MAULANA AZAD

NATIONAL INSTITUTE OF TECHNOLOGY
BHOPAL, 462003

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MAJOR PROJECT ON

Bird Species Classification Using Sound

UNDER THE GUIDANCE OF

Dr. Praveen Kaushik And Dr. Vaibhav Soni

SUBMITTED IN PARTIAL FULFILMENT FOR THE DEGREE OF BACHELOR OF TECHNOLOGY

SUBMITTED BY :

pranshu kumar choudhary

171112297

SESSION: 2020-21

MAULANA AZAD

NATIONAL INSTITUTE OF TECHNOLOGY BHOPAL, 462003

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that **pranshu kumar choudhary**, students of B.Tech. Final Year (VIII SEMESTER), have successfully completed their project entitled “Bird Species Classification Using Sound” in partial fulfillment of their major project in Computer Science & Engineering.

Dr. Praveen Kaushik And
Dr. Vaibhav Soni
(Project Guide)

MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY BHOPAL

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We, hereby, declare that the following report which is being presented in the Major Project Documentation entitled "**Bird Species Classification Using Sound**" is the partial fulfillment of the requirements of the final year (eighth semester) **Major Project** in the field of **Computer Science & Engineering**. It is an authentic documentation of our original work carried out under the guidance of **Dr. Praveen Kaushik And Dr. Vaibhav Soni**. The work presented here is carried out entirely at **Maulana Azad National Institute of Technology, Bhopal**. The following project and its report, in part or whole, has not been presented or submitted by us for any purpose in any other organization or institution.

We, hereby, declare that the facts mentioned above are true to the best of our knowledge. In case of unlikely discrepancy that may occur, we will be the ones to take responsibility.

NAME	**SCHOLAR NO.**	**SIGNATURE**
Pranshu kumar choudhary	**171112297**	Pranshu

ACKNOWLEDMENT

With due respect, we express our deep sense of gratitude and thanks to our respected guide **Dr. Praveen Kaushik And Dr. Vaibhav Soni**, for **her** valuable help and guidance. We are thankful for the encouragement that **he** has given us in completing this project successfully. **Her**rigorous evaluation and constructive criticism was of great assistance.

It is imperative for us to mention the fact that this major project could not have been accomplished without the periodic advice and suggestions of our project coordinator **Dr. S K Saritha** and **Dr. Manish Pandey**.

We are also grateful to our director **Dr. N. S. Raghuwanshi,** for permitting us to utilize all the necessary facilities in the college.

Needless to mention is the additional help and support extended by respected HOD,

Dr. Nilay Khare, in allowing us to use the departmental laboratories and other services.

We are also thankful to all the other faculty, staff members and laboratory attendants of our department for their kind co-operation and help. Last but certainly not the least, we would like to express our deep appreciation towards our family members and batch mates for providing the much needed support and encouragement.

ABSTRACT

Acoustic monitoring has gained widespread interest as an ecological tool for wildlife population assessment, conservation, and biodiversity research. Many species emit regular vocalizations or other acoustic signals that are species-specific, which enables monitoring via sound recognition. Identifying bird species in audio recordings is a challenging field of research. Accurate prediction of bird species from audio recordings is beneficial to bird conservation. Devising effective algorithms for bird species classification is a preliminary step toward extracting useful ecological data from recordings collected in the field. In this Synopsis we will define the problem and propose a Deep Learning based method to classify the bird species using their sounds. Convolutional neural networks (CNNs) are powerful toolkits of machine learning which have proven efficient in the field of image processing and sound recognition.

Keywords: Bioacoustics, Classification, Convolutional Neural Networks, Audio Features, Bird Sound Identification

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

Bioacoustics is a cross-disciplinary science that combines biology and acoustics. Usually it refers to the investigation of sound production, dispersion and reception in animals.

Public consciousness about environmental conservation and sustainable development has awakened in recent years. The timely and accurate detection of animals, birds and insects is of critical importance for conservation, ecology and epidemiology. The effective analysis of the natural soundscape is a constituent component of this task.

Recent high-profile advances in deep learning have improved performance across many application domains: One such domain is Bioacoustics signal analysis. In this synopsis we will present one such application of Deep Learning in Bioacoustics signal analysis. We will propose a method to classify Bird Species using the sound recordings.

