In [ ]:
```
import numpy as np
```

In [ ]:
```
import tensorflow as tf
```

In [ ]:
```
from tensorflow.keras.models import Sequential
```

In [ ]:
```
from tensorflow.keras.layers import LSTM, Dense, Dropout
```

In [ ]:
```
from tensorflow.keras.optimizers import Adam
```

In [ ]:
```
X_train = np.load("X_train.npy")
```

In [ ]:
```
y_train = np.load("y_train.npy")
```

In [ ]:
```
X_val = np.load("X_val.npy")
```

In [ ]:
```
y_val = np.load("y_val.npy")
```

In [ ]:
```
X_test = np.load("X_test.npy")
```

In [ ]:
```
y_test = np.load("y_test.npy")
```

In [ ]:
```
print(f"Training Data Shape: {X_train.shape}")
print(f"Training Labels Shape: {y_train.shape}")
print(f"Validation Data Shape: {X_val.shape}")
print(f"Validation Labels Shape: {y_val.shape}")
print(f"Testing Data Shape: {X_test.shape}")
print(f"Testing Labels Shape: {y_test.shape}")
```

```
Training Data Shape: (14000, 63)
Training Labels Shape: (14000,)
Validation Data Shape: (3000, 63)
Validation Labels Shape: (3000,)
Testing Data Shape: (3000, 63)
Testing Labels Shape: (3000,)
```

In [ ]:
```
X_train = X_train.reshape((X_train.shape[0], 1, X_train.shape[1]))  # 1 timestep
```

In [ ]:
```
X_val = X_val.reshape((X_val.shape[0], 1, X_val.shape[1]))  # 1 timestep
```

In [ ]:
```
X_test = X_test.reshape((X_test.shape[0], 1, X_test.shape[1]))  # 1 timestep
```

In [ ]:
```
model = Sequential()
```

In [ ]:
```
model.add(LSTM(128, input_shape=(X_train.shape[1], X_train.shape[2]), return_sequen
```

```
C:\Users\anshu\AppData\Roaming\Python\Python312\site-packages\keras\src\layers\rnn\r
nn.py:204: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a laye
r. When using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)
```

In [ ]:
```
model.add(Dropout(0.2))  # Dropout layer for regularization
```

In [ ]:
```
model.add(LSTM(64, return_sequences=False))
```

In [ ]:
```
model.add(Dropout(0.2))  # Dropout layer for regularization
```

In [ ]:
```
model.add(Dense(64, activation='relu'))
```

In [ ]:
```
model.add(Dropout(0.2))  # Dropout layer for regularization
```

In [ ]:
```
model.add(Dense(5, activation='softmax'))  # 5 output classes for the gestures
```

In [ ]:
```
model.compile(optimizer=Adam(), loss='sparse_categorical_crossentropy', metrics=['a
```

In [ ]:
```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 1, 128)	98,304
dropout (Dropout)	(None, 1, 128)	0
lstm_1 (LSTM)	(None, 64)	49,408
dropout_1 (Dropout)	(None, 64)	0
dense (Dense)	(None, 64)	4,160
dropout_2 (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 5)	325

Total params: 152,197 (594.52 KB)

Trainable params: 152,197 (594.52 KB)

Non-trainable params: 0 (0.00 B)

In [ ]:
```
history = model.fit(X_train, y_train, epochs=50, batch_size=64, validation_data=(X_
```

Loading [MathJax]/extensions/Safe.js

```
Epoch 1/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.3764 - loss: 1.3211 - val_acc
uracy: 0.7880 - val_loss: 0.4641
Epoch 2/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.7502 - loss: 0.5048 - val_acc
uracy: 0.8347 - val_loss: 0.3550
Epoch 3/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8173 - loss: 0.3822 - val_acc
uracy: 0.8777 - val_loss: 0.2824
Epoch 4/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8588 - loss: 0.3148 - val_acc
uracy: 0.8800 - val_loss: 0.2395
Epoch 5/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8797 - loss: 0.2517 - val_acc
uracy: 0.9187 - val_loss: 0.1796
Epoch 6/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9062 - loss: 0.2052 - val_acc
uracy: 0.9357 - val_loss: 0.1563
Epoch 7/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9217 - loss: 0.1868 - val_acc
uracy: 0.9480 - val_loss: 0.1351
Epoch 8/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9339 - loss: 0.1586 - val_acc
uracy: 0.9560 - val_loss: 0.1192
Epoch 9/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9378 - loss: 0.1549 - val_acc
uracy: 0.9597 - val_loss: 0.1160
Epoch 10/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9475 - loss: 0.1375 - val_acc
uracy: 0.9657 - val_loss: 0.0974
Epoch 11/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9515 - loss: 0.1200 - val_acc
uracy: 0.9627 - val_loss: 0.0985
Epoch 12/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9570 - loss: 0.1150 - val_acc
uracy: 0.9337 - val_loss: 0.1331
Epoch 13/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9553 - loss: 0.1171 - val_acc
uracy: 0.9643 - val_loss: 0.0893
Epoch 14/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9660 - loss: 0.0935 - val_acc
uracy: 0.9630 - val_loss: 0.0888
Epoch 15/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9647 - loss: 0.0971 - val_acc
uracy: 0.9713 - val_loss: 0.0723
Epoch 16/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9663 - loss: 0.0918 - val_acc
uracy: 0.9753 - val_loss: 0.0695
Epoch 17/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9678 - loss: 0.0871 - val_acc
uracy: 0.9710 - val_loss: 0.0698
Epoch 18/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9699 - loss: 0.0787 - val_acc
uracy: 0.9703 - val_loss: 0.0693
Epoch 19/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9678 - loss: 0.0866 - val_acc
```

