COMPARISON_MediaPipe+CNN+LSTM

June 21, 2025

[1]:
```
from modules.SignLanguageProcessor import load_and_preprocess_data,parse_frame
import os
```

[2]:
```
ROOT_PATH = ''
sequences,labels,label_map = load_and_preprocess_data(os.path.
 ↪join(ROOT_PATH,'data'))
```

[3]:
```
num_classes = len(label_map)
```

[4]:
```
len(labels)
```

[4]: 5643

[5]:
```
sequences.shape
```

[5]: (5643, 3, 61, 3)

[6]:
```
from sklearn.model_selection import train_test_split

X_train, X_temp, y_train, y_temp = train_test_split(
    sequences, labels, test_size=0.4, stratify=labels, random_state=42
)

X_val, X_test, y_val, y_test = train_test_split(
    X_temp, y_temp, test_size=0.5, stratify=y_temp, random_state=42
)
```

[7]:
```
import numpy as np
def normalize_landmark_data(X):
    """
    Normalize the landmark features (x, y) to have zero mean and unit variance␣
 ↪across the training set.
    Assumes X shape is (N, F, L, T), where F=3 (x, y, vis).
    """
    X = X.copy()
    # Flatten across all samples, landmarks, and frames
    x_vals = X[:, 0, :, :].flatten()
    y_vals = X[:, 1, :, :].flatten()
```

```
    # Compute mean and std
    x_mean, x_std = np.mean(x_vals), np.std(x_vals)
    y_mean, y_std = np.mean(y_vals), np.std(y_vals)

    # Normalize
    X[:, 0, :, :] = (X[:, 0, :, :] - x_mean) / x_std
    X[:, 1, :, :] = (X[:, 1, :, :] - y_mean) / y_std

    return X, (x_mean, x_std), (y_mean, y_std)

def apply_normalization(X, x_mean, x_std, y_mean, y_std):
    X = X.copy()
    X[:, 0, :, :] = (X[:, 0, :, :] - x_mean) / x_std
    X[:, 1, :, :] = (X[:, 1, :, :] - y_mean) / y_std
    return X
```

[8]:
```
def reshape_frames_for_cnn(X, y):
    X = X.transpose(0, 3, 2, 1)  # (N, T, L, F)
    X = X[..., np.newaxis]       # (N, T, L, F, 1)
    return X,y
```

[9]:
```
X_train_norm, (x_mean, x_std), (y_mean, y_std) = 
 ↪normalize_landmark_data(X_train)
X_val_norm  = apply_normalization(X_val, x_mean, x_std, y_mean, y_std)
X_test_norm = apply_normalization(X_test, x_mean, x_std, y_mean, y_std)

X_train_cnn, y_train_cnn = reshape_frames_for_cnn(X_train_norm, y_train)
X_val_cnn, y_val_cnn     = reshape_frames_for_cnn(X_val_norm, y_val)
X_test_cnn, y_test_cnn   = reshape_frames_for_cnn(X_test_norm, y_test)

print(X_train_cnn.shape)
print(y_train_cnn.shape)
```

```
(3385, 3, 61, 3, 1)
(3385,)
```

[10]:
```
input_shape = X_train_cnn.shape[1:]
print(input_shape)
```

```
(3, 61, 3, 1)
```

[11]:
```
import tensorflow as tf

train_ds = tf.data.Dataset.from_tensor_slices((X_train_cnn, y_train_cnn))
train_ds = train_ds.shuffle(buffer_size=1000).batch(64).prefetch(tf.data.
 ↪AUTOTUNE)
```

```
val_ds = tf.data.Dataset.from_tensor_slices((X_val_cnn, y_val_cnn))
val_ds = val_ds.batch(64).prefetch(tf.data.AUTOTUNE)

test_ds = tf.data.Dataset.from_tensor_slices((X_test_cnn, y_test_cnn))
test_ds = test_ds.batch(64).prefetch(tf.data.AUTOTUNE)
```

[19]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import TimeDistributed, Conv2D, MaxPooling2D,␣
↪Flatten,Input
from tensorflow.keras.layers import LSTM, Dropout, Dense, BatchNormalization

model = Sequential([
    Input((3, 61, 3, 1)),
    TimeDistributed(Conv2D(32, (3, 2), activation='relu', padding='same')),
    TimeDistributed(BatchNormalization()),
    TimeDistributed(MaxPooling2D(pool_size=(2, 1))),
    TimeDistributed(Dropout(0.25)),

    TimeDistributed(Conv2D(64, (3, 2), activation='relu', padding='same')),
    TimeDistributed(BatchNormalization()),
    TimeDistributed(MaxPooling2D(pool_size=(2, 1))),
    TimeDistributed(Flatten()),

