Google Isolated Sign Language Recognition

Inference

Code submitted for group project, DA526, 2023, IITG.

Team Members: Vikky Masih, Pallapu Mohan Krishna, Shania H, Rahul Bhardwaj, Prakhar Kumar Sonkar

In [1]:

```
import os
import math
import time
import copy

import numpy as np
import pandas as pd

import torch
from torch import nn
from torch.utils.data import DataLoader, Dataset
from sklearn.model_selection import StratifiedGroupKFold
from sklearn.metrics import confusion_matrix

import matplotlib.pyplot as plt
import seaborn as sns
from itables import init_notebook_mode
import itables.options as itable_opt
from tqdm.notebook import trange, tqdm

import cv2
import mediapipe as mp
from IPython.display import Video
```

In [2]:

```
device = "cuda" if torch.cuda.is_available() else "cpu"
device = "mps" if torch.has_mps else device
print(f"Using {device} device")

if device=="cuda":
    !nvidia-smi
```

```
Using cuda device
Sun May 14 12:35:38 2023
+---------------------------------------------------------------------------------------+
| NVIDIA-SMI 531.61                 Driver Version: 531.61       CUDA Version: 12.1     |
|-----------------------------------------+----------------------+----------------------+
| GPU  Name                      TCC/WDDM | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf            Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|=========================================+======================+======================|
|   0  NVIDIA GeForce RTX 4090       WDDM | 00000000:01:00.0  On |                  Off |
|  0%   34C    P8               13W / 450W|    576MiB / 24564MiB |      2%      Default |
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+

+---------------------------------------------------------------------------------------+
| Processes:                                                                            |
|  GPU   GI   CI        PID   Type   Process name                            GPU Memory |
|        ID   ID                                                             Usage      |
|=======================================================================================|
|    0   N/A  N/A      1580    C+G   C:\Windows\explorer.exe                   N/A      |
|    0   N/A  N/A      3548    C+G   ...es\Windows Firewall Control\wfc.exe    N/A      |
|    0   N/A  N/A      4920    C+G   ...__8wekyb3d8bbwe\WindowsTerminal.exe    N/A      |
|    0   N/A  N/A      9872    C+G   ...nt.CBS_cw5n1h2txyewy\SearchHost.exe    N/A      |
|    0   N/A  N/A      9912    C+G   ...2txyewy\StartMenuExperienceHost.exe    N/A      |
|    0   N/A  N/A     10968    C+G   ...les\LibreOffice\program\soffice.bin    N/A      |
|    0   N/A  N/A     14732    C+G   ...siveControlPanel\SystemSettings.exe    N/A      |
+---------------------------------------------------------------------------------------+
```

In [3]:

```
iskaggle = os.environ.get('KAGGLE_KERNEL_RUN_TYPE', '')
myDataDir='data/google_asl_data' if not iskaggle else os.path.join("/kaggle","input","asl-signs")
outputDir="data/output" if not iskaggle else "output"
videoDir="data/output/Inference/" if not iskaggle else "output/Inference"
```

In [4]:

```
# Module Aliases
mp_drawing = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles
mp_holistic = mp.solutions.holistic

# Calculating number of landmarks in each region
p=set()
for k in mp_holistic.FACEMESH_TESSELATION:
    for kk in k:
        p.add(kk)
f_len=len(p)
p_len=len(mp_holistic.PoseLandmark)
h_len=len(mp_holistic.HandLandmark)
```

