:
```
(78, 20, 126)
```

In [37]:
```
Y_train.shape
```

Out[37]:
```
(702, 26)
```

In [38]:
```
Y_test.shape
```

Out[38]:
```
(78, 26)
```

3. Training dataset using LSTM

Build LSTM architecture

In [39]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from tensorflow.keras.callbacks import TensorBoard
from tensorflow.keras.layers import LeakyReLU
from tensorflow.keras.optimizers import legacy as keras_legacy_optimizer
```

In [40]:
```
#storing TensorBoard logs
log_dir = os.path.join('Logs')

#integrating TensorBoard with the model training process.
tb_callback = TensorBoard(log_dir=log_dir)
```

In [41]:
```
#initializing an empty LSTM Neural Network model
model = Sequential()

# adding LSTM layer to model which as 64 units, and uses 'tanh'
# activation function
model.add(LSTM(64, return_sequences=True, activation='tanh',
               input_shape=(20, 126)))
# adding another LSTM layer with 128 units and'tanh' activation
# function
model.add(LSTM(128, return_sequences=True, activation='tanh'))
# adding LSTM layer to model which as 64 units and'tanh'
# activation function
model.add(LSTM(64, return_sequences=False, activation='tanh'))

# adding a dense layer with 64 units
model.add(Dense(64, activation='tanh'))
# adding a dense layer with 32 units
model.add(Dense(32, activation='tanh'))
# adding a dense layer with units equal to number of
# categories(alphabets) and 'softmax' activation function
model.add(Dense(alphabets.shape[0], activation='softmax'))
```

In [42]:
```
#checking the number of outputs in the final layer
alphabets.shape[0]
```

Out[42]:
```
26
```

In [43]:
```
res = [0.7, 0.2, 0.1]
```

In [44]:
```
alphabets[np.argmax(res)]
```

Out[44]:
```
'A'
```

In [45]:
```
#configuring the training process of the model
model.compile(optimizer=keras_legacy_optimizer.Adam(learning_rate=0.001)
              , loss='categorical_crossentropy',
              metrics=['categorical_accuracy'])
```

In [46]:
```
#display the summary of the LSTM model built
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 lstm (LSTM)                 (None, 20, 64)            48896     
                                                                 
 lstm_1 (LSTM)               (None, 20, 128)           98816     
                                                                 
 lstm_2 (LSTM)               (None, 64)                49408     
                                                                 
 dense (Dense)               (None, 64)                4160      
                                                                 
 dense_1 (Dense)             (None, 32)                2080      
                                                                 
 dense_2 (Dense)             (None, 26)                858       
                                                                 
=================================================================
Total params: 204218 (797.73 KB)
Trainable params: 204218 (797.73 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Train LSTM Neural Network