    LSTM(128, return_sequences=False),
    Dropout(0.5),
    Dense(num_classes, activation='softmax')
])

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',␣
↪metrics=['accuracy'])
```

[20]:
```
history = model.fit(train_ds,validation_data=val_ds, epochs=50, batch_size=64)
```

```
Epoch 1/50
53/53                 7s 60ms/step -
accuracy: 0.0507 - loss: 3.6923 - val_accuracy: 0.0806 - val_loss: 3.4596
Epoch 2/50
53/53                 3s 51ms/step -
accuracy: 0.1152 - loss: 3.2279 - val_accuracy: 0.0912 - val_loss: 3.1668
Epoch 3/50
53/53                 3s 48ms/step -
accuracy: 0.1367 - loss: 3.0772 - val_accuracy: 0.1293 - val_loss: 3.0201
Epoch 4/50
53/53                 3s 51ms/step -
accuracy: 0.1592 - loss: 2.9782 - val_accuracy: 0.1453 - val_loss: 2.9099
Epoch 5/50
53/53                 3s 52ms/step -
accuracy: 0.1810 - loss: 2.8905 - val_accuracy: 0.2002 - val_loss: 2.8233
```

```
Epoch 6/50
53/53            3s 48ms/step -
accuracy: 0.2183 - loss: 2.7815 - val_accuracy: 0.2117 - val_loss: 2.7393
Epoch 7/50
53/53            3s 47ms/step -
accuracy: 0.2265 - loss: 2.7112 - val_accuracy: 0.2427 - val_loss: 2.6644
Epoch 8/50
53/53            3s 54ms/step -
accuracy: 0.2851 - loss: 2.5682 - val_accuracy: 0.2799 - val_loss: 2.6393
Epoch 9/50
53/53            3s 51ms/step -
accuracy: 0.3132 - loss: 2.4698 - val_accuracy: 0.3003 - val_loss: 2.5923
Epoch 10/50
53/53            3s 51ms/step -
accuracy: 0.3256 - loss: 2.4252 - val_accuracy: 0.3020 - val_loss: 2.4991
Epoch 11/50
53/53            3s 50ms/step -
accuracy: 0.3545 - loss: 2.3281 - val_accuracy: 0.3029 - val_loss: 2.4149
Epoch 12/50
53/53            3s 54ms/step -
accuracy: 0.3917 - loss: 2.1795 - val_accuracy: 0.3339 - val_loss: 2.3556
Epoch 13/50
53/53            3s 49ms/step -
accuracy: 0.4134 - loss: 2.0929 - val_accuracy: 0.4039 - val_loss: 2.2474
Epoch 14/50
53/53            3s 51ms/step -
accuracy: 0.4289 - loss: 2.0593 - val_accuracy: 0.4172 - val_loss: 2.1792
Epoch 15/50
53/53            3s 52ms/step -
accuracy: 0.4478 - loss: 1.9414 - val_accuracy: 0.4198 - val_loss: 2.1114
Epoch 16/50
53/53            3s 54ms/step -
accuracy: 0.4768 - loss: 1.8474 - val_accuracy: 0.4057 - val_loss: 2.1733
Epoch 17/50
53/53            3s 55ms/step -
accuracy: 0.4839 - loss: 1.8162 - val_accuracy: 0.4756 - val_loss: 1.9977
Epoch 18/50
53/53            3s 57ms/step -
accuracy: 0.5030 - loss: 1.7098 - val_accuracy: 0.4774 - val_loss: 1.9651
Epoch 19/50
53/53            3s 53ms/step -
accuracy: 0.5434 - loss: 1.5988 - val_accuracy: 0.5102 - val_loss: 1.8605
Epoch 20/50
53/53            3s 61ms/step -
accuracy: 0.5671 - loss: 1.5555 - val_accuracy: 0.5261 - val_loss: 1.8027
Epoch 21/50
53/53            3s 61ms/step -
accuracy: 0.5879 - loss: 1.4889 - val_accuracy: 0.5536 - val_loss: 1.6867
```

