phw3

May 11, 2025

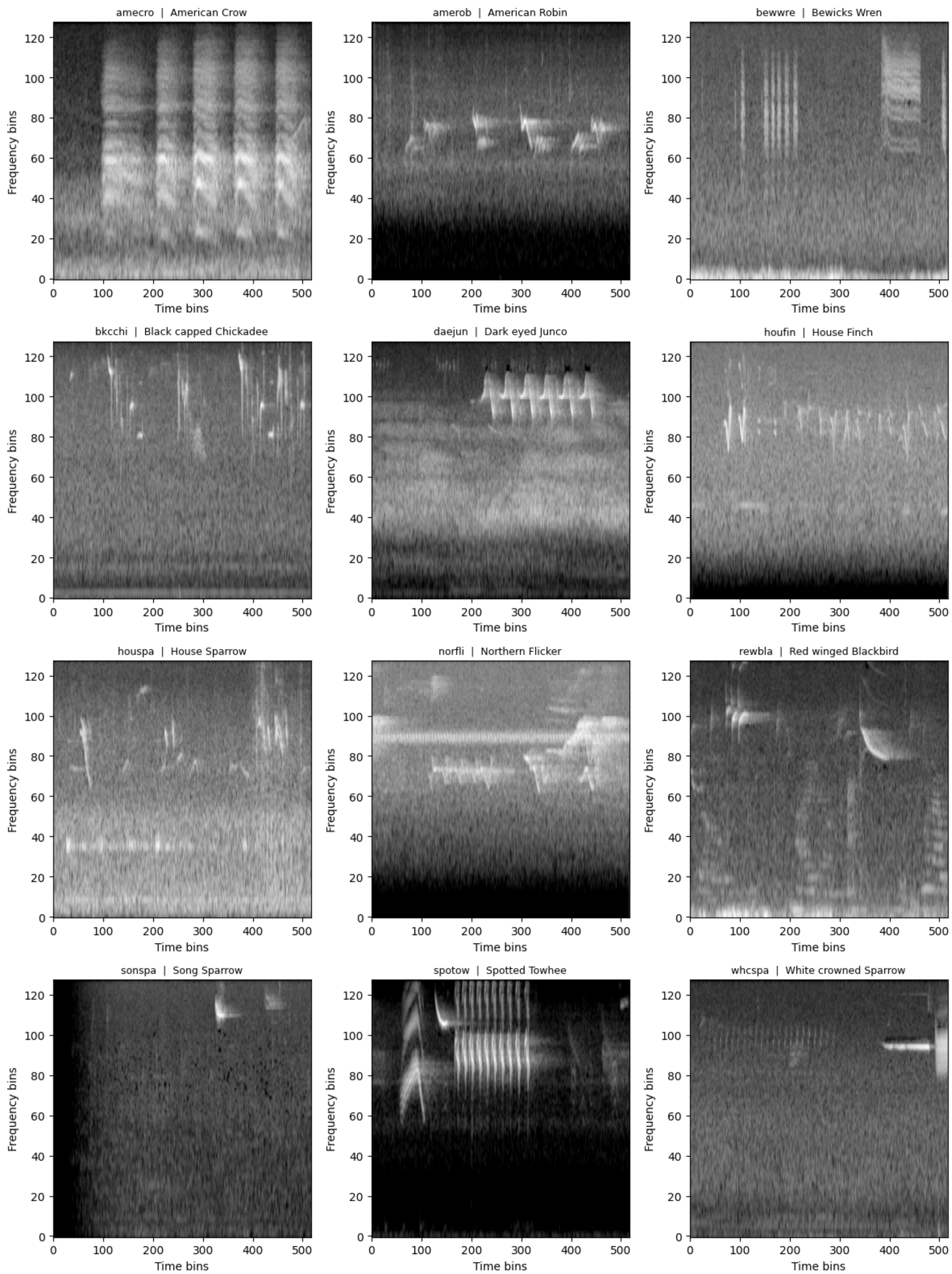
[2]:
```python
# Necessary libraries
import h5py
import numpy as np
import matplotlib.pyplot as plt
import math
from collections import Counter
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.utils import shuffle
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

from tensorflow.keras.utils import to_categorical

import librosa, librosa.display, numpy as np, matplotlib.pyplot as plt, pandas␣
↪as pd
from tensorflow.keras.models import load_model
import pathlib
```

[3]:
```python
# loading the file
data = h5py.File('birds/bird_spectrograms.hdf5', 'r')

bird_names = list(data.keys())
print(bird_names)

#labeling them for readability
names = {
    'amecro': 'American Crow',
    'amerob': 'American Robin',
    'bewwre': 'Bewicks Wren',
    'bkcchi': 'Black capped Chickadee',
    'daejun': 'Dark eyed Junco',
    'houfin': 'House Finch',
    'houspa': 'House Sparrow',
    'norfli': 'Northern Flicker',
    'rewbla': 'Red winged Blackbird',
    'sonspa': 'Song Sparrow',
```

```
    'spotow': 'Spotted Towhee',
    'whcspa': 'White crowned Sparrow'
}
print(names)

print("All bird species:")
all_birds = [names[code] for code in bird_names]
for bird in all_birds:
    print(bird)

for code in bird_names:
    shape = data[code].shape
    print(f"{code:7s} {names.get(code, 'Unknown'):25s} {shape}")

n_cols, n_rows = 3, math.ceil(len(bird_names) / 3)
fig, axes = plt.subplots(n_rows, n_cols, figsize=(12, 4 * n_rows))
axes = axes.flatten()

for i, code in enumerate(bird_names):
        spectro = data[code][:, :, 0]
        axes[i].imshow(spectro, aspect="auto", origin="lower", cmap="gray")
        axes[i].set_title(f"{code}  |  {names.get(code, '')}", fontsize=9)
        axes[i].set_xlabel("Time bins")
        axes[i].set_ylabel("Frequency bins")


for j in range(i + 1, len(axes)):
    axes[j].axis("off")

plt.tight_layout()
plt.show()
```

```
['amecro', 'amerob', 'bewwre', 'bkcchi', 'daejun', 'houfin', 'houspa', 'norfli',
'rewbla', 'sonspa', 'spotow', 'whcspa']
{'amecro': 'American Crow', 'amerob': 'American Robin', 'bewwre': 'Bewicks
Wren', 'bkcchi': 'Black capped Chickadee', 'daejun': 'Dark eyed Junco',
'houfin': 'House Finch', 'houspa': 'House Sparrow', 'norfli': 'Northern
Flicker', 'rewbla': 'Red winged Blackbird', 'sonspa': 'Song Sparrow', 'spotow':
'Spotted Towhee', 'whcspa': 'White crowned Sparrow'}
All bird species:
American Crow
American Robin
Bewicks Wren
Black capped Chickadee
Dark eyed Junco
House Finch
House Sparrow
Northern Flicker
```

```
Red winged Blackbird
Song Sparrow
Spotted Towhee
White crowned Sparrow
amecro  American Crow             (128, 517, 66)
amerob  American Robin            (128, 517, 172)
bewwre  Bewicks Wren              (128, 517, 144)
bkcchi  Black capped Chickadee    (128, 517, 45)
daejun  Dark eyed Junco           (128, 517, 125)
houfin  House Finch               (128, 517, 84)
houspa  House Sparrow             (128, 517, 630)
norfli  Northern Flicker          (128, 517, 37)
rewbla  Red winged Blackbird      (128, 517, 187)
sonspa  Song Sparrow              (128, 517, 263)
spotow  Spotted Towhee            (128, 517, 137)
whcspa  White crowned Sparrow     (128, 517, 91)
```




```
[4]: with data as f:
         song_sparrow = f['sonspa'][...].astype(np.float32) / 255.0
```

```
    white_sparrow = f['whcspa'][...].astype(np.float32) / 255.0

    # making into 2-d
    song_sparrow = np.transpose(song_sparrow, (2, 0, 1))
    white_sparrow = np.transpose(white_sparrow, (2, 0, 1))

    # making both the classes the same length
    minimum_length = min(song_sparrow.shape[0], white_sparrow.shape[0])
    song_sparrow = song_sparrow[:minimum_length]
    white_sparrow = white_sparrow[:minimum_length]

    # Labels, 0 for Song Sparrow, 1 for white crowned sparrow
    song_one = np.zeros(minimum_length, dtype=np.uint8)
    white_one = np.ones(minimum_length, dtype=np.uint8)

    x_class = np.concatenate([song_sparrow, white_sparrow], axis=0)
    y_class = np.concatenate([song_one, white_one], axis=0)
    x_class = x_class[..., np.newaxis]
    x_class, y_class = shuffle(x_class, y_class, random_state=42)

    # Splitting 30% train, 40% test, 30% validation set
    x_one, x_test, y_one, y_test = train_test_split(x_class, y_class,␣
 ↪test_size=0.4, stratify=y_class, random_state=42)
    x_train, x_valid, y_train, y_valid = train_test_split(x_one, y_one,␣
 ↪test_size=0.5, stratify=y_one, random_state=42)
```

[ ]:
```
# binary CNN model
model_one = tf.keras.Sequential([
    tf.keras.layers.Conv2D(16, (3, 3), activation='relu', input_shape=x_train.
 ↪shape[1:]),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Dense(1, activation='sigmoid')
])


model_one.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=1e-4),
    loss='binary_crossentropy',
    metrics=['accuracy', tf.keras.metrics.AUC(name='auc')]
)
```

```python
model_one.summary()

callbacks_one = [
    tf.keras.callbacks.EarlyStopping(patience=10, restore_best_weights=True),
    tf.keras.callbacks.ModelCheckpoint("lecture_cnn_sparrows.keras",␣
 ↪save_best_only=True)
]

history = model_one.fit(
    x_train, y_train,
    validation_data=(x_valid, y_valid),
    epochs=100,
    batch_size=32,
    callbacks=callbacks_one,
    verbose=1
)

plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.legend()
plt.title('Accuracy')

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend()
plt.title('Loss')

plt.tight_layout()
plt.show()

loss, acc, auc = model_one.evaluate(x_test, y_test, verbose=0)
print(f"\n Final Test Accuracy: {acc:.4f} | AUC: {auc:.4f}")

probability_one = model_one.predict(x_test)
prediction_one = (probability_one > 0.5).astype(int).flatten()

conf_matrix = confusion_matrix(y_test, prediction_one)
matrix_one = ConfusionMatrixDisplay(confusion_matrix=conf_matrix,␣
 ↪display_labels=['Song Sparrow', 'White crowned Sparrow'])
matrix_one.plot(cmap=plt.cm.Blues)
plt.title("Confusion Matrix")
plt.show()
```

```
Model: "sequential"
```

