# Kevin Tong - Bird Image Classification

## Packages

In [80]:
```python
import datetime
import time
from packaging import version
from collections import Counter
import numpy as np
import pandas as pd

import matplotlib as mpl  # EA
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.metrics import confusion_matrix, classification_report
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import mean_squared_error as MSE
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

import tensorflow as tf
from tensorflow.keras.utils import to_categorical
from tensorflow import keras
from tensorflow.keras import layers, models
from tensorflow.keras.models import Sequential
import tensorflow.keras.backend as k
from tensorflow.keras.utils import plot_model
from tensorflow.keras.layers import Conv2D, MaxPool2D, BatchNormalization, Dropout, Flatten, Dense
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping
from tensorflow.keras.preprocessing import image
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.layers import Dropout
import os
import cv2
from PIL import Image
from tensorflow.keras.regularizers import l2
```

In [2]:
```python
# from google.colab import drive
# drive.mount('/content/drive')
data_dir = '/content/drive/MyDrive/Northwestern MSDS/MSDS 458/458_Final_Notebooks/spectrogram_output'
```

## Preprocessing

In [3]:
```python
# Initialize a set to store unique classes (bird species)
unique_classes = set()

# Define a function to load and preprocess images
def load_and_preprocess_images(image_paths):
    images = []
    for path in image_paths:
        img = Image.open(path)
        img = img.resize((224, 224))  # Resize to your desired dimensions
        img = img.convert("RGB")  # Convert to RGB (remove alpha channel if present)
        img = np.array(img) / 255.0   # Normalize pixel values
        images.append(img)
    return np.array(images)

# Loop through directories and load images
all_images = []
for bird_folder in os.listdir(data_dir):
    bird_folder_path = os.path.join(data_dir, bird_folder)
    if os.path.isdir(bird_folder_path) and len(os.listdir(bird_folder_path)) > 1:
        # Add the current folder name to the set of unique classes
        unique_classes.add(bird_folder)

        image_paths = [os.path.join(bird_folder_path, img) for img in os.listdir(bird_folder_path)]
        images = load_and_preprocess_images(image_paths)
        all_images.extend(images)

all_images = np.array(all_images)

# Calculate the total number of classes
num_classes = len(unique_classes)
print("Number of classes:", num_classes)
```

```
Number of classes: 110
```

In [4]:
```python
# Load and preprocess labels
all_labels = []
for bird_folder in os.listdir(data_dir):
    bird_folder_path = os.path.join(data_dir, bird_folder)
    if os.path.isdir(bird_folder_path) and len(os.listdir(bird_folder_path)) > 1:
        label = bird_folder  # Assuming folder name is the label
        labels = [label] * len(os.listdir(bird_folder_path))
        all_labels.extend(labels)

# Convert labels to numerical format (e.g., one-hot encoding)
from sklearn.preprocessing import LabelEncoder, OneHotEncoder

label_encoder = LabelEncoder()
integer_encoded = label_encoder.fit_transform(all_labels)
onehot_encoder = OneHotEncoder(sparse=False)
encoded_labels = onehot_encoder.fit_transform(integer_encoded.reshape(-1, 1))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `sparse_output` is ignored unless you leave `sparse` to its default value.
  warnings.warn(
```

In [5]:
```python
from sklearn.model_selection import train_test_split

# Split data into training and temporary set (which includes validation and test data)
train_images, temp_images, train_labels, temp_labels = train_test_split(all_images, encoded_labels, test_size=0.2, random_state=42)

# Split temporary set into validation and test sets
val_images, test_images, val_labels, test_labels = train_test_split(temp_images, temp_labels, test_size=0.5, random_state=42)
```

In [6]:
```python
print("Train images shape:", train_images.shape)
print("Validation images shape:", val_images.shape)
print("Test images shape:", test_images.shape)
```

```
Train images shape: (1725, 224, 224, 3)
Validation images shape: (216, 224, 224, 3)
Test images shape: (216, 224, 224, 3)
```

## Neural Network

In [81]:
```python
start_time = time.time()

# Define your CNN model with Conv2D and Dropout layers
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3)),
    layers.MaxPooling2D((2, 2)),

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

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

    layers.Conv2D(256, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),

    layers.Flatten(),

    # Add dropout layers
    layers.Dropout(0.5),  # Add dropout after flattening
    layers.Dense(128, activation='relu', kernel_regularizer=l2(0.01)),  # L2 regularization

    layers.Dropout(0.5),  # Add dropout after the first dense layer
    layers.Dense(num_classes, activation='softmax')  # num_classes is the number of bird species
])

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

In [82]:
```python
# Print the model summary
model.summary()
```

```
Model: "sequential_13"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_25 (Conv2D)          (None, 222, 222, 32)      896

 max_pooling2d_25 (MaxPoolin  (None, 111, 111, 32)     0
 g2D)

 conv2d_26 (Conv2D)          (None, 109, 109, 64)      18496

 max_pooling2d_26 (MaxPoolin  (None, 54, 54, 64)       0
 g2D)

 conv2d_27 (Conv2D)          (None, 52, 52, 128)       73856

 max_pooling2d_27 (MaxPoolin  (None, 26, 26, 128)      0
 g2D)

 conv2d_28 (Conv2D)          (None, 24, 24, 256)       295168

 max_pooling2d_28 (MaxPoolin  (None, 12, 12, 256)      0
 g2D)

