In [ ]:
```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import VGG16
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
```

In [ ]:
```
# Define the paths to your train and test data folders
train_data_dir = 'train_data'
test_data_dir = 'test_data'

# Define the parameters for data augmentation for training data
train_datagen = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True
)

# Define the parameters for rescaling the testing data
test_datagen = ImageDataGenerator(rescale=1./255)

# Load the VGG16 pre-trained model without the top layer
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

# Generate batches of augmented data for training and validation
train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical'
)

test_generator = test_datagen.flow_from_directory(
    test_data_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical'
)

# num_classes is the number of bird species
num_classes = train_generator.num_classes
```

```
Found 150 images belonging to 16 classes.
Found 157 images belonging to 16 classes.
```

In [ ]:
```
# Freeze the layers of the base model
for layer in base_model.layers:
    layer.trainable = False

# Build your model by adding the base model and additional layers
model = Sequential()
# model.add(base_model)
model.add(Conv2D(64, (3, 3), activation='relu', input_shape = (224, 224, 3)))
model.add(MaxPooling2D((2, 2)))
# model.add(Conv2D(128, (3, 3), activation='relu'))
# model.add(MaxPooling2D((2, 2)))
# model.add(Conv2D(256, (3, 3), activation='relu'))
# model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
# model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))


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

# Train the model
model.fit(
    train_generator,
    batch_size=8,
    steps_per_epoch=len(train_generator),
    epochs=10,
    validation_data=test_generator,
    validation_steps=len(test_generator)
)
```

```
Epoch 1/10
5/5 [==============================] - 69s 15s/step - loss: 70.0222 - accuracy: 0.0733 - val_loss: 55.6618 - val_accuracy: 0.0637
Epoch 2/10
5/5 [==============================] - 58s 13s/step - loss: 29.3127 - accuracy: 0.1133 - val_loss: 14.2972 - val_accuracy: 0.1146
Epoch 3/10
5/5 [==============================] - 60s 13s/step - loss: 9.8447 - accuracy: 0.1133 - val_loss: 6.8923 - val_accuracy: 0.0637
Epoch 4/10
5/5 [==============================] - 56s 13s/step - loss: 4.4202 - accuracy: 0.2333 - val_loss: 4.5484 - val_accuracy: 0.1529
Epoch 5/10
5/5 [==============================] - 57s 12s/step - loss: 3.0465 - accuracy: 0.2533 - val_loss: 3.6620 - val_accuracy: 0.1465
Epoch 6/10
5/5 [==============================] - 57s 13s/step - loss: 2.4322 - accuracy: 0.3400 - val_loss: 3.4058 - val_accuracy: 0.1656
Epoch 7/10
5/5 [==============================] - 58s 13s/step - loss: 2.1477 - accuracy: 0.3133 - val_loss: 3.2219 - val_accuracy: 0.1911
Epoch 8/10
5/5 [==============================] - 58s 12s/step - loss: 2.0053 - accuracy: 0.3933 - val_loss: 3.2166 - val_accuracy: 0.1656
Epoch 9/10
5/5 [==============================] - 57s 13s/step - loss: 1.9197 - accuracy: 0.4267 - val_loss: 3.6637 - val_accuracy: 0.2293
Epoch 10/10
5/5 [==============================] - 58s 13s/step - loss: 1.7817 - accuracy: 0.4667 - val_loss: 3.5489 - val_accuracy: 0.2038
```

Out[ ]:
```
<keras.callbacks.History at 0x25c573b0f10>
```

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

In [ ]:
```
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.callbacks import EarlyStopping

some_tuned_model = Sequential()
some_tuned_model.add(base_model)
some_tuned_model.add(Flatten())
some_tuned_model.add(Dense(num_classes*4, activation='relu'))
some_tuned_model.add(BatchNormalization())
some_tuned_model.add(Dropout(0.2))
some_tuned_model.add(Dense(num_classes*2, activation='relu'))
some_tuned_model.add(Dense(num_classes, activation='softmax'))
early_stop = EarlyStopping(monitor='accuracy',patience=8)

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

# Train the model
some_tuned_model.fit(
    train_generator,
    batch_size=8,
    steps_per_epoch=len(train_generator),
    epochs=50,
    validation_data=test_generator,
    validation_steps=len(test_generator),
    callbacks = early_stop)
```

```
Epoch 1/50
5/5 [==============================] - 69s 15s/step - loss: 2.7354 - accuracy: 0.1533 - val_loss: 3.0531 - val_accuracy: 0.0892
Epoch 2/50
5/5 [==============================] - 59s 13s/step - loss: 2.0055 - accuracy: 0.4533 - val_loss: 3.0831 - val_accuracy: 0.1847
Epoch 3/50
5/5 [==============================] - 60s 14s/step - loss: 1.8466 - accuracy: 0.5333 - val_loss: 3.2313 - val_accuracy: 0.2166
Epoch 4/50
5/5 [==============================] - 68s 15s/step - loss: 1.5723 - accuracy: 0.6200 - val_loss: 3.1958 - val_accuracy: 0.2293
Epoch 5/50
5/5 [==============================] - 59s 13s/step - loss: 1.3324 - accuracy: 0.7733 - val_loss: 3.1559 - val_accuracy: 0.2293
Epoch 6/50
5/5 [==============================] - 63s 14s/step - loss: 1.2171 - accuracy: 0.7667 - val_loss: 3.1164 - val_accuracy: 0.2166
Epoch 7/50
5/5 [==============================] - 62s 14s/step - loss: 1.0213 - accuracy: 0.8200 - val_loss: 3.0625 - val_accuracy: 0.2229
Epoch 8/50
5/5 [==============================] - 61s 14s/step - loss: 0.9648 - accuracy: 0.8600 - val_loss: 3.0219 - val_accuracy: 0.2166
Epoch 9/50
5/5 [==============================] - 63s 15s/step - loss: 0.7899 - accuracy: 0.8933 - val_loss: 2.9374 - val_accuracy: 0.2357
Epoch 10/50
5/5 [==============================] - 62s 14s/step - loss: 0.7291 - accuracy: 0.9133 - val_loss: 2.8211 - val_accuracy: 0.2357
Epoch 11/50
5/5 [==============================] - 63s 14s/step - loss: 0.6398 - accuracy: 0.9400 - val_loss: 2.7622 - val_accuracy: 0.2548
Epoch 12/50
5/5 [==============================] - 61s 13s/step - loss: 0.5534 - accuracy: 0.9533 - val_loss: 2.6999 - val_accuracy: 0.2866
Epoch 13/50
5/5 [==============================] - 61s 14s/step - loss: 0.5063 - accuracy: 0.9600 - val_loss: 2.6656 - val_accuracy: 0.2611
Epoch 14/50
5/5 [==============================] - 61s 14s/step - loss: 0.4051 - accuracy: 0.9733 - val_loss: 2.6363 - val_accuracy: 0.2930
Epoch 15/50
5/5 [==============================] - 60s 13s/step - loss: 0.3861 - accuracy: 0.9800 - val_loss: 2.5871 - val_accuracy: 0.3439
Epoch 16/50
5/5 [==============================] - 58s 13s/step - loss: 0.3166 - accuracy: 0.9867 - val_loss: 2.5982 - val_accuracy: 0.3503
Epoch 17/50
5/5 [==============================] - 59s 13s/step - loss: 0.2757 - accuracy: 0.9933 - val_loss: 2.6313 - val_accuracy: 0.3121
Epoch 18/50
5/5 [==============================] - 58s 13s/step - loss: 0.2733 - accuracy: 0.9733 - val_loss: 2.5240 - val_accuracy: 0.3312
Epoch 19/50
5/5 [==============================] - 60s 13s/step - loss: 0.2301 - accuracy: 0.9933 - val_loss: 2.4737 - val_accuracy: 0.3312
Epoch 20/50
5/5 [==============================] - 61s 14s/step - loss: 0.1974 - accuracy: 1.0000 - val_loss: 2.4510 - val_accuracy: 0.3631
Epoch 21/50
5/5 [==============================] - 61s 14s/step - loss: 0.1919 - accuracy: 1.0000 - val_loss: 2.4458 - val_accuracy: 0.3758
Epoch 22/50
5/5 [==============================] - 62s 14s/step - loss: 0.1554 - accuracy: 1.0000 - val_loss: 2.4413 - val_accuracy: 0.3822
Epoch 23/50
5/5 [==============================] - 59s 13s/step - loss: 0.1433 - accuracy: 1.0000 - val_loss: 2.4132 - val_accuracy: 0.3885
Epoch 24/50
5/5 [==============================] - 59s 13s/step - loss: 0.1250 - accuracy: 1.0000 - val_loss: 2.4033 - val_accuracy: 0.3949
Epoch 25/50
5/5 [==============================] - 59s 14s/step - loss: 0.1141 - accuracy: 1.0000 - val_loss: 2.3942 - val_accuracy: 0.3885
Epoch 26/50
5/5 [==============================] - 61s 14s/step - loss: 0.1046 - accuracy: 1.0000 - val_loss: 2.3703 - val_accuracy: 0.3694
Epoch 27/50
5/5 [==============================] - 61s 14s/step - loss: 0.0838 - accuracy: 1.0000 - val_loss: 2.3659 - val_accuracy: 0.3503
Epoch 28/50
5/5 [==============================] - 61s 14s/step - loss: 0.0922 - accuracy: 1.0000 - val_loss: 2.3639 - val_accuracy: 0.3439
```

Out[ ]:
```
<keras.callbacks.History at 0x25c5b626310>
```

In [ ]:
```
some_tuned_model.save('vgg16_version.h5')
```

In [ ]:
```
print(train_generator.class_indices)
inverse_dict = {v: k for k, v in train_generator.class_indices.items()}
```

```
{'blasti': 0, 'bonegl': 1, 'brnkyt': 2, 'cbrtsh': 3, 'cmnmyn': 4, 'gretit': 5, 'hilpig': 6, 'himbul': 7, 'himgri': 8, 'hsparo': 9, 'indvul': 10, 'jglowl': 11, 'lbicrw': 12, 'mgprob': 13, 'rebimg': 14, 'wcrsrt': 15}
```

In [ ]:
```
print(inverse_dict)
```

```
{0: 'blasti', 1: 'bonegl', 2: 'brnkyt', 3: 'cbrtsh', 4: 'cmnmyn', 5: 'gretit', 6: 'hilpig', 7: 'himbul', 8: 'himgri', 9: 'hsparo', 10: 'indvul', 11: 'jglowl', 12: 'lbicrw', 13: 'mgprob', 14: 'rebimg', 15: 'wcrsrt'}
```

In [ ]:
```
import numpy as np
from tensorflow.keras.preprocessing import image

img1 = image.load_img('new_valid/great_tit.jpg', target_size=(224, 224))
img2 = image.load_img('new_valid/chestnut_bellied_rock_thrush.jpg', target_size=(224, 224))
img3 = image.load_img('new_valid/large_billed_crow.jpg', target_size=(224, 224))
img4 = image.load_img('new_valid/himalyan_bulbul.jpg', target_size=(224, 224))

predicted_indices = []
predicted_indices_vgg = []

for img in [img1, img2, img3, img4]:
    img = image.img_to_array(img)
    img = np.expand_dims(img, axis=0)
    pred = np.argmax(model.predict(img))
    pred2 = np.argmax(some_tuned_model.predict(img))

    predicted_indices.append(pred)
    predicted_indices_vgg.append(pred2)

predicted_labels = [inverse_dict[index] for index in predicted_indices]
predicted_labels_vgg = [inverse_dict[index] for index in predicted_indices_vgg]

print("for non-vgg model :\n", predicted_labels)
print("for vgg model :\n", predicted_labels_vgg)
```

```
1/1 [==============================] - 1s 534ms/step
1/1 [==============================] - 1s 534ms/step
1/1 [==============================] - 0s 248ms/step
1/1 [==============================] - 0s 120ms/step
1/1 [==============================] - 0s 96ms/step
1/1 [==============================] - 0s 103ms/step
1/1 [==============================] - 0s 94ms/step
1/1 [==============================] - 0s 99ms/step
1/1 [==============================] - 0s 92ms/step
for non-vgg model :
 ['himbul', 'rebimg', 'himbul', 'rebimg']
for vgg model :
 ['cmnmyn', 'cmnmyn', 'cmnmyn', 'cmnmyn']
```