In [47]:
```
# training model.
model.fit(X_train, Y_train, epochs=2000, callbacks=[tb_callback])
```

```
Epoch 1/2000
22/22 [==============================] - 3s 28ms/step - loss: 3.0652 - ca
tegorical_accuracy: 0.1054
Epoch 2/2000
22/22 [==============================] - 0s 22ms/step - loss: 2.4747 - ca
tegorical_accuracy: 0.2251
Epoch 3/2000
22/22 [==============================] - 0s 22ms/step - loss: 1.8936 - ca
tegorical_accuracy: 0.4744
Epoch 4/2000
22/22 [==============================] - 0s 22ms/step - loss: 1.4651 - ca
tegorical_accuracy: 0.5912
Epoch 5/2000
22/22 [==============================] - 0s 22ms/step - loss: 1.1583 - ca
```

```
tegorical_accuracy: 0.6624
Epoch 6/2000
22/22 [==============================] - 0s 22ms/step - loss: 1.0649 - ca
tegorical_accuracy: 0.6652
Epoch 7/2000
22/22 [==============================] - 0s 22ms/step - loss: 1.1052 - ca
tegorical_accuracy: 0.6538
Epoch 8/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.7860 - ca
tegorical_accuracy: 0.7963
Epoch 9/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.7356 - ca
tegorical_accuracy: 0.7735
Epoch 10/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.7068 - ca
tegorical_accuracy: 0.7963
Epoch 11/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.6499 - ca
tegorical_accuracy: 0.8091
Epoch 12/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.6083 - ca
tegorical_accuracy: 0.8120
Epoch 13/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.6275 - ca
tegorical_accuracy: 0.7934
Epoch 14/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.4829 - ca
tegorical_accuracy: 0.8732
Epoch 15/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.7237 - ca
tegorical_accuracy: 0.7707
Epoch 16/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.7950 - ca
tegorical_accuracy: 0.7194
Epoch 17/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.5116 - ca
tegorical_accuracy: 0.8405
Epoch 18/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.4025 - ca
tegorical_accuracy: 0.8960
Epoch 19/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.3492 - ca
tegorical_accuracy: 0.9003
Epoch 20/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.2972 - ca
tegorical_accuracy: 0.9188
Epoch 21/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.3342 - ca
tegorical_accuracy: 0.8903
Epoch 22/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.6668 - ca
tegorical_accuracy: 0.7764
Epoch 23/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.7885 - ca
tegorical_accuracy: 0.7578
Epoch 24/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.4628 - ca
```

```
tegorical_accuracy: 0.8305
Epoch 25/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.2994 - ca
tegorical_accuracy: 0.9259
Epoch 26/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.2802 - ca
tegorical_accuracy: 0.9117
Epoch 27/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.2383 - ca
tegorical_accuracy: 0.9274
Epoch 28/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.2266 - ca
tegorical_accuracy: 0.9444
Epoch 29/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.3774 - ca
tegorical_accuracy: 0.8946
Epoch 30/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.5645 - ca
tegorical_accuracy: 0.7977
Epoch 31/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.3710 - ca
tegorical_accuracy: 0.8661
Epoch 32/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.2069 - ca
tegorical_accuracy: 0.9530
Epoch 33/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.2067 - ca
tegorical_accuracy: 0.9402
Epoch 34/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.3108 - ca
tegorical_accuracy: 0.8803
Epoch 35/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.3661 - ca
tegorical_accuracy: 0.8704
Epoch 36/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.4370 - ca
tegorical_accuracy: 0.8476
Epoch 37/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.3485 - ca
tegorical_accuracy: 0.8832
Epoch 38/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.2752 - ca
tegorical_accuracy: 0.9017
Epoch 39/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.5014 - ca
tegorical_accuracy: 0.8319
Epoch 40/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.3365 - ca
tegorical_accuracy: 0.8704
Epoch 41/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.2405 - ca
tegorical_accuracy: 0.9231
Epoch 42/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.1795 - ca
tegorical_accuracy: 0.9416
Epoch 43/2000
22/22 [==============================] - 1s 30ms/step - loss: 0.1827 - ca
```

```
tegorical_accuracy: 0.9359
Epoch 44/2000
22/22 [==============================] - 1s 23ms/step - loss: 0.1709 - ca
tegorical_accuracy: 0.9473
Epoch 45/2000
22/22 [==============================] - 0s 23ms/step - loss: 0.3055 - ca
tegorical_accuracy: 0.9103
Epoch 46/2000
22/22 [==============================] - 0s 23ms/step - loss: 0.2152 - ca
tegorical_accuracy: 0.9202
Epoch 47/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.6790 - ca
tegorical_accuracy: 0.7821
Epoch 48/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.4105 - ca
tegorical_accuracy: 0.8575
Epoch 49/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.3134 - ca
tegorical_accuracy: 0.9046
Epoch 50/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.2404 - ca
tegorical_accuracy: 0.9145
Epoch 51/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.1823 - ca
tegorical_accuracy: 0.9672
Epoch 52/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.1706 - ca
tegorical_accuracy: 0.9487
Epoch 53/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.1775 - ca
tegorical_accuracy: 0.9444
Epoch 54/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.3124 - ca
tegorical_accuracy: 0.8818
Epoch 55/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.1590 - ca
tegorical_accuracy: 0.9430
Epoch 56/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.1432 - ca
tegorical_accuracy: 0.9473
Epoch 57/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.2515 - ca
tegorical_accuracy: 0.9060
Epoch 58/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.2222 - ca
tegorical_accuracy: 0.9103
Epoch 59/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.2740 - ca
tegorical_accuracy: 0.8917
Epoch 60/2000
22/22 [==============================] - 1s 23ms/step - loss: 0.2906 - ca
tegorical_accuracy: 0.9046
Epoch 61/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.1543 - ca
tegorical_accuracy: 0.9615
Epoch 62/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.1426 - ca
```

```
tegorical_accuracy: 0.9587
Epoch 63/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.3720 - ca
tegorical_accuracy: 0.8960
Epoch 64/2000
22/22 [==============================] - 0s 21ms/step - loss: 0.5481 - ca
tegorical_accuracy: 0.8006
Epoch 65/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.2268 - ca
tegorical_accuracy: 0.9473
Epoch 66/2000
22/22 [==============================] - 1s 28ms/step - loss: 0.1466 - ca
tegorical_accuracy: 0.9587
Epoch 67/2000
22/22 [==============================] - 0s 23ms/step - loss: 0.1403 - ca
tegorical_accuracy: 0.9644
Epoch 68/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.3549 - ca
tegorical_accuracy: 0.8989
Epoch 69/2000
22/22 [==============================] - 1s 23ms/step - loss: 0.2070 - ca
tegorical_accuracy: 0.9359
Epoch 70/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.1667 - ca
tegorical_accuracy: 0.9387
Epoch 71/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.1130 - ca
tegorical_accuracy: 0.9601
Epoch 72/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.0797 - ca
tegorical_accuracy: 0.9858
Epoch 73/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.1156 - ca
tegorical_accuracy: 0.9630
Epoch 74/2000
22/22 [==============================] - 0s 21ms/step - loss: 0.1827 - ca
tegorical_accuracy: 0.9373
Epoch 75/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.1560 - ca
tegorical_accuracy: 0.9459
Epoch 76/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.1485 - ca
tegorical_accuracy: 0.9573
Epoch 77/2000
22/22 [==============================] - 0s 21ms/step - loss: 0.0878 - ca
tegorical_accuracy: 0.9829
Epoch 78/2000
22/22 [==============================] - 0s 21ms/step - loss: 0.1712 - ca
tegorical_accuracy: 0.9444
Epoch 79/2000
22/22 [==============================] - 0s 21ms/step - loss: 0.3003 - ca
tegorical_accuracy: 0.9103
Epoch 80/2000
22/22 [==============================] - 0s 21ms/step - loss: 0.3389 - ca
tegorical_accuracy: 0.8704
Epoch 81/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.3195 - ca
```

```
tegorical_accuracy: 0.8932
Epoch 82/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.2212 - ca
tegorical_accuracy: 0.9231
Epoch 83/2000
22/22 [==============================] - 1s 23ms/step - loss: 0.1296 - ca
tegorical_accuracy: 0.9573
Epoch 84/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.1337 - ca
tegorical_accuracy: 0.9615
Epoch 85/2000
22/22 [==============================] - 1s 31ms/step - loss: 0.0817 - ca
tegorical_accuracy: 0.9815
Epoch 86/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.0828 - ca
tegorical_accuracy: 0.9815
Epoch 87/2000
22/22 [==============================] - 0s 23ms/step - loss: 0.0721 - ca
tegorical_accuracy: 0.9829
Epoch 88/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.2586 - ca
tegorical_accuracy: 0.9202
Epoch 89/2000
22/22 [==============================] - 1s 23ms/step - loss: 0.7592 - ca
tegorical_accuracy: 0.7877
Epoch 90/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.4573 - ca
tegorical_accuracy: 0.8390
Epoch 91/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.4075 - ca
tegorical_accuracy: 0.8547
Epoch 92/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.2551 - ca
tegorical_accuracy: 0.9174
Epoch 93/2000
22/22 [==============================] - 1s 42ms/step - loss: 0.2921 - ca
tegorical_accuracy: 0.8903
Epoch 94/2000
22/22 [==============================] - 1s 39ms/step - loss: 0.1949 - ca
tegorical_accuracy: 0.9402
Epoch 95/2000
22/22 [==============================] - 1s 42ms/step - loss: 0.1566 - ca
tegorical_accuracy: 0.9516
Epoch 96/2000
22/22 [==============================] - 1s 35ms/step - loss: 0.0983 - ca
tegorical_accuracy: 0.9729
Epoch 97/2000
22/22 [==============================] - 1s 35ms/step - loss: 0.1119 - ca
tegorical_accuracy: 0.9701
Epoch 98/2000
22/22 [==============================] - 1s 29ms/step - loss: 0.1200 - ca
tegorical_accuracy: 0.9672
Epoch 99/2000
22/22 [==============================] - 1s 36ms/step - loss: 0.0861 - ca
tegorical_accuracy: 0.9772
Epoch 100/2000
22/22 [==============================] - 1s 28ms/step - loss: 0.2281 - ca
```

```
tegorical_accuracy: 0.9387
Epoch 101/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.3699 - ca
tegorical_accuracy: 0.8533
Epoch 102/2000
22/22 [==============================] - 1s 32ms/step - loss: 0.1640 - ca
tegorical_accuracy: 0.9416
Epoch 103/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.0807 - ca
tegorical_accuracy: 0.9815
Epoch 104/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.0683 - ca
tegorical_accuracy: 0.9801
Epoch 105/2000
22/22 [==============================] - 1s 29ms/step - loss: 0.0565 - ca
tegorical_accuracy: 0.9829
Epoch 106/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.0632 - ca
tegorical_accuracy: 0.9858
Epoch 107/2000
22/22 [==============================] - 1s 32ms/step - loss: 0.2356 - ca
tegorical_accuracy: 0.9259
Epoch 108/2000
22/22 [==============================] - 1s 40ms/step - loss: 0.3136 - ca
tegorical_accuracy: 0.9174
Epoch 109/2000
22/22 [==============================] - 1s 39ms/step - loss: 0.3634 - ca
tegorical_accuracy: 0.9060
Epoch 110/2000
22/22 [==============================] - 1s 30ms/step - loss: 0.2069 - ca
tegorical_accuracy: 0.9202
Epoch 111/2000
22/22 [==============================] - 1s 41ms/step - loss: 0.0997 - ca
tegorical_accuracy: 0.9687
Epoch 112/2000
22/22 [==============================] - 1s 28ms/step - loss: 0.1181 - ca
tegorical_accuracy: 0.9687
Epoch 113/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.1146 - ca
tegorical_accuracy: 0.9615
Epoch 114/2000
22/22 [==============================] - 1s 30ms/step - loss: 0.0556 - ca
tegorical_accuracy: 0.9843
Epoch 115/2000
22/22 [==============================] - 1s 29ms/step - loss: 0.1389 - ca
tegorical_accuracy: 0.9544
Epoch 116/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.0525 - ca
tegorical_accuracy: 0.9872
Epoch 117/2000
22/22 [==============================] - 1s 29ms/step - loss: 0.0386 - ca
tegorical_accuracy: 0.9943
Epoch 118/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.0609 - ca
tegorical_accuracy: 0.9815
Epoch 119/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.6465 - ca
```

```
tegorical_accuracy: 0.8219
Epoch 120/2000
22/22 [==============================] - 1s 29ms/step - loss: 0.6466 - ca
tegorical_accuracy: 0.8034
Epoch 121/2000
22/22 [==============================] - 1s 28ms/step - loss: 0.3402 - ca
tegorical_accuracy: 0.8846
Epoch 122/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.1994 - ca
tegorical_accuracy: 0.9359
Epoch 123/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.1073 - ca
tegorical_accuracy: 0.9744
Epoch 124/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.0715 - ca
tegorical_accuracy: 0.9815
Epoch 125/2000
22/22 [==============================] - 1s 32ms/step - loss: 0.0896 - ca
tegorical_accuracy: 0.9744
Epoch 126/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.0908 - ca
tegorical_accuracy: 0.9729
Epoch 127/2000
22/22 [==============================] - 1s 28ms/step - loss: 0.2028 - ca
tegorical_accuracy: 0.9416
Epoch 128/2000
22/22 [==============================] - 1s 29ms/step - loss: 0.1477 - ca
tegorical_accuracy: 0.9316
Epoch 129/2000
22/22 [==============================] - 1s 28ms/step - loss: 0.0927 - ca
tegorical_accuracy: 0.9644
Epoch 130/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.1456 - ca
tegorical_accuracy: 0.9544
Epoch 131/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.2059 - ca
tegorical_accuracy: 0.9316
Epoch 132/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.3791 - ca
tegorical_accuracy: 0.8675
Epoch 133/2000
22/22 [==============================] - 1s 31ms/step - loss: 0.2079 - ca
tegorical_accuracy: 0.9231
Epoch 134/2000
22/22 [==============================] - 1s 29ms/step - loss: 0.1445 - ca
tegorical_accuracy: 0.9644
Epoch 135/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.0779 - ca
tegorical_accuracy: 0.9744
Epoch 136/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.0444 - ca
tegorical_accuracy: 0.9929
Epoch 137/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.0395 - ca
tegorical_accuracy: 0.9943
Epoch 138/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.0342 - ca
```

```
tegorical_accuracy: 0.9929
Epoch 139/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.0326 - ca
tegorical_accuracy: 0.9929
Epoch 140/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.0323 - ca
tegorical_accuracy: 0.9915
Epoch 141/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.0705 - ca
tegorical_accuracy: 0.9786
Epoch 142/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.0948 - ca
tegorical_accuracy: 0.9587
Epoch 143/2000
22/22 [==============================] - 1s 23ms/step - loss: 0.0646 - ca
tegorical_accuracy: 0.9872
Epoch 144/2000
22/22 [==============================] - 1s 23ms/step - loss: 0.0488 - ca
tegorical_accuracy: 0.9843
Epoch 145/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.0296 - ca
tegorical_accuracy: 0.9929
Epoch 146/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.0257 - ca
tegorical_accuracy: 0.9957
Epoch 147/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.0237 - ca
tegorical_accuracy: 0.9957
Epoch 148/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.0230 - ca
tegorical_accuracy: 0.9943
Epoch 149/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.0283 - ca
tegorical_accuracy: 0.9915
Epoch 150/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.0248 - ca
tegorical_accuracy: 0.9957
Epoch 151/2000
22/22 [==============================] - 1s 31ms/step - loss: 0.0819 - ca
tegorical_accuracy: 0.9786
