**FOLDER STRUCTURE**

**isl-organised/**

├── **main.ipynb** # Main notebook for preprocessing, training, and evaluation

├── **userInterface.py**  # Real-time webcam prediction script

│

├── **feature\_extraction/**

│ └── **feature\_extraction.py** # Extracts frames and prepares training input X.npy

│

├── **mediapipe/**

│ └── **mp.py** # Converts original videos to Media Pipe skeleton videos

│

├── **models/**

│ └── **model.py** # Defines and compiles the Spatiotemporal CNN model

│

├── **data/**

│ ├── **Days\_and\_Time/** # Raw input videos, sorted by class (Morning, Night, etc.)

│ └── **Days\_and\_Time\_skeleton/** # Skeleton videos created using Media Pipe

│

├── **X.npy** # Input features: (num\_videos, 30, height, width, 3)

├── **y.npy** # Labels: (num\_videos,)

├── **my\_sign\_model.h5** # Trained model weights

│

└── **README.md** # (Optional) Project documentation

**CODE**

**main.ipynb**

import zipfile

import os

*# Path to the ZIP file*

zip\_path = "data/dataset\_time.zip"

print(os.listdir("data"))

*# Directory to extract files*

extract\_to = "data/"

*# Open and extract*

with zipfile.ZipFile(zip\_path, 'r') as zip\_ref:

zip\_ref.extractall(extract\_to)

print("Extraction complete!")

**['.ipynb\_checkpoints', 'dataset\_time.zip', 'hand\_skeleton', 'hand\_skeleton.zip', 'hand\_skeleton\_images', 'hand\_skeleton\_images.zip']**

**Extraction complete!**

In [19]:

*## Mediapipe Code(To be called once only)*

import importlib

import sys

sys.path.append('feature\_extraction')

sys.path.append("mediapipe")

import mp

importlib.reload(mp)

mp.use\_mediapipe()

