```
# import os
# os.listdir("/kaggle/input/tensorflow230")
```

```
# !pip install /kaggle/input/tensorflow230/tensorflow_estimator-2.3.0-py2.py3-none-any.whl
# !pip install /kaggle/input/tensorflow230/tensorflow-2.3.0rc2-cp37-cp37m-manylinux2010_x86_64.whl
```

```
from google.colab import drive
drive.mount('/content/drive', force_remount = True)
```

```
Mounted at /content/drive
```

```
import numpy as np
import pandas as pd
import wave
from scipy.io import wavfile
import os
import librosa
from librosa.feature import melspectrogram
import warnings
from sklearn.utils import shuffle
from sklearn.utils import class_weight
from PIL import Image
from uuid import uuid4
import sklearn
from tqdm import tqdm

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras import layers
from tensorflow.keras import Input
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Flatten, Dropout, Activation
from tensorflow.keras.layers import BatchNormalization, GlobalAveragePooling2D
from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, EarlyStopping
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.layers import Dense, Flatten, Dropout, Activation, LSTM, SimpleRNN, Conv1D, Input, BatchNormalization, GlobalAveragePooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import EfficientNetB0


import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
```

Data Preprocessing

```
train_df = pd.read_csv('/content/drive/MyDrive/Birds/train.csv')

train_df = train_df.query("rating>=3")

birds_count = {}
for bird_species, count in zip(train_df.ebird_code.unique(), train_df.groupby("ebird_code")["ebird_code"].count().values):
    birds_count[bird_species] = count
most_represented_birds = [key for key,value in birds_count.items() if value >= 50]

train_df = train_df.query("ebird_code in @most_represented_birds")

len(train_df.ebird_code.unique())
```

```
11
```

```
birds_to_recognise = sorted(shuffle(most_represented_birds)[:20])
print(birds_to_recognise)
```

```
['aldfly', 'astfly', 'dusfly', 'grcfly', 'gryfly', 'hamfly', 'leafly', 'olsfly', 'pasfly', 'wilfly', 'yebfly']
```

```
train_df = shuffle(train_df)
train_df.head()
```

	rating	playback_used	ebird_code	channels	date	pitch	duration	filenam
633	4.0	no	leafly	2 (stereo)	11/8/2015	level	21	XC289161.mp
427	4.0	no	gryfly	2 (stereo)	6/9/2013	Not specified	89	XC137914.mp

```
len(train_df)
```

```
1003
```

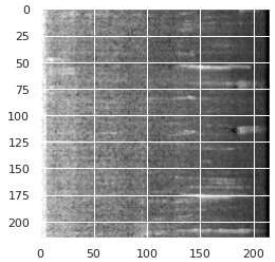
951 4.5 yes wilfly 2 (stereo) 12/7/2016 specified 6 XC384647.mp

```
def get_sample(filename, bird, output_folder):
    wave_data, wave_rate = librosa.load(filename)
    wave_data, _ = librosa.effects.trim(wave_data)
    #only take 5s samples and add them to the dataframe
    song_sample = []
    sample_length = 5*wave_rate
    samples_from_file = []
    #The variable below is chosen mainly to create a 216x216 image
    N_mels=216
    for idx in range(0,len(wave_data),sample_length):
        song_sample = wave_data[idx:idx+sample_length]
        if len(song_sample)>=sample_length:
            mel = melspectrogram(song_sample, n_mels=N_mels)
            db = librosa.power_to_db(mel)
            normalised_db = sklearn.preprocessing.minmax_scale(db)
            filename = str(uuid4())+".tif"
            db_array = (np.asarray(normalised_db)*255).astype(np.uint8)
            db_image =  Image.fromarray(np.array([db_array, db_array, db_array]).T)
            db_image.save("{}{}".format(output_folder,filename))

            samples_from_file.append({"song_sample":"{}{}".format(output_folder,filename),
                                      "bird":bird})
    return samples_from_file
```

```
%%time
warnings.filterwarnings("ignore")
samples_df = pd.DataFrame(columns=["song_sample","bird"])

#We limit the number of audio files being sampled to 1000 in this notebook to save time
#on top of having limited the number of bird species previously
sample_limit = 1005
sample_list = []

output_folder = "/content/drive/MyDrive/Birds/Mel2/"
os.mkdir(output_folder)
with tqdm(total=sample_limit) as pbar:
    for idx, row in train_df[:sample_limit].iterrows():
        pbar.update(1)
        try:
            audio_file_path = "/content/drive/MyDrive/Birds/train_audio/"
            audio_file_path += row.ebird_code

            if row.ebird_code in birds_to_recognise:
                sample_list += get_sample('{}/{}'.format(audio_file_path, row.filename), row.ebird_code, output_folder)
            else:
                sample_list += get_sample('{}/{}'.format(audio_file_path, row.filename), "nocall", output_folder)
        except:
            raise
            print("{} is corrupted".format(audio_file_path))

samples_df = pd.DataFrame(sample_list)
```

```
100%|██████████| 1003/1005 [1:08:05<00:08,  4.07s/it]CPU times: user 50min 42s, sys: 1h 6min 55s, total: 1h 57min 38s
Wall time: 1h 8min 5s
```

```
demo_img = Image.open(samples_df.iloc[0].song_sample)
plt.imshow(demo_img)
plt.show()
```




```
samples_df = shuffle(samples_df)
samples_df[:10]
```