Epoch 152/2000
22/22 [==============================] - 1s 23ms/step - loss: 0.0722 - ca
tegorical_accuracy: 0.9715
Epoch 153/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.3112 - ca
tegorical_accuracy: 0.9060
Epoch 154/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.1524 - ca
tegorical_accuracy: 0.9516
Epoch 155/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.0706 - ca
tegorical_accuracy: 0.9815
Epoch 156/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.0336 - ca
tegorical_accuracy: 0.9929
Epoch 157/2000
22/22 [==============================] - 1s 28ms/step - loss: 0.0290 - ca
```

```
tegorical_accuracy: 0.9929
Epoch 158/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.0289 - ca
tegorical_accuracy: 0.9957
Epoch 159/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.0305 - ca
tegorical_accuracy: 0.9915
Epoch 160/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.0362 - ca
tegorical_accuracy: 0.9886
Epoch 161/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.0535 - ca
tegorical_accuracy: 0.9858
Epoch 162/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.0573 - ca
tegorical_accuracy: 0.9829
Epoch 163/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.1328 - ca
tegorical_accuracy: 0.9687
Epoch 164/2000
22/22 [==============================] - 1s 30ms/step - loss: 0.1004 - ca
tegorical_accuracy: 0.9658
Epoch 165/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.0626 - ca
tegorical_accuracy: 0.9801
Epoch 166/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.0295 - ca
tegorical_accuracy: 0.9929
Epoch 167/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.0312 - ca
tegorical_accuracy: 0.9929
Epoch 168/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.0270 - ca
tegorical_accuracy: 0.9915
Epoch 169/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.0272 - ca
tegorical_accuracy: 0.9900
Epoch 170/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.0220 - ca
tegorical_accuracy: 0.9929
Epoch 171/2000
22/22 [==============================] - 1s 23ms/step - loss: 0.0217 - ca
tegorical_accuracy: 0.9957
Epoch 172/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.0182 - ca
tegorical_accuracy: 0.9957
Epoch 173/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.0196 - ca
tegorical_accuracy: 0.9943
Epoch 174/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.0406 - ca
tegorical_accuracy: 0.9858
Epoch 175/2000
22/22 [==============================] - 1s 24ms/step - loss: 1.0961 - ca
tegorical_accuracy: 0.7436
Epoch 176/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.5807 - ca
```

```
tegorical_accuracy: 0.8362
Epoch 177/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.3286 - ca
tegorical_accuracy: 0.8832
Epoch 178/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.2461 - ca
tegorical_accuracy: 0.9259
Epoch 179/2000
22/22 [==============================] - 1s 23ms/step - loss: 0.9087 - ca
tegorical_accuracy: 0.7464
Epoch 180/2000
22/22 [==============================] - 1s 23ms/step - loss: 0.3965 - ca
tegorical_accuracy: 0.8447
Epoch 181/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.2067 - ca
tegorical_accuracy: 0.9231
Epoch 182/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.1523 - ca
tegorical_accuracy: 0.9644
Epoch 183/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.1395 - ca
tegorical_accuracy: 0.9601
Epoch 184/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.1500 - ca
tegorical_accuracy: 0.9473
Epoch 185/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.1458 - ca
tegorical_accuracy: 0.9587
Epoch 186/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.3029 - ca
tegorical_accuracy: 0.9017
Epoch 187/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.4657 - ca
tegorical_accuracy: 0.8419
Epoch 188/2000
22/22 [==============================] - 1s 28ms/step - loss: 0.2460 - ca
tegorical_accuracy: 0.9060
Epoch 189/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.1430 - ca
tegorical_accuracy: 0.9416
Epoch 190/2000
22/22 [==============================] - 1s 43ms/step - loss: 0.0771 - ca
tegorical_accuracy: 0.9872
Epoch 191/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.0558 - ca
tegorical_accuracy: 0.9886
Epoch 192/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.0487 - ca
tegorical_accuracy: 0.9900
Epoch 193/2000
22/22 [==============================] - 1s 29ms/step - loss: 0.0526 - ca
tegorical_accuracy: 0.9858
Epoch 194/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.0461 - ca
tegorical_accuracy: 0.9886
Epoch 195/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.0379 - ca
```

```
tegorical_accuracy: 0.9915
Epoch 196/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.0958 - ca
tegorical_accuracy: 0.9715
Epoch 197/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.0610 - ca
tegorical_accuracy: 0.9801
Epoch 198/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.0508 - ca
tegorical_accuracy: 0.9858
Epoch 199/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.0515 - ca
tegorical_accuracy: 0.9843
Epoch 200/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.0646 - ca
tegorical_accuracy: 0.9815
Epoch 201/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.0837 - ca
tegorical_accuracy: 0.9744
Epoch 202/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.0618 - ca
tegorical_accuracy: 0.9815
Epoch 203/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.0614 - ca
tegorical_accuracy: 0.9829
Epoch 204/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.1870 - ca
tegorical_accuracy: 0.9444
Epoch 205/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.1240 - ca
tegorical_accuracy: 0.9587
Epoch 206/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.0708 - ca
tegorical_accuracy: 0.9715
Epoch 207/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.1016 - ca
tegorical_accuracy: 0.9715
Epoch 208/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.0868 - ca
tegorical_accuracy: 0.9672
Epoch 209/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.0503 - ca
tegorical_accuracy: 0.9858
Epoch 210/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.0486 - ca
tegorical_accuracy: 0.9900
Epoch 211/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.3241 - ca
tegorical_accuracy: 0.9060
Epoch 212/2000
22/22 [==============================] - 0s 23ms/step - loss: 0.4923 - ca
tegorical_accuracy: 0.8519
Epoch 213/2000
22/22 [==============================] - 1s 27ms/step - loss: 1.0960 - ca
tegorical_accuracy: 0.7251
Epoch 214/2000
22/22 [==============================] - 1s 23ms/step - loss: 0.5796 - ca
```

```
tegorical_accuracy: 0.8148
Epoch 215/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.2404 - ca
tegorical_accuracy: 0.9174
Epoch 216/2000
22/22 [==============================] - 1s 23ms/step - loss: 0.1612 - ca
tegorical_accuracy: 0.9544
Epoch 217/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.8294 - ca
tegorical_accuracy: 0.7920
Epoch 218/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.2350 - ca
tegorical_accuracy: 0.9459
Epoch 219/2000
22/22 [==============================] - 1s 31ms/step - loss: 0.1173 - ca
tegorical_accuracy: 0.9644
Epoch 220/2000
22/22 [==============================] - 1s 28ms/step - loss: 0.1083 - ca
tegorical_accuracy: 0.9558
Epoch 221/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.0686 - ca
tegorical_accuracy: 0.9815
Epoch 222/2000
22/22 [==============================] - 1s 29ms/step - loss: 0.0642 - ca
tegorical_accuracy: 0.9858
Epoch 223/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.1727 - ca
tegorical_accuracy: 0.9387
Epoch 224/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.0833 - ca
tegorical_accuracy: 0.9758
Epoch 225/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.0717 - ca
tegorical_accuracy: 0.9815
Epoch 226/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.0431 - ca
tegorical_accuracy: 0.9900
Epoch 227/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.0331 - ca
tegorical_accuracy: 0.9943
Epoch 228/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.0314 - ca
tegorical_accuracy: 0.9915
Epoch 229/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.0293 - ca
tegorical_accuracy: 0.9915
Epoch 230/2000
22/22 [==============================] - 1s 23ms/step - loss: 0.0333 - ca
tegorical_accuracy: 0.9900
Epoch 231/2000
22/22 [==============================] - 1s 28ms/step - loss: 0.0778 - ca
tegorical_accuracy: 0.9744
Epoch 232/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.0541 - ca
tegorical_accuracy: 0.9815
Epoch 233/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.0524 - ca
```

```
tegorical_accuracy: 0.9872
Epoch 234/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.1516 - ca
tegorical_accuracy: 0.9473
Epoch 235/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.1179 - ca
tegorical_accuracy: 0.9630
Epoch 236/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.0878 - ca
tegorical_accuracy: 0.9786
Epoch 237/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.0855 - ca
tegorical_accuracy: 0.9701
Epoch 238/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.0743 - ca
tegorical_accuracy: 0.9772
Epoch 239/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.0898 - ca
tegorical_accuracy: 0.9744
Epoch 240/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.0823 - ca
tegorical_accuracy: 0.9715
Epoch 241/2000
22/22 [==============================] - 1s 28ms/step - loss: 0.0990 - ca
tegorical_accuracy: 0.9558
Epoch 242/2000
22/22 [==============================] - 1s 23ms/step - loss: 0.2316 - ca
tegorical_accuracy: 0.9387
Epoch 243/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.1568 - ca
tegorical_accuracy: 0.9473
Epoch 244/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.0862 - ca
tegorical_accuracy: 0.9729
Epoch 245/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.0632 - ca
tegorical_accuracy: 0.9801
Epoch 246/2000
22/22 [==============================] - 0s 21ms/step - loss: 0.0364 - ca
tegorical_accuracy: 0.9872
Epoch 247/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.0268 - ca
tegorical_accuracy: 0.9929
Epoch 248/2000
22/22 [==============================] - 0s 21ms/step - loss: 0.0186 - ca
tegorical_accuracy: 0.9972
Epoch 249/2000
22/22 [==============================] - 0s 23ms/step - loss: 0.0195 - ca
tegorical_accuracy: 0.9957
Epoch 250/2000
22/22 [==============================] - 1s 23ms/step - loss: 0.0179 - ca
tegorical_accuracy: 0.9943
Epoch 251/2000
22/22 [==============================] - 1s 23ms/step - loss: 0.0181 - ca
tegorical_accuracy: 0.9957
Epoch 252/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.0171 - ca
```

```
tegorical_accuracy: 0.9957
Epoch 253/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.0314 - ca
tegorical_accuracy: 0.9886
Epoch 254/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.0188 - ca
tegorical_accuracy: 0.9929
Epoch 255/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.0134 - ca
tegorical_accuracy: 0.9972
Epoch 256/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.0219 - ca
tegorical_accuracy: 0.9929
Epoch 257/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.0677 - ca
tegorical_accuracy: 0.9829
Epoch 258/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.2724 - ca
tegorical_accuracy: 0.9245
Epoch 259/2000
22/22 [==============================] - 1s 23ms/step - loss: 0.1643 - ca
tegorical_accuracy: 0.9487
Epoch 260/2000
22/22 [==============================] - 1s 29ms/step - loss: 0.2659 - ca
tegorical_accuracy: 0.9060
Epoch 261/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.2799 - ca
tegorical_accuracy: 0.9145
Epoch 262/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.1190 - ca
tegorical_accuracy: 0.9644
Epoch 263/2000
22/22 [==============================] - 1s 29ms/step - loss: 0.0550 - ca
tegorical_accuracy: 0.9872
Epoch 264/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.0386 - ca
tegorical_accuracy: 0.9915
Epoch 265/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.0319 - ca
tegorical_accuracy: 0.9957
Epoch 266/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.0606 - ca
tegorical_accuracy: 0.9815
Epoch 267/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.0415 - ca
tegorical_accuracy: 0.9843
Epoch 268/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.0273 - ca
tegorical_accuracy: 0.9943
Epoch 269/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.0256 - ca
tegorical_accuracy: 0.9915
Epoch 270/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.0219 - ca
tegorical_accuracy: 0.9929
Epoch 271/2000
22/22 [==============================] - 1s 28ms/step - loss: 0.0184 - ca
```

```
tegorical_accuracy: 0.9957
Epoch 272/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.0319 - ca
tegorical_accuracy: 0.9886
Epoch 273/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.0203 - ca
tegorical_accuracy: 0.9900
Epoch 274/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.0178 - ca
tegorical_accuracy: 0.9929
Epoch 275/2000
22/22 [==============================] - 1s 28ms/step - loss: 0.0184 - ca
tegorical_accuracy: 0.9957
Epoch 276/2000
22/22 [==============================] - 1s 28ms/step - loss: 0.0401 - ca
tegorical_accuracy: 0.9858
Epoch 277/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.6837 - ca
tegorical_accuracy: 0.8590
Epoch 278/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.8013 - ca
tegorical_accuracy: 0.7892
Epoch 279/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.3045 - ca
tegorical_accuracy: 0.8946
Epoch 280/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.1796 - ca
tegorical_accuracy: 0.9487
Epoch 281/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.0745 - ca
tegorical_accuracy: 0.9801
Epoch 282/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.0678 - ca
tegorical_accuracy: 0.9758
Epoch 283/2000
22/22 [==============================] - 1s 23ms/step - loss: 0.1260 - ca
tegorical_accuracy: 0.9544
Epoch 284/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.0778 - ca
tegorical_accuracy: 0.9715
Epoch 285/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.0702 - ca
tegorical_accuracy: 0.9758
Epoch 286/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.0485 - ca
tegorical_accuracy: 0.9843
Epoch 287/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.0576 - ca
tegorical_accuracy: 0.9872
Epoch 288/2000
22/22 [==============================] - 1s 31ms/step - loss: 0.3069 - ca
tegorical_accuracy: 0.9131
Epoch 289/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.1254 - ca
tegorical_accuracy: 0.9658
Epoch 290/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.0575 - ca
```

```
tegorical_accuracy: 0.9843
Epoch 291/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.0323 - ca
tegorical_accuracy: 0.9886
Epoch 292/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.0232 - ca
tegorical_accuracy: 0.9972
Epoch 293/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.0188 - ca
tegorical_accuracy: 0.9986
Epoch 294/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.0179 - ca
tegorical_accuracy: 0.9943
Epoch 295/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.0164 - ca
tegorical_accuracy: 0.9972
Epoch 296/2000
22/22 [==============================] - 1s 23ms/step - loss: 0.0303 - ca
tegorical_accuracy: 0.9900
Epoch 297/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.0373 - ca
tegorical_accuracy: 0.9858
Epoch 298/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.0444 - ca
tegorical_accuracy: 0.9829
Epoch 299/2000
22/22 [==============================] - 1s 29ms/step - loss: 0.0226 - ca
tegorical_accuracy: 0.9929
Epoch 300/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.0170 - ca
tegorical_accuracy: 0.9957
Epoch 301/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.0158 - ca
tegorical_accuracy: 0.9943
Epoch 302/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.0251 - ca
tegorical_accuracy: 0.9915
Epoch 303/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.0331 - ca
tegorical_accuracy: 0.9886
Epoch 304/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.0793 - ca
tegorical_accuracy: 0.9815
Epoch 305/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.1508 - ca