**Class .ipynb\_checkpoints done**

**data\Days\_and\_Time\Afternoon\MVI\_4655.MOV**

**data\Days\_and\_Time\Afternoon\MVI\_4656.MOV**

**data\Days\_and\_Time\Afternoon\MVI\_4657.MOV**

**data\Days\_and\_Time\Afternoon\MVI\_4658.MOV**

**data\Days\_and\_Time\Afternoon\MVI\_5063.MOV**

**data\Days\_and\_Time\Afternoon\MVI\_5064.MOV**

**data\Days\_and\_Time\Afternoon\MVI\_5065.MOV**

**data\Days\_and\_Time\Afternoon\MVI\_5511.MOV**

**data\Days\_and\_Time\Afternoon\MVI\_5512.MOV**

**data\Days\_and\_Time\Afternoon\MVI\_5513.MOV**

**data\Days\_and\_Time\Afternoon\MVI\_5514.MOV**

**Class Afternoon done**

**data\Days\_and\_Time\Evening\MVI\_4659.MOV**

**data\Days\_and\_Time\Evening\MVI\_4660.MOV**

**data\Days\_and\_Time\Evening\MVI\_4661.MOV**

**data\Days\_and\_Time\Evening\MVI\_4662.MOV**

**data\Days\_and\_Time\Evening\MVI\_5070.MOV**

**data\Days\_and\_Time\Evening\MVI\_5071.MOV**

**data\Days\_and\_Time\Evening\MVI\_5072.MOV**

**data\Days\_and\_Time\Evening\MVI\_5515.MOV**

**data\Days\_and\_Time\Evening\MVI\_5517.MOV**

**data\Days\_and\_Time\Evening\MVI\_5518.MOV**

**data\Days\_and\_Time\Evening\MVI\_5519.MOV**

**Class Evening done**

**data\Days\_and\_Time\Morning\MVI\_4651.MOV**

**data\Days\_and\_Time\Morning\MVI\_4652.MOV**

**data\Days\_and\_Time\Morning\MVI\_4653.MOV**

**data\Days\_and\_Time\Morning\MVI\_4654.MOV**

**data\Days\_and\_Time\Morning\MVI\_5060.MOV**

**data\Days\_and\_Time\Morning\MVI\_5061.MOV**

**data\Days\_and\_Time\Morning\MVI\_5062.MOV**

**data\Days\_and\_Time\Morning\MVI\_5507.MOV**

**data\Days\_and\_Time\Morning\MVI\_5508.MOV**

**data\Days\_and\_Time\Morning\MVI\_5509.MOV**

**data\Days\_and\_Time\Morning\MVI\_5510.MOV**

**Class Morning done**

**data\Days\_and\_Time\Night\MVI\_4663.MOV**

**data\Days\_and\_Time\Night\MVI\_4664.MOV**

**data\Days\_and\_Time\Night\MVI\_4665.MOV**

**data\Days\_and\_Time\Night\MVI\_4666.MOV**

**data\Days\_and\_Time\Night\MVI\_5073.MOV**

**data\Days\_and\_Time\Night\MVI\_5074.MOV**

**data\Days\_and\_Time\Night\MVI\_5076.MOV**

**data\Days\_and\_Time\Night\MVI\_5520.MOV**

**data\Days\_and\_Time\Night\MVI\_5521.MOV**

**data\Days\_and\_Time\Night\MVI\_5522.MOV**

**data\Days\_and\_Time\Night\MVI\_5523.MOV**

**Class Night done**

**✅ Holistic video created successfully!**

In [31]:

*## Feature Extraction*

print("Testing...", flush=True)

import feature\_extraction as fe

importlib.reload(fe)

fe.combine\_features()

**Testing...**

**Afternoon**

**Evening**

**Morning**

**Night**

**(44, 30, 224, 224, 3)**

In [65]:

import numpy as np

X = np.load('X.npy')

y = np.load('y.npy')

from sklearn.preprocessing import LabelEncoder

encoded\_y = np.array([])

print(y)

for ele in y:

if(ele == 'Afternoon'):

encoded\_y = np.append(encoded\_y,0)

if(ele == 'Evening'):

encoded\_y = np.append(encoded\_y,1)

if(ele == 'Morning'):

encoded\_y = np.append(encoded\_y,2)

if(ele == 'Night'):

encoded\_y = np.append(encoded\_y,3)

print(encoded\_y)

**['Afternoon' 'Afternoon' 'Afternoon' 'Afternoon' 'Afternoon' 'Afternoon'**

**'Afternoon' 'Afternoon' 'Afternoon' 'Afternoon' 'Afternoon' 'Evening'**

**'Evening' 'Evening' 'Evening' 'Evening' 'Evening' 'Evening' 'Evening'**

**'Evening' 'Evening' 'Evening' 'Morning' 'Morning' 'Morning' 'Morning'**

**'Morning' 'Morning' 'Morning' 'Morning' 'Morning' 'Morning' 'Morning'**

**'Night' 'Night' 'Night' 'Night' 'Night' 'Night' 'Night' 'Night' 'Night'**

**'Night' 'Night']**

**[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 2. 2.**

**2. 2. 2. 2. 2. 2. 2. 2. 2. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3.]**

In [66]:

*##Train - Test Split*

from sklearn.model\_selection import train\_test\_split

print(X.shape)

*# Split into train and test (80/20)*

print(encoded\_y)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, encoded\_y, test\_size=0.2, stratify=encoded\_y, random\_state=42

)

*# Optional: Also split training set into training + validation*

X\_train, X\_val, y\_train, y\_val = train\_test\_split(

X\_train, y\_train, test\_size=0.1, stratify=y\_train, random\_state=42

)

print(X\_train.shape)

print(X\_val.shape)

print(X\_test.shape)

**(44, 30, 224, 224, 3)**

**[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 2. 2.**

**2. 2. 2. 2. 2. 2. 2. 2. 2. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3.]**

**(31, 30, 224, 224, 3)**

**(4, 30, 224, 224, 3)**

**(9, 30, 224, 224, 3)**

In [67]:

import sys

sys.path.append('models')

import model as model

model = model.build\_spatiotemporal\_cnn(num\_classes = 4)

**C:\Users\nebhw\AppData\Roaming\Python\Python312\site-packages\keras\src\layers\convolutional\base\_conv.py:107: UserWarning: Do not pass an `input\_shape`/`input\_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.**

**super().\_\_init\_\_(activity\_regularizer=activity\_regularizer, \*\*kwargs)**

In [68]:

model.fit(X\_train, y\_train,

validation\_data=(X\_val, y\_val),

epochs=20, batch\_size=2)

**Epoch 1/20**

**16/16 ━━━━━━━━━━━━━━━━━━━━ 18s 950ms/step - accuracy: 0.2559 - loss: 2.7449 - val\_accuracy: 0.2500 - val\_loss: 1.3863**

