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 z
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` t
o 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 [95]:
```python
start_time = time.time()

model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3)),
    layers.MaxPooling2D((2, 2)),

    layers.Conv2D(32, (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=tf.keras.regularizers.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 [96]:
```python
# Print the model summary
model.summary()
```

```
Model: "sequential_15"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_33 (Conv2D)          (None, 222, 222, 32)      896

 max_pooling2d_33 (MaxPoolin  (None, 111, 111, 32)     0
 g2D)

 conv2d_34 (Conv2D)          (None, 109, 109, 32)      9248

 max_pooling2d_34 (MaxPoolin  (None, 54, 54, 32)       0
 g2D)

 flatten_15 (Flatten)        (None, 93312)             0

 dropout_24 (Dropout)        (None, 93312)             0

 dense_36 (Dense)            (None, 128)               11944064

 dropout_25 (Dropout)        (None, 128)               0

 dense_37 (Dense)            (None, 110)               14190

=================================================================
Total params: 11,968,398
Trainable params: 11,968,398
Non-trainable params: 0
_________________________________________________________________
```

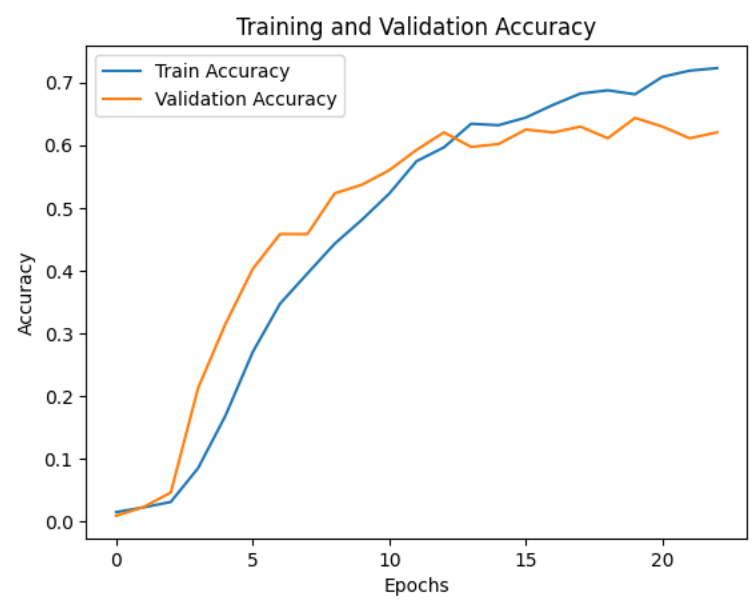
In [97]:
```python
# Train the model
history = model.fit(train_images, train_labels, epochs=200, batch_size=64,
                    validation_data=(val_images, val_labels),callbacks=[
                     tf.keras.callbacks.ModelCheckpoint("CNN_model.h5",save_best_only=True,save_weights_only=False)
                     ,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 [==============================] - 3s 44ms/step - loss: 5.9809 - accuracy: 0.0151 - val_loss: 5.2983 - val_acc
uracy: 0.0093
Epoch 2/200
27/27 [==============================] - 1s 32ms/step - loss: 5.0747 - accuracy: 0.0226 - val_loss: 4.9215 - val_acc
uracy: 0.0231
Epoch 3/200
27/27 [==============================] - 1s 32ms/step - loss: 4.8432 - accuracy: 0.0313 - val_loss: 4.7785 - val_acc
uracy: 0.0463
Epoch 4/200
27/27 [==============================] - 1s 33ms/step - loss: 4.7326 - accuracy: 0.0852 - val_loss: 4.7550 - val_acc
uracy: 0.2130
Epoch 5/200
27/27 [==============================] - 1s 32ms/step - loss: 4.7623 - accuracy: 0.1687 - val_loss: 4.5719 - val_acc
uracy: 0.3148
Epoch 6/200
27/27 [==============================] - 1s 33ms/step - loss: 4.5788 - accuracy: 0.2701 - val_loss: 4.5528 - val_acc
uracy: 0.4028
Epoch 7/200
27/27 [==============================] - 2s 84ms/step - loss: 4.4498 - accuracy: 0.3472 - val_loss: 4.4789 - val_acc
