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

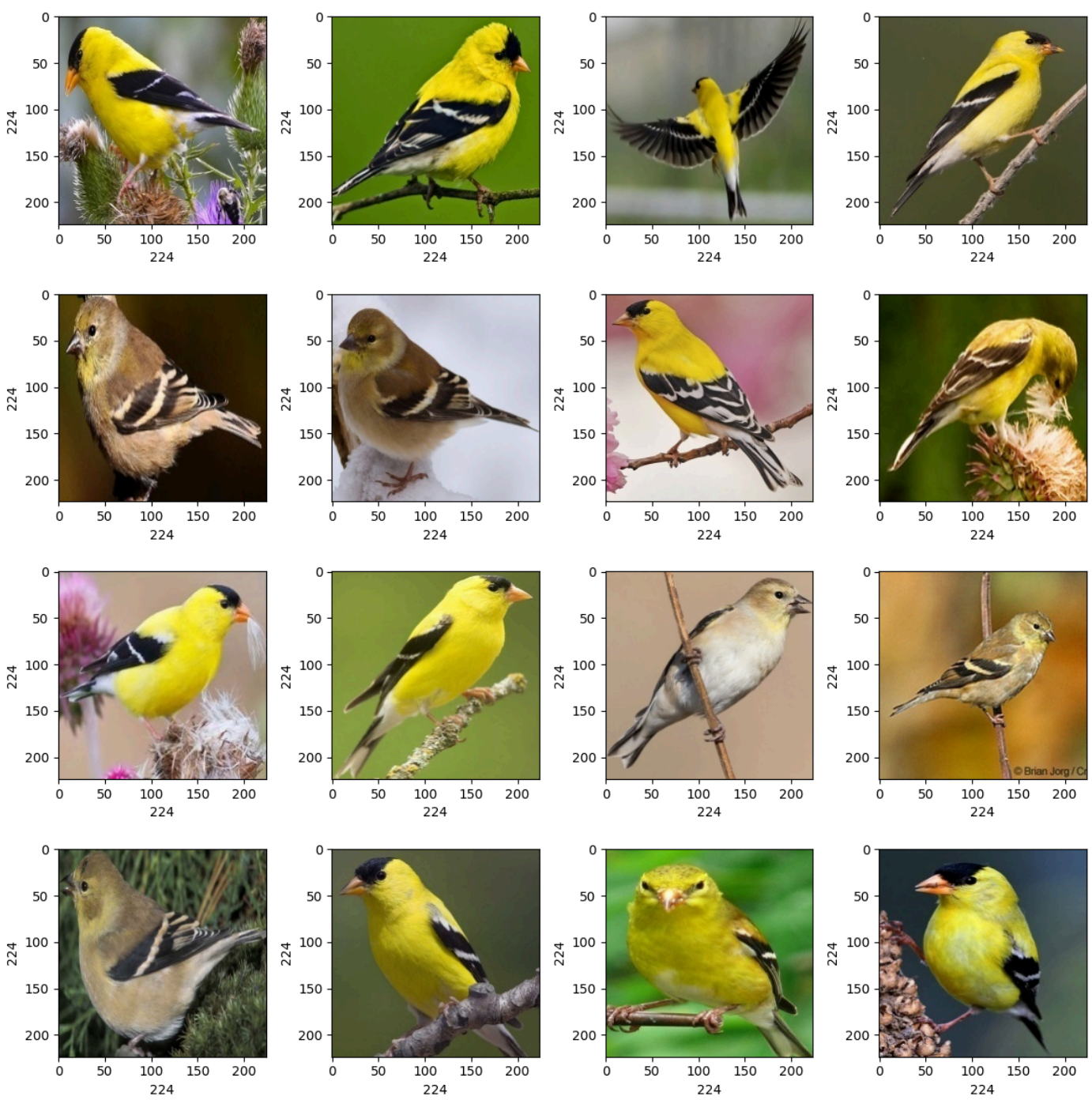
```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
!ls "/content/drive/My Drive"
```

```
'bandicam 2024-10-19 23-59-13-798.zip'  'mahir cs2001030.zip'
 'Colab Notebooks'                       monkey
 'Data science'                          monkey.ipynb
 'dataset sample pic.gdraw'              nlp_no
 'Getting started.pdf'                   projectXai.ipynb
  KDDTest-21.txt                        'Untitled drawing (1).gdraw'
  KDDTest+.txt                          'Untitled drawing (2).gdraw'
  KDDTrain+_20Percent.txt               'Untitled drawing.gdraw'
  KDDTrain+.txt
```

```
#Library imports
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.image import imread
import cv2
import random
from os import listdir
from sklearn.preprocessing import  LabelBinarizer
from keras.preprocessing import image
from keras.preprocessing.image import img_to_array, array_to_img
from keras.optimizers import Adam
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Activation, Flatten, Dropout, Dense
from sklearn.model_selection import train_test_split
```

```
#images
plt.figure(figsize=(12,12))
path = "/content/drive/MyDrive/Data science/project12/Data/Bird Speciees Dataset/AMERICAN GOLDFINCH"
for i in range(1,17):
    plt.subplot(4,4,i)
    plt.tight_layout()
    rand_img = imread(path +'/'+ random.choice(sorted(listdir(path))))
    plt.imshow(rand_img)
    plt.xlabel(rand_img.shape[1], fontsize = 10)
    plt.ylabel(rand_img.shape[0], fontsize = 10)
```



```
#Setting path
dir = "/content/drive/MyDrive/Data science/project12/Data/Bird Speciees Dataset"
root_dir = listdir(dir)
image_list, label_list = [], []
```

```
for directory in root_dir:
  for files in listdir(f"{dir}/{directory}"):
    image_path = f"{dir}/{directory}/{files}"
    image = cv2.imread(image_path)
    image = img_to_array(image)
    image_list.append(image)
    label_list.append(directory)


#Visualize
label_counts = pd.DataFrame(label_list).value_counts()
label_counts
```