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 126, 515, 16)      160       
                                                                 
 max_pooling2d (MaxPooling2D  (None, 63, 257, 16)      0         
 )                                                               
                                                                 
 conv2d_1 (Conv2D)           (None, 61, 255, 31)       4495      
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 30, 127, 31)      0         
 2D)                                                             
                                                                 
 flatten (Flatten)           (None, 118110)            0         
                                                                 
 dense (Dense)               (None, 32)                3779552   
                                                                 
 dropout (Dropout)           (None, 32)                0         
                                                                 
 dense_1 (Dense)             (None, 1)                 33        
                                                                 
=================================================================
Total params: 3,784,240
Trainable params: 3,784,240
Non-trainable params: 0
_________________________________________________________________
Epoch 1/100
2/2 [==============================] - 1s 645ms/step - loss: 0.6888 - accuracy:
0.5000 - auc: 0.6187 - val_loss: 0.6864 - val_accuracy: 0.4909 - val_auc: 0.7698
Epoch 2/100
2/2 [==============================] - 1s 438ms/step - loss: 0.7004 - accuracy:
0.5000 - auc: 0.5123 - val_loss: 0.6800 - val_accuracy: 0.4909 - val_auc: 0.7884
Epoch 3/100
2/2 [==============================] - 1s 454ms/step - loss: 0.6759 - accuracy:
0.5926 - auc: 0.7154 - val_loss: 0.6774 - val_accuracy: 0.6182 - val_auc: 0.7930
Epoch 4/100
2/2 [==============================] - 1s 421ms/step - loss: 0.6854 - accuracy:
0.6111 - auc: 0.6043 - val_loss: 0.6749 - val_accuracy: 0.6182 - val_auc: 0.7884
Epoch 5/100
2/2 [==============================] - 1s 390ms/step - loss: 0.6797 - accuracy:
0.6296 - auc: 0.6529 - val_loss: 0.6686 - val_accuracy: 0.5455 - val_auc: 0.7844
Epoch 6/100
2/2 [==============================] - 1s 397ms/step - loss: 0.6613 - accuracy:
0.6481 - auc: 0.7641 - val_loss: 0.6662 - val_accuracy: 0.5091 - val_auc: 0.7844
Epoch 7/100
2/2 [==============================] - 1s 400ms/step - loss: 0.6587 - accuracy:
0.6111 - auc: 0.7867 - val_loss: 0.6628 - val_accuracy: 0.5455 - val_auc: 0.7890
Epoch 8/100
```

```
2/2 [==============================] - 1s 410ms/step - loss: 0.6529 - accuracy:
0.6296 - auc: 0.7538 - val_loss: 0.6587 - val_accuracy: 0.6182 - val_auc: 0.7864
Epoch 9/100
2/2 [==============================] - 1s 394ms/step - loss: 0.6497 - accuracy:
0.7037 - auc: 0.7682 - val_loss: 0.6555 - val_accuracy: 0.6909 - val_auc: 0.7831
Epoch 10/100
2/2 [==============================] - 1s 414ms/step - loss: 0.6703 - accuracy:
0.6111 - auc: 0.6584 - val_loss: 0.6532 - val_accuracy: 0.7273 - val_auc: 0.7890
Epoch 11/100
2/2 [==============================] - 1s 398ms/step - loss: 0.6486 - accuracy:
0.7222 - auc: 0.7483 - val_loss: 0.6504 - val_accuracy: 0.6909 - val_auc: 0.7831
Epoch 12/100
2/2 [==============================] - 1s 396ms/step - loss: 0.6355 - accuracy:
0.7222 - auc: 0.7846 - val_loss: 0.6485 - val_accuracy: 0.6909 - val_auc: 0.7831
Epoch 13/100
2/2 [==============================] - 1s 411ms/step - loss: 0.6470 - accuracy:
0.6852 - auc: 0.7250 - val_loss: 0.6466 - val_accuracy: 0.7091 - val_auc: 0.7851
Epoch 14/100
2/2 [==============================] - 1s 442ms/step - loss: 0.6441 - accuracy:
0.5926 - auc: 0.7222 - val_loss: 0.6444 - val_accuracy: 0.7091 - val_auc: 0.7791
Epoch 15/100
2/2 [==============================] - 1s 560ms/step - loss: 0.6499 - accuracy:
0.6481 - auc: 0.7023 - val_loss: 0.6441 - val_accuracy: 0.6727 - val_auc: 0.7665
Epoch 16/100
2/2 [==============================] - 1s 613ms/step - loss: 0.6519 - accuracy:
0.6667 - auc: 0.7119 - val_loss: 0.6405 - val_accuracy: 0.6727 - val_auc: 0.7685
Epoch 17/100
2/2 [==============================] - 1s 547ms/step - loss: 0.6392 - accuracy:
0.7037 - auc: 0.7332 - val_loss: 0.6362 - val_accuracy: 0.6727 - val_auc: 0.7738
Epoch 18/100
2/2 [==============================] - 1s 600ms/step - loss: 0.6227 - accuracy:
0.7037 - auc: 0.7545 - val_loss: 0.6318 - val_accuracy: 0.7273 - val_auc: 0.7778
Epoch 19/100
2/2 [==============================] - 1s 568ms/step - loss: 0.6107 - accuracy:
0.7222 - auc: 0.7840 - val_loss: 0.6283 - val_accuracy: 0.7091 - val_auc: 0.7817
Epoch 20/100
2/2 [==============================] - 1s 600ms/step - loss: 0.5914 - accuracy:
0.7407 - auc: 0.8114 - val_loss: 0.6267 - val_accuracy: 0.7273 - val_auc: 0.7864
Epoch 21/100
2/2 [==============================] - 1s 567ms/step - loss: 0.6107 - accuracy:
0.6481 - auc: 0.7798 - val_loss: 0.6231 - val_accuracy: 0.7273 - val_auc: 0.7857
Epoch 22/100
2/2 [==============================] - 1s 571ms/step - loss: 0.6011 - accuracy:
0.7222 - auc: 0.7819 - val_loss: 0.6200 - val_accuracy: 0.6909 - val_auc: 0.7837
Epoch 23/100
2/2 [==============================] - 1s 580ms/step - loss: 0.6295 - accuracy:
0.6296 - auc: 0.7318 - val_loss: 0.6190 - val_accuracy: 0.6909 - val_auc: 0.7837
Epoch 24/100
```

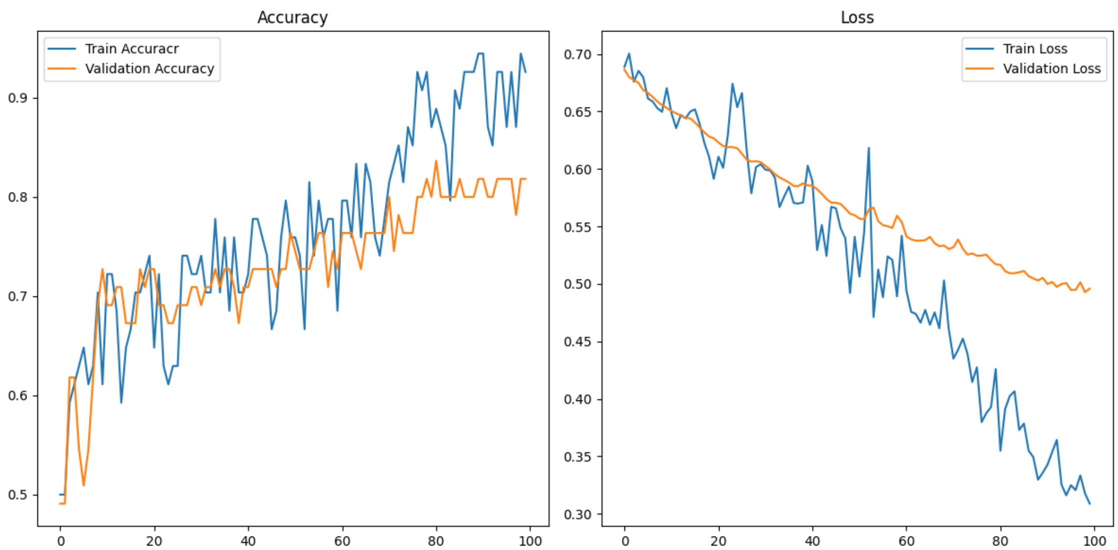
```
2/2 [==============================] - 1s 567ms/step - loss: 0.6742 - accuracy:
0.6111 - auc: 0.6255 - val_loss: 0.6190 - val_accuracy: 0.6727 - val_auc: 0.7791
Epoch 25/100
2/2 [==============================] - 1s 567ms/step - loss: 0.6538 - accuracy:
0.6296 - auc: 0.6818 - val_loss: 0.6181 - val_accuracy: 0.6727 - val_auc: 0.7784
Epoch 26/100
2/2 [==============================] - 1s 550ms/step - loss: 0.6661 - accuracy:
0.6296 - auc: 0.6536 - val_loss: 0.6133 - val_accuracy: 0.6909 - val_auc: 0.7870
Epoch 27/100
2/2 [==============================] - 1s 562ms/step - loss: 0.6168 - accuracy:
0.7407 - auc: 0.7373 - val_loss: 0.6083 - val_accuracy: 0.6909 - val_auc: 0.7917
Epoch 28/100
2/2 [==============================] - 1s 561ms/step - loss: 0.5789 - accuracy:
0.7407 - auc: 0.8045 - val_loss: 0.6064 - val_accuracy: 0.6909 - val_auc: 0.7903
Epoch 29/100
2/2 [==============================] - 1s 500ms/step - loss: 0.6017 - accuracy:
0.7222 - auc: 0.7730 - val_loss: 0.6068 - val_accuracy: 0.7091 - val_auc: 0.7917
Epoch 30/100
2/2 [==============================] - 1s 566ms/step - loss: 0.6041 - accuracy:
0.7222 - auc: 0.7778 - val_loss: 0.6060 - val_accuracy: 0.7091 - val_auc: 0.7903
Epoch 31/100
2/2 [==============================] - 1s 570ms/step - loss: 0.5994 - accuracy:
0.7407 - auc: 0.7922 - val_loss: 0.6027 - val_accuracy: 0.6909 - val_auc: 0.7950
Epoch 32/100
2/2 [==============================] - 1s 569ms/step - loss: 0.5988 - accuracy:
0.7037 - auc: 0.7599 - val_loss: 0.5996 - val_accuracy: 0.7091 - val_auc: 0.7956
Epoch 33/100
2/2 [==============================] - 1s 550ms/step - loss: 0.5926 - accuracy:
0.7037 - auc: 0.7997 - val_loss: 0.5955 - val_accuracy: 0.7091 - val_auc: 0.7970
Epoch 34/100
2/2 [==============================] - 1s 559ms/step - loss: 0.5669 - accuracy:
0.7778 - auc: 0.8182 - val_loss: 0.5926 - val_accuracy: 0.7273 - val_auc: 0.7989
Epoch 35/100
2/2 [==============================] - 1s 583ms/step - loss: 0.5762 - accuracy:
0.7037 - auc: 0.8086 - val_loss: 0.5905 - val_accuracy: 0.7091 - val_auc: 0.8009
Epoch 36/100
2/2 [==============================] - 1s 567ms/step - loss: 0.5846 - accuracy:
0.7593 - auc: 0.8059 - val_loss: 0.5883 - val_accuracy: 0.7273 - val_auc: 0.8009
Epoch 37/100
2/2 [==============================] - 1s 558ms/step - loss: 0.5704 - accuracy:
0.6852 - auc: 0.8251 - val_loss: 0.5852 - val_accuracy: 0.7273 - val_auc: 0.8049
Epoch 38/100
2/2 [==============================] - 1s 550ms/step - loss: 0.5699 - accuracy:
0.7593 - auc: 0.8141 - val_loss: 0.5851 - val_accuracy: 0.7091 - val_auc: 0.8036
Epoch 39/100
2/2 [==============================] - 1s 484ms/step - loss: 0.5710 - accuracy:
0.7037 - auc: 0.8045 - val_loss: 0.5876 - val_accuracy: 0.6727 - val_auc: 0.8029
Epoch 40/100
```

