

```

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import librosa
from glob import glob
import IPython.display as ipd
import tensorflow as tf
from PIL import Image

df =
pd.read_csv("C:/Users/raksh/Desktop/bird/bird_songs_metadata.csv")
df.head(3)

```

| | id | genus | species | subspecies | name \ |
|---|--------|------------|----------|------------|---------------|
| 0 | 557838 | Thryomanes | bewickii | NaN | Bewick's Wren |
| 1 | 557838 | Thryomanes | bewickii | NaN | Bewick's Wren |
| 2 | 557838 | Thryomanes | bewickii | NaN | Bewick's Wren |

| | recordist | country \ |
|---|------------------------|---------------|
| 0 | Whitney Neufeld-Kaiser | United States |
| 1 | Whitney Neufeld-Kaiser | United States |
| 2 | Whitney Neufeld-Kaiser | United States |

| | location | latitude | longitude |
|------------|---|----------|-----------|
| altitude \ | | | |
| 0 | Arlington, Snohomish County, Washington | 48.0708 | -122.1006 |
| 100 | | | |
| 1 | Arlington, Snohomish County, Washington | 48.0708 | -122.1006 |
| 100 | | | |
| 2 | Arlington, Snohomish County, Washington | 48.0708 | -122.1006 |
| 100 | | | |

| | sound_type | source_url | time |
|--------|----------------------------|-----------------------------|------------------|
| date \ | | | |
| 0 | adult, sex uncertain, song | //www.xeno-canto.org/557838 | 11:51 14-03-2020 |
| 1 | adult, sex uncertain, song | //www.xeno-canto.org/557838 | 11:51 14-03-2020 |
| 2 | adult, sex uncertain, song | //www.xeno-canto.org/557838 | 11:51 14-03-2020 |

| | remarks | filename |
|---|---|--------------|
| 0 | Recorded with Voice Record Pro on iPhone7, nor... | 557838-0.wav |
| 1 | Recorded with Voice Record Pro on iPhone7, nor... | 557838-1.wav |
| 2 | Recorded with Voice Record Pro on iPhone7, nor... | 557838-4.wav |

```

class_names = df["name"].unique()
print(class_names)

```

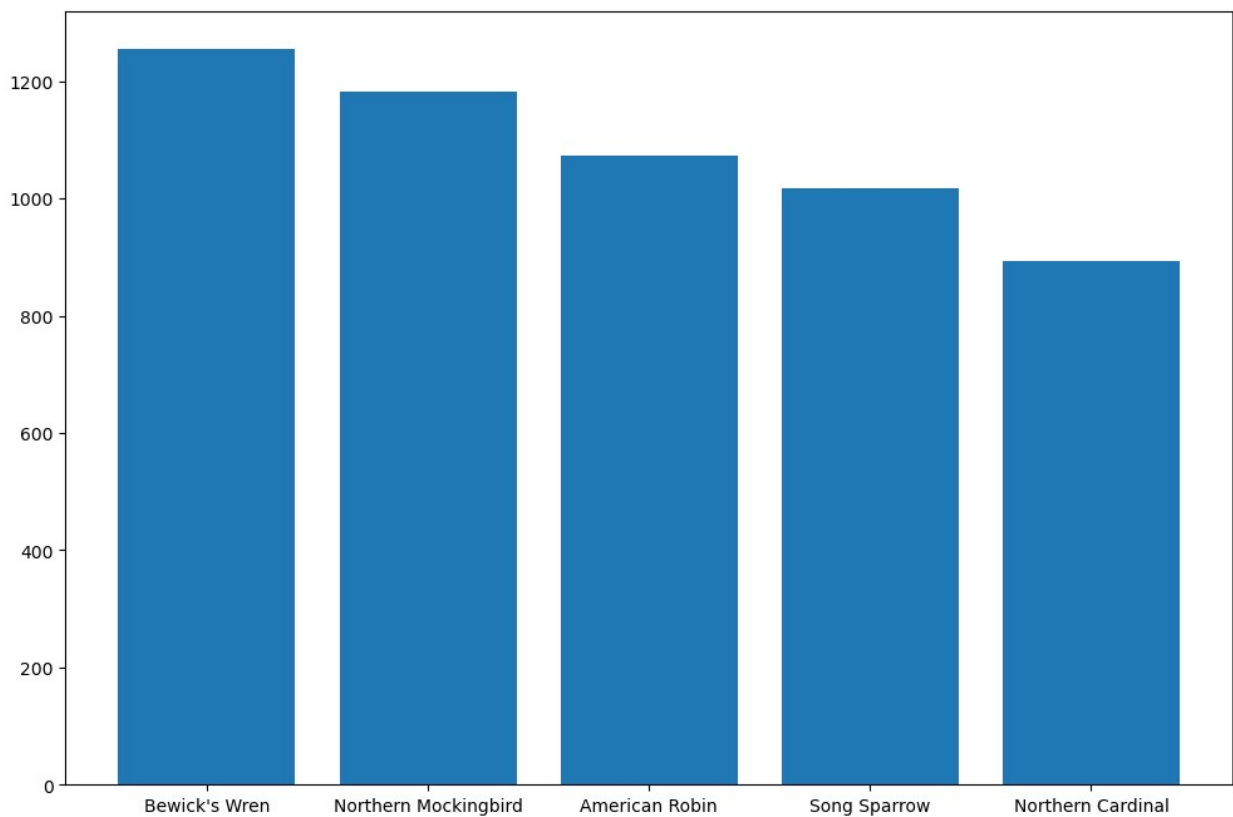
```
["Bewick's Wren" 'Northern Mockingbird' 'American Robin' 'Song Sparrow'  
 'Northern Cardinal']
```

```
df["name"].value_counts()
```

```
name  
Song Sparrow      1256  
Northern Mockingbird  1182  
Northern Cardinal  1074  
American Robin    1017  
Bewick's Wren      893  
Name: count, dtype: int64
```

```
fig, ax = plt.subplots(figsize=(12, 8))  
ax.bar(df["name"].unique(), df["name"].value_counts())
```

```
<BarContainer object of 5 artists>
```



```
path_to_wav = "C:/Users/raksh/Desktop/bird/wavfiles (1)/"
```

```
datafiles = glob(path_to_wav + "*")
```

```
def generate_spectrogram(file_audio, identifier):  
    audio_data, sample_rate = librosa.load(path_to_wav + file_audio)
```