0	count
AMERICAN GOLDFINCH	143
EMPEROR PENGUIN	139
DOWNY WOODPECKER	137
FLAMINGO	132
CARMINE BEE-EATER	131
BARN OWL	129

dtype: int64

```
#num classes
num_classes = len(label_counts)
num_classes
```

```
6
```

```
image_list[0].shape
```

```
(224, 224, 3)
```

```
label_list = np.array(label_list)
label_list.shape
```

```
(811,)
```

```
#Splitting dataset
x_train, x_test, y_train, y_test = train_test_split(image_list, label_list, test_size=0.2, random_state = 10)


#Normalize
x_train = np.array(x_train, dtype=np.float16) / 225.0
x_test = np.array(x_test, dtype=np.float16) / 225.0
x_train = x_train.reshape( -1, 224,224,3)
x_test = x_test.reshape( -1, 224,224,3)


#Labeling
lb = LabelBinarizer()
y_train = lb.fit_transform(y_train)
y_test = lb.fit_transform(y_test)
print(lb.classes_)
```

```
['AMERICAN GOLDFINCH' 'BARN OWL' 'CARMINE BEE-EATER' 'DOWNY WOODPECKER'
 'EMPEROR PENGUIN' 'FLAMINGO']
```

```
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size = 0.2)


#Model cnn
model = Sequential()
model.add(Conv2D(8, (3, 3), padding="same",input_shape=(224,224,3), activation="relu"))
model.add(MaxPooling2D(pool_size=(3, 3)))
model.add(Conv2D(16, (3, 3), padding="same", activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(32, (3, 3), padding="same", activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
model.add(Flatten())
model.add(Dense(32, activation="relu"))
model.add(Dense(num_classes, activation="softmax"))
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, 224, 224, 8)	224
max_pooling2d (MaxPooling2D)	(None, 74, 74, 8)	0
conv2d_1 (Conv2D)	(None, 74, 74, 16)	1,168
max_pooling2d_1 (MaxPooling2D)	(None, 37, 37, 16)	0
conv2d_2 (Conv2D)	(None, 37, 37, 32)	4,640
max_pooling2d_2 (MaxPooling2D)	(None, 18, 18, 32)	0
flatten (Flatten)	(None, 10368)	0
dense (Dense)	(None, 32)	331,808
dense_1 (Dense)	(None, 6)	198

Total params: 338,038 (1.29 MB)
Trainable params: 338,038 (1.29 MB)
Non-trainable params: 0 (0.00 B)

```
model.compile(loss = 'categorical_crossentropy', optimizer = Adam(0.0005),metrics=['accuracy'])