Pre-processing helper function

In [5]:

```
# number of landmarks per frame (features)
ROWS_PER_FRAME = 543

# Point cloud groups as per data file
face_indices=np.arange(0,468)
lhand_indices=np.arange(468,489)
pose_indices=np.arange(489,522)
rhand_indices=np.arange(522,543)

# Helper function for preprocessing
def data_transform(data):
    # Removing un-necessary dimensions
    dataZ=data.squeeze()

    # Normalizing by frame mean and std while ignoring NaN values
    dataZ=(dataZ-dataZ.nanmean(dim=(0,1)))/np.nanstd(dataZ,axis=(0,1))

    # Detecting missing features
    lhand_missing=(dataZ[:,lhand_indices,:].isnan().sum(dim=[1,2]))>0
    rhand_missing=(dataZ[:,rhand_indices,:].isnan().sum(dim=[1,2]))>0
    face_missing=(dataZ[:,face_indices,:].isnan().sum(dim=[1,2]))>0
    handsMissing=lhand_missing&rhand_missing
    face_or_hands_missing=handsMissing|face_missing

    # Filling up missing hand via mirroring other hand about y-axis
    fillMissingRight=np.where((~handsMissing)&(rhand_missing))[0]
    fillMissingLeft=np.where((~handsMissing)&(lhand_missing))[0]
    if len(fillMissingRight)>0:
        dataZ[fillMissingRight[:,np.newaxis],rhand_indices[np.newaxis,:],0]=    \
             -dataZ[fillMissingRight[:,np.newaxis],lhand_indices[np.newaxis,:],0]

        dataZ[fillMissingRight[:,np.newaxis],rhand_indices[np.newaxis,:],1]=    \
             dataZ[fillMissingRight[:,np.newaxis],lhand_indices[np.newaxis,:],1]

        dataZ[fillMissingRight[:,np.newaxis],rhand_indices[np.newaxis,:],2]=    \
             -dataZ[fillMissingRight[:,np.newaxis],lhand_indices[np.newaxis,:],2]
    if len(fillMissingLeft)>0:
        dataZ[fillMissingLeft[:,np.newaxis],lhand_indices[np.newaxis,:],0]=    \
             -dataZ[fillMissingLeft[:,np.newaxis],rhand_indices[np.newaxis,:],0]

        dataZ[fillMissingLeft[:,np.newaxis],lhand_indices[np.newaxis,:],1]=    \
             dataZ[fillMissingLeft[:,np.newaxis],rhand_indices[np.newaxis,:],1]

        dataZ[fillMissingLeft[:,np.newaxis],lhand_indices[np.newaxis,:],2]=    \
             -dataZ[fillMissingLeft[:,np.newaxis],rhand_indices[np.newaxis,:],2]

    # Removing frames without face or both-hands
    dataZ=dataZ[~face_or_hands_missing,:,:]

    # Replacing NaN(s) with zero
    return torch.tensor(np.nan_to_num(dataZ,0.0)).flatten(1)
```

LSTM based Neural Netwok

In [6]:

```
class Network_LSTM(nn.Module):
    def __init__(self,inSize,hiddenSize,outSize,rnnLayers=1,dropout=0,device='cpu'):
        super(Network_LSTM, self).__init__()
        self.inSize = inSize
        self.hiddenSize = hiddenSize
        self.outSize = outSize
        self.rnnLayers = rnnLayers
        self.device = device
        self.dropout = dropout

        # LSTM Layer
        self.rnn = nn.LSTM(inSize,hiddenSize,rnnLayers,batch_first=True,dropout=dropout)

        # Fully Connected Layer
        self.fc = nn.Linear(hiddenSize,outSize,bias=False)

    def forward(self, x):
        x, (hiddenState, cellState) = self.rnn(x)
        x = torch.vstack([k[-1,:] for k in nn.utils.rnn.unpack_sequence(x)])
        x = self.fc(x)
        return x
```

Loading Model

In [7]:

```
# Reading model information
state=torch.load(os.path.join(outputDir,'BestModel.pt'), map_location=torch.device(device))
m_hs,m_rl,_=state['HyperParameters']
classNames=state['ClassNames']

# Model instance
model=Network_LSTM(inSize=1629,\
                   hiddenSize=m_hs,\
                   outSize=10,\
                   rnnLayers=m_rl).to(device)

# Loading model weights
model.load_state_dict(state['State'])

# Setting model to evaluation mode
model.eval()
```

Out[7]:

```
Network_LSTM(
  (rnn): LSTM(1629, 256, batch_first=True)
  (fc): Linear(in_features=256, out_features=10, bias=False)
)
```