uracy: 0.4583
Epoch 8/200
27/27 [==============================] - 1s 33ms/step - loss: 4.2692 - accuracy: 0.3954 - val_loss: 4.3122 - val_acc
uracy: 0.4583
Epoch 9/200
27/27 [==============================] - 1s 51ms/step - loss: 4.1611 - accuracy: 0.4429 - val_loss: 4.2420 - val_acc
uracy: 0.5231
Epoch 10/200
27/27 [==============================] - 1s 23ms/step - loss: 4.0633 - accuracy: 0.4812 - val_loss: 4.2602 - val_acc
uracy: 0.5370
Epoch 11/200
27/27 [==============================] - 1s 33ms/step - loss: 3.8303 - accuracy: 0.5229 - val_loss: 3.9751 - val_acc
uracy: 0.5602
Epoch 12/200
27/27 [==============================] - 1s 23ms/step - loss: 3.6803 - accuracy: 0.5745 - val_loss: 4.0081 - val_acc
uracy: 0.5926
Epoch 13/200
27/27 [==============================] - 1s 22ms/step - loss: 3.7667 - accuracy: 0.5965 - val_loss: 4.1603 - val_acc
uracy: 0.6204
Epoch 14/200
27/27 [==============================] - 1s 22ms/step - loss: 3.6477 - accuracy: 0.6342 - val_loss: 3.9897 - val_acc
uracy: 0.5972
Epoch 15/200
27/27 [==============================] - 1s 34ms/step - loss: 3.5415 - accuracy: 0.6319 - val_loss: 3.9339 - val_acc
uracy: 0.6019
Epoch 16/200
27/27 [==============================] - 1s 34ms/step - loss: 3.4909 - accuracy: 0.6441 - val_loss: 3.8428 - val_acc
uracy: 0.6250
Epoch 17/200
27/27 [==============================] - 1s 23ms/step - loss: 3.3526 - accuracy: 0.6643 - val_loss: 3.8736 - val_acc
uracy: 0.6204
Epoch 18/200
27/27 [==============================] - 1s 33ms/step - loss: 3.2576 - accuracy: 0.6823 - val_loss: 3.8360 - val_acc
uracy: 0.6296
Epoch 19/200
27/27 [==============================] - 1s 22ms/step - loss: 3.3310 - accuracy: 0.6875 - val_loss: 3.9476 - val_acc
uracy: 0.6111
Epoch 20/200
27/27 [==============================] - 1s 22ms/step - loss: 3.3190 - accuracy: 0.6812 - val_loss: 4.0307 - val_acc
uracy: 0.6435
Epoch 21/200
27/27 [==============================] - 1s 23ms/step - loss: 3.3295 - accuracy: 0.7090 - val_loss: 4.0812 - val_acc
uracy: 0.6296
Epoch 22/200
27/27 [==============================] - 1s 23ms/step - loss: 3.2271 - accuracy: 0.7188 - val_loss: 3.9622 - val_acc
uracy: 0.6111
Epoch 23/200
27/27 [==============================] - 1s 23ms/step - loss: 3.1378 - accuracy: 0.7229 - val_loss: 3.8896 - val_acc
uracy: 0.6204
7/7 [==============================] - 0s 8ms/step - loss: 3.9971 - accuracy: 0.6435
Test accuracy: 0.6435185074806213
```

In [98]:
```python
end_time = time.time()
runtime = end_time - start_time
print('Total runtime:', runtime, 'seconds')
```

```
Total runtime: 25.649061679840088 seconds
```

In [101]:
```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 [102]:
```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()
```