**Epoch 2/20**

**16/16 ━━━━━━━━━━━━━━━━━━━━ 15s 910ms/step - accuracy: 0.2765 - loss: 1.3965 - val\_accuracy: 0.2500 - val\_loss: 1.3861**

**Epoch 3/20**

**16/16 ━━━━━━━━━━━━━━━━━━━━ 15s 917ms/step - accuracy: 0.5069 - loss: 1.3828 - val\_accuracy: 0.2500 - val\_loss: 1.4063**

**Epoch 4/20**

**16/16 ━━━━━━━━━━━━━━━━━━━━ 15s 917ms/step - accuracy: 0.1868 - loss: 1.4463 - val\_accuracy: 0.2500 - val\_loss: 1.3880**

**Epoch 5/20**

**16/16 ━━━━━━━━━━━━━━━━━━━━ 15s 905ms/step - accuracy: 0.3674 - loss: 1.3758 - val\_accuracy: 0.2500 - val\_loss: 1.3956**

**Epoch 6/20**

**16/16 ━━━━━━━━━━━━━━━━━━━━ 15s 918ms/step - accuracy: 0.2052 - loss: 1.3969 - val\_accuracy: 0.0000e+00 - val\_loss: 1.4119**

**Epoch 7/20**

**16/16 ━━━━━━━━━━━━━━━━━━━━ 15s 908ms/step - accuracy: 0.3497 - loss: 1.2771 - val\_accuracy: 0.2500 - val\_loss: 1.4449**

**Epoch 8/20**

**16/16 ━━━━━━━━━━━━━━━━━━━━ 15s 918ms/step - accuracy: 0.3454 - loss: 1.3283 - val\_accuracy: 0.2500 - val\_loss: 1.4824**

**Epoch 9/20**

**16/16 ━━━━━━━━━━━━━━━━━━━━ 15s 917ms/step - accuracy: 0.2465 - loss: 1.3229 - val\_accuracy: 0.2500 - val\_loss: 1.4400**

**Epoch 10/20**

**16/16 ━━━━━━━━━━━━━━━━━━━━ 14s 895ms/step - accuracy: 0.5783 - loss: 0.9620 - val\_accuracy: 0.0000e+00 - val\_loss: 1.4625**

**Epoch 11/20**

**16/16 ━━━━━━━━━━━━━━━━━━━━ 15s 912ms/step - accuracy: 0.5705 - loss: 1.0596 - val\_accuracy: 0.0000e+00 - val\_loss: 1.5725**

**Epoch 12/20**

**16/16 ━━━━━━━━━━━━━━━━━━━━ 15s 942ms/step - accuracy: 0.6170 - loss: 0.7617 - val\_accuracy: 0.5000 - val\_loss: 1.2265**

**Epoch 13/20**

**16/16 ━━━━━━━━━━━━━━━━━━━━ 15s 909ms/step - accuracy: 0.7712 - loss: 0.5797 - val\_accuracy: 0.7500 - val\_loss: 0.9181**

**Epoch 14/20**

**16/16 ━━━━━━━━━━━━━━━━━━━━ 15s 907ms/step - accuracy: 0.9053 - loss: 0.2964 - val\_accuracy: 0.5000 - val\_loss: 0.9176**

**Epoch 15/20**

**16/16 ━━━━━━━━━━━━━━━━━━━━ 15s 915ms/step - accuracy: 0.9460 - loss: 0.1576 - val\_accuracy: 0.7500 - val\_loss: 0.5556**

**Epoch 16/20**

**16/16 ━━━━━━━━━━━━━━━━━━━━ 14s 901ms/step - accuracy: 0.9942 - loss: 0.2616 - val\_accuracy: 1.0000 - val\_loss: 0.3179**

**Epoch 17/20**

**16/16 ━━━━━━━━━━━━━━━━━━━━ 15s 910ms/step - accuracy: 0.9796 - loss: 0.0866 - val\_accuracy: 0.7500 - val\_loss: 0.2928**

**Epoch 18/20**

**16/16 ━━━━━━━━━━━━━━━━━━━━ 14s 900ms/step - accuracy: 0.9707 - loss: 0.1367 - val\_accuracy: 1.0000 - val\_loss: 0.1069**