#Train
epochs = 50
batch_size = 128
history = model.fit(x_train, y_train, batch_size = batch_size, epochs = epochs, validation_data = (x_val, y_val))
```

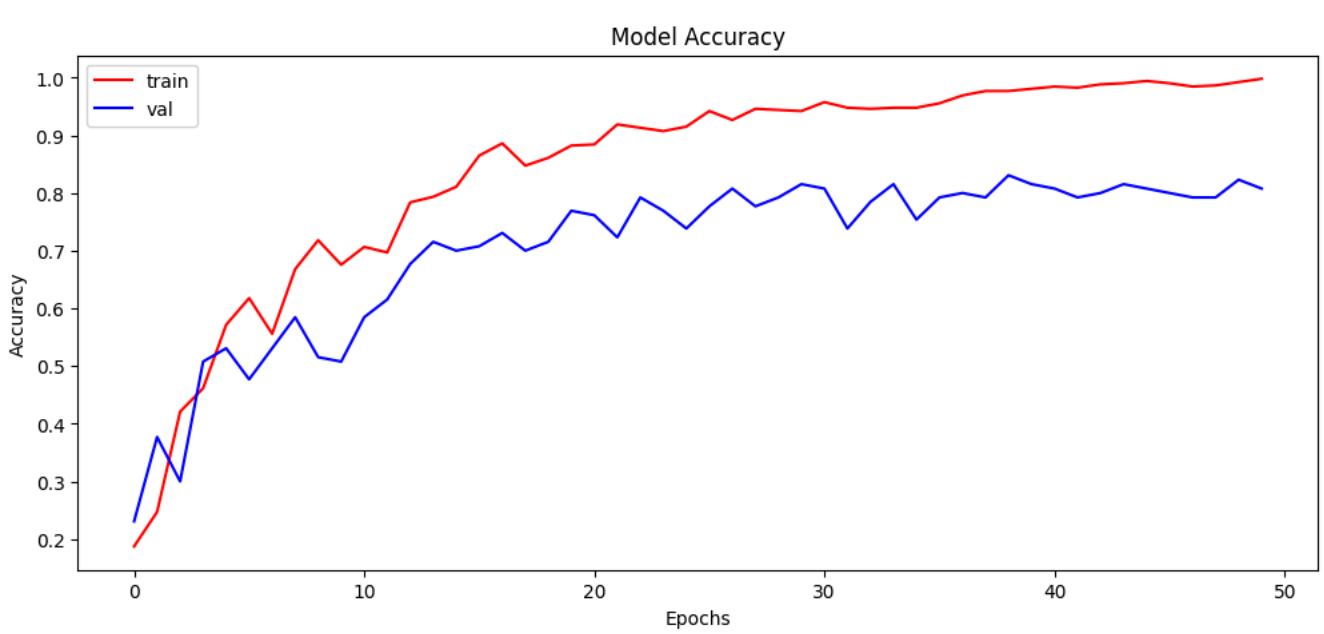
```
5/5 ━━━━━━━━━━━━━━━━━━━━ 1s 77ms/step - accuracy: 0.8379 - loss: 0.4780 - val_accuracy: 0.7000 - val_loss: 0.8015
Epoch 19/50
5/5 ━━━━━━━━━━━━━━━━━━━━ 1s 92ms/step - accuracy: 0.8608 - loss: 0.4508 - val_accuracy: 0.7154 - val_loss: 0.7137
Epoch 20/50
5/5 ━━━━━━━━━━━━━━━━━━━━ 1s 138ms/step - accuracy: 0.8719 - loss: 0.3957 - val_accuracy: 0.7692 - val_loss: 0.6865
Epoch 21/50
```