Video file for inference

In [8]:

```
# YouTube Videos taken from https://asl-kids.com/ solely for educational purposes
videoFiles=os.listdir(videoDir)
for k in videoFiles:
    print(k)
```

```
Bird in Sign Language- ASL Dictionary for kids.mp4
Cat in Sign Language, ASL Dictionary for kids.mp4
Chicken in Sign Language- ASL Dictionary for kids.mp4
Cow in Sign Language- ASL Dictionary for kids.mp4
Dog in Sign Language- ASL Dictionary for kids.mp4
Fish in Sign Language- ASL Dictionary for kids.mp4
Frog in Sign Language- ASL Dictionary for kids.mp4
Ladybug in Sign Language- ASL Dictionary for kids.mp4
Mouse in Sign Language- ASL Dictionary for kids.mp4
Pig in Sign Language- ASL Dictionary for kids.mp4
```

Video Sample

In [9]:

```
videoIndex=0
print(videoFiles[videoIndex])
Video(os.path.join(videoDir,videoFiles[videoIndex]),embed=True)
```

```
Bird in Sign Language- ASL Dictionary for kids.mp4
```

Out[9]:

Preprocessing Videos

In [10]:

```
videoLandmarks=[]
for video in videoFiles:
    landmarks=torch.tensor([])
    videoFile=os.path.join(videoDir,video)
    cap=cv2.VideoCapture(videoFile)

    with mp_holistic.Holistic() as holistic:
        while cap.isOpened():
            success, image = cap.read()
            if not success:
                break

            image.flags.writeable = False
            image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
            results = holistic.process(image)

            q=results.face_landmarks
            fl=[[k.x,k.y,k.z] for k in q.landmark] if q else [[float('NAN')]*3]*f_len
            q=results.left_hand_landmarks
            lhl=[[k.x,k.y,k.z] for k in q.landmark] if q else [[float('NAN')]*3]*h_len
            q=results.pose_landmarks
            pl=[[k.x,k.y,k.z] for k in q.landmark] if q else [[float('NAN')]*3]*p_len
            q=results.right_hand_landmarks
            rhl=[[k.x,k.y,k.z] for k in q.landmark] if q else [[float('NAN')]*3]*h_len
            landmarks=torch.cat((landmarks,torch.tensor(fl+lhl+pl+rhl).unsqueeze(0)),dim=0)
    cap.release()
    videoLandmarks.append(data_transform(landmarks))
```

In [11]:

```
batch=sorted(enumerate(videoLandmarks),key=lambda x:x[1].shape[0],reverse=True)
i,d=zip(*batch)
landmarksPacked=nn.utils.rnn.pack_sequence(d, enforce_sorted=True).to(device)
```

Inference

In [12]:

```
with torch.set_grad_enabled(False):
    output=model(landmarksPacked)
```

In [13]:

```
_,indices=torch.sort(output.to('cpu'),dim=1,descending=True)
indices

print('Top 3 Classes:\n'+'='*50)
for kkk,k in enumerate(i):
    videoFile=videoFiles[k]
    top3=[classNames[kk] for kk in indices[kkk,:3].tolist()]
    print(f'{videoFile}:\n    {top3}\n')
```

```
Top 3 Classes:
==================================================
Ladybug in Sign Language- ASL Dictionary for kids.mp4:
    ['fish', 'hen', 'cat']

Chicken in Sign Language- ASL Dictionary for kids.mp4:
    ['cat', 'fish', 'dog']

Dog in Sign Language- ASL Dictionary for kids.mp4:
    ['dog', 'cat', 'hen']

Frog in Sign Language- ASL Dictionary for kids.mp4:
    ['hen', 'fish', 'dog']

Cow in Sign Language- ASL Dictionary for kids.mp4:
    ['cow', 'dog', 'cat']

Cat in Sign Language, ASL Dictionary for kids.mp4:
    ['dog', 'cat', 'cow']

Fish in Sign Language- ASL Dictionary for kids.mp4:
    ['bug', 'cat', 'dog']

Mouse in Sign Language- ASL Dictionary for kids.mp4:
    ['mouse', 'bird', 'hen']

Bird in Sign Language- ASL Dictionary for kids.mp4:
    ['fish', 'cat', 'cow']

Pig in Sign Language- ASL Dictionary for kids.mp4:
    ['pig', 'frog', 'fish']
```