**Epoch 19/20**

**16/16 ━━━━━━━━━━━━━━━━━━━━ 17s 1s/step - accuracy: 1.0000 - loss: 0.0480 - val\_accuracy: 1.0000 - val\_loss: 0.1205**

**Epoch 20/20**

**16/16 ━━━━━━━━━━━━━━━━━━━━ 16s 1s/step - accuracy: 1.0000 - loss: 0.0477 - val\_accuracy: 1.0000 - val\_loss: 0.0576**

**Out[68]:**

**<keras.src.callbacks.history.History at 0x20f565495e0>**

In [69]:

model.save("my\_sign\_model.h5") *# .h5 format*

**WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save\_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my\_model.keras')` or `keras.saving.save\_model(model, 'my\_model.keras')`.**

In [70]:

y\_pred\_probs = model.predict(X\_test)

y\_pred = y\_pred\_probs.argmax(axis=1) *# convert from one-hot or probabilities to class index*

**1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step**

In [71]:

from sklearn.metrics import classification\_report

print(classification\_report(y\_test, y\_pred, digits=4))

**precision recall f1-score support**

**0.0 1.0000 1.0000 1.0000 2**

**1.0 1.0000 1.0000 1.0000 2**

**2.0 1.0000 1.0000 1.0000 3**

**3.0 1.0000 1.0000 1.0000 2**

**accuracy 1.0000 9**

**macro avg 1.0000 1.0000 1.0000 9**

**weighted avg 1.0000 1.0000 1.0000 9**

In [72]:

from sklearn.metrics import confusion\_matrix

import seaborn as sns

import matplotlib.pyplot as plt

cm = confusion\_matrix(y\_test, y\_pred)

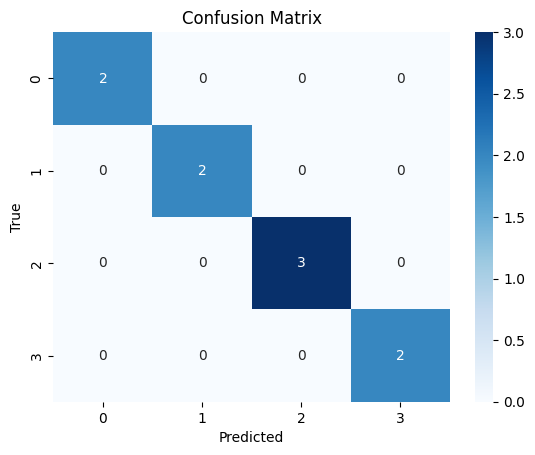
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')

plt.xlabel('Predicted')

plt.ylabel('True')

plt.title('Confusion Matrix')

plt.show()

****

In [73]:

from sklearn.metrics import ConfusionMatrixDisplay

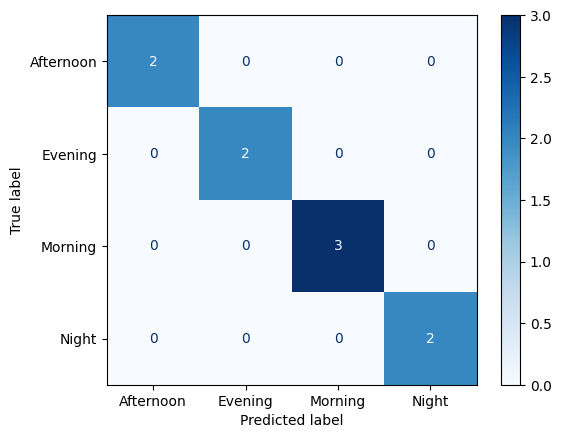
class\_names = ['Afternoon', 'Evening', 'Morning', 'Night']

disp = ConfusionMatrixDisplay(confusion\_matrix=cm, display\_labels=class\_names)

disp.plot(cmap='Blues')

**Out[73]:**

**<sklearn.metrics.\_plot.confusion\_matrix.ConfusionMatrixDisplay at 0x20f567dd7c0>**

****

In [1]:

pip install opencv-python mediapipe numpy tensorflow

**Defaulting to user installation because normal site-packages is not writeable**

**Requirement already satisfied: opencv-python in c:\users\nebhw\appdata\roaming\python\python312\site-packages (4.10.0.84)**

**Requirement already satisfied: mediapipe in c:\users\nebhw\appdata\roaming\python\python312\site-packages (0.10.20)**

**Requirement already satisfied: numpy in c:\users\nebhw\appdata\roaming\python\python312\site-packages (1.26.4)**

**Requirement already satisfied: tensorflow in c:\users\nebhw\appdata\roaming\python\python312\site-packages (2.18.0)**

**Requirement already satisfied: absl-py in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from mediapipe) (2.1.0)**