```
Epoch 22/50
53/53            3s 54ms/step -
accuracy: 0.5861 - loss: 1.4140 - val_accuracy: 0.5757 - val_loss: 1.6500
Epoch 23/50
53/53            3s 56ms/step -
accuracy: 0.6328 - loss: 1.3359 - val_accuracy: 0.6058 - val_loss: 1.5774
Epoch 24/50
53/53            4s 74ms/step -
accuracy: 0.6420 - loss: 1.3187 - val_accuracy: 0.5421 - val_loss: 1.6943
Epoch 25/50
53/53            4s 78ms/step -
accuracy: 0.6433 - loss: 1.2622 - val_accuracy: 0.5270 - val_loss: 1.6817
Epoch 26/50
53/53            4s 79ms/step -
accuracy: 0.6593 - loss: 1.2080 - val_accuracy: 0.5226 - val_loss: 1.6613
Epoch 27/50
53/53            4s 78ms/step -
accuracy: 0.6716 - loss: 1.1873 - val_accuracy: 0.6058 - val_loss: 1.5274
Epoch 28/50
53/53            4s 71ms/step -
accuracy: 0.6944 - loss: 1.1506 - val_accuracy: 0.5757 - val_loss: 1.5156
Epoch 29/50
53/53            3s 60ms/step -
accuracy: 0.7052 - loss: 1.1038 - val_accuracy: 0.5864 - val_loss: 1.5028
Epoch 30/50
53/53            3s 60ms/step -
accuracy: 0.7004 - loss: 1.0738 - val_accuracy: 0.6050 - val_loss: 1.4674
Epoch 31/50
53/53            3s 58ms/step -
accuracy: 0.7274 - loss: 0.9806 - val_accuracy: 0.6023 - val_loss: 1.4380
Epoch 32/50
53/53            3s 47ms/step -
accuracy: 0.7023 - loss: 1.0166 - val_accuracy: 0.5846 - val_loss: 1.4293
Epoch 33/50
53/53            3s 48ms/step -
accuracy: 0.7274 - loss: 0.9633 - val_accuracy: 0.6484 - val_loss: 1.3223
Epoch 34/50
53/53            3s 52ms/step -
accuracy: 0.7472 - loss: 0.9066 - val_accuracy: 0.6457 - val_loss: 1.3168
Epoch 35/50
53/53            3s 58ms/step -
accuracy: 0.7527 - loss: 0.8753 - val_accuracy: 0.6572 - val_loss: 1.2775
Epoch 36/50
53/53            3s 60ms/step -
accuracy: 0.7483 - loss: 0.8848 - val_accuracy: 0.6900 - val_loss: 1.2040
Epoch 37/50
53/53            4s 69ms/step -
accuracy: 0.7688 - loss: 0.8282 - val_accuracy: 0.6962 - val_loss: 1.1737
```

```
Epoch 38/50
53/53            3s 60ms/step -
accuracy: 0.7735 - loss: 0.8184 - val_accuracy: 0.6811 - val_loss: 1.1797
Epoch 39/50
53/53            3s 61ms/step -
accuracy: 0.7750 - loss: 0.7760 - val_accuracy: 0.6829 - val_loss: 1.1854
Epoch 40/50
53/53            3s 58ms/step -
accuracy: 0.7845 - loss: 0.7461 - val_accuracy: 0.6643 - val_loss: 1.2882
Epoch 41/50
53/53            3s 59ms/step -
accuracy: 0.7848 - loss: 0.7711 - val_accuracy: 0.6528 - val_loss: 1.3329
Epoch 42/50
53/53            3s 61ms/step -
accuracy: 0.7865 - loss: 0.7324 - val_accuracy: 0.6696 - val_loss: 1.2521
Epoch 43/50
53/53            3s 60ms/step -
accuracy: 0.7881 - loss: 0.7324 - val_accuracy: 0.6501 - val_loss: 1.2915
Epoch 44/50
53/53            3s 58ms/step -
accuracy: 0.8081 - loss: 0.6850 - val_accuracy: 0.6776 - val_loss: 1.1597
Epoch 45/50
53/53            3s 55ms/step -
accuracy: 0.8023 - loss: 0.6634 - val_accuracy: 0.6980 - val_loss: 1.1118
Epoch 46/50
53/53            3s 63ms/step -
accuracy: 0.7997 - loss: 0.6798 - val_accuracy: 0.6599 - val_loss: 1.1908
Epoch 47/50
53/53            3s 57ms/step -
accuracy: 0.8149 - loss: 0.6753 - val_accuracy: 0.6634 - val_loss: 1.1857
Epoch 48/50
53/53            3s 57ms/step -
accuracy: 0.8166 - loss: 0.6374 - val_accuracy: 0.6475 - val_loss: 1.2377
Epoch 49/50
53/53            3s 51ms/step -
accuracy: 0.8062 - loss: 0.6589 - val_accuracy: 0.6492 - val_loss: 1.2095
Epoch 50/50
53/53            3s 56ms/step -
accuracy: 0.8419 - loss: 0.5885 - val_accuracy: 0.6767 - val_loss: 1.1418
```

[21]:

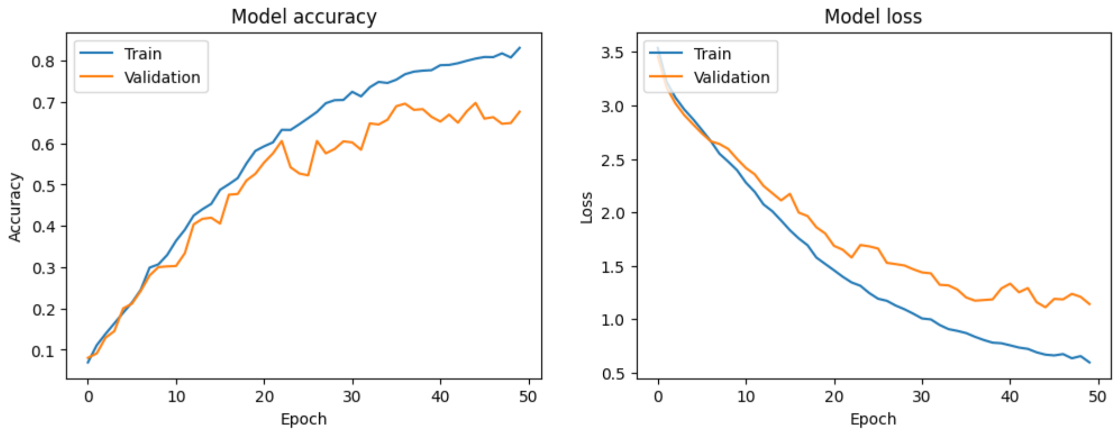
```
test_loss, test_accuracy = model.evaluate(test_ds)
print(f"Test Accuracy: {test_accuracy:.4f}")
print(f"Test Loss: {test_loss:.4f}")
```