tegorical_accuracy: 0.9530
Epoch 306/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.2082 - ca
tegorical_accuracy: 0.9402
Epoch 307/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.0762 - ca
tegorical_accuracy: 0.9801
Epoch 308/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.0590 - ca
tegorical_accuracy: 0.9829
Epoch 309/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.0342 - ca
```

```
tegorical_accuracy: 0.9915
Epoch 310/2000
22/22 [==============================] - 1s 29ms/step - loss: 0.1310 - ca
tegorical_accuracy: 0.9601
Epoch 311/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.1056 - ca
tegorical_accuracy: 0.9544
Epoch 312/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.0577 - ca
tegorical_accuracy: 0.9843
Epoch 313/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.1045 - ca
tegorical_accuracy: 0.9658
Epoch 314/2000
22/22 [==============================] - 1s 31ms/step - loss: 0.2786 - ca
tegorical_accuracy: 0.9060
Epoch 315/2000
22/22 [==============================] - 1s 29ms/step - loss: 0.1552 - ca
tegorical_accuracy: 0.9473
Epoch 316/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.0611 - ca
tegorical_accuracy: 0.9801
Epoch 317/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.0552 - ca
tegorical_accuracy: 0.9815
Epoch 318/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.0251 - ca
tegorical_accuracy: 0.9929
Epoch 319/2000
22/22 [==============================] - 1s 30ms/step - loss: 0.0197 - ca
tegorical_accuracy: 0.9943
Epoch 320/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.0217 - ca
tegorical_accuracy: 0.9915
Epoch 321/2000
22/22 [==============================] - 1s 29ms/step - loss: 0.0265 - ca
tegorical_accuracy: 0.9929
Epoch 322/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.0176 - ca
tegorical_accuracy: 0.9957
Epoch 323/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.0149 - ca
tegorical_accuracy: 0.9957
Epoch 324/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.0130 - ca
tegorical_accuracy: 0.9972
Epoch 325/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.0118 - ca
tegorical_accuracy: 0.9986
Epoch 326/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.0554 - ca
tegorical_accuracy: 0.9815
Epoch 327/2000
22/22 [==============================] - 1s 28ms/step - loss: 0.0833 - ca
tegorical_accuracy: 0.9729
Epoch 328/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.0386 - ca
```

```
tegorical_accuracy: 0.9900
Epoch 329/2000
22/22 [==============================] - 1s 29ms/step - loss: 0.0382 - ca
tegorical_accuracy: 0.9915
Epoch 330/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.0329 - ca
tegorical_accuracy: 0.9872
Epoch 331/2000
22/22 [==============================] - 1s 29ms/step - loss: 0.2460 - ca
tegorical_accuracy: 0.9245
Epoch 332/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.1560 - ca
tegorical_accuracy: 0.9459
Epoch 333/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.0881 - ca
tegorical_accuracy: 0.9744
Epoch 334/2000
22/22 [==============================] - 1s 29ms/step - loss: 0.0289 - ca
tegorical_accuracy: 0.9943
Epoch 335/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.0224 - ca
tegorical_accuracy: 0.9972
Epoch 336/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.0157 - ca
tegorical_accuracy: 0.9957
Epoch 337/2000
22/22 [==============================] - 1s 28ms/step - loss: 0.0141 - ca
tegorical_accuracy: 0.9957
Epoch 338/2000
22/22 [==============================] - 1s 28ms/step - loss: 0.0114 - ca
tegorical_accuracy: 0.9986
Epoch 339/2000
22/22 [==============================] - 1s 28ms/step - loss: 0.0119 - ca
tegorical_accuracy: 0.9986
Epoch 340/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.0082 - ca
tegorical_accuracy: 1.0000
Epoch 341/2000
22/22 [==============================] - 1s 30ms/step - loss: 0.0067 - ca
tegorical_accuracy: 1.0000
Epoch 342/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.0760 - ca
tegorical_accuracy: 0.9786
Epoch 343/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.3753 - ca
tegorical_accuracy: 0.9160
Epoch 344/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.3477 - ca
tegorical_accuracy: 0.8832
Epoch 345/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.0927 - ca
tegorical_accuracy: 0.9672
Epoch 346/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.0506 - ca
tegorical_accuracy: 0.9858
Epoch 347/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.2746 - ca
```

```
tegorical_accuracy: 0.9316
Epoch 348/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.1145 - ca
tegorical_accuracy: 0.9729
Epoch 349/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.0812 - ca
tegorical_accuracy: 0.9715
Epoch 350/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.0326 - ca
tegorical_accuracy: 0.9929
Epoch 351/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.0368 - ca
tegorical_accuracy: 0.9843
Epoch 352/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.0454 - ca
tegorical_accuracy: 0.9900
Epoch 353/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.0466 - ca
tegorical_accuracy: 0.9843
Epoch 354/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.0257 - ca
tegorical_accuracy: 0.9972
Epoch 355/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.0273 - ca
tegorical_accuracy: 0.9929
Epoch 356/2000
22/22 [==============================] - 1s 23ms/step - loss: 0.0304 - ca
tegorical_accuracy: 0.9886
Epoch 357/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.0159 - ca
tegorical_accuracy: 0.9957
Epoch 358/2000
22/22 [==============================] - 1s 23ms/step - loss: 0.0160 - ca
tegorical_accuracy: 0.9972
Epoch 359/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.0143 - ca
tegorical_accuracy: 0.9972
Epoch 360/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.0147 - ca
tegorical_accuracy: 0.9972
Epoch 361/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.0129 - ca
tegorical_accuracy: 0.9957
Epoch 362/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.0176 - ca
tegorical_accuracy: 0.9943
Epoch 363/2000
22/22 [==============================] - 1s 23ms/step - loss: 0.0115 - ca
tegorical_accuracy: 0.9972
Epoch 364/2000
22/22 [==============================] - 1s 29ms/step - loss: 0.0106 - ca
tegorical_accuracy: 0.9986
Epoch 365/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.0129 - ca
tegorical_accuracy: 0.9972
Epoch 366/2000
22/22 [==============================] - 1s 28ms/step - loss: 0.0089 - ca
```

```
tegorical_accuracy: 0.9972
Epoch 367/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.0087 - ca
tegorical_accuracy: 0.9986
Epoch 368/2000
22/22 [==============================] - 1s 28ms/step - loss: 0.0072 - ca
tegorical_accuracy: 0.9986
Epoch 369/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.0230 - ca
tegorical_accuracy: 0.9929
Epoch 370/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.1423 - ca
tegorical_accuracy: 0.9587
Epoch 371/2000
22/22 [==============================] - 1s 23ms/step - loss: 0.3403 - ca
tegorical_accuracy: 0.8960
Epoch 372/2000
22/22 [==============================] - 1s 23ms/step - loss: 0.5000 - ca
tegorical_accuracy: 0.8447
Epoch 373/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.1401 - ca
tegorical_accuracy: 0.9672
Epoch 374/2000
22/22 [==============================] - 1s 25ms/step - loss: 0.0678 - ca
tegorical_accuracy: 0.9815
Epoch 375/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.0892 - ca
tegorical_accuracy: 0.9843
Epoch 376/2000
22/22 [==============================] - 1s 23ms/step - loss: 0.0505 - ca
tegorical_accuracy: 0.9886
Epoch 377/2000
22/22 [==============================] - 1s 28ms/step - loss: 0.0675 - ca
tegorical_accuracy: 0.9829
Epoch 378/2000
22/22 [==============================] - 1s 28ms/step - loss: 0.0388 - ca
tegorical_accuracy: 0.9915
Epoch 379/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.0190 - ca
tegorical_accuracy: 0.9957
Epoch 380/2000
22/22 [==============================] - 1s 26ms/step - loss: 0.0159 - ca
tegorical_accuracy: 0.9972
Epoch 381/2000
22/22 [==============================] - 1s 27ms/step - loss: 0.0160 - ca
tegorical_accuracy: 0.9972
Epoch 382/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.0112 - ca
tegorical_accuracy: 1.0000
Epoch 383/2000
22/22 [==============================] - 1s 29ms/step - loss: 0.0108 - ca
tegorical_accuracy: 0.9986
Epoch 384/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.0148 - ca
tegorical_accuracy: 0.9957
Epoch 385/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.0106 - ca
```

```
tegorical_accuracy: 0.9986
Epoch 386/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.0104 - ca
tegorical_accuracy: 0.9986
Epoch 387/2000
22/22 [==============================] - 1s 23ms/step - loss: 0.0117 - ca
tegorical_accuracy: 0.9986
Epoch 388/2000
22/22 [==============================] - 1s 23ms/step - loss: 0.0167 - ca
tegorical_accuracy: 0.9957
Epoch 389/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.0129 - ca
tegorical_accuracy: 0.9972
Epoch 390/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.0077 - ca
tegorical_accuracy: 1.0000
Epoch 391/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.0136 - ca
tegorical_accuracy: 0.9957
Epoch 392/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.0097 - ca
tegorical_accuracy: 0.9986
Epoch 393/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.0064 - ca
tegorical_accuracy: 1.0000
Epoch 394/2000
22/22 [==============================] - 1s 24ms/step - loss: 0.0050 - ca
tegorical_accuracy: 1.0000
Epoch 395/2000
22/22 [==============================] - 1s 23ms/step - loss: 0.0045 - ca
tegorical_accuracy: 1.0000
Epoch 396/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.0043 - ca
tegorical_accuracy: 1.0000
Epoch 397/2000
22/22 [==============================] - 1s 23ms/step - loss: 0.0041 - ca
tegorical_accuracy: 1.0000
Epoch 398/2000
22/22 [==============================] - 1s 23ms/step - loss: 0.0040 - ca
tegorical_accuracy: 1.0000
Epoch 399/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.0039 - ca
tegorical_accuracy: 1.0000
Epoch 400/2000
22/22 [==============================] - 0s 22ms/step - loss: 0.0038 - ca
tegorical_accuracy: 1.0000
Epoch 401/2000
 1/22 [>.............................] - ETA: 0s - loss: 0.0043 - categor
ical_accuracy: 1.0000
```

```
-------------------------------------------------------------------------
--
KeyboardInterrupt                         Traceback (most recent call las
t)
Cell In[47], line 2
      1 # training model.
----> 2 model.fit(X_train, Y_train, epochs=2000, callbacks=[tb_callback])
```

```
File /Applications/miniconda3/lib/python3.10/site-packages/keras/src/util
s/traceback_utils.py:65, in filter_traceback.<locals>.error_handler(*args
, **kwargs)
     63 filtered_tb = None
     64 try:
---> 65     return fn(*args, **kwargs)
     66 except Exception as e:
     67     filtered_tb = _process_traceback_frames(e.__traceback__)

File /Applications/miniconda3/lib/python3.10/site-packages/keras/src/engi
ne/training.py:1742, in Model.fit(self, x, y, batch_size, epochs, verbose
, callbacks, validation_split, validation_data, shuffle, class_weight, sa
mple_weight, initial_epoch, steps_per_epoch, validation_steps, validation
_batch_size, validation_freq, max_queue_size, workers, use_multiprocessin
g)
   1734 with tf.profiler.experimental.Trace(
   1735     "train",
   1736     epoch_num=epoch,
   (...)
   1739     _r=1,
   1740 ):
   1741     callbacks.on_train_batch_begin(step)
-> 1742     tmp_logs = self.train_function(iterator)
   1743     if data_handler.should_sync:
   1744         context.async_wait()

File /Applications/miniconda3/lib/python3.10/site-packages/tensorflow/pyt
hon/util/traceback_utils.py:150, in filter_traceback.<locals>.error_handl
er(*args, **kwargs)
    148 filtered_tb = None
    149 try:
--> 150   return fn(*args, **kwargs)
    151 except Exception as e:
    152   filtered_tb = _process_traceback_frames(e.__traceback__)

File /Applications/miniconda3/lib/python3.10/site-packages/tensorflow/pyt
hon/eager/polymorphic_function/polymorphic_function.py:820, in Function._
_call__(self, *args, **kwds)
    817 compiler = "xla" if self._jit_compile else "nonXla"
    819 with OptionalXlaContext(self._jit_compile):
--> 820   result = self._call(*args, **kwds)
    822 new_tracing_count = self.experimental_get_tracing_count()
    823 without_tracing = (tracing_count == new_tracing_count)

File /Applications/miniconda3/lib/python3.10/site-packages/tensorflow/pyt
hon/eager/polymorphic_function/polymorphic_function.py:848, in Function._
call(self, *args, **kwds)
    845   self._lock.release()
    846   # In this case we have created variables on the first call, so
we run the
    847   # defunned version which is guaranteed to never create variable
s.
--> 848   return self._no_variable_creation_fn(*args, **kwds)  # pylint:
disable=not-callable
    849 elif self._variable_creation_fn is not None:
    850   # Release the lock early so that multiple threads can perform t
he call
```

```
    851     # in parallel.
    852     self._lock.release()

File /Applications/miniconda3/lib/python3.10/site-packages/tensorflow/pyt
hon/eager/polymorphic_function/tracing_compiler.py:132, in TracingCompile
r.__call__(self, *args, **kwargs)
    128 with self._lock:
    129   (concrete_function, filtered_flat_args) = self._maybe_define_fu
nction(
    130       args, kwargs
    131   )
--> 132 return concrete_function._call_flat(
    133     filtered_flat_args, captured_inputs=concrete_function.capture
d_inputs
    134 )

File /Applications/miniconda3/lib/python3.10/site-packages/tensorflow/pyt
hon/eager/polymorphic_function/concrete_function.py:1368, in ConcreteFunc
tion._call_flat(self, args, captured_inputs)
   1364 possible_gradient_type = gradients_util.PossibleTapeGradientTypes
(args)
   1365 if (possible_gradient_type == gradients_util.POSSIBLE_GRADIENT_TY
PES_NONE
   1366     and executing_eagerly):
   1367   # No tape is watching; skip to running the function.
-> 1368   return self._build_call_outputs(self._inference_function(*args)
)
   1369 forward_backward = self._select_forward_and_backward_functions(
   1370     args,
   1371     possible_gradient_type,
   1372     executing_eagerly)
   1373 forward_function, args_with_tangents = forward_backward.forward()

File /Applications/miniconda3/lib/python3.10/site-packages/tensorflow/pyt
hon/eager/polymorphic_function/atomic_function.py:222, in AtomicFunction.
__call__(self, *args)
    220 with record.stop_recording():
    221   if self._bound_context.executing_eagerly():
--> 222     outputs = self._bound_context.call_function(
    223         self.name,
    224         list(args),
    225         len(self.function_type.flat_outputs),
    226     )
    227   else:
    228     outputs = make_call_op_in_graph(
    229         self,
    230         list(args),
    231         self._bound_context.function_call_options.as_attrs(),
    232     )

File /Applications/miniconda3/lib/python3.10/site-packages/tensorflow/pyt
hon/eager/context.py:1479, in Context.call_function(self, name, tensor_in
puts, num_outputs)
   1477 cancellation_context = cancellation.context()
   1478 if cancellation_context is None:
-> 1479   outputs = execute.execute(
   1480       name.decode("utf-8"),
```

```
   1481         num_outputs=num_outputs,
   1482         inputs=tensor_inputs,
   1483         attrs=attrs,
   1484         ctx=self,
   1485     )
   1486 else:
   1487   outputs = execute.execute_with_cancellation(
   1488       name.decode("utf-8"),
   1489       num_outputs=num_outputs,
   (...)
   1493       cancellation_manager=cancellation_context,
   1494   )

File /Applications/miniconda3/lib/python3.10/site-packages/tensorflow/pyt
hon/eager/execute.py:53, in quick_execute(op_name, num_outputs, inputs, a
ttrs, ctx, name)
     51 try:
     52   ctx.ensure_initialized()
---> 53   tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name, o
p_name,
     54                                       inputs, attrs, num_outputs)
     55 except core._NotOkStatusException as e:
     56   if name is not None:

KeyboardInterrupt:
```

