Hand Gesture Recognition Model

```
import os

import zipfile

import numpy as np

import tensorflow as tf

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.models import Sequential

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

from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint

import matplotlib.pyplot as plt

from PIL import Image


# Step 1: Unzip the datasets

def unzip_data(zip_path, extract_to):

    with zipfile.ZipFile(zip_path, 'r') as zip_ref:

        zip_ref.extractall(extract_to)


data_paths = [

    '/Users/ronitvyas/Downloads/leapGestRecog/F1.zip',

    '/Users/ronitvyas/Downloads/leapGestRecog/F2.zip',

    '/Users/ronitvyas/Downloads/leapGestRecog/F3.zip',

    '/Users/ronitvyas/Downloads/leapGestRecog/LG1.zip',

    '/Users/ronitvyas/Downloads/leapGestRecog/LG2.zip'

]
```

```
extract_paths = [
    '/Users/ronitvyas/Downloads/leapGestRecog/extract/F1',
    '/Users/ronitvyas/Downloads/leapGestRecog/extract/F2',
    '/Users/ronitvyas/Downloads/leapGestRecog/extract/F3',
    '/Users/ronitvyas/Downloads/leapGestRecog/extract/LG1',
    '/Users/ronitvyas/Downloads/leapGestRecog/extract/LG2'
]


for zip_path, extract_path in zip(data_paths, extract_paths):
    unzip_data(zip_path, extract_path)


# Custom function to filter valid image files
def is_valid_image(file_path):
    try:
        Image.open(file_path)
        return True
    except:
        return False


# Remove invalid files
for category in ['F1', 'F2', 'F3', 'LG1', 'LG2']:
    category_path = os.path.join('/Users/ronitvyas/Downloads/leapGestRecog/extract', category)
    for root, dirs, files in os.walk(category_path):
        for file in files:
            file_path = os.path.join(root, file)
            if not is_valid_image(file_path):
```

```
            os.remove(file_path)


# Step 2: Load and preprocess the data
data_dir = '/Users/ronitvyas/Downloads/leapGestRecog/extract'
image_size = (128, 128)
batch_size = 32


datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)


train_generator = datagen.flow_from_directory(
    data_dir,
    target_size=image_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='training'
)


validation_generator = datagen.flow_from_directory(
    data_dir,
    target_size=image_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation'
)


# Step 3: Build the CNN model
```

```python
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(5, activation='softmax')
])


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


# Step 4: Train the model
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
model_checkpoint = ModelCheckpoint('hand_gesture_model.h5', save_best_only=True)


history = model.fit(
    train_generator,
    epochs=50,
    validation_data=validation_generator,
    callbacks=[early_stopping, model_checkpoint]
)
```

```
# Step 5: Evaluate the model

plt.plot(history.history['accuracy'], label='accuracy')

plt.plot(history.history['val_accuracy'], label='val_accuracy')

plt.xlabel('Epoch')

plt.ylabel('Accuracy')

plt.legend(loc='lower right')

plt.title('Model Accuracy')

plt.savefig('accuracy_plot.png')

plt.show()


plt.plot(history.history['loss'], label='loss')

plt.plot(history.history['val_loss'], label='val_loss')

plt.xlabel('Epoch')

plt.ylabel('Loss')

plt.legend(loc='upper right')

plt.title('Model Loss')

plt.savefig('loss_plot.png')

plt.show()
```

Model Interpretation

Hand Gesture Recognition Model Evaluation:

1. Model Accuracy:

   - The training accuracy starts at around 55% and shows a slight improvement over epochs.

   - The validation accuracy starts at around 37% but does not improve significantly, indicating potential overfitting.

2. Model Loss:

   - The training loss decreases over epochs, indicating that the model is learning.

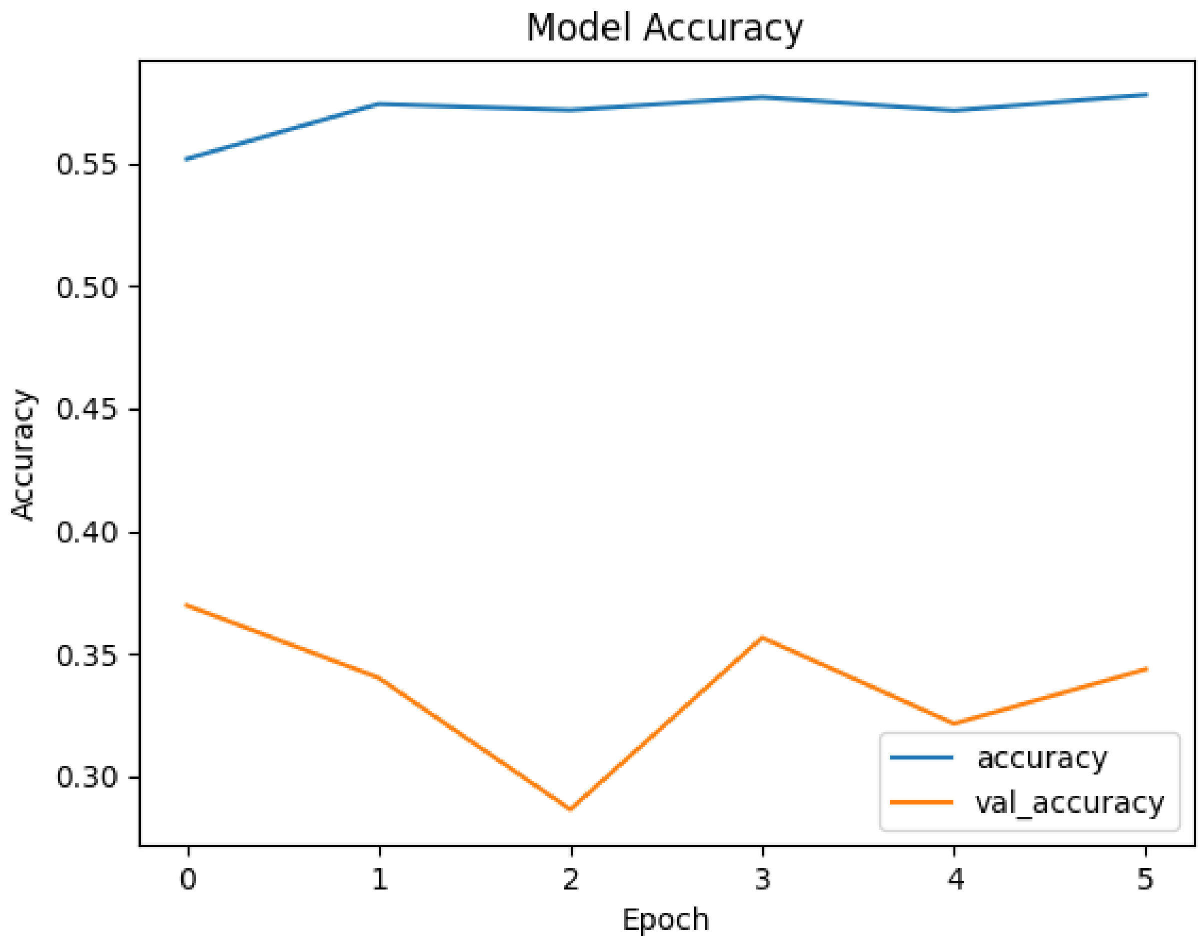
   - The validation loss increases significantly, suggesting overfitting.