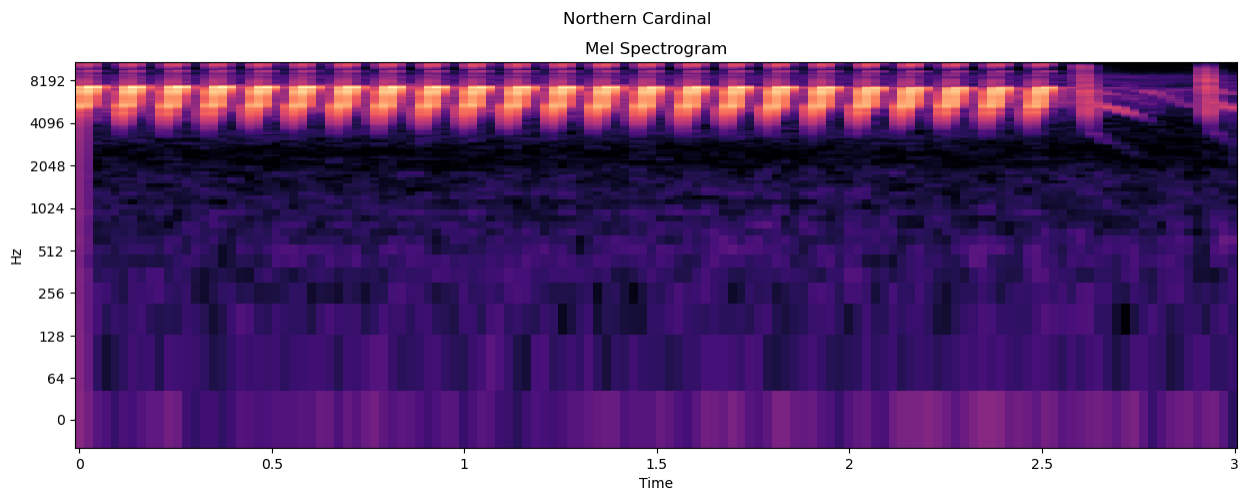
```

    spec_mel = librosa.feature.melspectrogram(y=audio_data,
sr=sample_rate)
    spec_mel = librosa.power_to_db(spec_mel, ref=np.max)
    figure, axis = plt.subplots(figsize=(15, 5))
    axis.set_title("Mel Spectrogram")
    plt.suptitle(identifier)
    librosa.display.specshow(spec_mel, x_axis='time', y_axis='log',
ax=axis)
    return ipd.Audio(path_to_wav + file_audio, rate=sample_rate)

i = np.random.randint(0, df.shape[0])
generate_spectrogram(df.loc[i, "filename"], df.loc[i, "name"])

<IPython.lib.display.Audio object>