	song_sample	bird
1024	/content/drive/MyDrive/Birds/Mel2/424c4aaa-018...	grcfly
4361	/content/drive/MyDrive/Birds/Mel2/67620c9f-ea9...	leafly
6225	/content/drive/MyDrive/Birds/Mel2/0ac94bd9-69b...	aldfly
4019	/content/drive/MyDrive/Birds/Mel2/95e91168-ed1...	astfly
2424	/content/drive/MyDrive/Birds/Mel2/df3fa9e1-0ed...	grcfly
1018	/content/drive/MyDrive/Birds/Mel2/42676dfe-a80...	grcfly
4179	/content/drive/MyDrive/Birds/Mel2/f473e3db-d34...	grcfly
6175	/content/drive/MyDrive/Birds/Mel2/64a391e1-0ea...	pasfly
9665	/content/drive/MyDrive/Birds/Mel2/c1681e15-d17...	pasfly

Creating the model

```
training_percentage = 0.9
training_item_count = int(len(samples_df)*training_percentage)
validation_item_count = len(samples_df)-int(len(samples_df)*training_percentage)
training_df = samples_df[:training_item_count]
validation_df = samples_df[training_item_count:]
```

```
classes_to_predict = sorted(samples_df.bird.unique())
input_shape = (216,216, 3)
effnet_layers = EfficientNetB0(weights=None, include_top=False, input_shape=input_shape)

for layer in effnet_layers.layers:
    layer.trainable = True

dropout_dense_layer = 0.3

model = Sequential()
model.add(effnet_layers)

model.add(GlobalAveragePooling2D())
model.add(Dense(256, use_bias=False))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(dropout_dense_layer))

model.add(Dense(len(classes_to_predict), activation="softmax"))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 efficientnetb0 (Functional)  (None, 7, 7, 1280)       4049571

 global_average_pooling2d (G  (None, 1280)             0
 lobalAveragePooling2D)

 dense (Dense)               (None, 256)               327680

 batch_normalization (BatchN  (None, 256)              1024
 ormalization)

 activation (Activation)     (None, 256)               0

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 11)                2827

=================================================================
Total params: 4,381,102
Trainable params: 4,338,567
Non-trainable params: 42,535
_________________________________________________________________
```

```
callbacks = [ReduceLROnPlateau(monitor='val_loss', patience=2, verbose=1, factor=0.7),
             EarlyStopping(monitor='val_loss', patience=5),
             ModelCheckpoint(filepath='best_model.h5', monitor='val_loss', save_best_only=True)]

model.compile(loss="categorical_crossentropy", optimizer='adam')
```

```
#  import cv2
#  import base64
#  import json
#  import numpy as np
# class_weights = class_weight.compute_class_weight("balanced", classes_to_predict, samples_df.bird.values)
# class_weights_dict = {i : class_weights[i] for i,label in enumerate(classes_to_predict)}
```

```
training_batch_size = 32
validation_batch_size = 32
target_size = (216,216)

train_datagen = ImageDataGenerator(
    rescale=1. / 255
)

train_generator = train_datagen.flow_from_dataframe(
    dataframe = training_df,
```

```
    x_col='song_sample',
    y_col='bird',
    directory='/',
    target_size=target_size,
    batch_size=training_batch_size,
    shuffle=True,
    class_mode='categorical')


validation_datagen = ImageDataGenerator(rescale=1. / 255)
validation_generator = validation_datagen.flow_from_dataframe(
    dataframe = validation_df,
    x_col='song_sample',
    y_col='bird',
    directory='/',
    target_size=target_size,
    shuffle=False,
    batch_size=validation_batch_size,
    class_mode='categorical')
```