```
2/2 [==============================] - 1s 484ms/step - loss: 0.6029 - accuracy:
0.7037 - auc: 0.7695 - val_loss: 0.5858 - val_accuracy: 0.7091 - val_auc: 0.8022
Epoch 41/100
2/2 [==============================] - 1s 500ms/step - loss: 0.5892 - accuracy:
0.7222 - auc: 0.7785 - val_loss: 0.5856 - val_accuracy: 0.7091 - val_auc: 0.8036
Epoch 42/100
2/2 [==============================] - 1s 569ms/step - loss: 0.5294 - accuracy:
0.7778 - auc: 0.8512 - val_loss: 0.5823 - val_accuracy: 0.7273 - val_auc: 0.8056
Epoch 43/100
2/2 [==============================] - 1s 484ms/step - loss: 0.5513 - accuracy:
0.7778 - auc: 0.8388 - val_loss: 0.5785 - val_accuracy: 0.7273 - val_auc: 0.8069
Epoch 44/100
2/2 [==============================] - 1s 500ms/step - loss: 0.5244 - accuracy:
0.7593 - auc: 0.8567 - val_loss: 0.5739 - val_accuracy: 0.7273 - val_auc: 0.8069
Epoch 45/100
2/2 [==============================] - 1s 520ms/step - loss: 0.5671 - accuracy:
0.7407 - auc: 0.7805 - val_loss: 0.5707 - val_accuracy: 0.7273 - val_auc: 0.8082
Epoch 46/100
2/2 [==============================] - 1s 504ms/step - loss: 0.5658 - accuracy:
0.6667 - auc: 0.7915 - val_loss: 0.5706 - val_accuracy: 0.7273 - val_auc: 0.8075
Epoch 47/100
2/2 [==============================] - 1s 487ms/step - loss: 0.5485 - accuracy:
0.6852 - auc: 0.8073 - val_loss: 0.5695 - val_accuracy: 0.7091 - val_auc: 0.8102
Epoch 48/100
2/2 [==============================] - 1s 487ms/step - loss: 0.5396 - accuracy:
0.7593 - auc: 0.8189 - val_loss: 0.5656 - val_accuracy: 0.7273 - val_auc: 0.8102
Epoch 49/100
2/2 [==============================] - 1s 500ms/step - loss: 0.4921 - accuracy:
0.7963 - auc: 0.8896 - val_loss: 0.5611 - val_accuracy: 0.7273 - val_auc: 0.8095
Epoch 50/100
2/2 [==============================] - 1s 494ms/step - loss: 0.5410 - accuracy:
0.7593 - auc: 0.8073 - val_loss: 0.5598 - val_accuracy: 0.7636 - val_auc: 0.8102
Epoch 51/100
2/2 [==============================] - 1s 483ms/step - loss: 0.5063 - accuracy:
0.7593 - auc: 0.8567 - val_loss: 0.5569 - val_accuracy: 0.7455 - val_auc: 0.8082
Epoch 52/100
2/2 [==============================] - 1s 499ms/step - loss: 0.5449 - accuracy:
0.7407 - auc: 0.7956 - val_loss: 0.5563 - val_accuracy: 0.7273 - val_auc: 0.8082
Epoch 53/100
2/2 [==============================] - 1s 438ms/step - loss: 0.6184 - accuracy:
0.6667 - auc: 0.7270 - val_loss: 0.5647 - val_accuracy: 0.7273 - val_auc: 0.8102
Epoch 54/100
2/2 [==============================] - 1s 437ms/step - loss: 0.4712 - accuracy:
0.8148 - auc: 0.8978 - val_loss: 0.5664 - val_accuracy: 0.7273 - val_auc: 0.8102
Epoch 55/100
2/2 [==============================] - 1s 516ms/step - loss: 0.5126 - accuracy:
0.7407 - auc: 0.8237 - val_loss: 0.5546 - val_accuracy: 0.7455 - val_auc: 0.8089
Epoch 56/100
```

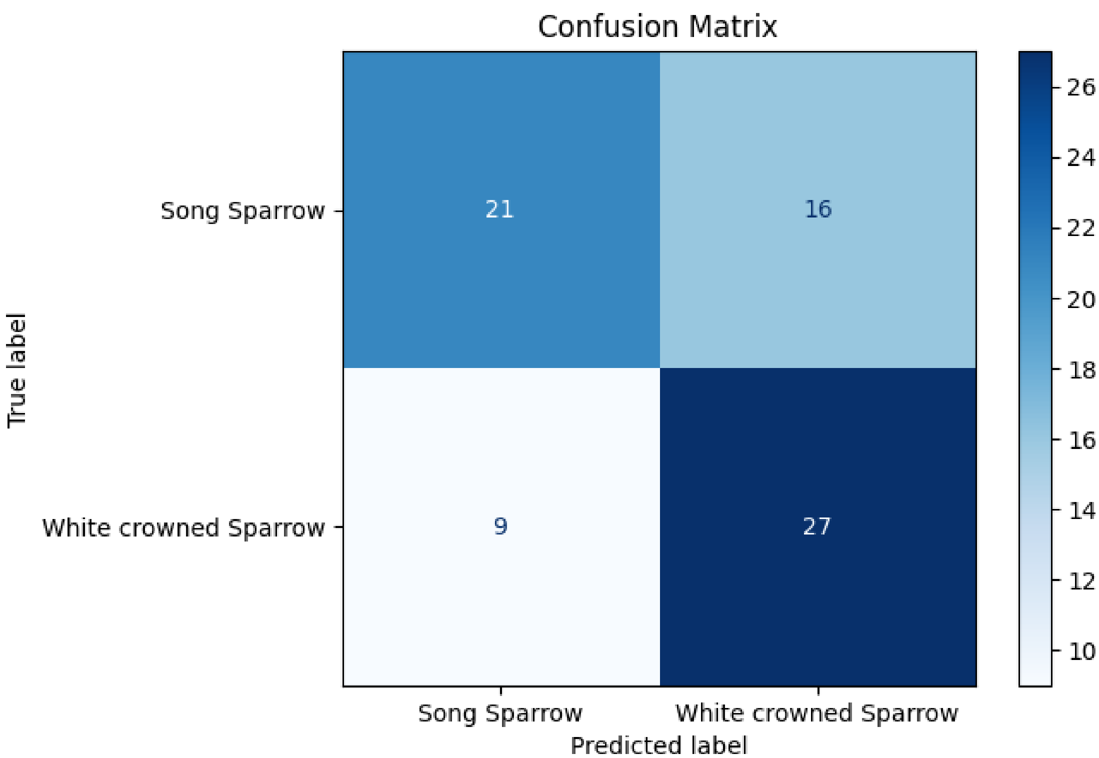
```
2/2 [==============================] - 1s 500ms/step - loss: 0.4885 - accuracy:
0.7963 - auc: 0.8656 - val_loss: 0.5511 - val_accuracy: 0.7636 - val_auc: 0.8115
Epoch 57/100
2/2 [==============================] - 1s 498ms/step - loss: 0.5240 - accuracy:
0.7593 - auc: 0.8381 - val_loss: 0.5503 - val_accuracy: 0.7636 - val_auc: 0.8122
Epoch 58/100
2/2 [==============================] - 1s 537ms/step - loss: 0.5209 - accuracy:
0.7778 - auc: 0.8333 - val_loss: 0.5487 - val_accuracy: 0.7091 - val_auc: 0.8128
Epoch 59/100
2/2 [==============================] - 1s 434ms/step - loss: 0.4892 - accuracy:
0.7778 - auc: 0.8827 - val_loss: 0.5594 - val_accuracy: 0.7455 - val_auc: 0.8135
Epoch 60/100
2/2 [==============================] - 1s 434ms/step - loss: 0.5419 - accuracy:
0.6852 - auc: 0.8121 - val_loss: 0.5539 - val_accuracy: 0.7273 - val_auc: 0.8148
Epoch 61/100
2/2 [==============================] - 1s 499ms/step - loss: 0.4943 - accuracy:
0.7963 - auc: 0.8683 - val_loss: 0.5414 - val_accuracy: 0.7636 - val_auc: 0.8181
Epoch 62/100
2/2 [==============================] - 1s 485ms/step - loss: 0.4757 - accuracy:
0.7963 - auc: 0.8717 - val_loss: 0.5388 - val_accuracy: 0.7636 - val_auc: 0.8188
Epoch 63/100
2/2 [==============================] - 1s 497ms/step - loss: 0.4740 - accuracy:
0.7593 - auc: 0.8704 - val_loss: 0.5376 - val_accuracy: 0.7636 - val_auc: 0.8168
Epoch 64/100
2/2 [==============================] - 1s 437ms/step - loss: 0.4663 - accuracy:
0.8333 - auc: 0.8923 - val_loss: 0.5377 - val_accuracy: 0.7455 - val_auc: 0.8175
Epoch 65/100
2/2 [==============================] - 1s 450ms/step - loss: 0.4774 - accuracy:
0.7593 - auc: 0.8731 - val_loss: 0.5381 - val_accuracy: 0.7273 - val_auc: 0.8181
Epoch 66/100
2/2 [==============================] - 1s 450ms/step - loss: 0.4645 - accuracy:
0.8333 - auc: 0.8841 - val_loss: 0.5410 - val_accuracy: 0.7636 - val_auc: 0.8208
Epoch 67/100
2/2 [==============================] - 1s 467ms/step - loss: 0.4753 - accuracy:
0.8148 - auc: 0.8813 - val_loss: 0.5354 - val_accuracy: 0.7636 - val_auc: 0.8228
Epoch 68/100
2/2 [==============================] - 1s 466ms/step - loss: 0.4614 - accuracy:
0.7593 - auc: 0.8786 - val_loss: 0.5329 - val_accuracy: 0.7636 - val_auc: 0.8241
Epoch 69/100
2/2 [==============================] - 1s 419ms/step - loss: 0.5030 - accuracy:
0.7407 - auc: 0.8381 - val_loss: 0.5334 - val_accuracy: 0.7636 - val_auc: 0.8228
Epoch 70/100
2/2 [==============================] - 1s 487ms/step - loss: 0.4609 - accuracy:
0.7778 - auc: 0.9053 - val_loss: 0.5304 - val_accuracy: 0.7636 - val_auc: 0.8254
Epoch 71/100
2/2 [==============================] - 1s 522ms/step - loss: 0.4351 - accuracy:
0.8148 - auc: 0.9225 - val_loss: 0.5320 - val_accuracy: 0.8000 - val_auc: 0.8228
Epoch 72/100
```