```

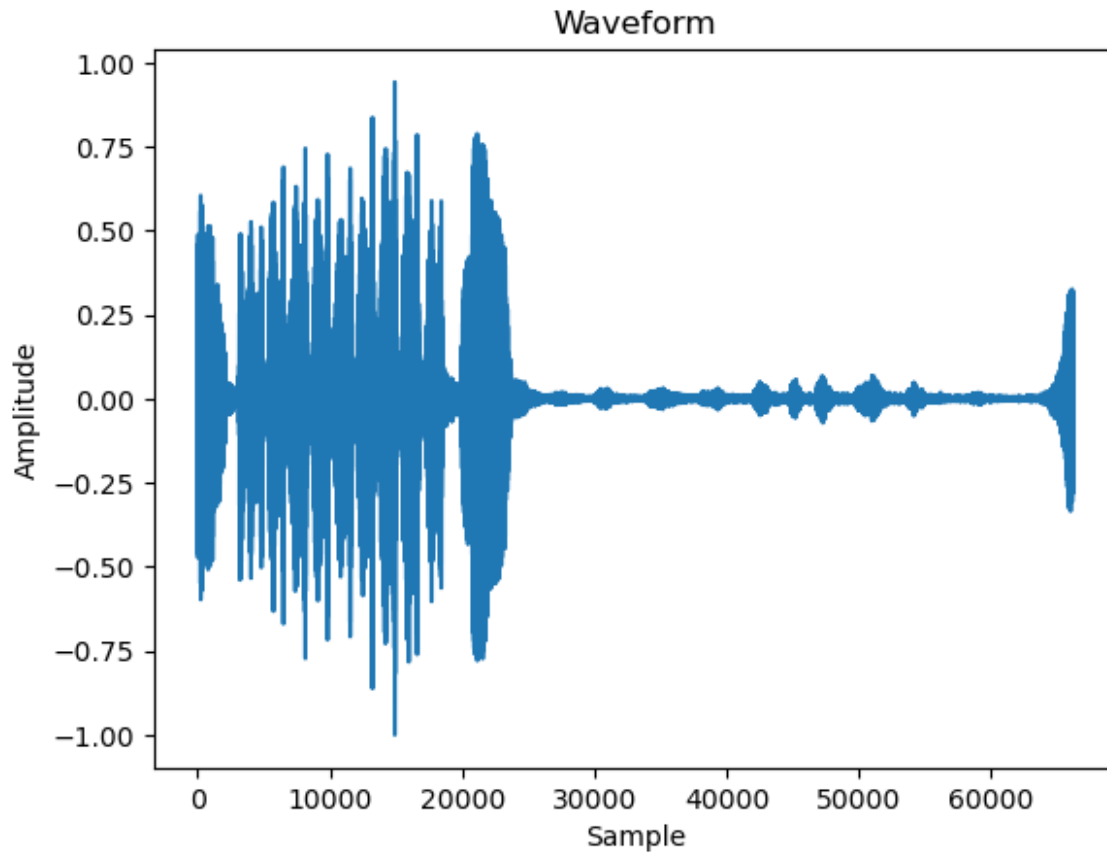


```

filename = "C:/Users/raksh/Desktop/bird/wavfiles (1)/12578-8.wav"
audio_data, sample_rate = librosa.load(filename)

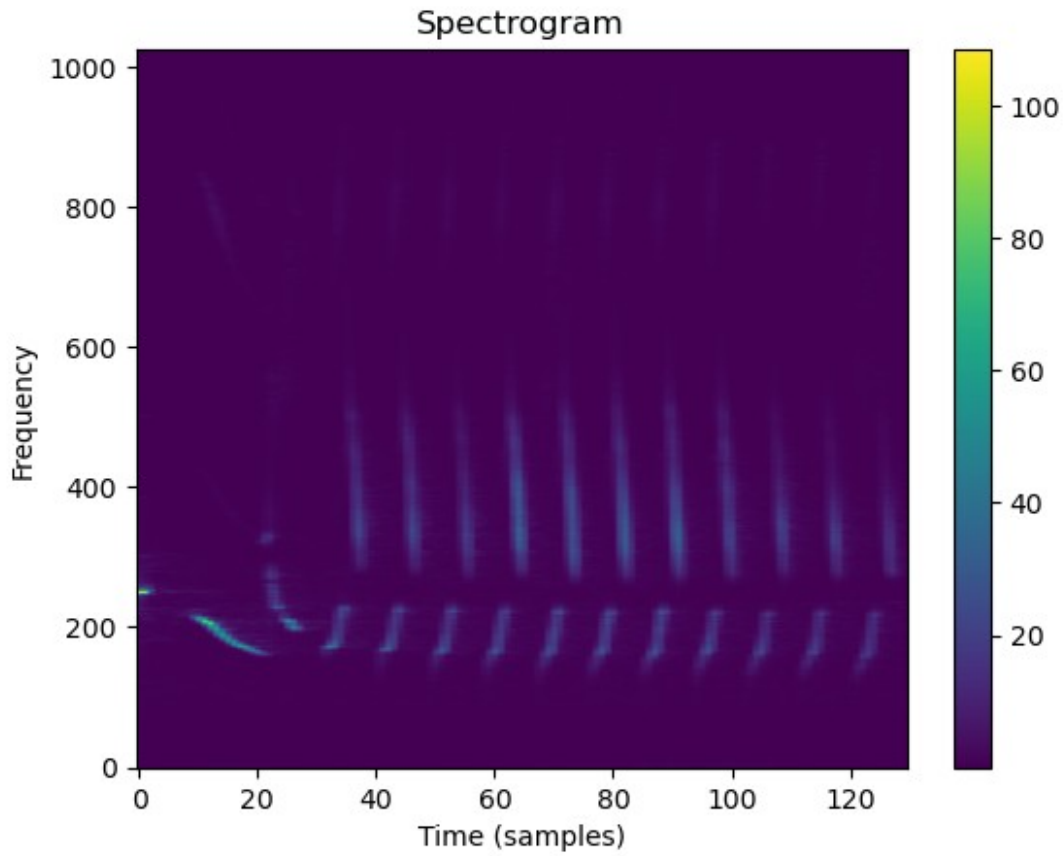
plt.plot(audio_data)
plt.title("Waveform")
plt.xlabel("Sample")
plt.ylabel("Amplitude")
plt.show()