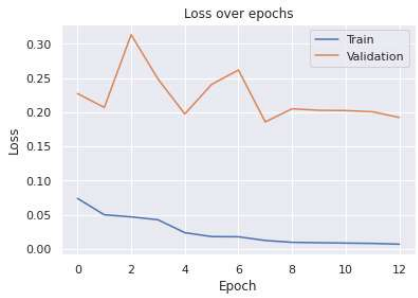
```
Found 10894 validated image filenames belonging to 11 classes.
Found 1211 validated image filenames belonging to 11 classes.
```

Train

```
history = model.fit(train_generator,
          epochs = 20,
          validation_data=validation_generator,
#           class_weight=class_weights_dict,
          callbacks=callbacks)
```

```
Epoch 1/20
341/341 [==============================] - 64s 187ms/step - loss: 0.0738 - val_loss: 0.2271 - lr: 3.4300e-04
Epoch 2/20
341/341 [==============================] - 64s 186ms/step - loss: 0.0498 - val_loss: 0.2068 - lr: 3.4300e-04
Epoch 3/20
341/341 [==============================] - 64s 186ms/step - loss: 0.0467 - val_loss: 0.3132 - lr: 3.4300e-04
Epoch 4/20
341/341 [==============================] - ETA: 0s - loss: 0.0425
Epoch 4: ReduceLROnPlateau reducing learning rate to 0.00024009999469853935.
341/341 [==============================] - 63s 185ms/step - loss: 0.0425 - val_loss: 0.2484 - lr: 3.4300e-04
Epoch 5/20
341/341 [==============================] - 64s 187ms/step - loss: 0.0236 - val_loss: 0.1973 - lr: 2.4010e-04
Epoch 6/20
341/341 [==============================] - 64s 187ms/step - loss: 0.0180 - val_loss: 0.2405 - lr: 2.4010e-04
Epoch 7/20
341/341 [==============================] - ETA: 0s - loss: 0.0176
Epoch 7: ReduceLROnPlateau reducing learning rate to 0.00016806999628897755.
341/341 [==============================] - 64s 186ms/step - loss: 0.0176 - val_loss: 0.2616 - lr: 2.4010e-04
Epoch 8/20
341/341 [==============================] - 64s 188ms/step - loss: 0.0122 - val_loss: 0.1856 - lr: 1.6807e-04
Epoch 9/20
341/341 [==============================] - 63s 185ms/step - loss: 0.0096 - val_loss: 0.2049 - lr: 1.6807e-04
Epoch 10/20
341/341 [==============================] - ETA: 0s - loss: 0.0090
Epoch 10: ReduceLROnPlateau reducing learning rate to 0.00011764899536501615.
341/341 [==============================] - 64s 187ms/step - loss: 0.0090 - val_loss: 0.2027 - lr: 1.6807e-04
Epoch 11/20
341/341 [==============================] - 64s 186ms/step - loss: 0.0084 - val_loss: 0.2022 - lr: 1.1765e-04
Epoch 12/20
341/341 [==============================] - ETA: 0s - loss: 0.0080
Epoch 12: ReduceLROnPlateau reducing learning rate to 8.235429777414538e-05.
341/341 [==============================] - 64s 187ms/step - loss: 0.0080 - val_loss: 0.2005 - lr: 1.1765e-04
Epoch 13/20
341/341 [==============================] - 64s 186ms/step - loss: 0.0068 - val_loss: 0.1921 - lr: 8.2354e-05
```

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Loss over epochs')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='best')
plt.show()
```



```
preds = model.predict_generator(validation_generator)
validation_df = pd.DataFrame(columns=["prediction", "groundtruth", "correct_prediction"])

for pred, groundtruth in zip(preds[:16], validation_generator.__getitem__(0)[1]):
    validation_df = validation_df.append({"prediction":classes_to_predict[np.argmax(pred)],
                                          "groundtruth":classes_to_predict[np.argmax(groundtruth)],
                                          "correct_prediction":np.argmax(pred)==np.argmax(groundtruth)}, ignore_index=True)
validation_df
```

	prediction	groundtruth	correct_prediction
0	wilfly	wilfly	True
1	dusfly	dusfly	True
2	hamfly	hamfly	True
3	grcfly	grcfly	True
4	dusfly	dusfly	True
5	aldfly	aldfly	True
6	dusfly	dusfly	True
7	grcfly	grcfly	True
8	grcfly	grcfly	True
9	aldfly	aldfly	True
10	olsfly	olsfly	True
11	astfly	astfly	True
12	leafly	leafly	True
13	astfly	astfly	True
14	astfly	astfly	True
15	wilfly	wilfly	True

```
# !rm -rf /kaggle/working/melspectrogram_dataset
```

Running It on the recordings

```
model.load_weights("best_model.h5")
```

```
def predict_on_melspectrogram(song_sample, sample_length):
    N_mels=216

    if len(song_sample)>=sample_length:
        mel = melspectrogram(song_sample, n_mels=N_mels)
        db = librosa.power_to_db(mel)
        normalised_db = sklearn.preprocessing.minmax_scale(db)
        db_array = (np.asarray(normalised_db)*255).astype(np.uint8)

        prediction = model.predict(np.array([np.array([db_array, db_array, db_array]).T]))
        predicted_bird = classes_to_predict[np.argmax(prediction)]
        return predicted_bird
    else:
        return "nocall"
```

```
def predict_submission(df, audio_file_path):

    loaded_audio_sample = []
    previous_filename = ""
    wave_data = []
    wave_rate = None
    sample_length = None

    for idx,row in df.iterrows():
        #I added this exception as I've heard that some files may be corrupted.
        try:
            if previous_filename == "" or previous_filename!=row.audio_id:
                filename = '{}/{}.mp3'.format(audio_file_path, row.audio_id)
                wave_data, wave_rate = librosa.load(filename)
                sample_length = 5*wave_rate
            previous_filename = row.audio_id

            #basically allows to check if we are running the examples or the test set.
            if "site" in df.columns:
                if row.site=="site_1" or row.site=="site_2":
                    song_sample = np.array(wave_data[int(row.seconds-5)*wave_rate:int(row.seconds)*wave_rate])
                elif row.site=="site_3":
                    #for now, I only take the first 5s of the samples from site_3 as they are groundtruthed at file level
                    song_sample = np.array(wave_data[0:sample_length])
            else:
                #same as the first condition but I isolated it for later and it is for the example file
                song_sample = np.array(wave_data[int(row.seconds-5)*wave_rate:int(row.seconds)*wave_rate])

            predicted_bird = predict_on_melspectrogram(song_sample, sample_length)
            df.at[idx,"birds"] = predicted_bird
        except:
            df.at[idx,"birds"] = "nocall"
    return df
```

```
from google.colab import drive
drive.mount('/content/drive', force_remount = True)
```

```
Mounted at /content/drive
```

```
# audio_file_path = "/content/drive/MyDrive/Birds/test_audio/"
# example_df = pd.read_csv("/content/drive/MyDrive/Birds/test_summary1.csv")
# #Ajusting the example filenames and creating the audio_id column to match with the test file.
# # example_df["audio_id"] = [ "BLKFR-10-CPL_20190611_093000.pt540" if filename=="BLKFR-10-CPL" else "ORANGE-7-CAP_20190606_093000.pt623" for filename in example_df["filename"]]
# example_df["audio_id"] = ["2.pt540" if filename=="2" for filename in example_df["filename"] ]
# if os.path.exists(audio_file_path):
#     example_df = predict_submission(example_df, audio_file_path)
# example_df
```

```
audio_file_path = "/content/drive/MyDrive/Birds/example_test_audio"
example_df = pd.read_csv("/content/drive/MyDrive/Birds/example_test_audio_summary.csv")
#Ajusting the example filenames and creating the audio_id column to match with the test file.
example_df["audio_id"] = [ "BLKFR-10-CPL_20190611_093000.pt540" if filename=="BLKFR-10-CPL" else "ORANGE-7-CAP_20190606_093000.pt623" for filename in example_df["filename"]]

if os.path.exists(audio_file_path):
    example_df = predict_submission(example_df, audio_file_path)
example_df
```

```
1/1 [==============================] - 2s 2s/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 49ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 48ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 38ms/step
```

```
# test_file_path = "/kaggle/input/birdsong-recognition/test_audio"
# test_df = pd.read_csv("/kaggle/input/birdsong-recognition/test.csv")
# # submission_df = pd.read_csv("/kaggle/input/birdsong-recognition/sample_submission.csv")

# if os.path.exists(test_file_path):
#     submission_df = predict_submission(test_df, test_file_path)
```

```
#     submission_df = predict_submission(test_df, test_file_path)

# print(submission_df)
# # submission_df[["row_id","birds"]].to_csv('submission.csv', index=False)
# # submission_df.head()
```

```
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 36ms/step
```

✓ 22s completed at 17:27