```
2/2 [==============================] - 1s 533ms/step - loss: 0.4428 - accuracy:
0.8333 - auc: 0.9198 - val_loss: 0.5386 - val_accuracy: 0.7455 - val_auc: 0.8254
Epoch 73/100
2/2 [==============================] - 1s 500ms/step - loss: 0.4525 - accuracy:
0.8519 - auc: 0.9053 - val_loss: 0.5306 - val_accuracy: 0.7818 - val_auc: 0.8254
Epoch 74/100
2/2 [==============================] - 1s 550ms/step - loss: 0.4391 - accuracy:
0.8148 - auc: 0.9047 - val_loss: 0.5255 - val_accuracy: 0.7636 - val_auc: 0.8267
Epoch 75/100
2/2 [==============================] - 1s 500ms/step - loss: 0.4147 - accuracy:
0.8704 - auc: 0.9396 - val_loss: 0.5265 - val_accuracy: 0.7636 - val_auc: 0.8261
Epoch 76/100
2/2 [==============================] - 1s 569ms/step - loss: 0.4275 - accuracy:
0.8519 - auc: 0.9294 - val_loss: 0.5246 - val_accuracy: 0.7636 - val_auc: 0.8307
Epoch 77/100
2/2 [==============================] - 1s 503ms/step - loss: 0.3799 - accuracy:
0.9259 - auc: 0.9479 - val_loss: 0.5247 - val_accuracy: 0.8000 - val_auc: 0.8307
Epoch 78/100
2/2 [==============================] - 1s 500ms/step - loss: 0.3876 - accuracy:
0.9074 - auc: 0.9588 - val_loss: 0.5255 - val_accuracy: 0.8000 - val_auc: 0.8320
Epoch 79/100
2/2 [==============================] - 1s 568ms/step - loss: 0.3929 - accuracy:
0.9259 - auc: 0.9568 - val_loss: 0.5208 - val_accuracy: 0.8182 - val_auc: 0.8333
Epoch 80/100
2/2 [==============================] - 1s 583ms/step - loss: 0.4260 - accuracy:
0.8704 - auc: 0.9335 - val_loss: 0.5171 - val_accuracy: 0.8000 - val_auc: 0.8366
Epoch 81/100
2/2 [==============================] - 1s 566ms/step - loss: 0.3549 - accuracy:
0.8889 - auc: 0.9609 - val_loss: 0.5167 - val_accuracy: 0.8364 - val_auc: 0.8360
Epoch 82/100
2/2 [==============================] - 1s 565ms/step - loss: 0.3912 - accuracy:
0.8704 - auc: 0.9492 - val_loss: 0.5112 - val_accuracy: 0.8000 - val_auc: 0.8353
Epoch 83/100
2/2 [==============================] - 1s 549ms/step - loss: 0.4026 - accuracy:
0.8519 - auc: 0.9369 - val_loss: 0.5093 - val_accuracy: 0.8000 - val_auc: 0.8327
Epoch 84/100
2/2 [==============================] - 1s 488ms/step - loss: 0.4066 - accuracy:
0.7963 - auc: 0.9218 - val_loss: 0.5094 - val_accuracy: 0.8000 - val_auc: 0.8287
Epoch 85/100
2/2 [==============================] - 1s 496ms/step - loss: 0.3730 - accuracy:
0.9074 - auc: 0.9616 - val_loss: 0.5102 - val_accuracy: 0.8000 - val_auc: 0.8287
Epoch 86/100
2/2 [==============================] - 1s 489ms/step - loss: 0.3786 - accuracy:
0.8889 - auc: 0.9616 - val_loss: 0.5113 - val_accuracy: 0.8182 - val_auc: 0.8280
Epoch 87/100
2/2 [==============================] - 1s 549ms/step - loss: 0.3551 - accuracy:
0.9259 - auc: 0.9698 - val_loss: 0.5068 - val_accuracy: 0.8000 - val_auc: 0.8307
Epoch 88/100
```

```
2/2 [==============================] - 1s 666ms/step - loss: 0.3493 - accuracy:
0.9259 - auc: 0.9534 - val_loss: 0.5048 - val_accuracy: 0.8000 - val_auc: 0.8300
Epoch 89/100
2/2 [==============================] - 1s 606ms/step - loss: 0.3297 - accuracy:
0.9259 - auc: 0.9794 - val_loss: 0.5031 - val_accuracy: 0.8000 - val_auc: 0.8313
Epoch 90/100
2/2 [==============================] - 1s 546ms/step - loss: 0.3358 - accuracy:
0.9444 - auc: 0.9746 - val_loss: 0.5053 - val_accuracy: 0.8182 - val_auc: 0.8360
Epoch 91/100
2/2 [==============================] - 1s 617ms/step - loss: 0.3427 - accuracy:
0.9444 - auc: 0.9781 - val_loss: 0.5000 - val_accuracy: 0.8182 - val_auc: 0.8300
Epoch 92/100
2/2 [==============================] - 1s 583ms/step - loss: 0.3537 - accuracy:
0.8704 - auc: 0.9616 - val_loss: 0.5017 - val_accuracy: 0.8000 - val_auc: 0.8307
Epoch 93/100
2/2 [==============================] - 1s 551ms/step - loss: 0.3643 - accuracy:
0.8519 - auc: 0.9664 - val_loss: 0.4975 - val_accuracy: 0.8000 - val_auc: 0.8327
Epoch 94/100
2/2 [==============================] - 1s 484ms/step - loss: 0.3254 - accuracy:
0.9259 - auc: 0.9691 - val_loss: 0.4999 - val_accuracy: 0.8182 - val_auc: 0.8360
Epoch 95/100
2/2 [==============================] - 1s 503ms/step - loss: 0.3161 - accuracy:
0.9259 - auc: 0.9883 - val_loss: 0.5008 - val_accuracy: 0.8182 - val_auc: 0.8446
Epoch 96/100
2/2 [==============================] - 1s 603ms/step - loss: 0.3249 - accuracy:
0.8704 - auc: 0.9726 - val_loss: 0.4948 - val_accuracy: 0.8182 - val_auc: 0.8433
Epoch 97/100
2/2 [==============================] - 1s 486ms/step - loss: 0.3207 - accuracy:
0.9259 - auc: 0.9746 - val_loss: 0.4949 - val_accuracy: 0.8182 - val_auc: 0.8419
Epoch 98/100
2/2 [==============================] - 1s 500ms/step - loss: 0.3335 - accuracy:
0.8704 - auc: 0.9794 - val_loss: 0.5015 - val_accuracy: 0.7818 - val_auc: 0.8433
Epoch 99/100
2/2 [==============================] - 1s 545ms/step - loss: 0.3177 - accuracy:
0.9444 - auc: 0.9733 - val_loss: 0.4928 - val_accuracy: 0.8182 - val_auc: 0.8439
Epoch 100/100
2/2 [==============================] - 1s 495ms/step - loss: 0.3089 - accuracy:
0.9259 - auc: 0.9877 - val_loss: 0.4960 - val_accuracy: 0.8182 - val_auc: 0.8439
```