```



```
filename = "C:/Users/raksh/Desktop/bird/wavfiles (1)/11713-6.wav"
audio_data, sample_rate = librosa.load(filename)
spectrogram = librosa.stft(audio_data)
spectrogram = np.abs(spectrogram)

plt.imshow(spectrogram, origin='lower', aspect='auto')
plt.title("Spectrogram")
plt.xlabel("Time (samples)")
plt.ylabel("Frequency")
plt.colorbar()
plt.show()
```



```
def process_audio(audio_file):
    audio_data, sample_rate = librosa.load(audio_file, duration=10)
    mel_spec = librosa.feature.melspectrogram(y=audio_data,
sr=sample_rate)
    mel_spec = librosa.power_to_db(mel_spec, ref=np.max)
    return mel_spec

filename = "C:/Users/raksh/Desktop/bird/wavfiles (1)/11713-6.wav"
print(len(process_audio(filename)))
print(len(process_audio(filename)[0]))

128
130

df_train = pd.DataFrame({"name": df["name"], "audiopath": path_to_wav
+ df["filename"]})

# Assuming `process_audio` is a function that generates mel
spectrograms
df_train["mel_spec"] = df_train["audiopath"].apply(lambda x:
process_audio(x))

# Using factorize to encode class labels
df_train["class"] = df_train["name"].factorize()[0]
```

```

from sklearn.utils import shuffle

df_train = shuffle(df_train)
df_train.shape

(5422, 4)

(train_x, train_y) = df_train["mel_spec"][:5000].values,
df_train["class"][:5000].values
(test_x, test_y) = df_train["mel_spec"][5000:].values,
df_train["class"][5000:].values

from keras.utils import to_categorical

test_y = to_categorical(test_y, num_classes=len(class_names))
train_y = to_categorical(train_y, num_classes=len(class_names))

train_x = np.stack(train_x[:])
test_x = np.stack(test_x[:])

train_x = tf.keras.utils.normalize(train_x)
test_x = tf.keras.utils.normalize(test_x)

train_dataset = tf.data.Dataset.from_tensor_slices((train_x, train_y))
test_dataset = tf.data.Dataset.from_tensor_slices((test_x, test_y))

train_dataset = train_dataset.batch(10)
test_dataset = test_dataset.batch(10)
train_dataset = train_dataset.prefetch(tf.data.AUTOTUNE)
test_dataset = test_dataset.prefetch(tf.data.AUTOTUNE)

import keras
from keras.layers import Conv2D, MaxPooling2D, BatchNormalization,
Flatten, Dense, Reshape, InputLayer, Dropout
from keras.models import Sequential

model = keras.models.Sequential()

model.add(InputLayer(input_shape=(128, 130)))
model.add(Reshape((128, 130, 1)))
model.add(Conv2D(64, (8, 8), input_shape=(128, 130),
activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(16, (2, 2), activation='relu'))
model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(128, activation='relu'))
model.add(Dense(5, activation='softmax'))

C:\Users\raksh\Documents\program\Lib\site-packages\keras\src\layers\
core\input_layer.py:25: UserWarning: Argument `input_shape` is

```

deprecated. Use `shape` instead.

```
warnings.warn(  
C:\Users\raksh\Documents\program\Lib\site-packages\keras\src\layers\  
convolutional\base_conv.py:107: UserWarning: Do not pass an  
`input_shape`/`input_dim` argument to a layer. When using Sequential  
models, prefer using an `Input(shape)` object as the first layer in  
the model instead.  
super().__init__(activity_regularizer=activity_regularizer,  
**kwargs)
```

```
model.summary()
```

Model: "sequential"

| Layer (type) Param # | Output Shape |
|---|----------------------|
| 0 reshape (Reshape) | (None, 128, 130, 1) |
| 4,160 conv2d (Conv2D) | (None, 121, 123, 64) |
| 256 batch_normalization (BatchNormalization) | (None, 121, 123, 64) |
| 0 max_pooling2d (MaxPooling2D) | (None, 60, 61, 64) |
| 4,112 conv2d_1 (Conv2D) | (None, 59, 60, 16) |
| 0 flatten (Flatten) | (None, 56640) |
| 0 dropout (Dropout) | (None, 56640) |

| | |
|-----------------|-------------|
| dense (Dense) | (None, 128) |
| 7,250,048 | |
| dense_1 (Dense) | (None, 5) |
| 645 | |

Total params: 7,259,221 (27.69 MB)

Trainable params: 7,259,093 (27.69 MB)

Non-trainable params: 128 (512.00 B)

```
model.compile('adam', loss='categorical_crossentropy',
metrics=[tf.keras.metrics.Recall(),tf.keras.metrics.Precision(),
'accuracy'])
```

```
hist = model.fit(train_dataset, epochs=10,
validation_data=test_dataset)
```

Epoch 1/10

```
500/500 _____ 168s 321ms/step - accuracy: 0.5586 -
loss: 1.0562 - precision: 0.6357 - recall: 0.4041 - val_accuracy:
0.2488 - val_loss: 1.6515 - val_precision: 0.2833 - val_recall: 0.2417
```

Epoch 2/10

```
500/500 _____ 152s 303ms/step - accuracy: 0.8148 -
loss: 0.5025 - precision: 0.8446 - recall: 0.7641 - val_accuracy:
0.5190 - val_loss: 2.2613 - val_precision: 0.5293 - val_recall: 0.5142
```

Epoch 3/10

```
500/500 _____ 150s 299ms/step - accuracy: 0.9210 -
loss: 0.2269 - precision: 0.9316 - recall: 0.9027 - val_accuracy:
0.4621 - val_loss: 6.5809 - val_precision: 0.4697 - val_recall: 0.4597
```

Epoch 4/10

```
500/500 _____ 149s 298ms/step - accuracy: 0.9549 -
loss: 0.1328 - precision: 0.9580 - recall: 0.9518 - val_accuracy:
0.7796 - val_loss: 0.7927 - val_precision: 0.7956 - val_recall: 0.7749
```

Epoch 5/10

```
500/500 _____ 148s 297ms/step - accuracy: 0.9652 -
loss: 0.1058 - precision: 0.9660 - recall: 0.9635 - val_accuracy:
0.6967 - val_loss: 1.6839 - val_precision: 0.6986 - val_recall: 0.6919
```

