```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import zipfile
import os

# Path in your Drive
zip_path = '/content/drive/MyDrive/SignLanguageDataset/train.zip'
csv_path = '/content/drive/MyDrive/SignLanguageDataset/Training_set.csv'

# Unzip only if not already extracted
extract_path = '/content/train_images'
if not os.path.exists(extract_path):
    with zipfile.ZipFile(zip_path, 'r') as zip_ref:
        zip_ref.extractall(extract_path)

# Load CSV from Drive
import pandas as pd
train_df = pd.read_csv(csv_path)
```

```
# STEP 1: Imports
import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Dropout, Flatten, GlobalAveragePooling2D, BatchNormalization
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import EfficientNetB0
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
import pandas as pd
import numpy as np
import os
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelBinarizer
import cv2
from tqdm import tqdm
```

```
import os
train_df['filepath'] = train_df['filename'].apply(lambda x: os.path.join('/content/train_images/train', x))
```

```
train_df['label_str'] = train_df['label'].astype(str)
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.1,
    rotation_range=10,
    zoom_range=0.1,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True
)

train_gen = train_datagen.flow_from_dataframe(
    train_df,
    x_col='filepath',
    y_col='label_str',
    target_size=(200, 200),
    class_mode='categorical',
    subset='training',
    batch_size=32,
    shuffle=True
)

val_gen = train_datagen.flow_from_dataframe(
    train_df,
    x_col='filepath',
    y_col='label_str',
    target_size=(200, 200),
    class_mode='categorical',
    subset='validation',
    batch_size=32,
    shuffle=False
)
```

```
Found 54810 validated image filenames belonging to 29 classes.
Found 6090 validated image filenames belonging to 29 classes.
```

```
# Save class indices for future use
import json

class_indices_path = 'class_indices.json'
with open(class_indices_path, 'w') as f:
    json.dump(train_gen.class_indices, f)

# Download to your system
from google.colab import files
files.download(class_indices_path)
```

```
import tensorflow as tf
print("GPU Available:", tf.config.list_physical_devices('GPU'))
```

```
GPU Available: [PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
```

```
import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Dropout, GlobalAveragePooling2D, BatchNormalization
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications import MobileNetV2

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, BatchNormalization

# Get number of classes from generator
num_classes = len(train_gen.class_indices)

model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 3)),
    MaxPooling2D(2, 2),
    BatchNormalization(),

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

    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),
    BatchNormalization(),

    Flatten(),
    Dense(256, activation='relu'),
    Dropout(0.5),
    Dense(num_classes, activation='softmax')
])

# Compile the model
model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=1e-4),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

# Show model summary
model.summary()
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 126, 126, 32)	896
max_pooling2d (MaxPooling2D)	(None, 63, 63, 32)	0
batch_normalization (BatchNormalization)	(None, 63, 63, 32)	128
conv2d_1 (Conv2D)	(None, 61, 61, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 30, 30, 64)	0
batch_normalization_1 (BatchNormalization)	(None, 30, 30, 64)	256
conv2d_2 (Conv2D)	(None, 28, 28, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 14, 14, 128)	0
batch_normalization_2 (BatchNormalization)	(None, 14, 14, 128)	512
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 256)	6,422,784
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 29)	7,453

Total params: 6,524,381 (24.89 MB)
Trainable params: 6,523,933 (24.89 MB)

```
# Optional: callbacks to improve training
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint

checkpoint = ModelCheckpoint('best_model.keras', monitor='val_accuracy', save_best_only=True, mode='max', verbose=1)
earlystop = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True, verbose=1)