save model

In [48]:

```python
# saving the model
model.save('action.h5')
```

```
/Applications/miniconda3/lib/python3.10/site-packages/keras/src/engine/tr
aining.py:3000: UserWarning: You are saving your model as an HDF5 file vi
a `model.save()`. This file format is considered legacy. We recommend usi
ng instead the native Keras format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
```

In [49]:

```python
# loading the saved model
model.load_weights('action.h5')
```

4. Evaluation and Testing

Make predictions

In [50]:

```python
# storing results inside 'res' variable
res = model.predict(X_test)
```

```
3/3 [==============================] - 1s 29ms/step
```

In [51]:

```python
alphabets[np.argmax(res[0])]
```

Out[51]:

```
'B'
```

In [52]:
```
alphabets[np.argmax(Y_test[0])]
```
Out[52]: 'B'

In [53]:
```
alphabets[np.argmax(res[1])]
```
Out[53]: 'E'

In [54]:
```
alphabets[np.argmax(Y_test[1])]
```
Out[54]: 'E'

In [55]:
```
alphabets[np.argmax(res[2])]
```
Out[55]: 'D'

In [56]:
```
alphabets[np.argmax(Y_test[2])]
```
Out[56]: 'D'

In [57]:
```
alphabets[np.argmax(res[3])]
```
Out[57]: 'J'

In [58]:
```
alphabets[np.argmax(Y_test[3])]
```
Out[58]: 'J'

Test data

In [59]:
```
# making predictions on test data using trained LSTM model
ypredict = model.predict(X_test)
```
```
3/3 [==============================] - 0s 8ms/step
```

In [60]:
```
# extracting true labels from 'Y_test'
ytrue = np.argmax(Y_test, axis=1).tolist()

# extracting predicted labels from 'yhat'
ypredict = np.argmax(ypredict, axis=1).tolist()
```

Evaluate using confusion matrix

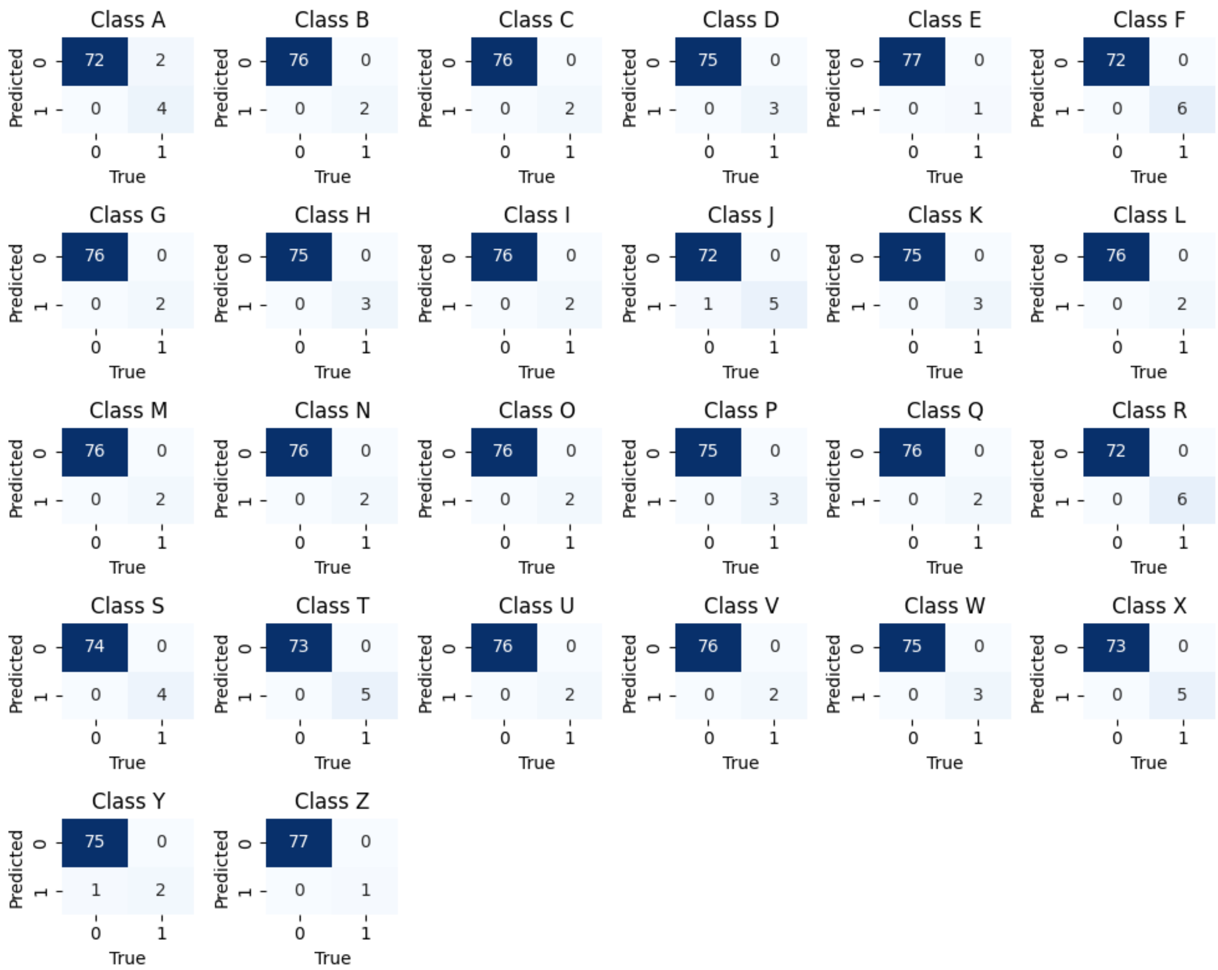
In [61]:
```
from sklearn.metrics import multilabel_confusion_matrix, accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_score, recall_score, f1_score

import seaborn as sns
```

In [62]:
```
# calculating multilabel confusion matrix
multilabel_confusion_matrix(ytrue, ypredict)
```

Out[62]:
```
array([[[72,  2],
        [ 0,  4]],

       [[76,  0],
        [ 0,  2]],

       [[76,  0],
        [ 0,  2]],

       [[75,  0],
        [ 0,  3]],

       [[77,  0],
        [ 0,  1]],

       [[72,  0],
        [ 0,  6]],

       [[76,  0],
        [ 0,  2]],

       [[75,  0],
        [ 0,  3]],

       [[76,  0],
        [ 0,  2]],

       [[72,  0],
        [ 1,  5]],

       [[75,  0],
        [ 0,  3]],

       [[76,  0],
        [ 0,  2]],

       [[76,  0],
        [ 0,  2]],

       [[76,  0],
        [ 0,  2]],

       [[76,  0],
        [ 0,  2]],

       [[75,  0],
        [ 0,  3]],

       [[76,  0],
        [ 0,  2]],

       [[72,  0],
        [ 0,  6]],

       [[74,  0],
        [ 0,  4]],
```

```
       [[73,  0],
        [ 0,  5]],

       [[76,  0],
        [ 0,  2]],

       [[76,  0],
        [ 0,  2]],

       [[75,  0],
        [ 0,  3]],

       [[73,  0],
        [ 0,  5]],

       [[75,  0],
        [ 1,  2]],

       [[77,  0],
        [ 0,  1]]])
```

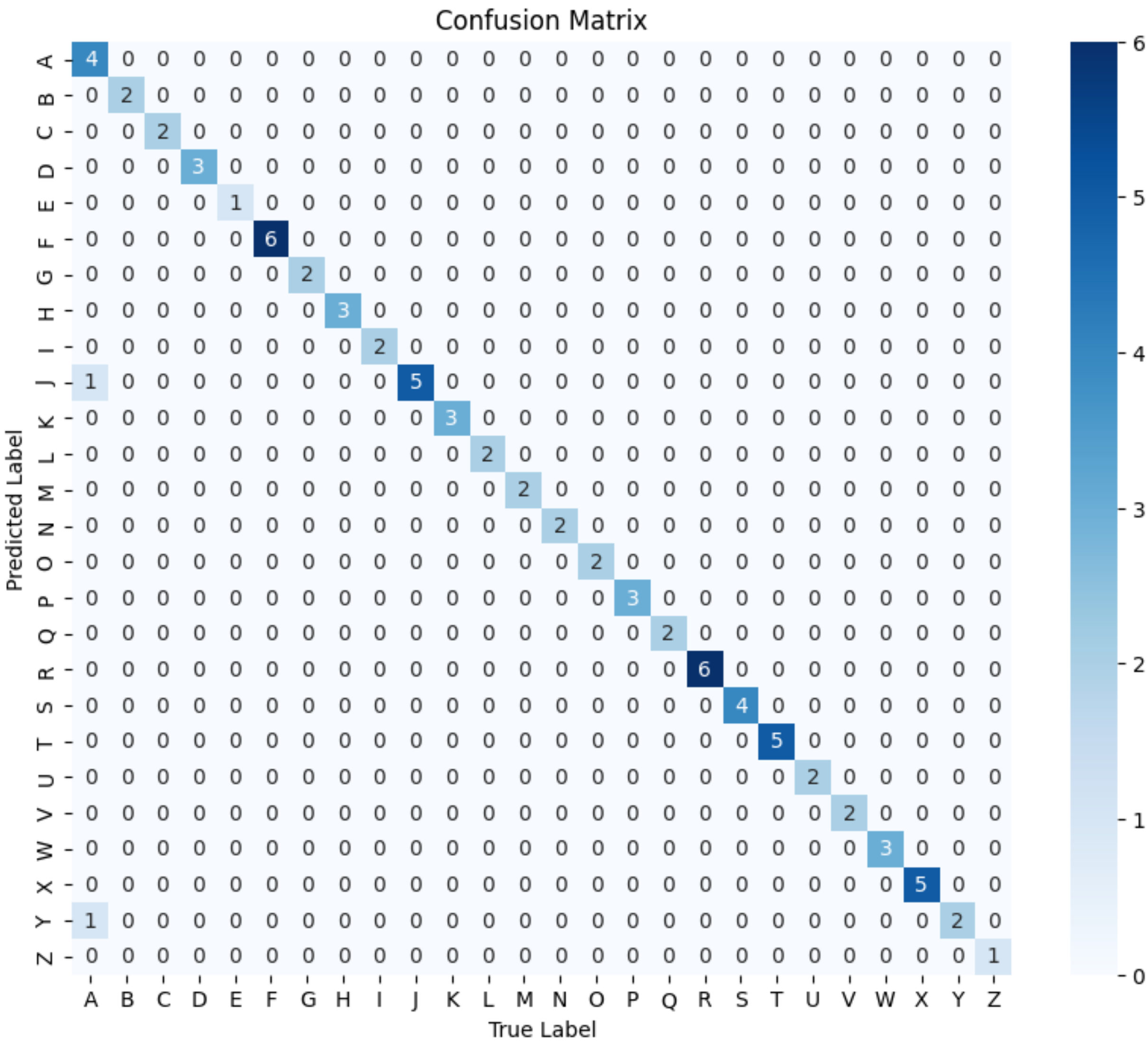
In [63]:
```
# storing confusion matrix in variable
multilabel_cm = multilabel_confusion_matrix(ytrue, ypredict)

# Define the class labels
class_labels = ['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']

# Create a function to plot a confusion matrix heatmap
def plot_confusion_matrix(conf_matrix, class_labels):
    plt.figure(figsize=(10, 8))
    for i, cm in enumerate(conf_matrix):
        plt.subplot(5, 6, i + 1)
        sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False)
        plt.title(f"Class {class_labels[i]}")
        plt.xlabel("True")
        plt.ylabel("Predicted")
    plt.tight_layout()
    plt.show()

# Plot the confusion matrix heatmap
plot_confusion_matrix(multilabel_cm, class_labels)
```