Epoch 6/10

```
500/500 _____ 149s 298ms/step - accuracy: 0.9831 -
loss: 0.0505 - precision: 0.9838 - recall: 0.9831 - val_accuracy:
0.4431 - val_loss: 6.9870 - val_precision: 0.4439 - val_recall: 0.4408
```

Epoch 7/10

```
500/500 _____ 148s 297ms/step - accuracy: 0.9774 -
loss: 0.0677 - precision: 0.9775 - recall: 0.9773 - val_accuracy:
0.7227 - val_loss: 1.3816 - val_precision: 0.7314 - val_recall: 0.7227
```

Epoch 8/10


```

500/500 ————— 147s 294ms/step - accuracy: 0.9807 -
loss: 0.0626 - precision: 0.9818 - recall: 0.9807 - val_accuracy:
0.4502 - val_loss: 11.9156 - val_precision: 0.4487 - val_recall:
0.4455
Epoch 9/10
500/500 ————— 150s 300ms/step - accuracy: 0.9856 -
loss: 0.0454 - precision: 0.9861 - recall: 0.9854 - val_accuracy:
0.6777 - val_loss: 2.2371 - val_precision: 0.6827 - val_recall: 0.6730
Epoch 10/10
500/500 ————— 152s 303ms/step - accuracy: 0.9899 -
loss: 0.0356 - precision: 0.9899 - recall: 0.9897 - val_accuracy:
0.3531 - val_loss: 14.6638 - val_precision: 0.3515 - val_recall:
0.3507

```

Save the model to a file

```
model.save('bird_sound_classification_model.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 considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')` or
`keras.saving.save_model(model, 'my_model.keras')`.

```
model.evaluate(test_dataset)
```

```

43/43 ————— 2s 40ms/step - accuracy: 0.3472 - loss:
14.0894 - precision: 0.3458 - recall: 0.3450

```

```

[14.663841247558594,
 0.3507108986377716,
 0.35154393315315247,
 0.3530805706977844]

```

```

pred_y = model.predict(test_x)
true_y = test_y.argmax(axis=1, keepdims=True)

```

```
14/14 ————— 2s 129ms/step
```

```

print(pred_y)
print(true_y)

```

```

[[6.9032223e-14 1.1925868e-15 5.9674535e-14 1.0000000e+00 5.9552114e-
09]
 [3.9875583e-22 5.1240882e-16 2.2828763e-14 1.0000000e+00 7.4895344e-
14]
 [8.2445224e-18 5.2689821e-18 1.3873992e-11 1.0000000e+00 1.5736290e-
15]
 ...
 [4.5625557e-16 6.8809799e-14 7.0644930e-09 1.0000000e+00 2.5091005e-
14]
 [3.0444321e-15 3.5197277e-15 1.2094655e-06 9.9999881e-01 1.5353808e-

```

```
15]
[2.2554762e-17 3.8286978e-09 2.0808821e-15 1.0000000e+00 3.3472818e-
16]]
[[2]
[4]
[2]
[0]
[2]
[2]
[3]
[1]
[2]
[2]
[1]
[4]
[3]
[4]
[2]
[4]
[3]
[1]
[3]
[0]
[4]
[2]
[3]
[1]
[2]
[4]
[2]
[1]
[4]
[2]
[0]
[4]
[4]
[0]
[0]
[2]
[4]
[1]
[3]
[1]
[3]
[0]
[3]
[1]
[4]
[3]
```

[2]
[1]
[4]
[1]
[1]
[1]
[2]
[0]
[3]
[1]
[2]
[2]
[3]
[3]
[4]
[2]
[4]
[1]
[2]
[4]
[3]
[4]
[3]
[1]
[1]
[0]
[3]
[4]
[2]
[2]
[4]
[3]
[4]
[1]
[3]
[1]
[1]
[3]
[1]
[0]
[3]
[2]
[4]
[1]
[2]
[2]
[3]
[0]
[4]

[2]
[3]
[2]
[0]
[1]
[1]
[2]
[2]
[0]
[3]
[0]
[3]
[0]
[4]
[4]
[2]
[3]
[4]
[1]
[1]
[3]
[2]
[3]
[3]
[3]
[4]
[0]
[1]
[4]
[2]
[3]
[4]
[0]
[1]
[3]
[4]
[0]
[3]
[1]
[2]
[3]
[2]
[1]
[3]
[4]
[4]
[0]
[4]
[4]

[4]
[4]
[0]
[3]
[4]
[3]
[0]
[4]
[2]
[0]
[3]
[0]
[3]
[3]
[3]
[1]
[0]
[4]
[4]
[4]
[1]
[2]
[3]
[1]
[3]
[3]
[1]
[3]
[3]
[2]
[2]
[2]
[4]
[2]
[3]
[1]
[3]
[2]
[0]
[0]
[4]
[1]
[3]
[3]
[2]
[3]
[3]
[0]
[4]
[3]