# ✅ Train the model
history = model.fit(
    train_gen,
    validation_data=val_gen,
    epochs=10,
    callbacks=[checkpoint, earlystop]
)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__init__(**
  self._warn_if_super_not_called()
Epoch 1/10
1713/1713 ━━━━━━━━━━━━━━━━━━━━ 0s 147ms/step - accuracy: 0.1707 - loss: 3.0419
Epoch 1: val_accuracy improved from -inf to 0.45977, saving model to best_model.keras
1713/1713 ━━━━━━━━━━━━━━━━━━━━ 287s 163ms/step - accuracy: 0.1707 - loss: 3.0416 - val_accuracy: 0.4598 - val_loss: 1.7302
Epoch 2/10
1713/1713 ━━━━━━━━━━━━━━━━━━━━ 0s 144ms/step - accuracy: 0.4328 - loss: 1.8424
Epoch 2: val_accuracy improved from 0.45977 to 0.67044, saving model to best_model.keras
1713/1713 ━━━━━━━━━━━━━━━━━━━━ 272s 159ms/step - accuracy: 0.4328 - loss: 1.8423 - val_accuracy: 0.6704 - val_loss: 1.0434
Epoch 3/10
1713/1713 ━━━━━━━━━━━━━━━━━━━━ 0s 146ms/step - accuracy: 0.5712 - loss: 1.3459
Epoch 3: val_accuracy improved from 0.67044 to 0.67603, saving model to best_model.keras
1713/1713 ━━━━━━━━━━━━━━━━━━━━ 276s 161ms/step - accuracy: 0.5712 - loss: 1.3459 - val_accuracy: 0.6760 - val_loss: 1.0308
Epoch 4/10
1713/1713 ━━━━━━━━━━━━━━━━━━━━ 0s 145ms/step - accuracy: 0.6469 - loss: 1.0745
Epoch 4: val_accuracy improved from 0.67603 to 0.75567, saving model to best_model.keras
1713/1713 ━━━━━━━━━━━━━━━━━━━━ 321s 160ms/step - accuracy: 0.6469 - loss: 1.0745 - val_accuracy: 0.7557 - val_loss: 0.7356
Epoch 5/10
1713/1713 ━━━━━━━━━━━━━━━━━━━━ 0s 145ms/step - accuracy: 0.7061 - loss: 0.8812
Epoch 5: val_accuracy improved from 0.75567 to 0.85304, saving model to best_model.keras
1713/1713 ━━━━━━━━━━━━━━━━━━━━ 274s 160ms/step - accuracy: 0.7061 - loss: 0.8812 - val_accuracy: 0.8530 - val_loss: 0.4794
Epoch 6/10
1713/1713 ━━━━━━━━━━━━━━━━━━━━ 0s 144ms/step - accuracy: 0.7490 - loss: 0.7524
Epoch 6: val_accuracy improved from 0.85304 to 0.89080, saving model to best_model.keras
1713/1713 ━━━━━━━━━━━━━━━━━━━━ 273s 159ms/step - accuracy: 0.7490 - loss: 0.7523 - val_accuracy: 0.8908 - val_loss: 0.3535
Epoch 7/10
1713/1713 ━━━━━━━━━━━━━━━━━━━━ 0s 146ms/step - accuracy: 0.7875 - loss: 0.6291
Epoch 7: val_accuracy improved from 0.89080 to 0.89606, saving model to best_model.keras
1713/1713 ━━━━━━━━━━━━━━━━━━━━ 325s 161ms/step - accuracy: 0.7875 - loss: 0.6291 - val_accuracy: 0.8961 - val_loss: 0.3212
Epoch 8/10
1713/1713 ━━━━━━━━━━━━━━━━━━━━ 0s 146ms/step - accuracy: 0.8163 - loss: 0.5413
Epoch 8: val_accuracy improved from 0.89606 to 0.91297, saving model to best_model.keras
1713/1713 ━━━━━━━━━━━━━━━━━━━━ 276s 161ms/step - accuracy: 0.8163 - loss: 0.5413 - val_accuracy: 0.9130 - val_loss: 0.2599
Epoch 9/10
1713/1713 ━━━━━━━━━━━━━━━━━━━━ 0s 145ms/step - accuracy: 0.8374 - loss: 0.4691
Epoch 9: val_accuracy did not improve from 0.91297
1713/1713 ━━━━━━━━━━━━━━━━━━━━ 276s 161ms/step - accuracy: 0.8374 - loss: 0.4691 - val_accuracy: 0.9062 - val_loss: 0.2786
Epoch 10/10
1713/1713 ━━━━━━━━━━━━━━━━━━━━ 0s 143ms/step - accuracy: 0.8584 - loss: 0.4199
Epoch 10: val_accuracy improved from 0.91297 to 0.92742, saving model to best_model.keras
1713/1713 ━━━━━━━━━━━━━━━━━━━━ 272s 158ms/step - accuracy: 0.8584 - loss: 0.4199 - val_accuracy: 0.9274 - val_loss: 0.2180
```