In [64]:
```
confusion_matrix(ytrue, ypredict)
```

```
Out[64]: array([[4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                 0, 0, 0, 0],
                [0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                 0, 0, 0, 0],
                [0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                 0, 0, 0, 0],
                [0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                 0, 0, 0, 0],
                [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                 0, 0, 0, 0],
                [0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                 0, 0, 0, 0],
                [0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                 0, 0, 0, 0],
                [0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                 0, 0, 0, 0],
                [0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                 0, 0, 0, 0],
                [1, 0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                 0, 0, 0, 0],
                [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                 0, 0, 0, 0],
                [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                 0, 0, 0, 0],
                [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                 0, 0, 0, 0],
                [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0,
                 0, 0, 0, 0],
                [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0,
                 0, 0, 0, 0],
                [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0,
                 0, 0, 0, 0],
                [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0,
                 0, 0, 0, 0],
                [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0,
                 0, 0, 0, 0],
                [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0,
                 0, 0, 0, 0],
                [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 0,
                 0, 0, 0, 0],
                [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0,
                 0, 0, 0, 0],
                [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2,
                 0, 0, 0, 0],
                [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                 3, 0, 0, 0],
                [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                 0, 5, 0, 0],
                [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                 0, 0, 2, 0],
                [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                 0, 0, 0, 1]])
```

In [65]:
```
# variable storing confusion matrix
cm = confusion_matrix(ytrue, ypredict)

# Visualize the confusion matrix using a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
        xticklabels=['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'],
        yticklabels=['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'])
plt.xlabel('True Label')
plt.ylabel('Predicted Label')
plt.title('Confusion Matrix')
plt.show()
```

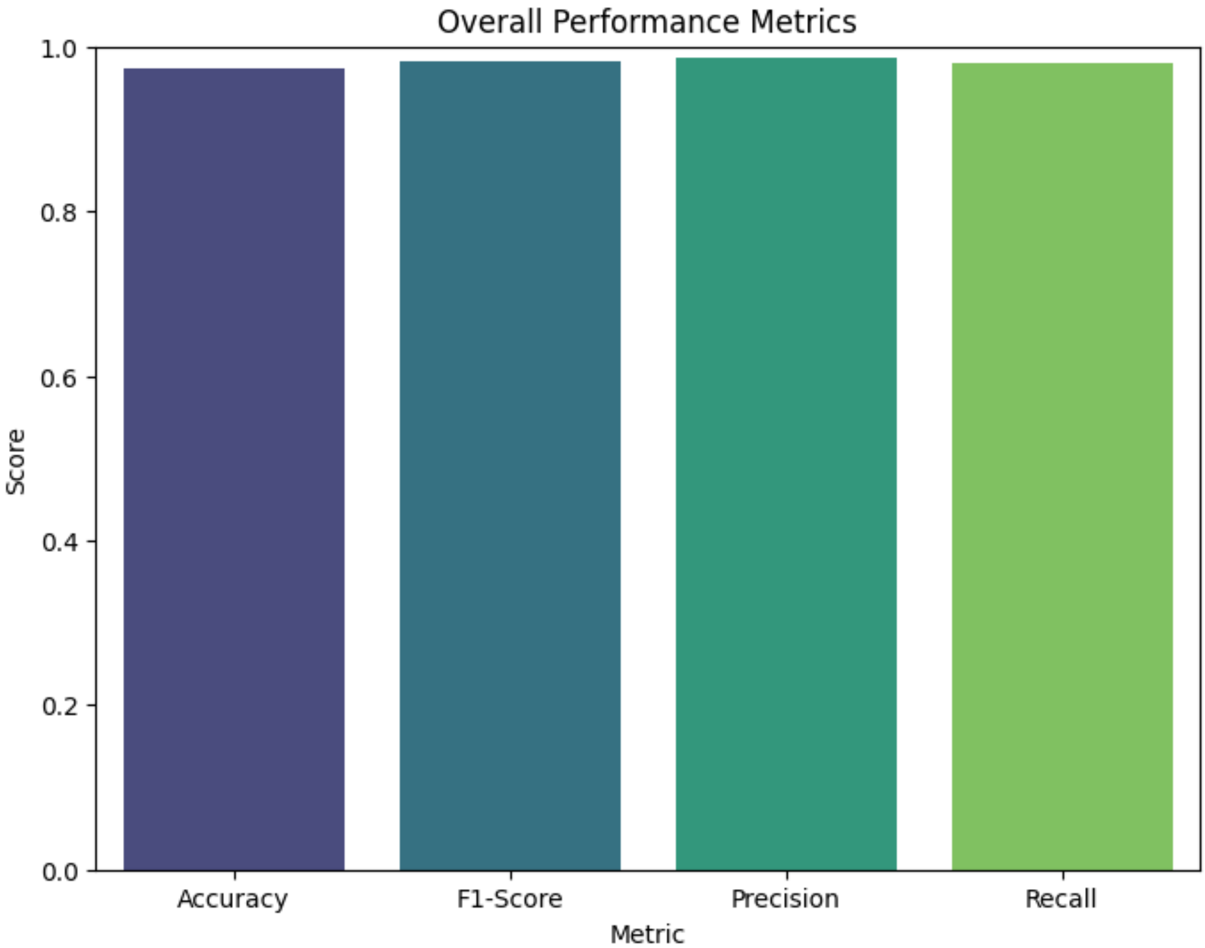


Evauate using evaluation metrics (accuracy, f1-score, precision, recall)

In [66]:
```
# calculating the accuracy, F1, precision, recall for all classes
accuracy = accuracy_score(ytrue, ypredict)
f1 = f1_score(ytrue, ypredict, average='macro')
precision = precision_score(ytrue, ypredict, average='macro')
recall = recall_score(ytrue, ypredict, average='macro')

print(f"accuracy: {accuracy}")
print(f"F1-score: {f1}")
print(f"precision-score: {precision}")
print(f"recall-score: {recall}")
```

```
accuracy: 0.9743589743589743
F1-score: 0.9811188811188812
precision-score: 0.9871794871794871
recall-score: 0.980769230769231
```

In [67]:
```
import pandas as pd

# Creating a DataFrame to store the metrics
metrics_df = pd.DataFrame({
    'Metric': ['Accuracy', 'F1-Score', 'Precision', 'Recall'],
    'Score': [accuracy, f1, precision, recall]
})

# Plotting the metrics using Seaborn
plt.figure(figsize=(8, 6))
sns.barplot(x='Metric', y='Score', data=metrics_df, palette='viridis')
plt.title('Overall Performance Metrics')
plt.ylim(0, 1)
plt.show()
```

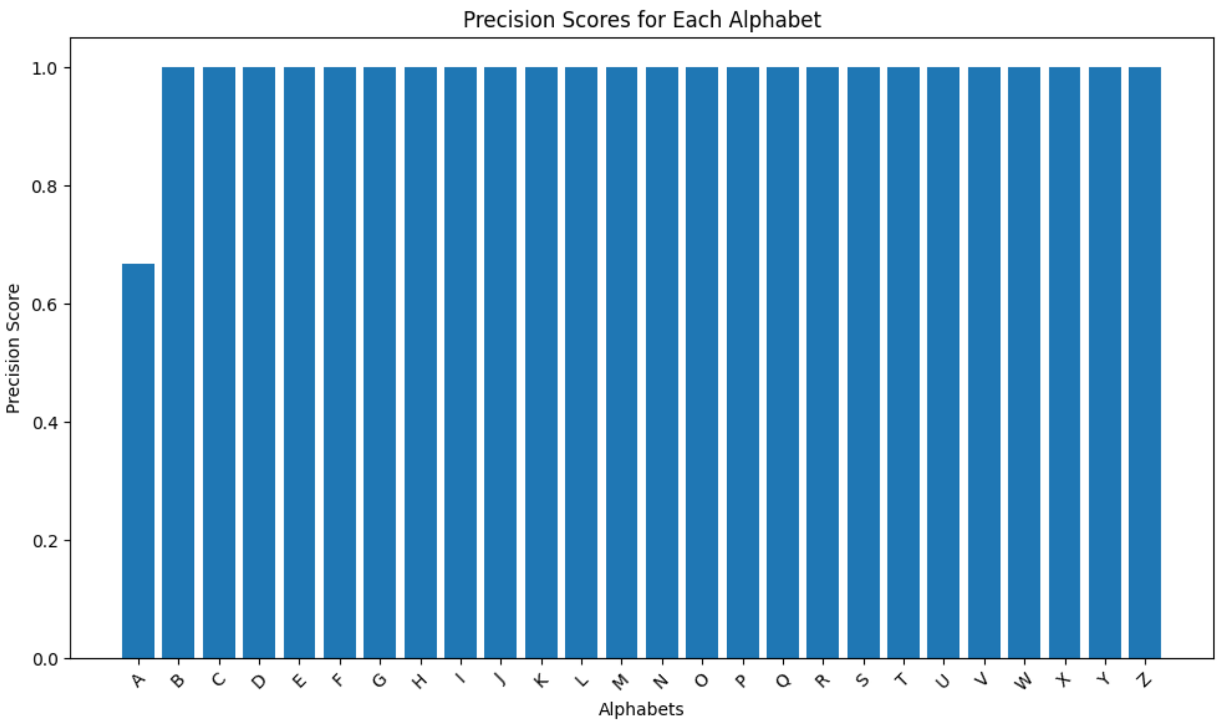


In [68]:
```
# Calculating precision score for each class (alphabet)
precision_scores = precision_score(ytrue, ypredict, average=None)

# Printing precision scores for each class
for idx, alphabet in enumerate(alphabets):
    print(f"Precision for Alphabet {alphabet}:
             {precision_scores[idx]:.4f}")
```

```
Precision for Alphabet A: 0.6667
Precision for Alphabet B: 1.0000
Precision for Alphabet C: 1.0000
Precision for Alphabet D: 1.0000
Precision for Alphabet E: 1.0000
Precision for Alphabet F: 1.0000
Precision for Alphabet G: 1.0000
Precision for Alphabet H: 1.0000
Precision for Alphabet I: 1.0000
Precision for Alphabet J: 1.0000
Precision for Alphabet K: 1.0000
Precision for Alphabet L: 1.0000
Precision for Alphabet M: 1.0000
Precision for Alphabet N: 1.0000
Precision for Alphabet O: 1.0000
Precision for Alphabet P: 1.0000
Precision for Alphabet Q: 1.0000
Precision for Alphabet R: 1.0000
Precision for Alphabet S: 1.0000
Precision for Alphabet T: 1.0000
Precision for Alphabet U: 1.0000
Precision for Alphabet V: 1.0000
Precision for Alphabet W: 1.0000
Precision for Alphabet X: 1.0000
Precision for Alphabet Y: 1.0000
Precision for Alphabet Z: 1.0000
```

