```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelBinarizer
import tensorflow as tf
from tensorflow.keras import layers, models
```

```
from google.colab import files
uploaded = files.upload()
```

Choose Files 2 files

- **sign_mnist_train.zip**(application/x-zip-compressed) - 25761228 bytes, last modified: 7/20/2025 - 100% done
- **sign_mnist_test.zip**(application/x-zip-compressed) - 6688512 bytes, last modified: 7/20/2025 - 100% done

```
Saving sign_mnist_train.zip to sign_mnist_train.zip
Saving sign_mnist_test.zip to sign_mnist_test.zip
```

```
import zipfile

with zipfile.ZipFile("sign_mnist_train.zip", 'r') as zip_ref:
    zip_ref.extractall()

with zipfile.ZipFile("sign_mnist_test.zip", 'r') as zip_ref:
    zip_ref.extractall()
```

```
import os
print(os.listdir())
```

```
['.config', 'sign_mnist_train.csv', 'sign_mnist_test.zip', 'sign_mnist_train.zip', 'sign_mnist_test.csv', 'sample_data']
```

```
import pandas as pd

train_df = pd.read_csv("sign_mnist_train.csv")
test_df = pd.read_csv("sign_mnist_test.csv")

print("Train shape:", train_df.shape)
print("Test shape:", test_df.shape)
train_df.head()
```

```
Train shape: (27455, 785)
Test shape: (7172, 785)
```

	label	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel775	pixel776	pixel777	pixel778	pixel779	p
0	3	107	118	127	134	139	143	146	150	153	...	207	207	207	207	206	
1	6	155	157	156	156	156	157	156	158	158	...	69	149	128	87	94	
2	2	187	188	188	187	187	186	187	188	187	...	202	201	200	199	198	
3	2	211	211	212	212	211	210	211	210	210	...	235	234	233	231	230	
4	13	164	167	170	172	176	179	180	184	185	...	92	105	105	108	133	

5 rows × 785 columns

```
import numpy as np
from sklearn.preprocessing import LabelBinarizer

X_train = train_df.drop("label", axis=1).values
y_train = train_df["label"].values

X_test = test_df.drop("label", axis=1).values
y_test = test_df["label"].values

X_train = X_train / 255.0
X_test = X_test / 255.0


X_train = X_train.reshape(-1, 28, 28, 1)
X_test = X_test.reshape(-1, 28, 28, 1)
X_test = X_test.reshape(-1, 28, 28, 1)

encoder = LabelBinarizer()
y_train = encoder.fit_transform(y_train)
y_test = encoder.transform(y_test)

print("X_train shape:", X_train.shape)
print("y_train shape:", y_train.shape)
```

```
X_train shape: (27455, 28, 28, 1)
y_train shape: (27455, 24)
```

```
import tensorflow as tf
from tensorflow.keras import layers, models

model = models.Sequential([
    layers.Conv2D(32, (3,3), activation='relu', input_shape=(28,28,1)),
    layers.MaxPooling2D(2,2),

    layers.Conv2D(64, (3,3), activation='relu'),
    layers.MaxPooling2D(2,2),

    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.3),
    layers.Dense(24, activation='softmax')  # 24 gesture classes (A–Y without J/Z)
])

model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.summary()
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`inpu
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Model: "sequential"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ conv2d (Conv2D)                 │ (None, 26, 26, 32)     │           320 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d (MaxPooling2D)    │ (None, 13, 13, 32)     │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_1 (Conv2D)               │ (None, 11, 11, 64)     │        18,496 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d_1 (MaxPooling2D)  │ (None, 5, 5, 64)       │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ flatten (Flatten)               │ (None, 1600)           │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense (Dense)                   │ (None, 128)            │       204,928 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout (Dropout)               │ (None, 128)            │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_1 (Dense)                 │ (None, 24)             │         3,096 │
└─────────────────────────────────┴────────────────────────┴───────────────┘
 Total params: 226,840 (886.09 KB)
 Trainable params: 226,840 (886.09 KB)
 Non-trainable params: 0 (0.00 B)
```

```
history = model.fit(X_train, y_train,
                    epochs=10,
                    batch_size=64,
                    validation_split=0.1)
```

```
Epoch 1/10
387/387 ━━━━━━━━━━━━━━━━━━━━ 14s 33ms/step - accuracy: 0.3106 - loss: 2.3081 - val_accuracy: 0.9020 - val_loss: 0.4114
Epoch 2/10
387/387 ━━━━━━━━━━━━━━━━━━━━ 20s 32ms/step - accuracy: 0.8326 - loss: 0.5194 - val_accuracy: 0.9763 - val_loss: 0.1176
Epoch 3/10
387/387 ━━━━━━━━━━━━━━━━━━━━ 13s 32ms/step - accuracy: 0.9234 - loss: 0.2352 - val_accuracy: 0.9869 - val_loss: 0.0493
Epoch 4/10
387/387 ━━━━━━━━━━━━━━━━━━━━ 20s 32ms/step - accuracy: 0.9550 - loss: 0.1395 - val_accuracy: 1.0000 - val_loss: 0.0177
Epoch 5/10
387/387 ━━━━━━━━━━━━━━━━━━━━ 12s 32ms/step - accuracy: 0.9739 - loss: 0.0861 - val_accuracy: 1.0000 - val_loss: 0.0077
Epoch 6/10
387/387 ━━━━━━━━━━━━━━━━━━━━ 20s 32ms/step - accuracy: 0.9825 - loss: 0.0594 - val_accuracy: 1.0000 - val_loss: 0.0050
```

```
Epoch 7/10
387/387 ━━━━━━━━━━━━━━━━━━━━ 12s 32ms/step - accuracy: 0.9789 - loss: 0.0682 - val_accuracy: 1.0000 - val_loss: 0.0018
Epoch 8/10
387/387 ━━━━━━━━━━━━━━━━━━━━ 21s 32ms/step - accuracy: 0.9892 - loss: 0.0379 - val_accuracy: 1.0000 - val_loss: 0.0025
Epoch 9/10
387/387 ━━━━━━━━━━━━━━━━━━━━ 20s 32ms/step - accuracy: 0.9880 - loss: 0.0357 - val_accuracy: 1.0000 - val_loss: 0.0024
Epoch 10/10
387/387 ━━━━━━━━━━━━━━━━━━━━ 21s 34ms/step - accuracy: 0.9917 - loss: 0.0268 - val_accuracy: 1.0000 - val_loss: 0.0013
```

```
loss, accuracy = model.evaluate(X_test, y_test)
print(f"Test Accuracy: {accuracy * 100:.2f}%")
```

```
225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9226 - loss: 0.2733
Test Accuracy: 92.53%
```