```
Final Test Accuracy: 0.6575 | AUC: 0.6821
3/3 [==============================] - 0s 67ms/step
```



[ ]:
```python
# MULTI-CLASS MODEL
x_all, y_all = [], []
all_birds  = []

with data as f:
    for idx, key in enumerate(f.keys()):
        spec = f[key][...].astype(np.float32) / 255.0
        spec = np.transpose(spec, (2, 0, 1))
        x_all.append(spec)
        y_all.append(np.full(spec.shape[0], idx, dtype=np.uint8))
        all_birds.append(key)

x_class = np.concatenate(x_all, axis=0)
y_class = np.concatenate(y_all, axis=0)
x_class = x_class[..., np.newaxis]

x_class, y_class = shuffle(x_class, y_class, random_state=42)

x_one, x_test, y_one, y_test = train_test_split(x_class, y_class, test_size=0.
↪4, stratify=y_class, random_state=42)
x_train, x_valid, y_train, y_valid = train_test_split(x_one, y_one, test_size=0.
↪5, stratify=y_one, random_state=42)

x_multi, x_test, y_multi, y_test = train_test_split(x_class, y_class,
                                               test_size=0.40,␣
↪stratify=y_class, random_state=42)

x_train, x_valid, y_train, y_valid = train_test_split(x_multi, y_multi,
                                                     test_size=0.50,␣
↪stratify=y_multi, random_state=42)

num_class = len(all_birds)
print("Input shape: ", x_train.shape[1:], " | classes:", num_class)
```

```
Input shape:  (128, 517, 1)  | classes: 12
```

[18]:
```python
model_two = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, 3, activation='relu', input_shape=x_train.
↪shape[1:]),
    tf.keras.layers.MaxPooling2D(2),

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

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

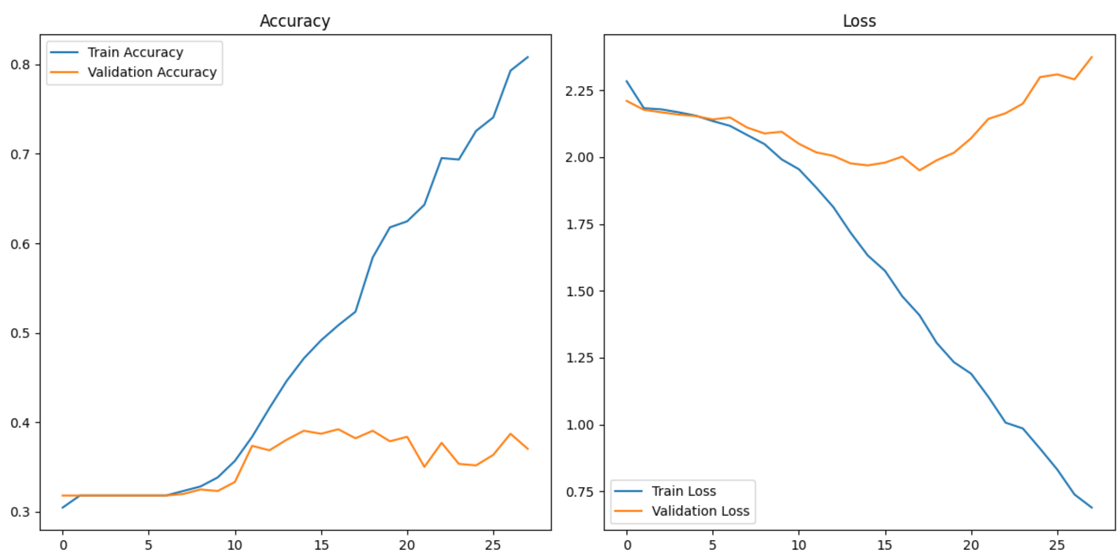
```
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(num_classes, activation='softmax')
])

model_two.compile(
    optimizer=tf.keras.optimizers.Adam(1e-4),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

model_two.summary()


callbacks_two = [
    tf.keras.callbacks.EarlyStopping(patience=10, restore_best_weights=True),
    tf.keras.callbacks.ModelCheckpoint('multiclass_birds.keras',␣
 ↪save_best_only=True)
]

history = model_two.fit(
    x_train, y_train,
    validation_data=(x_valid, y_valid),
    epochs=150,
    batch_size=32,
    callbacks=callbacks_two,
    verbose=1
)


plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Loss')
plt.legend()

plt.tight_layout()
plt.show()
```

```python
loss, acc = model_two.evaluate(x_test, y_test, verbose=0)
print(f"\nFinal Test Accuracy: {acc:.4f}")

probability_two  = model_two.predict(x_test)
prediction_two = np.argmax(probability_two , axis=1)
conf_mat = confusion_matrix(y_test, prediction_two)

ConfusionMatrixDisplay(conf_mat, display_labels=all_birds).plot(
    xticks_rotation=90, cmap=plt.cm.Blues)
plt.title('Multiclass Confusion Matrix')
plt.tight_layout()
plt.show()
```