```
[1]
[0]
[4]
[4]
[1]
[1]
[3]
[2]
[2]
[3]
[4]
[0]
[3]
[2]
[4]
[4]
[3]
[3]
[2]
[3]
[0]
[3]
[0]
[4]
[3]
[3]
[2]
[3]
[1]
[1]
[1]
[1]
[3]
[2]
[4]
[2]
[1]
[1]
[2]
[4]
[1]
[4]
[3]
[0]
[1]
[0]
[0]
[2]
[4]
```

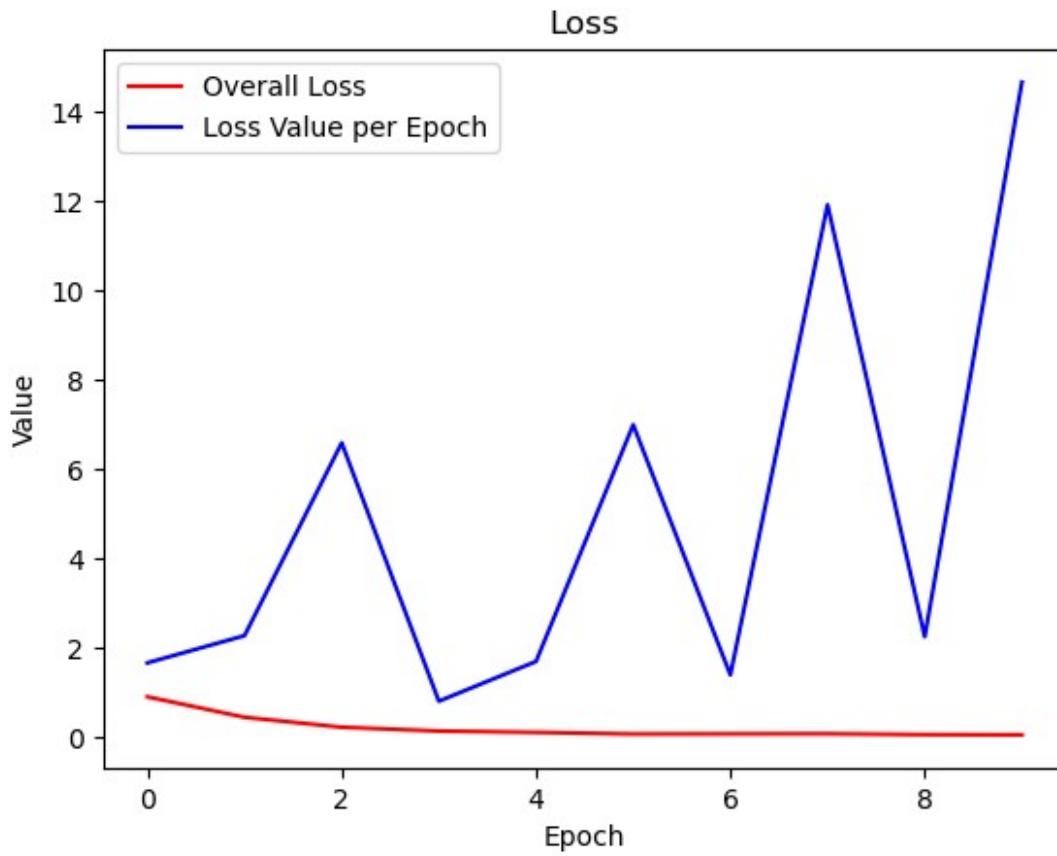
[3]
[0]
[1]
[0]
[1]
[1]
[0]
[4]
[3]
[0]
[1]
[4]
[3]
[3]
[4]
[3]
[1]
[0]
[2]
[3]
[3]
[3]
[2]
[4]
[0]
[4]
[4]
[1]
[4]
[4]
[0]
[4]
[3]
[0]
[0]
[2]
[1]
[1]
[1]
[2]
[1]
[1]
[2]
[4]
[4]
[1]
[3]
[2]
[3]

[4]
[2]
[4]
[0]
[0]
[1]
[4]
[3]
[3]
[3]
[0]
[3]
[3]
[4]
[4]
[2]
[2]
[3]
[3]
[1]
[4]
[1]
[1]
[0]
[0]
[2]
[3]
[0]
[4]
[1]
[3]
[1]
[0]
[4]
[4]
[4]
[3]
[1]
[4]
[2]
[3]
[4]
[0]
[3]
[4]
[4]
[1]
[2]
[2]

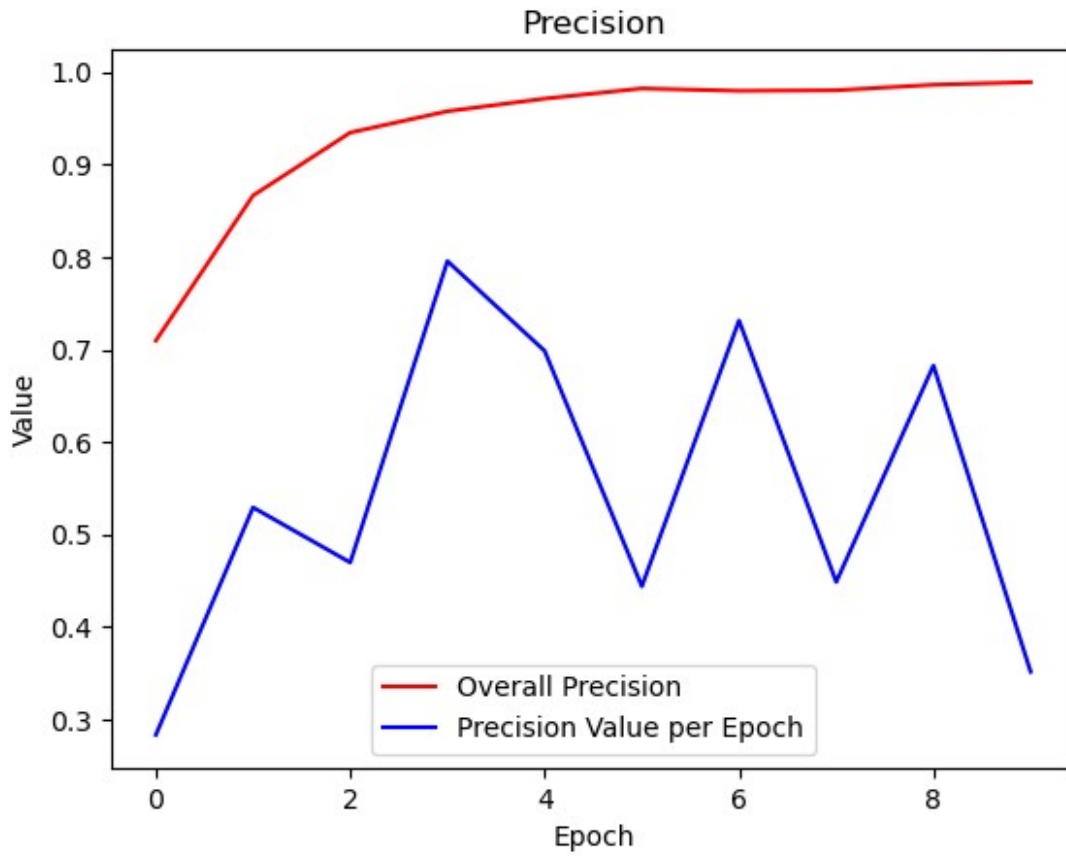
[1]
[3]
[1]
[1]
[1]
[3]
[4]
[0]
[2]
[1]
[4]
[0]
[1]
[2]
[3]
[3]
[4]
[3]
[0]
[4]
[0]
[4]
[2]
[1]
[2]
[2]
[3]
[1]
[0]
[1]
[4]
[2]
[4]
[4]
[4]
[0]
[0]
[0]
[2]
[4]
[4]
[2]
[4]
[2]
[2]
[0]
[4]
[0]
[2]
[0]

```
[2]  
[4]  
[0]  
[2]  
[2]  
[0]  
[4]  
[1]  
[1]  
[3]  
[3]  
[2]  
[3]  
[4]  
[4]  
[0]  
[4]  
[0]  
[2]  
[3]  
[1]  
[2]  
[0]  
[1]  
[3]  
[3]  
[3]  
[1]  
[1]  
[1]  
[3]  
[4]  
[3]]
```

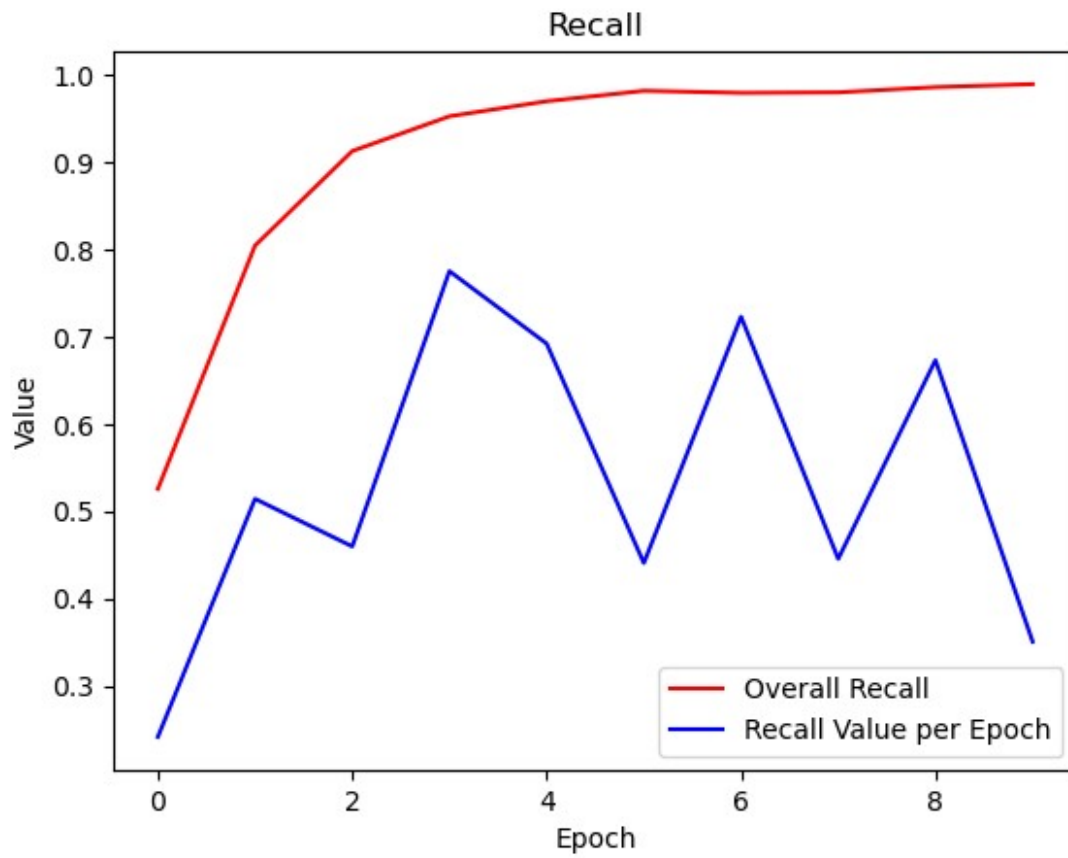
```
plt.title('Loss')  
plt.plot(hist.history['loss'], 'r', label='Overall Loss')  
plt.plot(hist.history['val_loss'], 'b', label='Loss Value per Epoch')  
plt.xlabel('Epoch')  
plt.ylabel('Value')  
plt.legend()  
plt.show()
```



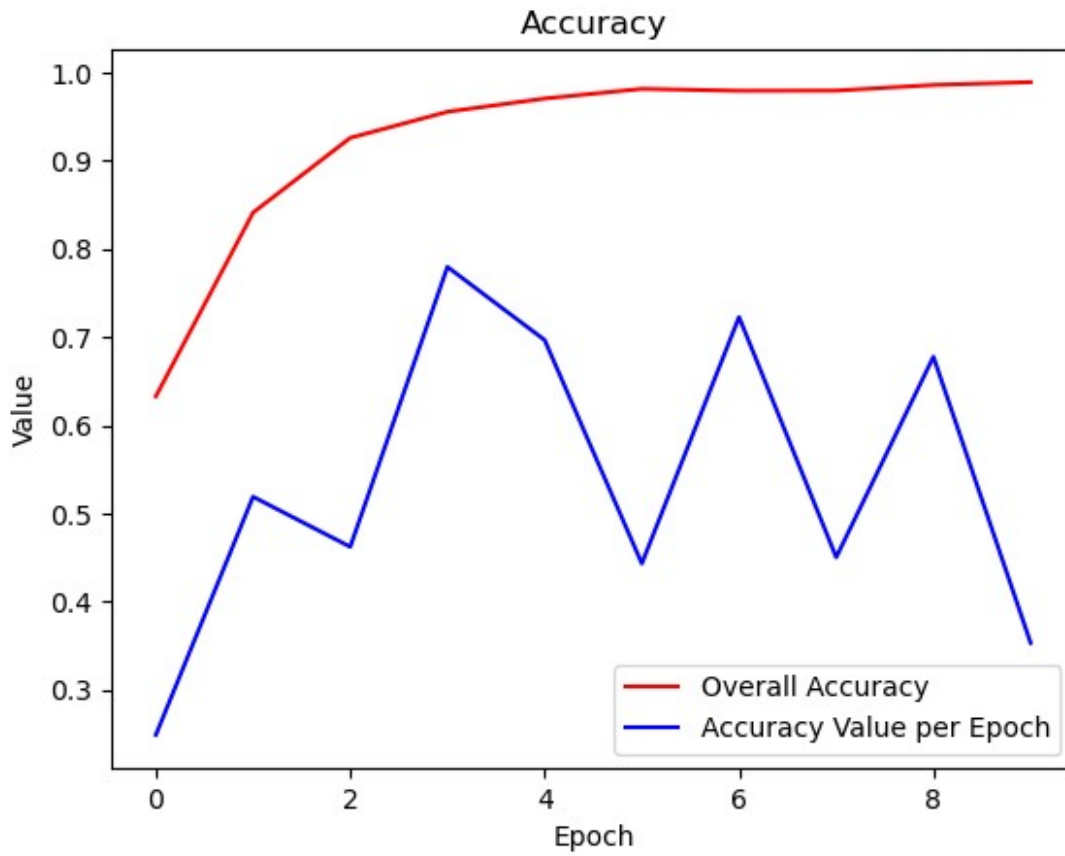
```
plt.title('Precision')
plt.plot(hist.history['precision'], 'r', label='Overall Precision')
plt.plot(hist.history['val_precision'], 'b', label='Precision Value per Epoch')
plt.xlabel('Epoch')
plt.ylabel('Value')
plt.legend()
plt.show()
```