**Requirement already satisfied: attrs>=19.1.0 in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from mediapipe) (24.2.0)**

**Requirement already satisfied: flatbuffers>=2.0 in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from mediapipe) (24.3.25)**

**Requirement already satisfied: jax in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from mediapipe) (0.4.37)**

**Requirement already satisfied: jaxlib in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from mediapipe) (0.4.36)**

**Requirement already satisfied: matplotlib in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from mediapipe) (3.9.2)**

**Requirement already satisfied: opencv-contrib-python in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from mediapipe) (4.10.0.84)**

**Requirement already satisfied: protobuf<5,>=4.25.3 in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from mediapipe) (4.25.5)**

**Requirement already satisfied: sounddevice>=0.4.4 in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from mediapipe) (0.5.1)**

**Requirement already satisfied: sentencepiece in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from mediapipe) (0.2.0)**

**Requirement already satisfied: tensorflow-intel==2.18.0 in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from tensorflow) (2.18.0)**

**Requirement already satisfied: astunparse>=1.6.0 in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from tensorflow-intel==2.18.0->tensorflow) (1.6.3)**

**Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from tensorflow-intel==2.18.0->tensorflow) (0.6.0)**

**Requirement already satisfied: google-pasta>=0.1.1 in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from tensorflow-intel==2.18.0->tensorflow) (0.2.0)**

**Requirement already satisfied: libclang>=13.0.0 in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from tensorflow-intel==2.18.0->tensorflow) (18.1.1)**

**Requirement already satisfied: opt-einsum>=2.3.2 in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from tensorflow-intel==2.18.0->tensorflow) (3.4.0)**

**Requirement already satisfied: packaging in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from tensorflow-intel==2.18.0->tensorflow) (24.1)**

**Requirement already satisfied: requests<3,>=2.21.0 in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from tensorflow-intel==2.18.0->tensorflow) (2.32.3)**

**Requirement already satisfied: setuptools in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from tensorflow-intel==2.18.0->tensorflow) (75.3.0)**

**Requirement already satisfied: six>=1.12.0 in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from tensorflow-intel==2.18.0->tensorflow) (1.16.0)**

**Requirement already satisfied: termcolor>=1.1.0 in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from tensorflow-intel==2.18.0->tensorflow) (2.5.0)**

**Requirement already satisfied: typing-extensions>=3.6.6 in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from tensorflow-intel==2.18.0->tensorflow) (4.12.2)**

**Requirement already satisfied: wrapt>=1.11.0 in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from tensorflow-intel==2.18.0->tensorflow) (1.17.0)**

**Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from tensorflow-intel==2.18.0->tensorflow) (1.68.0)**

**Requirement already satisfied: tensorboard<2.19,>=2.18 in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from tensorflow-intel==2.18.0->tensorflow) (2.18.0)**

**Requirement already satisfied: keras>=3.5.0 in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from tensorflow-intel==2.18.0->tensorflow) (3.7.0)**

**Requirement already satisfied: h5py>=3.11.0 in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from tensorflow-intel==2.18.0->tensorflow) (3.12.1)**

**Requirement already satisfied: ml-dtypes<0.5.0,>=0.4.0 in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from tensorflow-intel==2.18.0->tensorflow) (0.4.1)**

**Requirement already satisfied: CFFI>=1.0 in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from sounddevice>=0.4.4->mediapipe) (1.17.1)**

**Requirement already satisfied: scipy>=1.10 in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from jax->mediapipe) (1.14.1)**

**Requirement already satisfied: contourpy>=1.0.1 in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from matplotlib->mediapipe) (1.3.0)**

**Requirement already satisfied: cycler>=0.10 in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from matplotlib->mediapipe) (0.12.1)**

**Requirement already satisfied: fonttools>=4.22.0 in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from matplotlib->mediapipe) (4.54.1)**

**Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from matplotlib->mediapipe) (1.4.7)**

**Requirement already satisfied: pillow>=8 in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from matplotlib->mediapipe) (11.0.0)**

**Requirement already satisfied: pyparsing>=2.3.1 in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from matplotlib->mediapipe) (3.2.0)**

**Requirement already satisfied: python-dateutil>=2.7 in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from matplotlib->mediapipe) (2.9.0.post0)**

**Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from astunparse>=1.6.0->tensorflow-intel==2.18.0->tensorflow) (0.45.1)**

**Requirement already satisfied: pycparser in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from CFFI>=1.0->sounddevice>=0.4.4->mediapipe) (2.22)**

**Requirement already satisfied: rich in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from keras>=3.5.0->tensorflow-intel==2.18.0->tensorflow) (13.9.4)**

**Requirement already satisfied: namex in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from keras>=3.5.0->tensorflow-intel==2.18.0->tensorflow) (0.0.8)**

**Requirement already satisfied: optree in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from keras>=3.5.0->tensorflow-intel==2.18.0->tensorflow) (0.13.1)**

**Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from requests<3,>=2.21.0->tensorflow-intel==2.18.0->tensorflow) (3.4.0)**

**Requirement already satisfied: idna<4,>=2.5 in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from requests<3,>=2.21.0->tensorflow-intel==2.18.0->tensorflow) (3.10)**

**Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from requests<3,>=2.21.0->tensorflow-intel==2.18.0->tensorflow) (2.2.3)**

**Requirement already satisfied: certifi>=2017.4.17 in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from requests<3,>=2.21.0->tensorflow-intel==2.18.0->tensorflow) (2024.8.30)**

**Requirement already satisfied: markdown>=2.6.8 in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from tensorboard<2.19,>=2.18->tensorflow-intel==2.18.0->tensorflow) (3.7)**

**Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from tensorboard<2.19,>=2.18->tensorflow-intel==2.18.0->tensorflow) (0.7.2)**

**Requirement already satisfied: werkzeug>=1.0.1 in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from tensorboard<2.19,>=2.18->tensorflow-intel==2.18.0->tensorflow) (3.1.3)**

**Requirement already satisfied: MarkupSafe>=2.1.1 in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from werkzeug>=1.0.1->tensorboard<2.19,>=2.18->tensorflow-intel==2.18.0->tensorflow) (3.0.2)**

**Requirement already satisfied: markdown-it-py>=2.2.0 in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from rich->keras>=3.5.0->tensorflow-intel==2.18.0->tensorflow) (3.0.0)**

**Requirement already satisfied: pygments<3.0.0,>=2.13.0 in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from rich->keras>=3.5.0->tensorflow-intel==2.18.0->tensorflow) (2.18.0)**

**Requirement already satisfied: mdurl~=0.1 in c:\users\nebhw\appdata\roaming\python\python312\site-packages (from markdown-it-py>=2.2.0->rich->keras>=3.5.0->tensorflow-intel==2.18.0->tensorflow) (0.1.2)**

**Note: you may need to restart the kernel to use updated packages.**

**[notice] A new release of pip is available: 24.2 -> 25.1.1**

**[notice] To update, run: python.exe -m pip install --upgrade pip**

In [ ]:

%run userInterface.py

**WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile\_metrics` will be empty until you train or evaluate the model.**

**1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 209ms/step**

**Predicted Sign: Afternoon (99.91%)**

**1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 162ms/step**

**Predicted Sign: Afternoon (99.94%)**

**1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 156ms/step**

**Predicted Sign: Afternoon (99.72%)**

**1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 154ms/step**

**Predicted Sign: Afternoon (99.99%)**

**1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 165ms/step**

**Predicted Sign: Afternoon (100.00%)**

**1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 163ms/step**

**Predicted Sign: Afternoon (87.45%)**

**1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 158ms/step**

**Predicted Sign: Evening (87.07%)**

**1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 155ms/step**

**Predicted Sign: Afternoon (26.70%)**

**1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 198ms/step**

**Predicted Sign: Night (77.93%)**

**1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 154ms/step**

**Predicted Sign: Morning (81.96%)**

**1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 157ms/step**

**Predicted Sign: Morning (85.60%)**

**1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 163ms/step**

**Predicted Sign: Morning (88.95%)**

**In [ ]:**

**feature\_extraction/feature\_extraction.py**

import cv2

import os

import numpy as np

def extract\_video\_frames(video\_path, num\_frames=30, target\_size=(224, 224)):

cap = cv2.VideoCapture(video\_path)

total\_frames = int(cap.get(cv2.CAP\_PROP\_FRAME\_COUNT))

step = max(total\_frames // num\_frames, 1)

frames = []

count = 0

while len(frames) < num\_frames and cap.isOpened():

ret, frame = cap.read()

if not ret:

break

if count % step == 0:

frame = cv2.resize(frame, target\_size)

frame = frame / 255.0 # Normalize pixel values

frames.append(frame)

count += 1

cap.release()

