task 3

May 12, 2025

[1]:

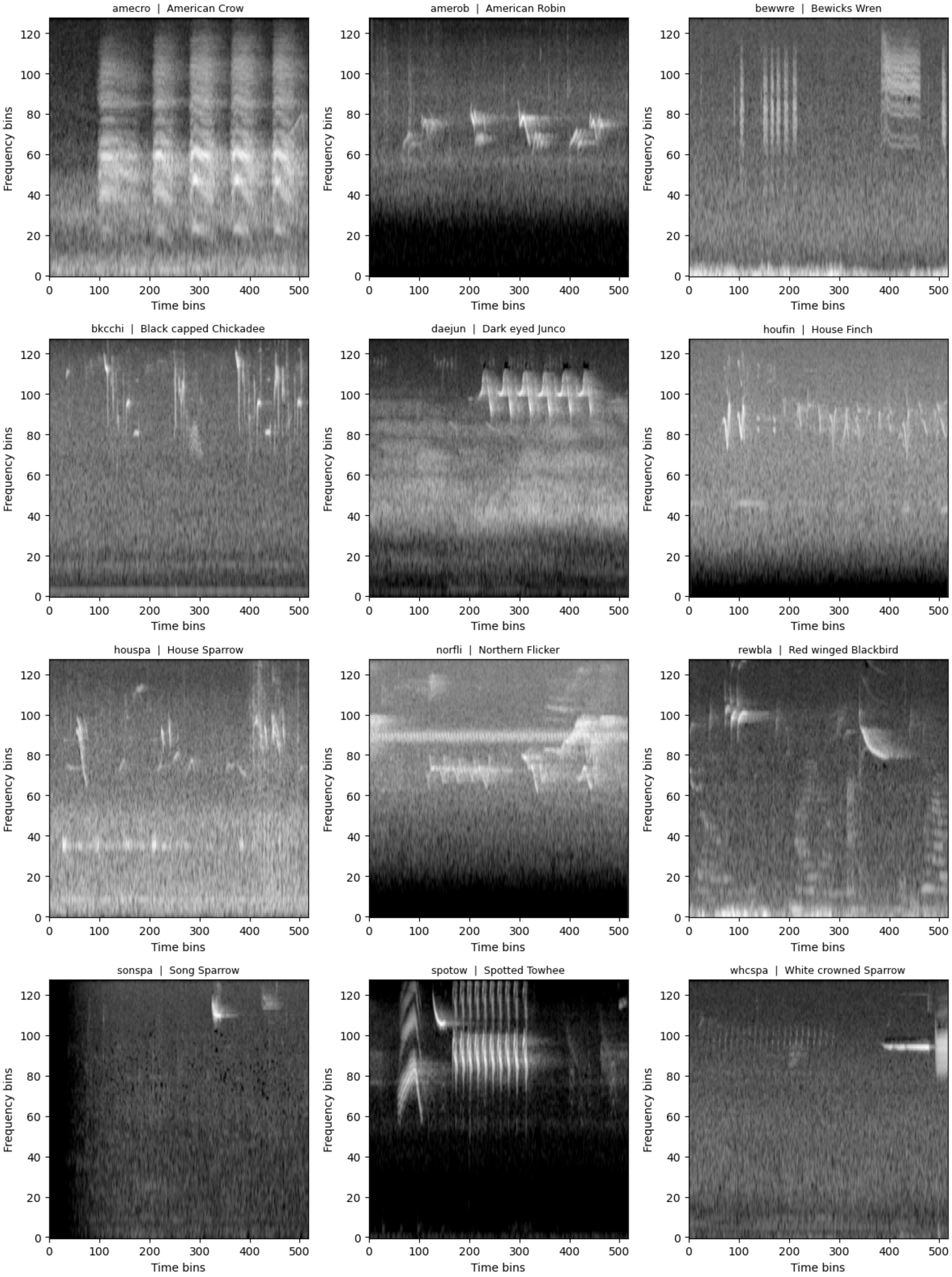
```
# Necessary libraries
import h5py
import numpy as np
import matplotlib.pyplot as plt
import math
from collections import Counter
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.utils import shuffle
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics 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
```

[2]:

```
# loading the file
data = h5py.File('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)
```



[3]:
```
import os
import h5py
```

```python
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
from tensorflow.keras.models import load_model

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')
    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 [==============================] - 97s 2s/step - loss: 2.2666 - accuracy:
0.3029
Epoch 2/5
62/62 [==============================] - 88s 1s/step - loss: 2.2182 - accuracy:
0.3165
Epoch 3/5
```

```
62/62 [==============================] - 95s 2s/step - loss: 2.1717 - accuracy:
0.3150
Epoch 4/5
62/62 [==============================] - 89s 1s/step - loss: 2.1132 - accuracy:
0.3246
Epoch 5/5
62/62 [==============================] - 89s 1s/step - loss: 2.0288 - accuracy:
0.3523
Test Spectrogram 1 predicted as sonspa
Test Spectrogram 2 predicted as sonspa
Test Spectrogram 3 predicted as sonspa
```

[4]:

```python
import numpy as np
import pandas as pd
import librosa

bird_codes = label_encoder.classes_
bird_names = [names[c] for c in bird_codes]

hop = int(time_frames * 0.5)
rows = []

for clip_name, path in audios.items():
    y, sr = librosa.load(path, sr=22050)
    s, e = time_windows[clip_name]
    seg = y[int(s*sr):int(e*sr)]

    S = librosa.feature.melspectrogram(
        y=seg, sr=sr,
        n_fft=2048, hop_length=512, n_mels=mel_bins
    )
    S_db = librosa.power_to_db(S, ref=np.max)
    S_norm = np.clip((S_db + 80) / 80, 0, 1)

    n_cols = S_norm.shape[1]
    if n_cols < time_frames:
        pad = time_frames - n_cols
        S_norm = np.pad(S_norm, ((0,0),(0,pad)), mode='constant')
        n_cols = time_frames

    starts = range(0, n_cols - time_frames + 1, hop)
    patches = np.stack([S_norm[:, i:i+time_frames] for i in starts])[..., None]

    P = model_three.predict(patches, verbose=0)
    average = P.mean(axis=0)
    top_three = np.argsort(average)[-3:][::-1]
    rows.append({
        "clip":         clip_name,
        "top1_species": bird_names[top_three[0]],
        "top1_probability":    float(average[top_three[0]]),
        "top2_species": bird_names[top_three[1]],
        "top2_probabaility":    float(average[top_three[1]]),
        "top3_species": bird_names[top_three[2]],
        "top3_probabality":    float(average[top_three[2]]),
    })

df_top3 = pd.DataFrame(rows)
print("The top 3 external clip predictions")
display(df_top3)
```

```
The top 3 external clip predictions

    clip   top1_species  top1_probability    top2_species  top2_probabaility  \
0  test1  House Sparrow          0.805277  American Robin           0.126353
1  test2  House Sparrow          0.643778  American Robin           0.204123
2  test3  House Sparrow          0.853333  American Robin           0.108637

   top3_species  top3_probabality
0  Song Sparrow          0.023167
1  Song Sparrow          0.053612
2  Song Sparrow          0.013637
```