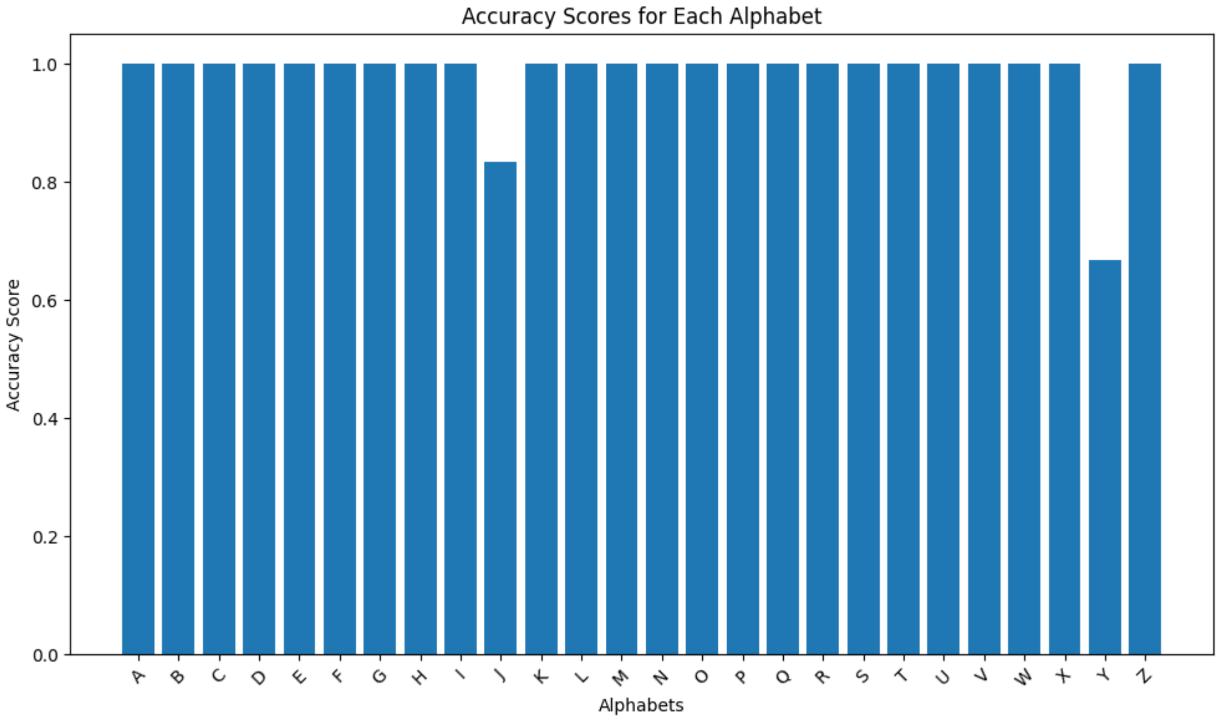
In [73]:

```python
import matplotlib.pyplot as plt
# Plotting the bar graph
plt.figure(figsize=(10, 6))
plt.bar(alphabets, precision_scores)
plt.xlabel('Alphabets')
plt.ylabel('Precision Score')
plt.title('Precision Scores for Each Alphabet')
plt.xticks(rotation=45)
plt.tight_layout()

# Display the plot
plt.show()
```



In [80]:
```
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score

alphabet_accuracy = {}

for idx, alphabet in enumerate(alphabets):
    indices = [i for i, y in enumerate(ytrue) if y == idx]
    class_ytrue = [ytrue[i] for i in indices]
    class_ypredict = [ypredict[i] for i in indices]
    accuracy = accuracy_score(class_ytrue, class_ypredict)
    alphabet_accuracy[alphabet] = accuracy

# Printing accuracy for each class
for alphabet, accuracy in alphabet_accuracy.items():
    print(f"Accuracy for Alphabet {alphabet}: {accuracy:.4f}")

# Plotting the bar graph
plt.figure(figsize=(10, 6))
plt.bar(alphabet_accuracy.keys(), alphabet_accuracy.values())
plt.xlabel('Alphabets')
plt.ylabel('Accuracy Score')
plt.title('Accuracy Scores for Each Alphabet')
plt.xticks(rotation=45)
plt.tight_layout()

# Display the plot
plt.show()
```

```
Accuracy for Alphabet A: 1.0000
Accuracy for Alphabet B: 1.0000
Accuracy for Alphabet C: 1.0000
Accuracy for Alphabet D: 1.0000
Accuracy for Alphabet E: 1.0000
Accuracy for Alphabet F: 1.0000
Accuracy for Alphabet G: 1.0000
Accuracy for Alphabet H: 1.0000
Accuracy for Alphabet I: 1.0000
Accuracy for Alphabet J: 0.8333
Accuracy for Alphabet K: 1.0000
Accuracy for Alphabet L: 1.0000
Accuracy for Alphabet M: 1.0000
Accuracy for Alphabet N: 1.0000
Accuracy for Alphabet O: 1.0000
Accuracy for Alphabet P: 1.0000
Accuracy for Alphabet Q: 1.0000
Accuracy for Alphabet R: 1.0000
Accuracy for Alphabet S: 1.0000
Accuracy for Alphabet T: 1.0000
Accuracy for Alphabet U: 1.0000
Accuracy for Alphabet V: 1.0000
Accuracy for Alphabet W: 1.0000
Accuracy for Alphabet X: 1.0000
Accuracy for Alphabet Y: 0.6667
Accuracy for Alphabet Z: 1.0000
```

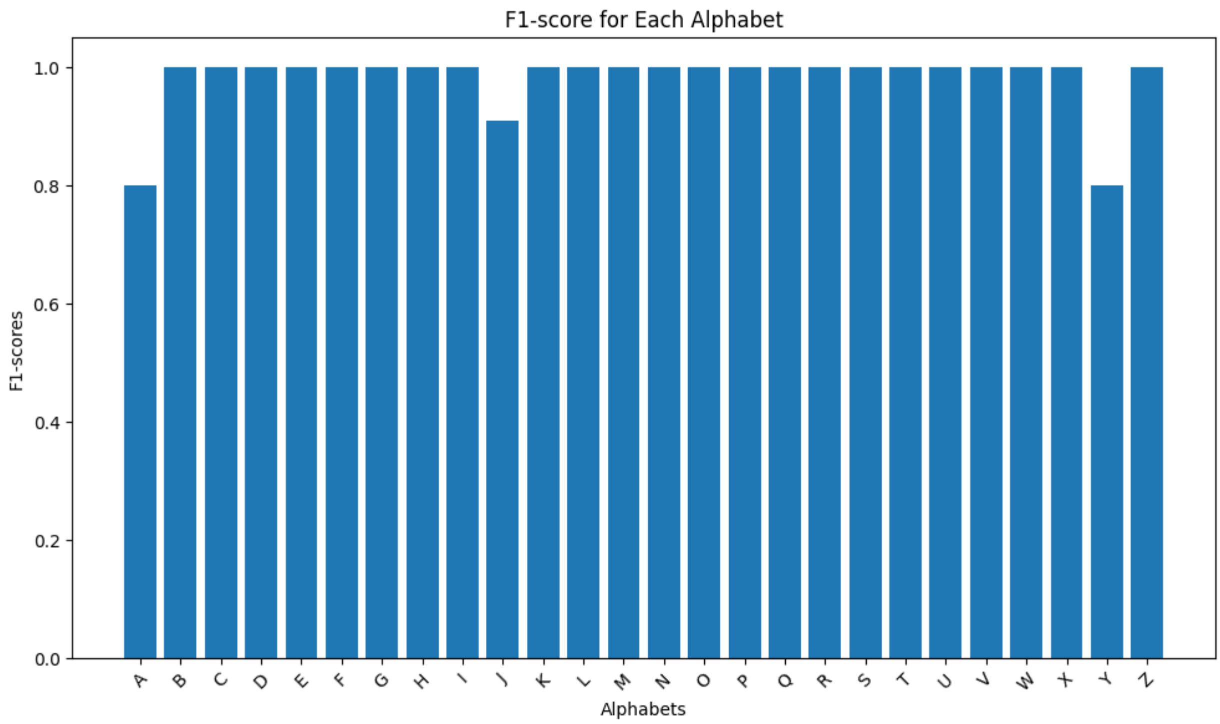


In [70]:
```
# Calculating F1 score for each class (alphabet)
f1_scores = f1_score(ytrue, ypredict, average=None)

# Printing F1 scores for each class
for idx, alphabet in enumerate(alphabets):
    print(f"F1 Score for Alphabet {alphabet}: {f1_scores[idx]:.4f}")
```

```
F1 Score for Alphabet A: 0.8000
F1 Score for Alphabet B: 1.0000
F1 Score for Alphabet C: 1.0000
F1 Score for Alphabet D: 1.0000
F1 Score for Alphabet E: 1.0000
F1 Score for Alphabet F: 1.0000
F1 Score for Alphabet G: 1.0000
F1 Score for Alphabet H: 1.0000
F1 Score for Alphabet I: 1.0000
F1 Score for Alphabet J: 0.9091
F1 Score for Alphabet K: 1.0000
F1 Score for Alphabet L: 1.0000
F1 Score for Alphabet M: 1.0000
F1 Score for Alphabet N: 1.0000
F1 Score for Alphabet O: 1.0000
F1 Score for Alphabet P: 1.0000
F1 Score for Alphabet Q: 1.0000
F1 Score for Alphabet R: 1.0000
F1 Score for Alphabet S: 1.0000
F1 Score for Alphabet T: 1.0000
F1 Score for Alphabet U: 1.0000
F1 Score for Alphabet V: 1.0000
F1 Score for Alphabet W: 1.0000
F1 Score for Alphabet X: 1.0000
F1 Score for Alphabet Y: 0.8000
F1 Score for Alphabet Z: 1.0000
```

In [84]:

```
import matplotlib.pyplot as plt
# Plotting the bar graph
plt.figure(figsize=(10, 6))
plt.bar(alphabets, f1_scores)
plt.xlabel('Alphabets')
plt.ylabel('F1-scores')
plt.title('F1-score for Each Alphabet')
plt.xticks(rotation=45)
plt.tight_layout()

# Display the plot
plt.show()
```