```
uracy: 0.9783 - val_loss: 0.0623
Epoch 20/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9647 - loss: 0.0828 - val_acc
uracy: 0.9723 - val_loss: 0.0704
Epoch 21/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9695 - loss: 0.0816 - val_acc
uracy: 0.9777 - val_loss: 0.0563
Epoch 22/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9720 - loss: 0.0767 - val_acc
uracy: 0.9753 - val_loss: 0.0608
Epoch 23/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9718 - loss: 0.0726 - val_acc
uracy: 0.9777 - val_loss: 0.0578
Epoch 24/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9759 - loss: 0.0667 - val_acc
uracy: 0.9790 - val_loss: 0.0574
Epoch 25/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9728 - loss: 0.0761 - val_acc
uracy: 0.9817 - val_loss: 0.0500
Epoch 26/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9768 - loss: 0.0649 - val_acc
uracy: 0.9830 - val_loss: 0.0441
Epoch 27/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9768 - loss: 0.0649 - val_acc
uracy: 0.9823 - val_loss: 0.0452
Epoch 28/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9798 - loss: 0.0582 - val_acc
uracy: 0.9850 - val_loss: 0.0444
Epoch 29/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9803 - loss: 0.0542 - val_acc
uracy: 0.9873 - val_loss: 0.0405
Epoch 30/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9797 - loss: 0.0577 - val_acc
uracy: 0.9873 - val_loss: 0.0399
Epoch 31/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9811 - loss: 0.0557 - val_acc
uracy: 0.9870 - val_loss: 0.0397
Epoch 32/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9814 - loss: 0.0521 - val_acc
uracy: 0.9803 - val_loss: 0.0593
Epoch 33/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9817 - loss: 0.0505 - val_acc
uracy: 0.9883 - val_loss: 0.0350
Epoch 34/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9830 - loss: 0.0458 - val_acc
uracy: 0.9833 - val_loss: 0.0419
Epoch 35/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9793 - loss: 0.0528 - val_acc
uracy: 0.9813 - val_loss: 0.0459
Epoch 36/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9844 - loss: 0.0479 - val_acc
uracy: 0.9877 - val_loss: 0.0385
Epoch 37/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9828 - loss: 0.0476 - val_acc
uracy: 0.9877 - val_loss: 0.0369
Epoch 38/50
```

```
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9839 - loss: 0.0435 - val_acc
uracy: 0.9897 - val_loss: 0.0322
Epoch 39/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9836 - loss: 0.0446 - val_acc
uracy: 0.9877 - val_loss: 0.0350
Epoch 40/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9863 - loss: 0.0382 - val_acc
uracy: 0.9917 - val_loss: 0.0298
Epoch 41/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9854 - loss: 0.0420 - val_acc
uracy: 0.9780 - val_loss: 0.0534
Epoch 42/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9859 - loss: 0.0407 - val_acc
uracy: 0.9873 - val_loss: 0.0354
Epoch 43/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9846 - loss: 0.0414 - val_acc
uracy: 0.9923 - val_loss: 0.0288
Epoch 44/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9856 - loss: 0.0383 - val_acc
uracy: 0.9903 - val_loss: 0.0323
Epoch 45/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9863 - loss: 0.0389 - val_acc
uracy: 0.9883 - val_loss: 0.0313
Epoch 46/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9864 - loss: 0.0346 - val_acc
uracy: 0.9897 - val_loss: 0.0293
Epoch 47/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9838 - loss: 0.0411 - val_acc
uracy: 0.9937 - val_loss: 0.0270
Epoch 48/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9889 - loss: 0.0339 - val_acc
uracy: 0.9917 - val_loss: 0.0302
Epoch 49/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9887 - loss: 0.0329 - val_acc
uracy: 0.9903 - val_loss: 0.0270
Epoch 50/50
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9885 - loss: 0.0334 - val_acc
uracy: 0.9937 - val_loss: 0.0229
```

In [ ]:
```
test_loss, test_acc = model.evaluate(X_test, y_test)
```

```
94/94 ━━━━━━━━━━━━━━━━━━━━ 0s 868us/step - accuracy: 0.9885 - loss: 0.0394
```

In [ ]:
```
print(f"Test Accuracy: {test_acc:.4f}")
```

```
Test Accuracy: 0.9927
```

In [ ]:
```
model.save('gesture_gesture_model_v3.h5')
```

```
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.savin
g.save_model(model, 'my_model.keras')`.
```

In [ ]:
```
print("Model saved to 'gesture_gesture_model_v3.h5'")
```

Loading [MathJax]/extensions/Safe.js

```
o 'gesture_gesture_model_v3.h5'
```