 flatten_13 (Flatten)        (None, 36864)             0

 dropout_20 (Dropout)        (None, 36864)             0

 dense_32 (Dense)            (None, 128)               4718720

 dropout_21 (Dropout)        (None, 128)               0

 dense_33 (Dense)            (None, 110)               14190

=================================================================
Total params: 5,121,326
Trainable params: 5,121,326
Non-trainable params: 0
_________________________________________________________________
```

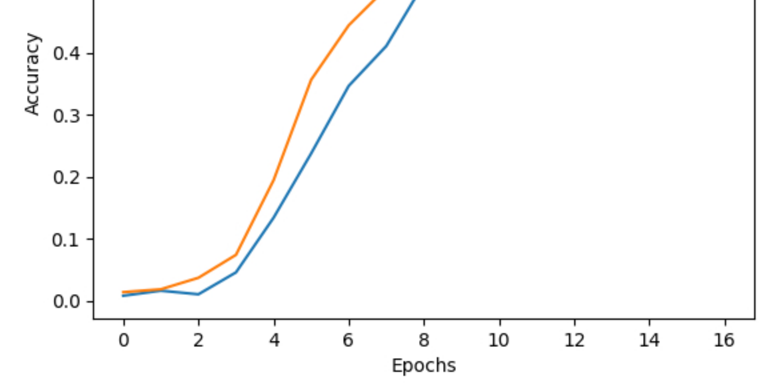
In [83]:
```python
# Train the model
history = model.fit(train_images, train_labels, epochs=200, batch_size=64,
                    validation_data=(val_images, val_labels),callbacks=[
                     tf.keras.callbacks.ModelCheckpoint("CNN_model.h5",save_best_only=True,save_weights_only=False)
                     ,tf.keras.callbacks.EarlyStopping(monitor='val_accuracy', patience=3),
                    ])

# Evaluate the model
test_loss, test_acc = model.evaluate(test_images, test_labels)
print('Test accuracy:', test_acc)
```

```
Epoch 1/200
27/27 [==============================] - 4s 45ms/step - loss: 5.4463 - accuracy: 0.0081 - val_loss: 4.8315 - val_acc
uracy: 0.0139
Epoch 2/200
27/27 [==============================] - 1s 32ms/step - loss: 4.7292 - accuracy: 0.0162 - val_loss: 4.6889 - val_acc
uracy: 0.0185
Epoch 3/200
27/27 [==============================] - 1s 32ms/step - loss: 4.6654 - accuracy: 0.0104 - val_loss: 4.6285 - val_acc
uracy: 0.0370
Epoch 4/200
27/27 [==============================] - 1s 34ms/step - loss: 4.6231 - accuracy: 0.0458 - val_loss: 4.4964 - val_acc
uracy: 0.0741
Epoch 5/200
27/27 [==============================] - 1s 33ms/step - loss: 4.3992 - accuracy: 0.1339 - val_loss: 4.1727 - val_acc
uracy: 0.1944
Epoch 6/200
27/27 [==============================] - 1s 57ms/step - loss: 4.1889 - accuracy: 0.2377 - val_loss: 3.9558 - val_acc
uracy: 0.3565
Epoch 7/200
27/27 [==============================] - 1s 33ms/step - loss: 3.9525 - accuracy: 0.3467 - val_loss: 3.8855 - val_acc
uracy: 0.4444
Epoch 8/200
27/27 [==============================] - 1s 33ms/step - loss: 3.7772 - accuracy: 0.4110 - val_loss: 3.6632 - val_acc
uracy: 0.5046
Epoch 9/200
27/27 [==============================] - 1s 32ms/step - loss: 3.4919 - accuracy: 0.5119 - val_loss: 3.6156 - val_acc
uracy: 0.5231
Epoch 10/200
27/27 [==============================] - 1s 32ms/step - loss: 3.3162 - accuracy: 0.5635 - val_loss: 3.5278 - val_acc
uracy: 0.6065
Epoch 11/200
27/27 [==============================] - 1s 33ms/step - loss: 3.2468 - accuracy: 0.5936 - val_loss: 3.4887 - val_acc
uracy: 0.6111
Epoch 12/200
27/27 [==============================] - 1s 28ms/step - loss: 3.1277 - accuracy: 0.6417 - val_loss: 3.5537 - val_acc
uracy: 0.6111
Epoch 13/200
27/27 [==============================] - 1s 28ms/step - loss: 3.1570 - accuracy: 0.6539 - val_loss: 3.5126 - val_acc
uracy: 0.6343
Epoch 14/200
27/27 [==============================] - 1s 27ms/step - loss: 2.9224 - accuracy: 0.6835 - val_loss: 3.5541 - val_acc
uracy: 0.6574
Epoch 15/200
27/27 [==============================] - 1s 33ms/step - loss: 2.8715 - accuracy: 0.7154 - val_loss: 3.4441 - val_acc
uracy: 0.6574
Epoch 16/200
27/27 [==============================] - 1s 27ms/step - loss: 2.8329 - accuracy: 0.7235 - val_loss: 3.5652 - val_acc
uracy: 0.6435
Epoch 17/200
27/27 [==============================] - 1s 33ms/step - loss: 2.8068 - accuracy: 0.7310 - val_loss: 3.3746 - val_acc
uracy: 0.6574
7/7 [==============================] - 0s 9ms/step - loss: 3.6227 - accuracy: 0.6296
Test accuracy: 0.6296296119689941
```

In [84]:
```python
end_time = time.time()
runtime = end_time - start_time
print('Total runtime:', runtime, 'seconds')
```

```
Total runtime: 22.145737648010254 seconds
```

In [86]:
```python
# Plot training and validation accuracy
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Training and Validation Accuracy')
plt.legend()
plt.show()
```




In [87]:
```python
# Plot test accuracy
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(len(history.history['val_accuracy']), test_acc, 'ro', label='Test Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Test Accuracy')
plt.legend()
plt.show()
```