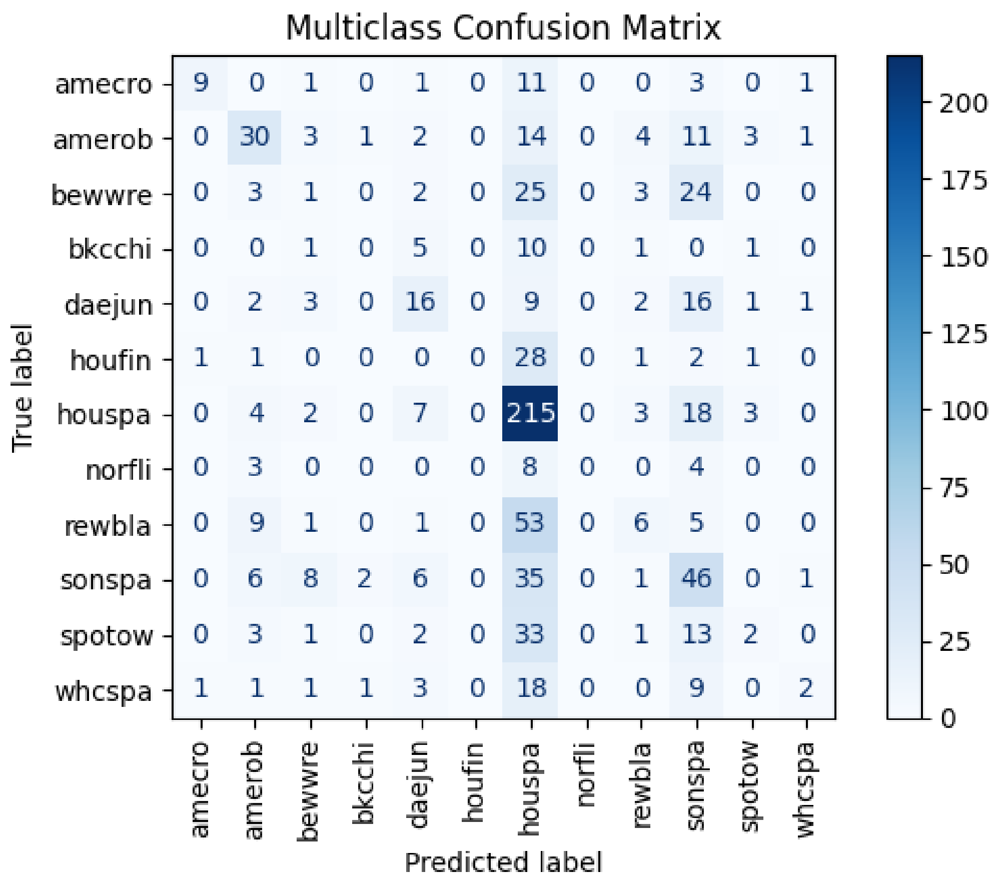
```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_18 (Conv2D)          (None, 126, 515, 32)      320

 max_pooling2d_18 (MaxPoolin  (None, 63, 257, 32)      0
 g2D)

 conv2d_19 (Conv2D)          (None, 61, 255, 64)       18496

 max_pooling2d_19 (MaxPoolin  (None, 30, 127, 64)      0
 g2D)

 conv2d_20 (Conv2D)          (None, 28, 125, 128)      73856

 max_pooling2d_20 (MaxPoolin  (None, 14, 62, 128)      0
 g2D)

 flatten_8 (Flatten)         (None, 111104)            0

 dense_16 (Dense)            (None, 128)               14221440

 dense_17 (Dense)            (None, 12)                1548

=================================================================
Total params: 14,315,660
Trainable params: 14,315,660
Non-trainable params: 0
_________________________________________________________________
Epoch 1/150
19/19 [==============================] - 33s 2s/step - loss: 2.2840 - accuracy:
0.3047 - val_loss: 2.2106 - val_accuracy: 0.3182
Epoch 2/150
19/19 [==============================] - 54s 3s/step - loss: 2.1832 - accuracy:
```

```
0.3182 - val_loss: 2.1773 - val_accuracy: 0.3182
Epoch 3/150
19/19 [==============================] - 60s 3s/step - loss: 2.1791 - accuracy:
0.3182 - val_loss: 2.1681 - val_accuracy: 0.3182
Epoch 4/150
19/19 [==============================] - 59s 3s/step - loss: 2.1681 - accuracy:
0.3182 - val_loss: 2.1595 - val_accuracy: 0.3182
Epoch 5/150
19/19 [==============================] - 58s 3s/step - loss: 2.1556 - accuracy:
0.3182 - val_loss: 2.1539 - val_accuracy: 0.3182
Epoch 6/150
19/19 [==============================] - 52s 3s/step - loss: 2.1351 - accuracy:
0.3182 - val_loss: 2.1412 - val_accuracy: 0.3182
Epoch 7/150
19/19 [==============================] - 46s 2s/step - loss: 2.1172 - accuracy:
0.3182 - val_loss: 2.1486 - val_accuracy: 0.3182
Epoch 8/150
19/19 [==============================] - 33s 2s/step - loss: 2.0828 - accuracy:
0.3232 - val_loss: 2.1106 - val_accuracy: 0.3199
Epoch 9/150
19/19 [==============================] - 33s 2s/step - loss: 2.0490 - accuracy:
0.3283 - val_loss: 2.0888 - val_accuracy: 0.3249
Epoch 10/150
19/19 [==============================] - 33s 2s/step - loss: 1.9917 - accuracy:
0.3384 - val_loss: 2.0950 - val_accuracy: 0.3232
Epoch 11/150
19/19 [==============================] - 41s 2s/step - loss: 1.9549 - accuracy:
0.3569 - val_loss: 2.0502 - val_accuracy: 0.3333
Epoch 12/150
19/19 [==============================] - 34s 2s/step - loss: 1.8866 - accuracy:
0.3838 - val_loss: 2.0182 - val_accuracy: 0.3737
Epoch 13/150
19/19 [==============================] - 33s 2s/step - loss: 1.8133 - accuracy:
0.4158 - val_loss: 2.0049 - val_accuracy: 0.3687
Epoch 14/150
19/19 [==============================] - 33s 2s/step - loss: 1.7176 - accuracy:
0.4461 - val_loss: 1.9768 - val_accuracy: 0.3805
Epoch 15/150
19/19 [==============================] - 36s 2s/step - loss: 1.6323 - accuracy:
0.4714 - val_loss: 1.9693 - val_accuracy: 0.3906
Epoch 16/150
19/19 [==============================] - 32s 2s/step - loss: 1.5748 - accuracy:
0.4916 - val_loss: 1.9797 - val_accuracy: 0.3872
Epoch 17/150
19/19 [==============================] - 32s 2s/step - loss: 1.4796 - accuracy:
0.5084 - val_loss: 2.0024 - val_accuracy: 0.3923
Epoch 18/150
19/19 [==============================] - 40s 2s/step - loss: 1.4090 - accuracy:
```

```
0.5236 - val_loss: 1.9506 - val_accuracy: 0.3822
Epoch 19/150
19/19 [==============================] - 49s 3s/step - loss: 1.3052 - accuracy:
0.5842 - val_loss: 1.9886 - val_accuracy: 0.3906
Epoch 20/150
19/19 [==============================] - 33s 2s/step - loss: 1.2330 - accuracy:
0.6178 - val_loss: 2.0165 - val_accuracy: 0.3788
Epoch 21/150
19/19 [==============================] - 29s 2s/step - loss: 1.1900 - accuracy:
0.6246 - val_loss: 2.0711 - val_accuracy: 0.3838
Epoch 22/150
19/19 [==============================] - 22s 1s/step - loss: 1.1034 - accuracy:
0.6431 - val_loss: 2.1433 - val_accuracy: 0.3502
Epoch 23/150
19/19 [==============================] - 22s 1s/step - loss: 1.0069 - accuracy:
0.6953 - val_loss: 2.1644 - val_accuracy: 0.3771
Epoch 24/150
19/19 [==============================] - 23s 1s/step - loss: 0.9852 - accuracy:
0.6936 - val_loss: 2.2003 - val_accuracy: 0.3535
Epoch 25/150
19/19 [==============================] - 23s 1s/step - loss: 0.9097 - accuracy:
0.7256 - val_loss: 2.2995 - val_accuracy: 0.3519
Epoch 26/150
19/19 [==============================] - 23s 1s/step - loss: 0.8314 - accuracy:
0.7407 - val_loss: 2.3097 - val_accuracy: 0.3636
Epoch 27/150
19/19 [==============================] - 23s 1s/step - loss: 0.7382 - accuracy:
0.7929 - val_loss: 2.2911 - val_accuracy: 0.3872
Epoch 28/150
19/19 [==============================] - 31s 2s/step - loss: 0.6892 - accuracy:
0.8081 - val_loss: 2.3744 - val_accuracy: 0.3704
```

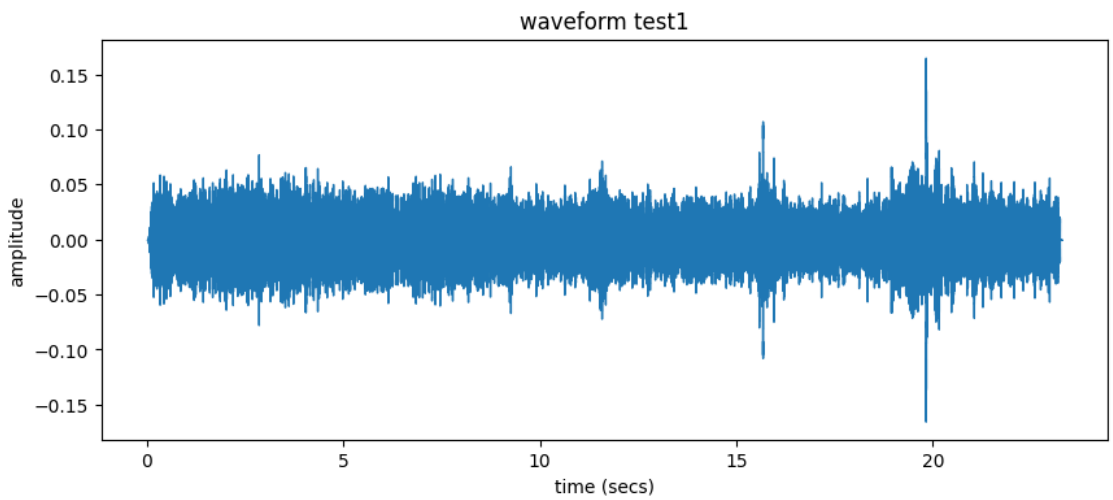


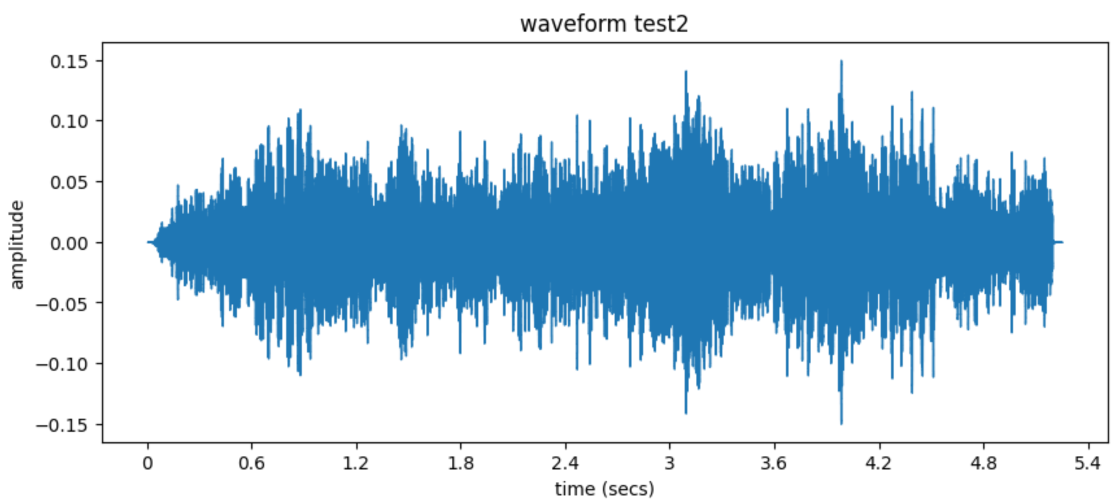
```
Final Test Accuracy: 0.4124
25/25 [==============================] - 9s 359ms/step
```

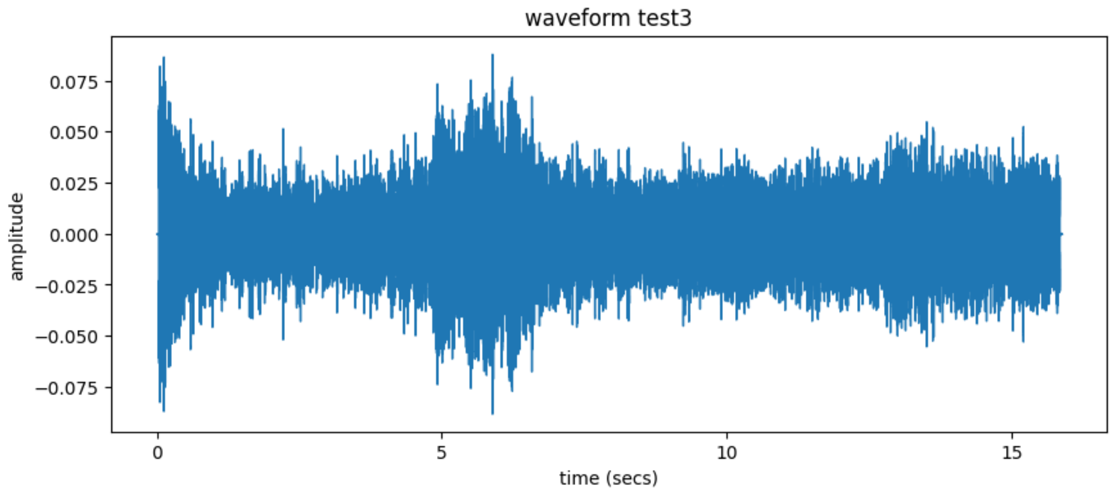



```
[ ]: # EXTERNAL TEST DATA
audios = {
    "test1": "/Users/alekh/Desktop/birds/test1.wav",
    "test2": "/Users/alekh/Desktop/birds/test2.wav",
    "test3": "/Users/alekh/Desktop/birds/test3.wav"
}



def waveforms(path, name):
    audio, sr = librosa.load(path, sr=22050)

    plt.figure(figsize=(10, 4))
    librosa.display.waveshow(audio, sr=sr)
```