```
 1/18            0s 17ms/step -
accuracy: 0.6094 - loss: 1.1868

18/18            0s 9ms/step -
accuracy: 0.6504 - loss: 1.1274
```

```
Test Accuracy: 0.6723
Test Loss: 1.0921
```

[22]:
```python
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
```

[23]:
```python
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
# Plot training & validation loss values
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```



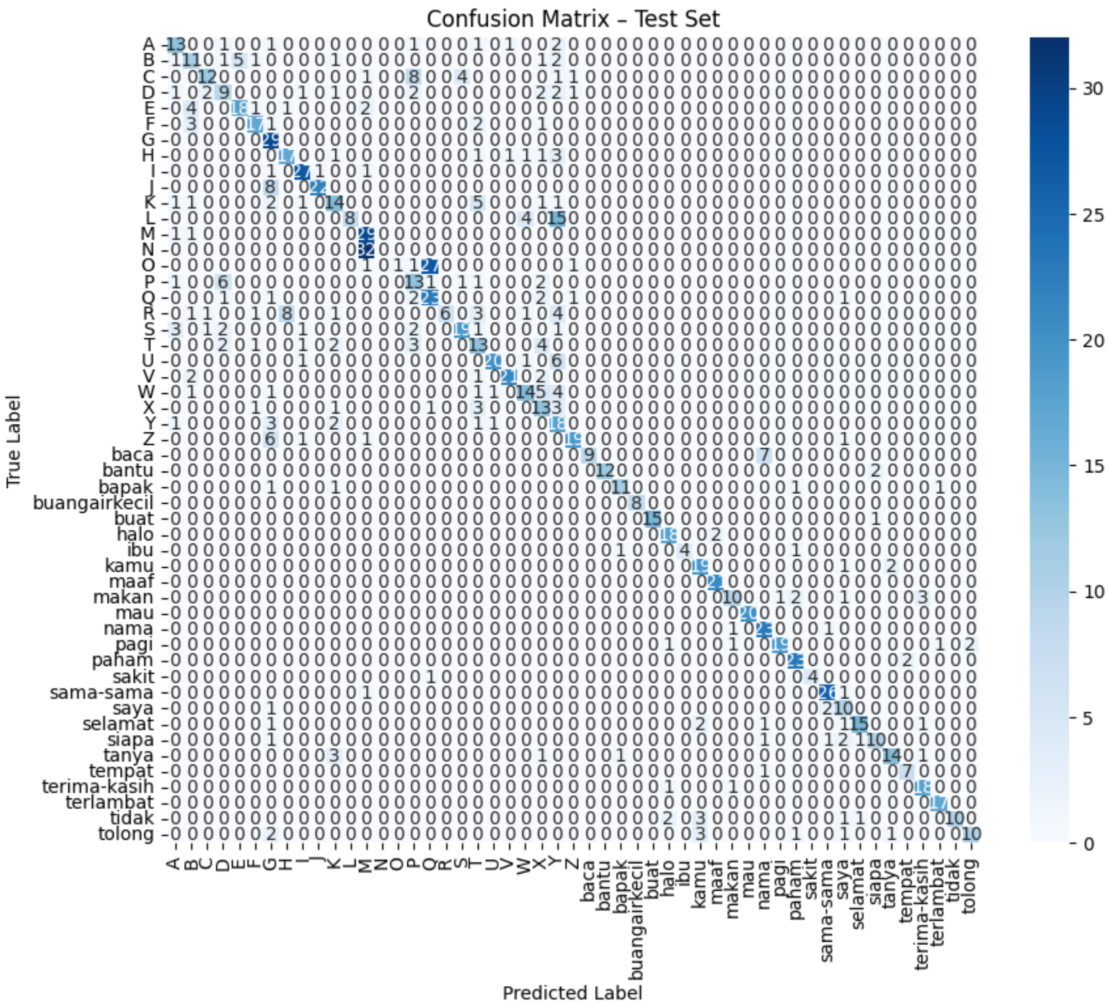
[24]:
```python
y_true, y_pred = [], []
target_names = [label_map[i] for i in range(len(label_map))]
for X_batch, y_batch in test_ds:
    y_true.append(y_batch.numpy())

    batch_pred = model.predict(X_batch, verbose=0)
    y_pred.append(np.argmax(batch_pred, axis=1))
```

```python
y_true = np.concatenate(y_true)
y_pred = np.concatenate(y_pred)

print(classification_report(
    y_true, y_pred,
    digits=3,
    target_names=target_names
))

cm = confusion_matrix(y_true, y_pred, labels=range(len(label_map)))
labels = [label_map[i] for i in range(len(label_map))]

plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=labels, yticklabels=labels)
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix - Test Set")
plt.show()
```

```
c:\Users\chris\.conda\envs\ASLR\Lib\site-
packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
c:\Users\chris\.conda\envs\ASLR\Lib\site-
packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
c:\Users\chris\.conda\envs\ASLR\Lib\site-
packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
              precision    recall  f1-score   support

           A      0.591     0.650     0.619        20
