**Pre-processing Code**

list\_of\_files = {}  
*# MP\_Data for gestures, MP\_Data\_Char for Alphabets*  
**for** (dirpath, dirnames, filenames) **in** os.walk(**"./data/MP\_Data\_Char\_Test"**):  
    *# print(dirpath, "\n", dirnames, "\n", len(filenames))*  
    **if**(len(filenames)!=30):  
         print(dirpath, **"\n"**, len(filenames))

[8]

**with** open(**"../signs/alphabets.pkl"**,**'rb'**) **as** f:  
    actions = pickle.load(f)  
  
print(actions)

['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'U', 'V', 'W', 'X', 'Y']

[6]

actions = [chr(i) **for** i **in** range(ord(**'A'**), ord(**'Y'**)+1) **if** chr(i)!=**"T"** **and** chr(i)!=**"Z"**]  
print(actions)

['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'U', 'V', 'W', 'X', 'Y']

[10]

**with** open(**"../signs/gestures.pkl"**,**'rb'**) **as** f:  
    actions = pickle.load(f)  
print(actions)

['Hello' 'Love You' 'Understand' 'Thanks' 'Some' 'Home' 'name' 'my' 'how']

[24]

actions = np.array([**'Hello'**, **'Love You'**, **'Understand'**, **'Thanks'**, **'Some'**, **'Home'**, **'name'**, **'my'**, **'how'**, **'Sorry'**, **"Help me"**, **"Yes"**, **"No"**, **"eat"**, **"friend"**])  
*# actions = np.array(['Sorry', "Help me", "Yes", "No", "eat", "friend"])*

[25]

label\_map = {label:num **for** num, label **in** enumerate(actions)}  
print(label\_map)

{'Hello': 0, 'Love You': 1, 'Understand': 2, 'Thanks': 3, 'Some': 4, 'Home': 5, 'name': 6, 'my': 7, 'how': 8, 'Sorry': 9, 'Help me': 10, 'Yes': 11, 'No': 12, 'eat': 13, 'friend': 14}

[29]

data = []  
labels = []  
  
**for** action **in** actions:  
    **for** (dirpath, dirnames, filenames) **in** os.walk(os.path.join(**"../data/Gesture\_Landmarks"**, action)):  
        **if**(len(dirnames)==0):  
            *# print(dirpath, filenames)*  
            **for** filename **in** filenames:  
                data.append(np.load( os.path.join(dirpath, filename), allow\_pickle=**True** ))  
                labels.append(label\_map[action])  
  
data = np.array(data)  
labels = np.array(labels)  
print(data.shape)  
print(labels.shape)

**---------------------------------------------------------------------------**  
**KeyboardInterrupt** Traceback (most recent call last)  
**~\AppData\Local\Temp\ipykernel\_18412\1124636950.py** in <module> 7 **# print(dirpath, filenames)** 8 **for** filename **in** filenames**:** **----> 9**data**.**append**(**np**.**load**(** os**.**path**.**join**(**dirpath**,** filename**),** allow\_pickle**=True** **))** 10 labels**.**append**(**label\_map**[**action**])** 11  
**D:\Software\_Installs\Anaconda\envs\capstone\lib\site-packages\numpy\lib\npyio.py** in load**(file, mmap\_mode, allow\_pickle, fix\_imports, encoding)** 414 own\_fid **=** **False** 415 **else:** **--> 416**fid **=** stack**.**enter\_context**(**open**(**os\_fspath**(**file**),** **"rb"))** 417 own\_fid **=** **True** 418  
**KeyboardInterrupt**:

[34]

unique, counts = np.unique(labels, return\_counts=**True**)  
print(dict(zip(unique, counts)))

{0: 749, 1: 750, 2: 742, 3: 750, 4: 748, 5: 695, 6: 750, 7: 750, 8: 750, 9: 737, 10: 750, 11: 125, 12: 406, 13: 116, 14: 749}

[35]

*# from tensorflow.keras.utils.np\_utils import to\_categorical*  
**import** tensorflow **as** tf  
  
lab = tf.keras.utils.to\_categorical(labels)  
print(lab.shape)

(9567, 15)

[36]

**from** sklearn.model\_selection **import** train\_test\_split  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(data, lab, test\_size=0.2, random\_state=42)  
  
unique, counts = np.unique(np.argmax(y\_train, axis=1), return\_counts=**True**)  
print(**"Training: "**, dict(zip(unique, counts)))  
  
unique, counts = np.unique(np.argmax(y\_test, axis=1), return\_counts=**True**)  
print(**"Test: "**, dict(zip(unique, counts)))

Training: {0: 591, 1: 600, 2: 593, 3: 600, 4: 602, 5: 548, 6: 590, 7: 602, 8: 618, 9: 589, 10: 593, 11: 104, 12: 327, 13: 94, 14: 602}  
Test: {0: 158, 1: 150, 2: 149, 3: 150, 4: 146, 5: 147, 6: 160, 7: 148, 8: 132, 9: 148, 10: 157, 11: 21, 12: 79, 13: 22, 14: 147}

[37]

print(**"Training: "**, X\_train.shape, y\_train.shape)  
print(**"Test: "**, X\_test.shape, y\_test.shape)

Training: (7653, 21, 4) (7653, 15)  
Test: (1914, 21, 4) (1914, 15)