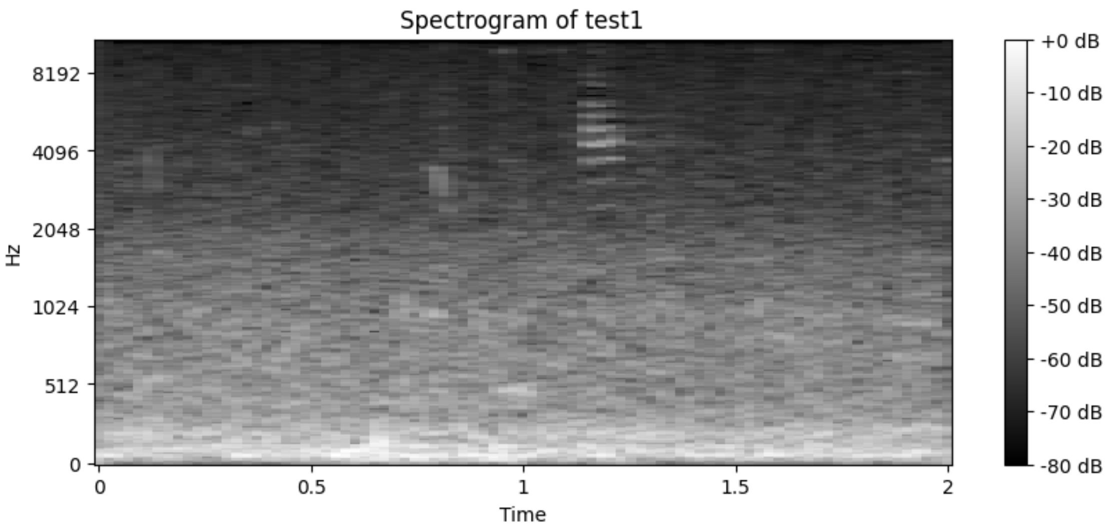
```python
    plt.title(f"waveform {name}")
    plt.xlabel("time (secs)")
    plt.ylabel("amplitude")
    plt.show()

    return audio, sr


audio_list = {}
for name, path in audios.items():
    audio, sr = waveforms(path, name)
    audio_list[name] = (audio, sr)
```



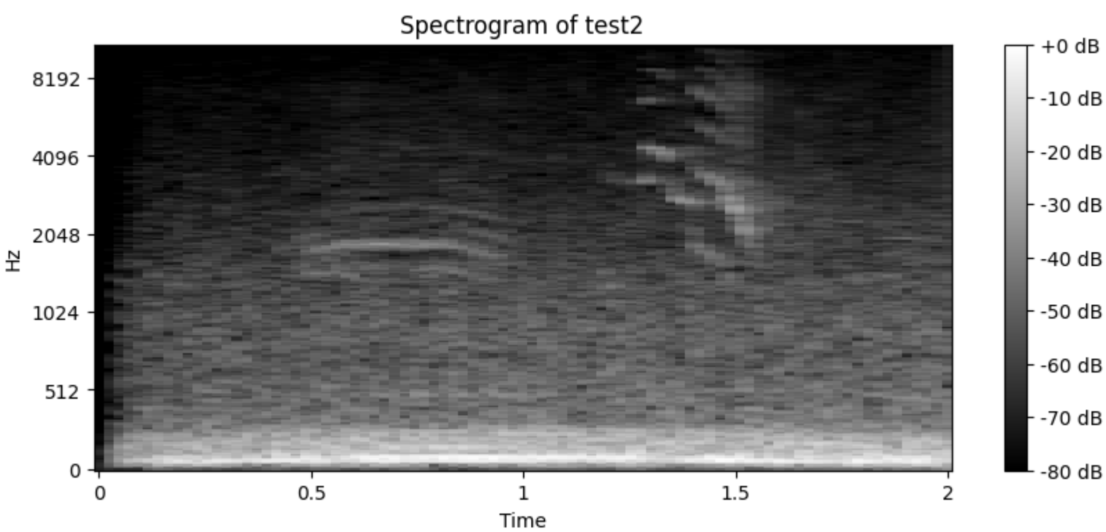




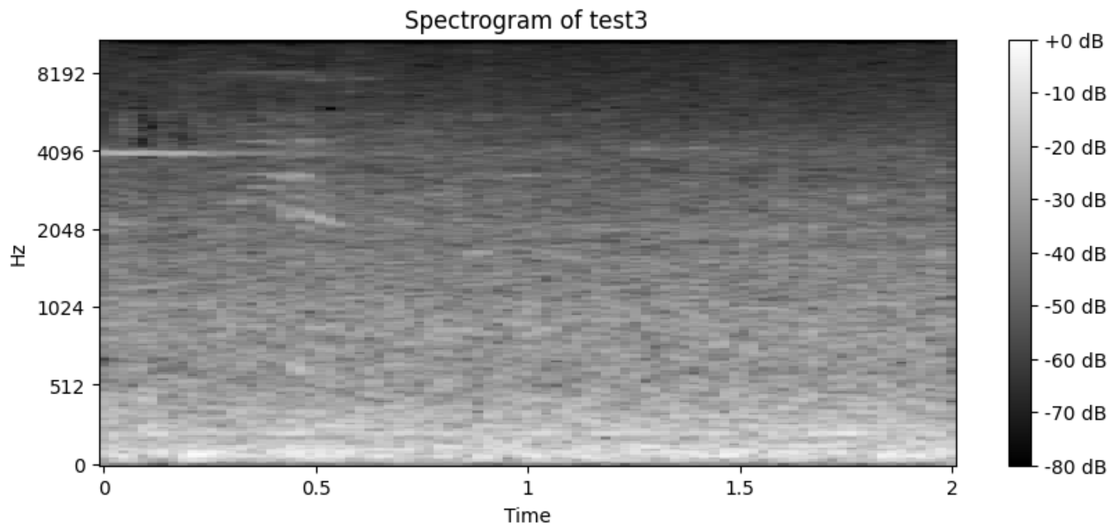
[13]:
```
def generate_spectrograms(audio, sr, start_sec, end_sec, name):
    start_sample = int(start_sec * sr)
    end_sample = int(end_sec * sr)
    window = audio[start_sample:end_sample]

    S = librosa.feature.melspectrogram(y=window, sr=sr, n_fft=2048,␣
 ↪hop_length=512, n_mels=256)
    S_DB = librosa.power_to_db(S, ref=np.max)

    plt.figure(figsize=(10, 4))
    librosa.display.specshow(S_DB, sr=sr, hop_length=512, x_axis='time',␣
 ↪y_axis='mel', cmap= 'gray')
    plt.colorbar(format='%+2.0f dB')
    plt.title(f'Spectrogram of {name}')
    plt.show()

time_windows = {
    "test1": (15, 17),
    "test2": (0, 2),
    "test3": (2, 4)
}

for name, (audio, sr) in audio_list.items():
    start_sec, end_sec = time_windows[name]
    generate_spectrograms(audio, sr, start_sec, end_sec, name)
```



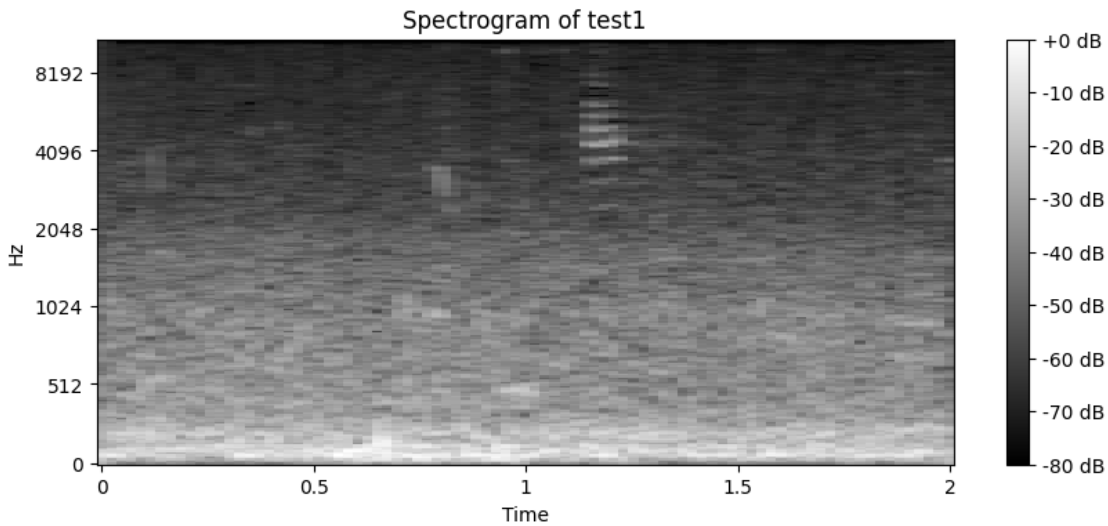
Spectrogram of test1
8192
4096
2048
1024
512
0
Hz
0
0.5
1
1.5
2
Time
+0 dB
-10 dB
-20 dB
-30 dB
-40 dB
-50 dB
-60 dB
-70 dB
-80 dB