# Pad if fewer frames

while len(frames) < num\_frames:

frames.append(frames[-1])

return frames # shape: (num\_frames, H, W, 3)

DATASET\_PATH = "data/Days\_and\_Time\_skeleton"

NUM\_FRAMES = 30

TARGET\_SIZE = (224, 224)

def combine\_features():

X, y = [], []

class\_names = sorted([

name for name in os.listdir(DATASET\_PATH)

if os.path.isdir(os.path.join(DATASET\_PATH, name)) and not name.startswith('.')

])

class\_map = {name: name for idx, name in enumerate(class\_names)}

for class\_name in class\_names:

class\_path = os.path.join(DATASET\_PATH, class\_name)

print(class\_name)

for file in os.listdir(class\_path):

if file.endswith(".mp4"):

print("heloo")

video\_path = os.path.join(class\_path, file)

frames = extract\_video\_frames(video\_path, NUM\_FRAMES, TARGET\_SIZE)

X.append(frames)

y.append(class\_map[class\_name])

X = np.array(X) # shape: (num\_videos, num\_frames, H, W, 3)

y = np.array(y) # shape: (num\_videos,)

print(X.shape)

np.save("X.npy", X)

np.save("y.npy", y)

**mediapipe/mp.py**

import cv2

import mediapipe as mp

import os

import numpy as np

# Initialize MediaPipe Holistic and Drawing

mp\_drawing = mp.solutions.drawing\_utils

mp\_holistic = mp.solutions.holistic

# Input/output video paths

def use\_mediapipe(dataset = "Days\_and\_Time"):

dataPath = os.path.join("data",dataset);

output = os.path.join("data",dataset + "\_skeleton")

for className in os.listdir(dataPath):

classPath = os.path.join(dataPath,className)

outputPath = os.path.join(output,className)

os.makedirs(outputPath,exist\_ok = True)

for video in os.listdir(classPath):

output\_video\_path = os.path.join(outputPath, os.path.splitext(video)[0] + "\_skeleton.mp4")

# Capture input video

if(video.endswith(".MOV")):

input\_video\_path = os.path.join(classPath,video)

print(input\_video\_path)

cap = cv2.VideoCapture(input\_video\_path)

width = int(cap.get(cv2.CAP\_PROP\_FRAME\_WIDTH))

height = int(cap.get(cv2.CAP\_PROP\_FRAME\_HEIGHT))

fps = cap.get(cv2.CAP\_PROP\_FPS)

# Video writer

fourcc = cv2.VideoWriter\_fourcc(\*'mp4v')

out = cv2.VideoWriter(output\_video\_path, fourcc, fps, (width, height))

# Initialize Holistic model

with mp\_holistic.Holistic(

static\_image\_mode=False,

model\_complexity=1,

enable\_segmentation=False,

refine\_face\_landmarks=True,

min\_detection\_confidence=0.5,

min\_tracking\_confidence=0.5

) as holistic:

while cap.isOpened():

ret, frame = cap.read()

if not ret:

break

# Convert BGR to RGB

image\_rgb = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

# Create white background

skeleton\_frame = 255 \* np.ones((height, width, 3), dtype=np.uint8)

# Process the frame

results = holistic.process(image\_rgb)

#Draw landmarks on the original BGR frame

mp\_drawing.draw\_landmarks(skeleton\_frame, results.face\_landmarks, mp\_holistic.FACEMESH\_TESSELATION)

mp\_drawing.draw\_landmarks(skeleton\_frame, results.left\_hand\_landmarks, mp\_holistic.HAND\_CONNECTIONS)

mp\_drawing.draw\_landmarks(skeleton\_frame, results.right\_hand\_landmarks, mp\_holistic.HAND\_CONNECTIONS)

mp\_drawing.draw\_landmarks(skeleton\_frame, results.pose\_landmarks, mp\_holistic.POSE\_CONNECTIONS)

# Write frame to output video

out.write(skeleton\_frame)

cap.release()

out.release()

print("Class " + className + " done")

print("✅ Holistic video created successfully!")