Recognizing birds by their song in the wild is a challenging task. With the arrival of convolutional neural networks (CNNs, ConvNets) automated processing of field recordings made a huge leap forward. Generating deep features based on visual representations of audio recordings has proven to be very effective when applied to the classification of audio events such as bird sounds.

Audio classification systems typically begin by extracting acoustic features from audio signals. Such features often pertain to individual frames (i.e., very short segments of signal). For example, one commonly used feature is the spectrum of a signal frame, which describes the intensity of (a short segment of) the signal as a function of frequency. To apply many standard algorithms for classification, it is necessary to represent a sound, which contains multiple frames, using a fixed-length vector.

2.Theoretical Aspects.

Our goal is to automatically identify which species of bird is present in an audio recording using Deep Learning Techniques by learning from a collection of labeled examples.

‘

The problem includes identifying the features or extracting the features from the audio by processing the audio using various transform. The commonly used method used for identifying the features from the audio includes transform such as Fourier Transform, Short Time Fourier Transform, Mel-Frequency Cepstral Coefficients (MFCC).

After identifying features, Deep learning techniques can be applied to make the classification.

Existing bird species distribution data are collected by manual surveys, which are labor intensive, and require observers trained in bird recognition. Automated bird population surveys could provide vast amounts of useful data for species distribution modeling, while requiring less effort and expense than human surveys.

3.Research Objective

This project will involve processing of audio signals and extracting the features from the signal using methods which will be critical for classification by applying Deep Learning techniques. How to extract the most representative features from the speech signal is also a difficult point in the project.

4.Literature Review & Research Gaps Identified

research papers and discovered that most of the work happened to be initiated by the various AI challenges, such as BirdCLEF and DCASE. Fortunately, winners of those challenges usually describe their approaches, so after checking the leader boards some interesting insights were obtained: almost all winning solutions used Convolutional Neural Networks (CNNs) or Recurrent Convolutional Neural Network (RCNNs) the gap between CNN-based models and shallow, feature-based approaches remained considerably high even though many of the recordings were quite noisy the CNNs worked well without any additional noise removal and many teams claimed that noise reduction techniques did not help data augmentation techniques seemed to be widely used, especially the techniques used in audio processing such as time or frequency shift some winning teams successfully approached it with semi-supervised learning methods (pseudo-labeling) and some increased AUC by model ensemble