```
Restoring model weights from the end of the best epoch: 10.
```

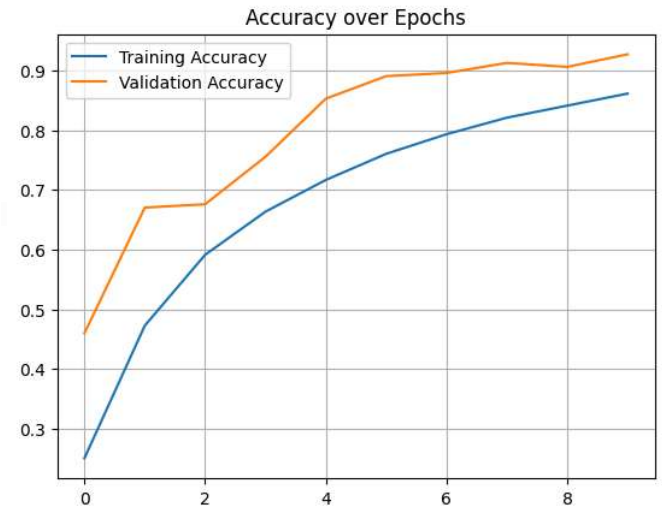
```
val_loss, val_acc = model.evaluate(val_gen)
print(f"Validation Accuracy: {val_acc*100:.2f}%")
print(f"Validation Loss: {val_loss:.4f}")
```

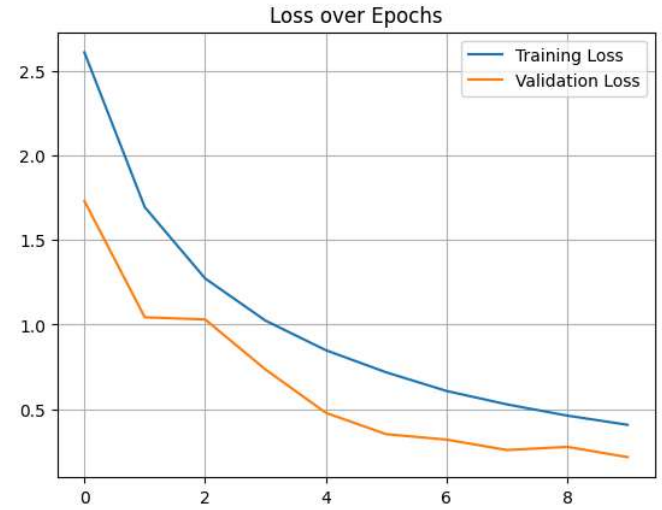
```
191/191 ━━━━━━━━━━━━━━━━━━━━ 26s 138ms/step - accuracy: 0.9257 - loss: 0.2205
Validation Accuracy: 92.78%
Validation Loss: 0.2165
```

```
import matplotlib.pyplot as plt

# Accuracy
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.legend()
plt.title("Accuracy over Epochs")
plt.grid(True)
plt.show()

# Loss
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend()
plt.title("Loss over Epochs")
plt.grid(True)
plt.show()
```






```
import zipfile, os, pandas as pd
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Paths
test_zip_path = '/content/drive/MyDrive/SignLanguageDataset/test.zip'
test_csv_path = '/content/drive/MyDrive/SignLanguageDataset/Testing_set.csv'
test_extract_path = '/content/test_images'

# Unzip if not already
```

```
if not os.path.exists(test_extract_path):
    with zipfile.ZipFile(test_zip_path, 'r') as zip_ref:
        zip_ref.extractall(test_extract_path)

# Load CSV
test_df = pd.read_csv(test_csv_path)

# File paths
test_df['filepath'] = test_df['filename'].apply(lambda x: os.path.join('/content/test_images/test', x))

# Create test generator (no labels!)
test_datagen = ImageDataGenerator(rescale=1./255)
test_gen = test_datagen.flow_from_dataframe(
    test_df,
    x_col='filepath',
    y_col=None,                # ⛔ No label column
    target_size=(200, 200),
    class_mode=None,           # ⛔ No class_mode
    batch_size=32,
    shuffle=False
)
```

```
Found 26100 validated image filenames.
```

```
# Predict
predictions = model.predict(test_gen)

# Get class indices from the generator used in training
class_indices = train_gen.class_indices
inv_class_indices = {v: k for k, v in class_indices.items()}

# Get top prediction for each sample
predicted_classes = [inv_class_indices[i] for i in predictions.argmax(axis=1)]

# Add to dataframe
test_df['predicted_label'] = predicted_classes

# Preview
test_df[['filename', 'predicted_label']].head()
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__init__(**
  self._warn_if_super_not_called()
816/816 ━━━━━━━━━━━━━━━━━━━━ 25s 30ms/step
```

	filename	predicted_label
0	Image_1.jpg	M
1	Image_2.jpg	G
2	Image_3.jpg	I
3	Image_4.jpg	E
4	Image_5.jpg	D

```
test_df[['filename', 'predicted_label']].to_csv('test_predictions.csv', index=False)
from google.colab import files
files.download('test_predictions.csv')
```

```
import matplotlib.pyplot as plt
import cv2

# Show first 5 images and predictions
for i in range(5):
    img_path = test_df.iloc[i]['filepath']
    label = test_df.iloc[i]['predicted_label']

    img = cv2.imread(img_path)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    plt.imshow(img)
    plt.title(f'Predicted: {label}')
    plt.axis('off')
    plt.show()
```

Predicted: M

Predicted: G

Predicted: I

Predicted: E

Predicted: D

```
model.save('sign_language_model.keras')


from google.colab import files
```