```
Epoch 39/50
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 91ms/step - accuracy: 0.9771 - loss: 0.1095 - val_accuracy: 0.8308 - val_loss: 0.5385
Epoch 40/50
5/5 ━━━━━━━━━━━━━━━━━━━━ 1s 93ms/step - accuracy: 0.9773 - loss: 0.0973 - val_accuracy: 0.8154 - val_loss: 0.5457
Epoch 41/50
5/5 ━━━━━━━━━━━━━━━━━━━━ 1s 98ms/step - accuracy: 0.9834 - loss: 0.0953 - val_accuracy: 0.8077 - val_loss: 0.5235
Epoch 42/50
5/5 ━━━━━━━━━━━━━━━━━━━━ 1s 101ms/step - accuracy: 0.9854 - loss: 0.0810 - val_accuracy: 0.7923 - val_loss: 0.5387
Epoch 43/50
5/5 ━━━━━━━━━━━━━━━━━━━━ 1s 108ms/step - accuracy: 0.9903 - loss: 0.0713 - val_accuracy: 0.8000 - val_loss: 0.5450
Epoch 44/50
5/5 ━━━━━━━━━━━━━━━━━━━━ 1s 106ms/step - accuracy: 0.9865 - loss: 0.0749 - val_accuracy: 0.8154 - val_loss: 0.5134
Epoch 45/50
5/5 ━━━━━━━━━━━━━━━━━━━━ 1s 107ms/step - accuracy: 0.9938 - loss: 0.0689 - val_accuracy: 0.8077 - val_loss: 0.5387
Epoch 46/50
5/5 ━━━━━━━━━━━━━━━━━━━━ 1s 92ms/step - accuracy: 0.9923 - loss: 0.0664 - val_accuracy: 0.8000 - val_loss: 0.5847
Epoch 47/50
```

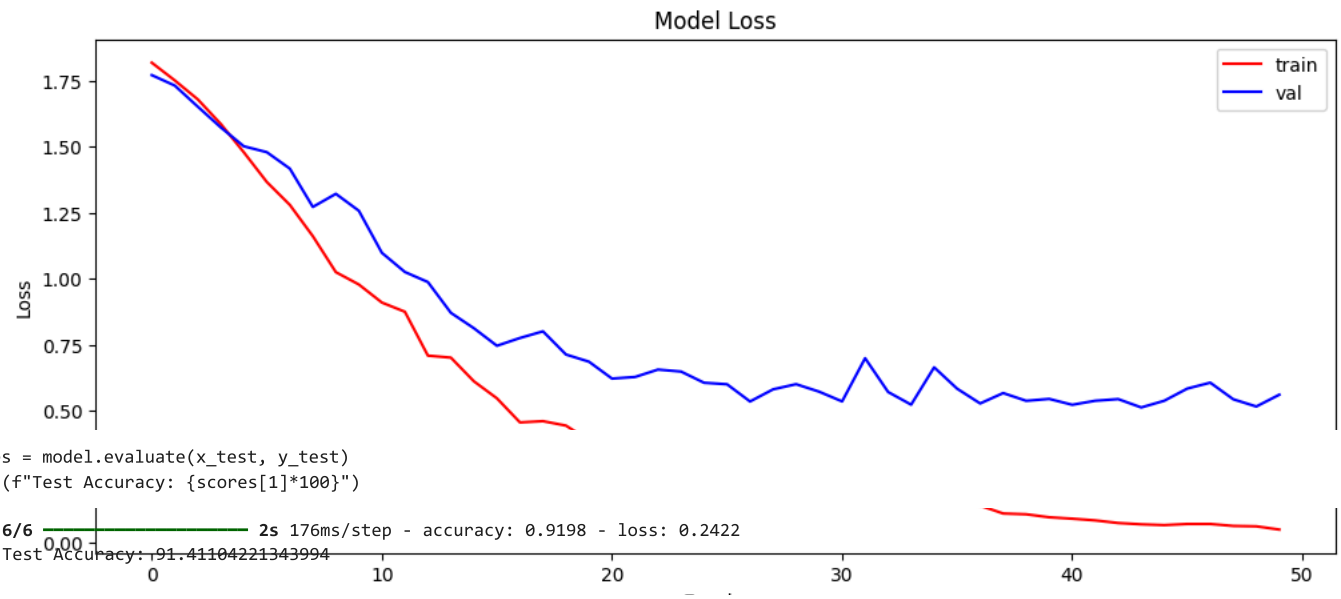
```
#Save
model.save("/content/drive/MyDrive/Data science/project12/bird_species.h5")
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is consi
```

```
#Plot
plt.figure(figsize=(12, 5))
plt.plot(history.history['accuracy'], color='r')
plt.plot(history.history['val_accuracy'], color='b')
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epochs')
plt.legend(['train', 'val'])
plt.show()
```




```
#Plot loss
plt.figure(figsize=(12, 5))
plt.plot(history.history['loss'], color='r')
plt.plot(history.history['val_loss'], color='b')
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epochs')
plt.legend(['train', 'val'])
plt.show()
```




```
scores = model.evaluate(x_test, y_test)
print(f"Test Accuracy: {scores[1]*100}")
```

```
6/6 ━━━━━━━━━━━━━━━━━━━━ 2s 176ms/step - accuracy: 0.9198 - loss: 0.2422
Test Accuracy: 91.41104221343994
```

```
y_pred = model.predict(x_test)
```

```
6/6 ━━━━━━━━━━━━━━━━━━━━ 1s 71ms/step
```

```
img = array_to_img(x_test[5])
img
```