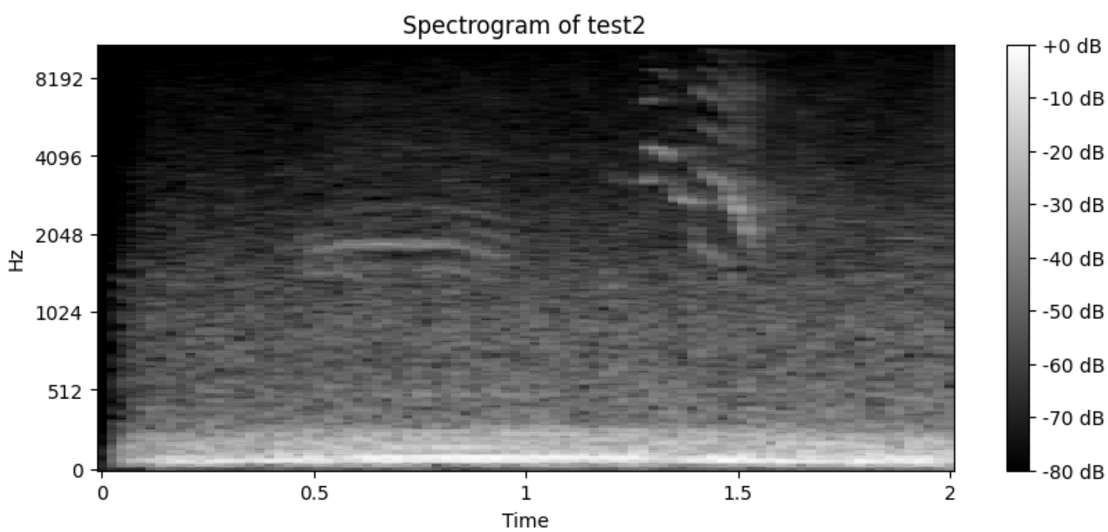


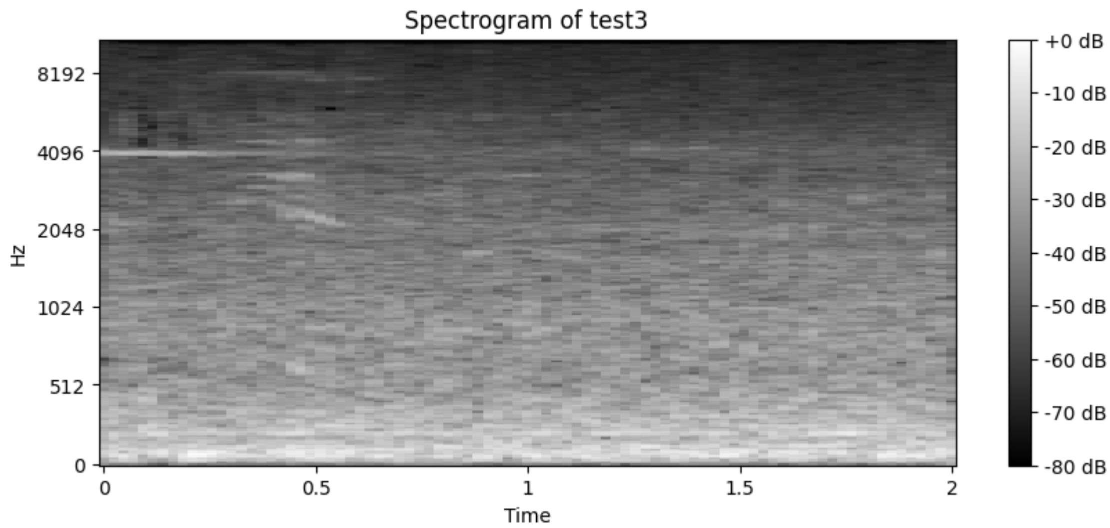
Spectrogram of test2
8192
4096
2048
1024
512
0
Hz
0
0.5
1
1.5
2
Time
+0 dB
-10 dB
-20 dB
-30 dB
-40 dB
-50 dB
-60 dB
-70 dB
-80 dB



[14]:
```python
def spectro_array(audio, sr, start_sec, end_sec, n_mels=256, n_fft=2048,␣
↪hop_length=512):
    start_sample = int(start_sec * sr)
    end_sample = int(end_sec * sr)
    window = audio[start_sample:end_sample]
    S = librosa.feature.melspectrogram(y=window, sr=sr, n_fft=n_fft,␣
↪hop_length=hop_length, n_mels=n_mels)
    S_DB = librosa.power_to_db(S, ref=np.max)
    return S_DB

with h5py.File('/Users/alekh/Desktop/birds/spectrograms.h5', 'w') as hf:
    for name, (audio, sr) in audio_list.items():
        start_sec, end_sec = time_windows[name]
        S_DB = spectro_array(audio, sr, start_sec, end_sec)
        hf.create_dataset(name, data=S_DB)
        plt.figure(figsize=(10, 4))
        librosa.display.specshow(S_DB, sr=sr, hop_length=512, x_axis='time',␣
↪y_axis='mel', cmap='grey')
        plt.colorbar(format='%+2.0f dB')
        plt.title(f'Spectrogram of {name}')
        plt.show()
```


Spectrogram of test1
Hz
8192
4096
2048
1024
512
0
0
0.5
1
1.5
2
Time
+0 dB
-10 dB
-20 dB
-30 dB
-40 dB
-50 dB
-60 dB
-70 dB
-80 dB


Spectrogram of test2
Hz
8192
4096
2048
1024
512
0
0
0.5
1
1.5
2
Time
+0 dB
-10 dB
-20 dB
-30 dB
-40 dB
-50 dB
-60 dB
-70 dB
-80 dB



[16]:
```
import os
import h5py
import numpy as np
import librosa
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense,␣
 ↪Dropout
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelEncoder

spectro_path = '/Users/alekh/Desktop/birds/bird_spectrograms.hdf5'
audios = {
    "test1": "/Users/alekh/Desktop/birds/test1.wav",
    "test2": "/Users/alekh/Desktop/birds/test2.wav",
    "test3": "/Users/alekh/Desktop/birds/test3.wav"
}
time_windows = {
    "test1": (15, 17),
    "test2": (0, 2),
    "test3": (2, 4)
}

spectrogram_list = []
labels = []
with h5py.File(spectro_path, 'r') as f:
    species = list(f.keys())
    for key in species:
        data = f[key][...]
```

```
        if data.ndim == 3:
            for i in range(data.shape[2]):
                spectrogram_list.append(data[:, :, i])
                labels.append(key)
        elif data.ndim == 2:
            spectrogram_list.append(data)
            labels.append(key)
        else:
            raise ValueError(f"ndim={data.ndim}")

spectrogram_list = np.array(spectrogram_list)
labels = np.array(labels)
label_encoder = LabelEncoder().fit(labels)
labels_encoded = label_encoder.transform(labels)
labels_onehot = to_categorical(labels_encoded)
mel_bins, time_frames = spectrogram_list.shape[1], spectrogram_list.shape[2]
spectrogram_list = spectrogram_list.reshape(-1, mel_bins, time_frames, 1).
 ↪astype(np.float32) / 255.0

model_three = Sequential([
    Conv2D(32, (3,3), activation='relu', input_shape=(mel_bins, time_frames,␣
 ↪1)),
    MaxPooling2D((2,2)),
    Conv2D(64, (3,3), activation='relu'),
    MaxPooling2D((2,2)),
    Conv2D(128, (3,3), activation='relu'),
    MaxPooling2D((2,2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(len(species), activation='softmax')
])
model_three.compile(optimizer='adam', loss='categorical_crossentropy',␣
 ↪metrics=['accuracy'])
model_three.fit(spectrogram_list, labels_onehot, epochs=5, batch_size=32)

test_data = []
for name, path in audios.items():
    audio, sr = librosa.load(path, sr=22050)
    start, end = time_windows[name]
    clip = audio[int(start*sr):int(end*sr)]
    S = librosa.feature.melspectrogram(y=clip, sr=sr, n_fft=2048,␣
 ↪hop_length=512, n_mels=mel_bins)
    S_db = librosa.power_to_db(S, ref=np.max)
    if S_db.shape[1] < time_frames:
        S_db = np.pad(S_db, ((0,0),(0, time_frames - S_db.shape[1])),␣
 ↪'constant')
```

```python
    else:
        S_db = S_db[:, :time_frames]
    test_data.append(S_db)

test_data = np.array(test_data).reshape(-1, mel_bins, time_frames, 1).astype(np.
 ↪float32) / 255.0
test_predictions = model_three.predict(test_data, verbose=0)
test_one = np.argmax(test_predictions, axis=1)
test_class = label_encoder.inverse_transform(test_one)
for i, pred in enumerate(test_class, 1):
    print(f"Test Spectrogram {i} predicted as {pred}")
```

```
Epoch 1/5
62/62 [==============================] - 73s 1s/step - loss: 2.2660 - accuracy:
0.2968
Epoch 2/5
62/62 [==============================] - 63s 1s/step - loss: 2.2259 - accuracy:
0.3019
Epoch 3/5
62/62 [==============================] - 65s 1s/step - loss: 2.2022 - accuracy:
0.3125
Epoch 4/5
62/62 [==============================] - 65s 1s/step - loss: 2.1840 - accuracy:
0.3150
Epoch 5/5
62/62 [==============================] - 66s 1s/step - loss: 2.1032 - accuracy:
0.3241
Test Spectrogram 1 predicted as sonspa
Test Spectrogram 2 predicted as sonspa
Test Spectrogram 3 predicted as sonspa
```