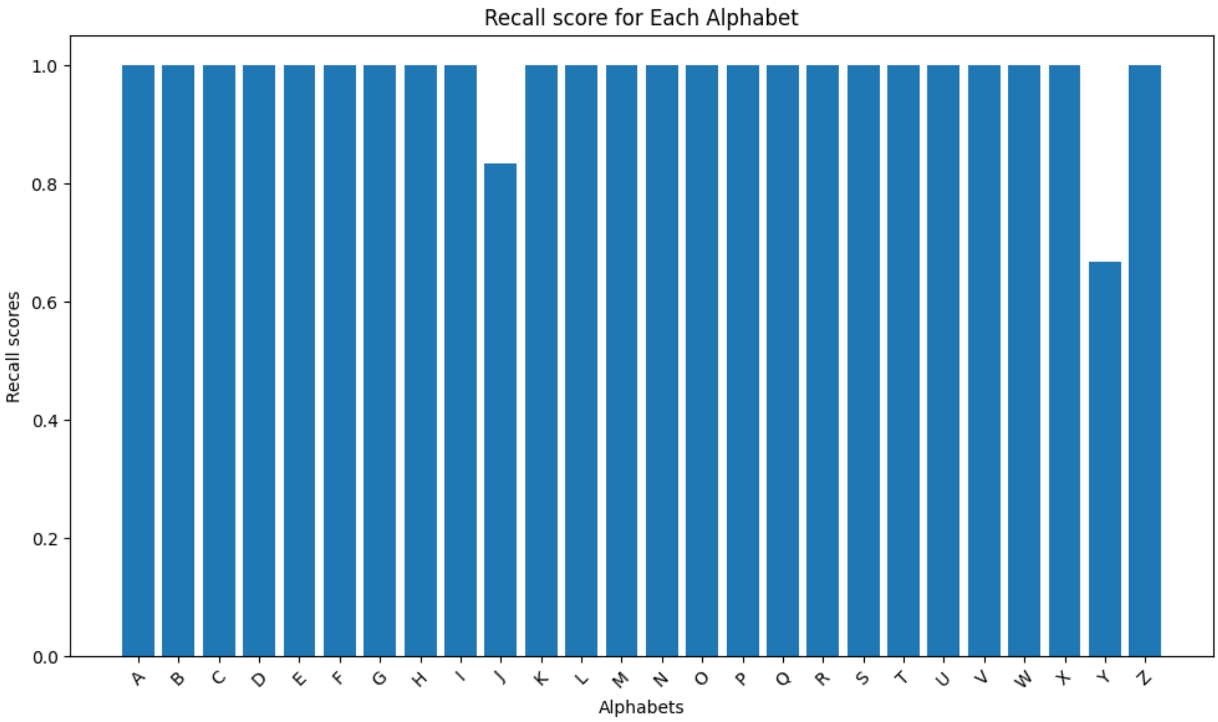


In [71]:

```
# Calculating recall score for each class (alphabet)
recall_scores = recall_score(ytrue, ypredict, average=None)

# Printing recall scores for each class
for idx, alphabet in enumerate(alphabets):
    print(f"Recall for Alphabet {alphabet}: {recall_scores[idx]:.4f}")
```

```
Recall for Alphabet A: 1.0000
Recall for Alphabet B: 1.0000
Recall for Alphabet C: 1.0000
Recall for Alphabet D: 1.0000
Recall for Alphabet E: 1.0000
Recall for Alphabet F: 1.0000
Recall for Alphabet G: 1.0000
Recall for Alphabet H: 1.0000
Recall for Alphabet I: 1.0000
Recall for Alphabet J: 0.8333
Recall for Alphabet K: 1.0000
Recall for Alphabet L: 1.0000
Recall for Alphabet M: 1.0000
Recall for Alphabet N: 1.0000
Recall for Alphabet O: 1.0000
Recall for Alphabet P: 1.0000
Recall for Alphabet Q: 1.0000
Recall for Alphabet R: 1.0000
Recall for Alphabet S: 1.0000
Recall for Alphabet T: 1.0000
Recall for Alphabet U: 1.0000
Recall for Alphabet V: 1.0000
Recall for Alphabet W: 1.0000
Recall for Alphabet X: 1.0000
Recall for Alphabet Y: 0.6667
Recall for Alphabet Z: 1.0000
```

In [85]:
```
import matplotlib.pyplot as plt
# Plotting the bar graph
plt.figure(figsize=(10, 6))
plt.bar(alphabets, recall_scores)
plt.xlabel('Alphabets')
plt.ylabel('Recall scores')
plt.title('Recall score for Each Alphabet')
plt.xticks(rotation=45)
plt.tight_layout()

# Display the plot
plt.show()
```



Train Data

In [ ]:
```
# making predictions on train data using trained model
# yhat = model.predict(X_train)
```

In [ ]:
```
# ytrue = np.argmax(Y_train, axis=1).tolist()
# yhat = np.argmax(yhat, axis=1).tolist()
```

5. Realtime testing

In [86]:
```
# function to perform real-time testing
colors = [(255,153,153), (255,178,102), (255, 255, 102),
          (153, 255,51), (51,153,255), (255,153,204), (255,153,153),
          (255,178,102), (255, 255, 102), (153, 255,51), (51,153,255),
          (255,153,204), (255,153,153), (255,178,102), (255, 255, 102),
          (153, 255,51), (51,153,255), (255,153,204), (255,153,153),
          (255,178,102), (255, 255, 102), (153, 255,51), (51,153,255),
          (255,153,204), (255,153,153), (255,178,102)]
def prob_viz(res, alphabets, input_frame, colors):
    output_frame = input_frame.copy()
    for num, prob in enumerate(res):
        cv2.rectangle(output_frame, (0, 25+num*25),
                      (int(prob*100), 45+num*25), colors[num], -1)
        cv2.putText(output_frame, alphabets[num], (0, 40+num*25),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0,0,0), 1,
                    cv2.LINE_AA)

    return output_frame
```

In [94]:
```
# making real time predictions
sequence = []
sentence = []
threshold = 0.9

cam = cv2.VideoCapture(0)
# Set mediapipe model
with mediapipe_holistic.Holistic(min_detection_confidence=0.9,
                min_tracking_confidence=0.9) as holistic:
    while cam.isOpened():

        # Read feed
        return_value, image_frame = cam.read()

        # Make detections
        image, detected_landmarks = detection_function(image_frame,
                                    holistic)
        print(detected_landmarks)

        # Draw landmarks
        draw_styled_landmarks(image, detected_landmarks)

        # making Prediction
        keypoints = mediapipe_keypoints(detected_landmarks)
        sequence.append(keypoints)
        sequence = sequence[-30:]

        if len(sequence) == 30:
            res = model.predict(np.expand_dims(sequence, axis=0))[0]
            print(alphabets[np.argmax(res)])


            if res[np.argmax(res)] > threshold:
                if len(sentence) > 0:
                    if alphabets[np.argmax(res)] != sentence[-1]:
                        sentence.append(alphabets[np.argmax(res)])
                else:
```

```
                    sentence.append(alphabets[np.argmax(res)])

                if len(sentence) > 5:
                    sentence = sentence[-5:]

                # displaying probability bar
                image = prob_viz(res, alphabets, image, colors)

            cv2.rectangle(image, (0,0), (1280, 20), (245, 117, 16), -1)
            cv2.putText(image, ' '.join(sentence), (3,20),
                           cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255),
                        1, cv2.LINE_AA)

            # displaying on screen
            cv2.imshow('Test model', image)

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

```
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 17ms/step
X
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 14ms/step
X
<class 'mediapipe.python.solution_base.SolutionOutputs'>
```

```
1/1 [==============================] - 0s 15ms/step
X
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 16ms/step
X
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 13ms/step
X
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 13ms/step
A
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 13ms/step
A
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 15ms/step
A
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 13ms/step
A
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 13ms/step
A
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 15ms/step
A
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 13ms/step
A
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 14ms/step
A
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 13ms/step
A
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 14ms/step
A
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 14ms/step
A
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 15ms/step
A
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 13ms/step
A
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 13ms/step
A
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 15ms/step
A
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 13ms/step
A
<class 'mediapipe.python.solution_base.SolutionOutputs'>
```

```
1/1 [==============================] - 0s 14ms/step
A
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 15ms/step
A
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 13ms/step
A
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 13ms/step
A
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 17ms/step
A
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 13ms/step
X
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 13ms/step
X
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 13ms/step
X
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 13ms/step
X
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 13ms/step
X
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 14ms/step
X
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 14ms/step
X
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 14ms/step
X
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 13ms/step
X
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 13ms/step
X
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 14ms/step
X
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 14ms/step
W
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 13ms/step
W
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 13ms/step
W
<class 'mediapipe.python.solution_base.SolutionOutputs'>
```

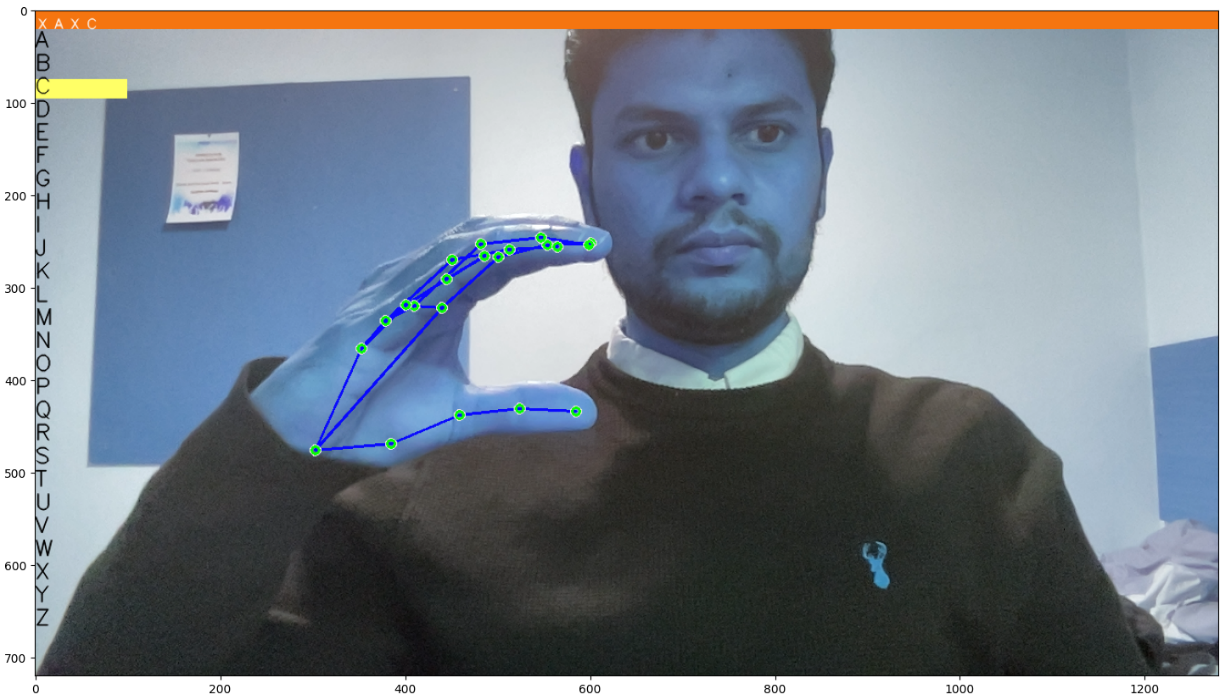
```
1/1 [==============================] - 0s 13ms/step
C
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 13ms/step
C
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 14ms/step
C
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 13ms/step
C
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 13ms/step
C
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 13ms/step
C
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 14ms/step
C
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 14ms/step
C
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 13ms/step
C
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 12ms/step
C
<class 'mediapipe.python.solution_base.SolutionOutputs'>
1/1 [==============================] - 0s 13ms/step
C
```

In [95]:

```
plt.figure(figsize=(18,18))
plt.imshow(prob_viz(res, alphabets, image, colors))
```

Out[95]:

```
<matplotlib.image.AxesImage at 0x2f3cb4370>
```


X A X C
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
0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200

In [ ]: