In [1]:

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.models import Sequential
```

In [2]:

```
# Data Augmentation
train_gen = ImageDataGenerator(
    rescale=1./255,
    horizontal_flip=True,
    shear_range=0.2
)

test_gen = ImageDataGenerator(rescale=1./255)
```

In [3]:

```
train = train_gen.flow_from_directory(
    'C:\\Users\Downloads\\train_data'        ,
    target_size=(224, 224),
    class_mode='categorical',
    batch_size=8
)

test = test_gen.flow_from_directory(
    'C:\\Users\\Downloads\\test_data'       ,
    target_size=(224, 224),
    class_mode='categorical',
    batch_size=8
)
```

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

In

```
train.class_indices
```

[4]:

```
Out[4]: {'blasti': 0, 'bonegl':
          1,
          'brhkyt': 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,                    In                    [6]:
'wcrsrt': 15}
```

```
#CNN MODEL
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3)))
model.add(MaxPooling2D((2, 2)))

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

model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(train.num_classes, activation='softmax'))
```

In

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

In [7]:

```
model.summary()
```

[8]:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
        conv2d (Conv2D)              (None, 222, 222, 32)      896
max_pooling2d (MaxPooling2D  (None, 111, 111, 32)     0          )

      conv2d_1 (Conv2D)            (None, 109, 109, 64)      18496
          max_pooling2d_1 (MaxPooling  (None, 54, 54, 64)        0
                                                                2D)

      conv2d_2 (Conv2D)            (None, 52, 52, 128)       73856
          max_pooling2d_2 (MaxPooling  (None, 26, 26, 128)       0
                                                                2D)

flatten  (Flatten)               (None,  86528)                  0
dense (Dense)                 (None, 128)                 11075712
dense_1 (Dense)                (None,  16)                    2064
=================================================================
Total params: 11,171,024
Trainable params: 11,171,024
Non-trainable params: 0
_________________________________________________________________
```

```
# Training the model
model.fit(train, batch_size=8, validation_data=test, epochs=10)
```

[9]:

```
Epoch 1/10
19/19 [==============================] - 67s 4s/step - loss: 3.0439 - accuracy: 0.1267 - val_loss: 2.7268 - val_a
ccuracy: 0.1274
Epoch 2/10
```

```
In          ⍰
            19/19 [==============================] - 64s 3s/step - loss: 2.5666 - accuracy: 0.2000 - val_loss: 2.7404 - val_a
            ccuracy: 0.1274
            Epoch 3/10
            19/19 [==============================] - 63s 3s/step - loss: 2.5092 - accuracy: 0.1867 - val_loss: 2.7207 - val_a
            ccuracy: 0.1338
            Epoch 4/10
            19/19 [==============================] - 64s 3s/step - loss: 2.4048 - accuracy: 0.2067 - val_loss: 2.6721 - val_a
            ccuracy: 0.1720
            Epoch 5/10
            19/19 [==============================] - 77s 4s/step - loss: 2.2932 - accuracy: 0.2067 - val_loss: 2.7198 - val_a
            ccuracy: 0.1083
            Epoch 6/10
            19/19 [==============================] - 72s 4s/step - loss: 2.1982 - accuracy: 0.2800 - val_loss: 2.7705 - val_a
            ccuracy: 0.1592
            Epoch 7/10
            19/19 [==============================] - 67s 4s/step - loss: 1.9299 - accuracy: 0.4133 - val_loss: 3.1716 - val_a
            ccuracy: 0.1783
            Epoch 8/10
            19/19 [==============================] - 70s 4s/step - loss: 1.8512 - accuracy: 0.4267 - val_loss: 3.2591 - val_a
            ccuracy: 0.2102
            Epoch 9/10
            19/19 [==============================] - 69s 4s/step - loss: 1.3655 - accuracy: 0.5533 - val_loss: 3.9734 - val_a
            ccuracy: 0.2357
            Epoch 10/10
            19/19 [==============================] - 71s 4s/step - loss: 1.2378 - accuracy: 0.6000 - val_loss: 3.3789 - val_a
            ccuracy: 0.2229

    Out[9]: <keras.callbacks.History at 0x1ddbb80c400>
```

In [10]:

```
model.save('birdSpeciesModel.h5')
```

In [11]:

```
# Testing

import numpy as np
from tensorflow.keras.preprocessing import image
```

In

In [17]:

```
#Testing-1
from tensorflow.keras.preprocessing import image
import numpy as np
from PIL import Image

# Load and resize the image
img = Image.open('C:\\Users\\mohan\\Downloads\\test_data\\jglowl\\_D32_13516.jpg')
img = img.resize((224, 224))  # Resize the image to match the input size expected by the model

# Convert the image to an array
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)

# Normalize the pixel values
img_array = img_array / 255.0

# Make predictions
pred = np.argmax(model.predict(img_array))

# Define class labels
output = ['blasti', 'bonegl', 'brhkyt', 'cbrtsh', 'cmnmyn', 'gretit', 'hilpig', 'himbul', 'himgri', 'hsparo',
          'indvul', 'jglowl', 'lbicrw', 'mgprob', 'rebimg', 'wcrst']

# Print the predicted class index and corresponding bird species
print(pred)
print(output[pred])
```

```
1/1 [==============================] - 0s 139ms/step
4
           cmnmyn
```

In [22]: ⍰

```
#Testing-2
from tensorflow.keras.preprocessing import image
import numpy as np
from PIL import Image

# Load and resize the image
img = Image.open('C:\\Users\\mohan\\Downloads\\test_data\\brhkyt\\D72_0475.jpg')
img = img.resize((224, 224))  # Resize the image to match the input size expected by the model

# Convert the image to an array
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)

# Normalize the pixel values
img_array = img_array / 255.0

# Make predictions
pred = np.argmax(model.predict(img_array))

# Define class labels
output = ['blasti', 'bonegl', 'brhkyt', 'cbrtsh', 'cmnmyn', 'gretit', 'hilpig', 'himbul', 'himgri', 'hsparo',
          'indvul', 'jglowl', 'lbicrw', 'mgprob', 'rebimg', 'wcrst']

# Print the predicted class index and corresponding bird species
print(pred)
print(output[pred])
```

```
1/1 [==============================] - 0s 36ms/step
12
lbicrw
```

In [21]:

```
#Testing-3
from tensorflow.keras.preprocessing import image
import numpy as np
from PIL import Image

# Load and resize the image
img = Image.open('C:\\Users\\mohan\\Downloads\\test_data\\wcrsrt\\100_4464.JPG')
img = img.resize((224, 224))  # Resize the image to match the input size expected by the model

# Convert the image to an array
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)

# Normalize the pixel values
img_array = img_array / 255.0

# Make predictions
pred = np.argmax(model.predict(img_array))

# Define class labels
output = ['blasti', 'bonegl', 'brhkyt', 'cbrtsh', 'cmnmyn', 'gretit', 'hilpig', 'himbul', 'himgri', 'hsparo',
          'indvul', 'jglowl', 'lbicrw', 'mgprob', 'rebimg', 'wcrst']

# Print the predicted class index and corresponding bird species
print(pred)
print(output[pred])
```

```
1/1 [==============================] - 0s 49ms/step
9
hsparo
```

In [23]:

```
#Testing-4
from tensorflow.keras.preprocessing import image
import numpy as np
from PIL import Image

# Load and resize the image
img = Image.open('C:\\Users\\mohan\\Downloads\\test_data\\indvul\\IMG_5489.JPG')
img = img.resize((224, 224))  # Resize the image to match the input size expected by the model

# Convert the image to an array
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)

# Normalize the pixel values
img_array = img_array / 255.0

# Make predictions
pred = np.argmax(model.predict(img_array))

# Define class labels
output = ['blasti', 'bonegl', 'brhkyt', 'cbrtsh', 'cmnmyn', 'gretit', 'hilpig', 'himbul', 'himgri', 'hsparo',
          'indvul', 'jglowl', 'lbicrw', 'mgprob', 'rebimg', 'wcrst']

# Print the predicted class index and corresponding bird species
print(pred)
print(output[pred])
```

```
1/1 [==============================] - 0s 58ms/step
10
indvul
```

[25]:

In

```
#Testing-5
from tensorflow.keras.preprocessing import image
import numpy as np
from PIL import Image

# Load and resize the image
img = Image.open('C:\\Users\\mohan\\Downloads\\test_data\\himgri\\_D32_10311.jpg')
img = img.resize((224, 224))  # Resize the image to match the input size expected by the model

# Convert the image to an array
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)

# Normalize the pixel values
img_array = img_array / 255.0

# Make predictions
pred = np.argmax(model.predict(img_array))

# Define class labels
output = ['blasti', 'bonegl', 'brhkyt', 'cbrtsh', 'cmnmyn', 'gretit', 'hilpig', 'himbul', 'himgri', 'hsparo',
          'indvul', 'jglowl', 'lbicrw', 'mgprob', 'rebimg', 'wcrst']

# Print the predicted class index and corresponding bird species
print(pred)
print(output[pred])
```

```
 1/1 [==============================] - 0s 50ms/step
 8
himgri
```

In [ ]:

```

```