**models/model.py**

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv3D, MaxPooling3D, Flatten, Dense, Dropout

from tensorflow.keras.optimizers import Adam

def build\_spatiotemporal\_cnn(input\_shape=(30, 224, 224, 3), num\_classes=8):

model = Sequential()

model.add(Conv3D(32, kernel\_size=(3, 3, 3), activation='relu', input\_shape=input\_shape))

model.add(MaxPooling3D(pool\_size=(1, 2, 2)))

model.add(Conv3D(64, kernel\_size=(3, 3, 3), activation='relu'))

model.add(MaxPooling3D(pool\_size=(2, 2, 2)))

model.add(Conv3D(128, kernel\_size=(3, 3, 3), activation='relu'))

model.add(MaxPooling3D(pool\_size=(2, 2, 2)))

model.add(Flatten())

model.add(Dense(256, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(num\_classes, activation='softmax'))

model.compile(optimizer=Adam(learning\_rate=0.0001),

loss='sparse\_categorical\_crossentropy',

metrics=['accuracy'])

return model

**userInterface.py**

import cv2

import numpy as np

import mediapipe as mp

from tensorflow.keras.models import load\_model

from collections import deque

import time

# Load model

model = load\_model("my\_sign\_model.h5")

class\_names = ['Afternoon','Evening','Morning','Night']

# Setup MediaPipe

mp\_drawing = mp.solutions.drawing\_utils

mp\_holistic = mp.solutions.holistic

# Webcam

cap = cv2.VideoCapture(0)

SEQUENCE\_LENGTH = 30

frame\_buffer = deque(maxlen=SEQUENCE\_LENGTH)

collecting = False

countdown\_started = False

countdown\_start\_time = None

countdown\_duration = 3

prediction\_made = False

with mp\_holistic.Holistic(

static\_image\_mode=False,

model\_complexity=1,

enable\_segmentation=False,

refine\_face\_landmarks=True) as holistic:

while cap.isOpened():

ret, frame = cap.read()

if not ret:

break

frame = cv2.flip(frame, 1)

rgb = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

results = holistic.process(rgb)

white\_canvas = 255 \* np.ones\_like(frame)

mp\_drawing.draw\_landmarks(white\_canvas, results.face\_landmarks, mp\_holistic.FACEMESH\_TESSELATION)

mp\_drawing.draw\_landmarks(white\_canvas, results.pose\_landmarks, mp\_holistic.POSE\_CONNECTIONS)

mp\_drawing.draw\_landmarks(white\_canvas, results.left\_hand\_landmarks, mp\_holistic.HAND\_CONNECTIONS)

mp\_drawing.draw\_landmarks(white\_canvas, results.right\_hand\_landmarks, mp\_holistic.HAND\_CONNECTIONS)

key = cv2.waitKey(1) & 0xFF

if key == ord(' '):

collecting = True

countdown\_started = True

countdown\_start\_time = time.time()

prediction\_made = False

frame\_buffer.clear()

if key == ord('q'):

break

# Countdown logic

if countdown\_started:

elapsed = time.time() - countdown\_start\_time

remaining = int(countdown\_duration - elapsed)

if remaining > 0:

cv2.putText(white\_canvas, f"Starting in {remaining}", (60, 60), cv2.FONT\_HERSHEY\_SIMPLEX, 1.5, (0, 0, 255), 4)

cv2.imshow('Landmark View', white\_canvas)

continue

else:

countdown\_started = False

if collecting:

processed\_frame = cv2.resize(white\_canvas, (224, 224))

frame\_buffer.append(processed\_frame)

if len(frame\_buffer) == SEQUENCE\_LENGTH and not prediction\_made:

input\_data = np.array(frame\_buffer, dtype=np.float32) / 255.0

input\_data = np.expand\_dims(input\_data, axis=0)

prediction = model.predict(input\_data)[0]

predicted\_index = np.argmax(prediction)

predicted\_class = class\_names[predicted\_index]

confidence = prediction[predicted\_index] \* 100

prediction\_made = True

collecting = False # Stop collecting

print(f"Predicted Sign: {predicted\_class} ({confidence:.2f}%)")

# Display prediction if available

if prediction\_made:

cv2.putText(white\_canvas, f'Sign: {predicted\_class} ({confidence:.2f}%)',

(10, 40), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 0, 255), 3)

cv2.imshow('Landmark View', white\_canvas)

cv2.waitKey(5000)

cv2.imshow('Landmark View', white\_canvas)

cap.release()

cv2.destroyAllWindows()