3. Recommendations:

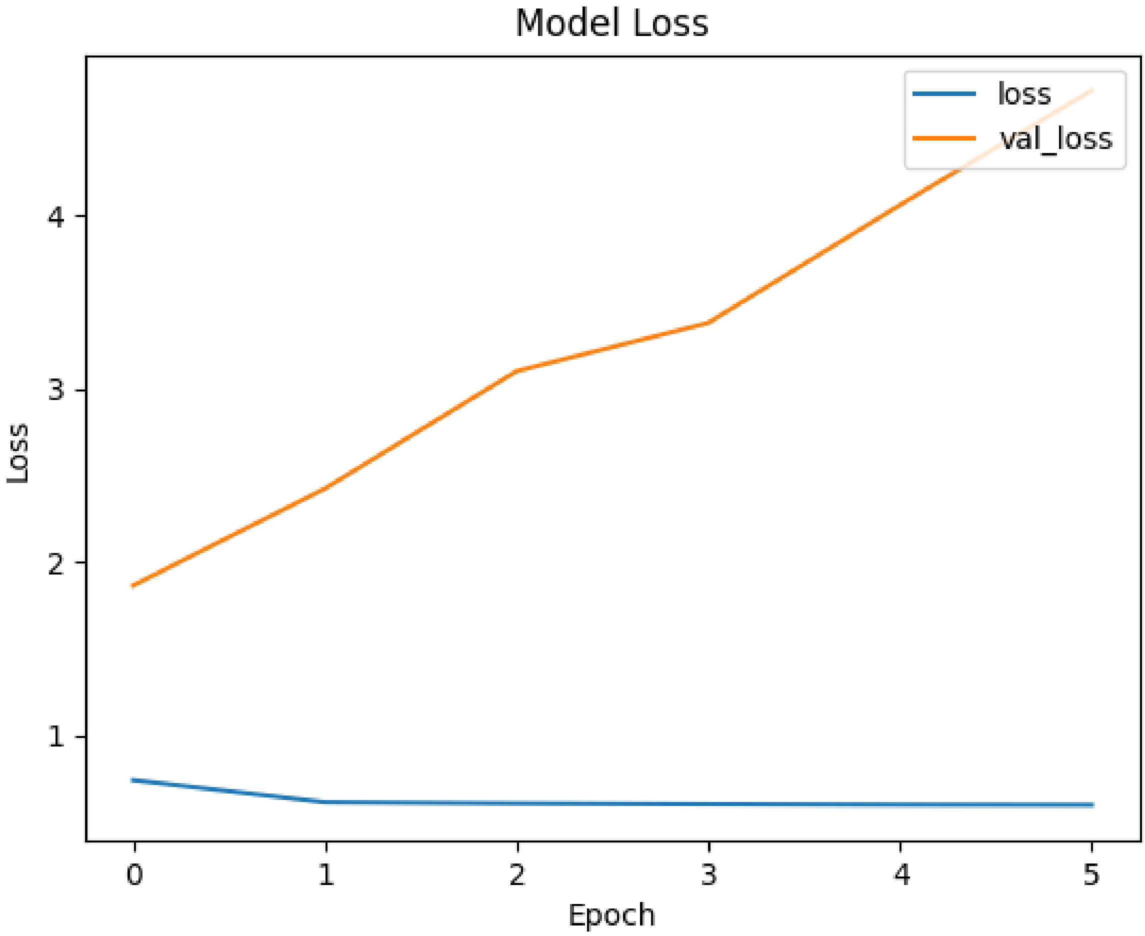
   - Improve data augmentation techniques to provide more varied training data.

   - Consider using a more complex model architecture or fine-tuning hyperparameters.

   - Collect more data or use transfer learning to improve model generalization.



Model Accuracy
0.55
0.50
0.45
0.40
0.35
0.30
Accuracy
0
1
2
3
4
5
Epoch
accuracy
val_accuracy


Model Loss
loss
val_loss
4
3
2
1
Loss
0
1
2
3
4
5
Epoch