```
plt.title('Recall')
plt.plot(hist.history['recall'], 'r', label='Overall Recall')
plt.plot(hist.history['val_recall'], 'b', label='Recall Value per Epoch')
plt.xlabel('Epoch')
plt.ylabel('Value')
plt.legend()
plt.show()
```



```
plt.title('Accuracy')
plt.plot(hist.history['accuracy'], 'r', label='Overall Accuracy')
plt.plot(hist.history['val_accuracy'], 'b', label='Accuracy Value per Epoch')
plt.xlabel('Epoch')
plt.ylabel('Value')
plt.legend()
plt.show()
```



```
audio_file = "C:/Users/raksh/Desktop/bird/wavfiles (1)/11713-6.wav"
audio_data, sample_rate = librosa.load(audio_file, duration=3)
mel_spec = librosa.feature.melspectrogram(y=audio_data,
sr=sample_rate)
mel_spec = librosa.power_to_db(mel_spec, ref=np.max)
mel_spec = tf.expand_dims(mel_spec, axis=0)
```

```
# Make a prediction
```

```
prediction = model.predict(mel_spec)
```

```
# Get the predicted class index
```

```
predicted_class_index = np.argmax(prediction, axis=1)[0]
```

```
# Get the corresponding bird name
```

```
bird_name = class_names[predicted_class_index]
```

```
# Print the prediction and bird name
```

```
print(f"Predicted Class Index: {predicted_class_index}")
```

```
print(f"Predicted Bird Name: {bird_name}")
```

```
print(f"Prediction Scores: {prediction}")
```

```
1/1 ————— 0s 40ms/step
```

```
Predicted Class Index: 3
```

Predicted Bird Name: Song Sparrow
Prediction Scores: [[0. 0. 0. 1. 0.]]