           B      0.458     0.478     0.468        23
           C      0.750     0.444     0.558        27
           D      0.409     0.429     0.419        21
           E      0.783     0.692     0.735        26
           F      0.773     0.708     0.739        24
           G      0.492     1.000     0.659        29
           H      0.654     0.680     0.667        25
           I      0.818     0.900     0.857        30
```

J	0.957	0.733	0.830	30
K	0.519	0.538	0.528	26
L	1.000	0.296	0.457	27
M	0.426	0.935	0.586	31
N	0.000	0.000	0.000	32
O	1.000	0.032	0.062	31
P	0.406	0.520	0.456	25
Q	0.434	0.742	0.548	31
R	1.000	0.231	0.375	26
S	0.792	0.633	0.704	30
T	0.394	0.500	0.441	26
U	0.909	0.714	0.800	28
V	0.913	0.808	0.857	26
W	0.667	0.519	0.583	27
X	0.371	0.591	0.456	22
Y	0.290	0.692	0.409	26
Z	0.826	0.679	0.745	28
baca	1.000	0.562	0.720	16
bantu	1.000	0.857	0.923	14
bapak	0.846	0.733	0.786	15
buangairkecil	1.000	1.000	1.000	8
buat	1.000	0.938	0.968	16
halo	0.818	0.900	0.857	20
ibu	1.000	0.667	0.800	6
kamu	0.704	0.864	0.776	22
maaf	0.913	1.000	0.955	21
makan	0.769	0.588	0.667	17
mau	1.000	1.000	1.000	20
nama	0.697	0.920	0.793	25
pagi	0.950	0.792	0.864	24
paham	0.821	0.920	0.868	25
sakit	1.000	0.800	0.889	5
sama-sama	0.867	0.929	0.897	28
saya	0.500	0.769	0.606	13
selamat	0.882	0.714	0.789	21
siapa	0.769	0.625	0.690	16
tanya	0.824	0.700	0.757	20
tempat	0.778	0.875	0.824	8
terima-kasih	0.783	0.900	0.837	20
terlambat	0.895	1.000	0.944	17
tidak	1.000	0.588	0.741	17
tolong	0.833	0.556	0.667	18
accuracy			0.672	1129
macro avg	0.751	0.693	0.690	1129
weighted avg	0.725	0.672	0.660	1129


Confusion Matrix – Test Set
True Label
Predicted Label