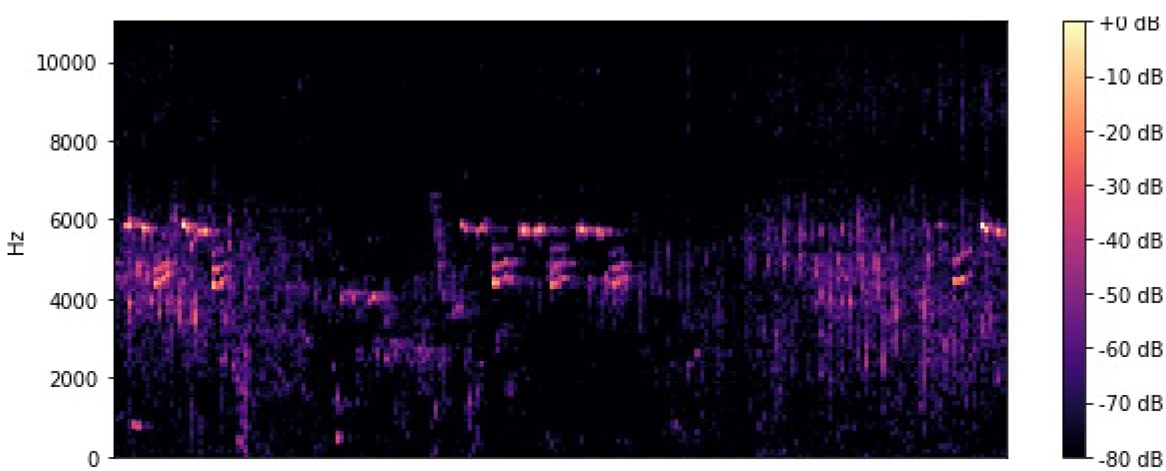
5.Proposed work and Methodology

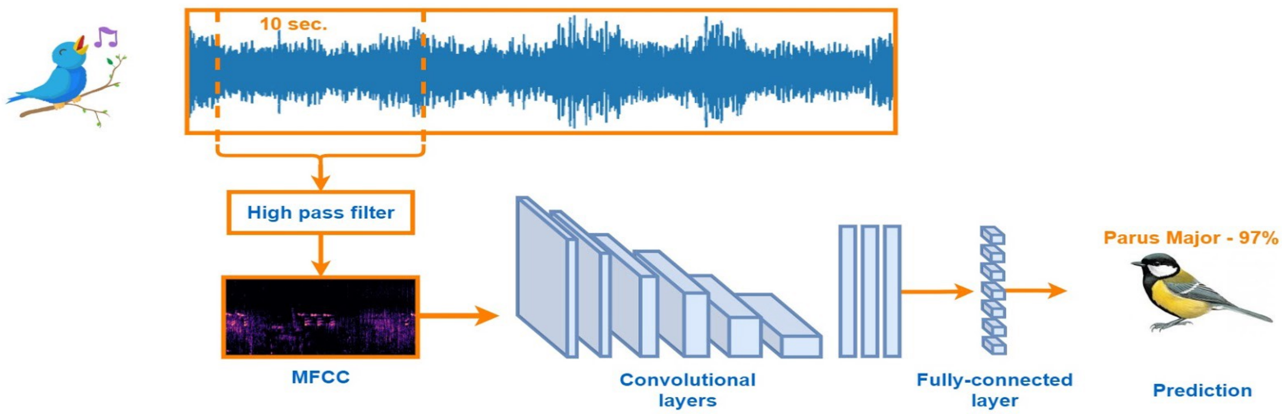
5.1Proposed Architecture

The methodology are architecture we propose consists of following steps. First, we will extract spectrograms from all audio recordings. Secondly, we will extend our training set through dataset augmentation techniques. Next, we will try to find the best CNN architecture with respect to number of classes.

5.2 Spectogram Extraction

Our strategy for the task was to treat bird sound classification as image classification; hence we need to visualize bird sounds. Each sound we hear is composed of multiple sound frequencies at the same time. The trick of a spectrogram is to visualize also those frequencies in one plot, instead of visualizing only the amplitude as in the waveform. Mel scale is known as an audio scale of sound pitches that seem to be in equal distance from each other for listeners.



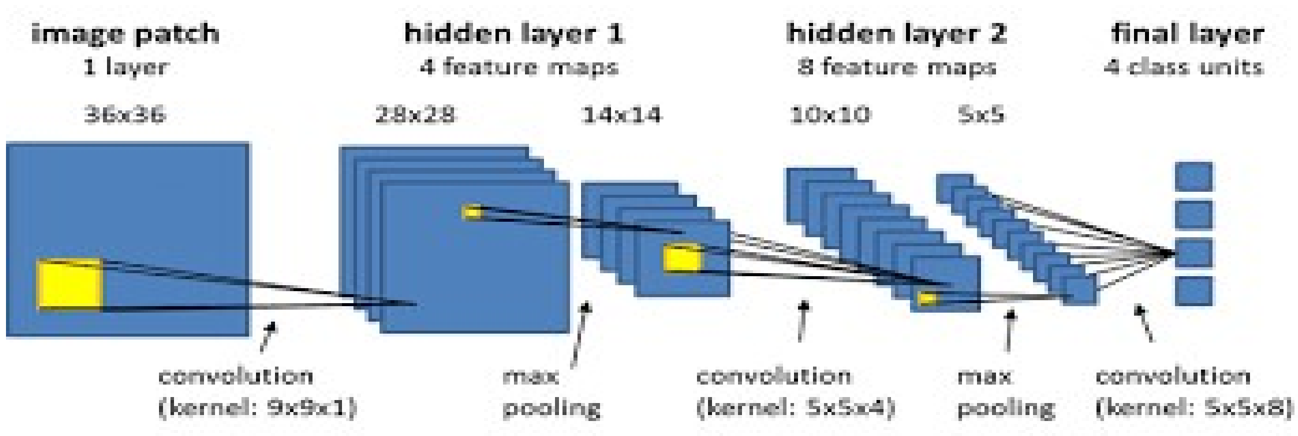


5.2 Convolutional Neural Networks

Using Convolutional neural networks, we can achieve the task of bird species classification without the need for hand crafted features. Convolutional neural networks (CNNs) have been widely used for the task of image classification] . The 3 channel (RGB) matrix representation of an image is fed into a CNN which is trained to predict the image class. In this study, the sound wave can be represented as a spectrogram, which in turn can be treated as an image. A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. ConvNets have the ability to learn these filters/characteristics.

5.3 Convolution layer - the kernel

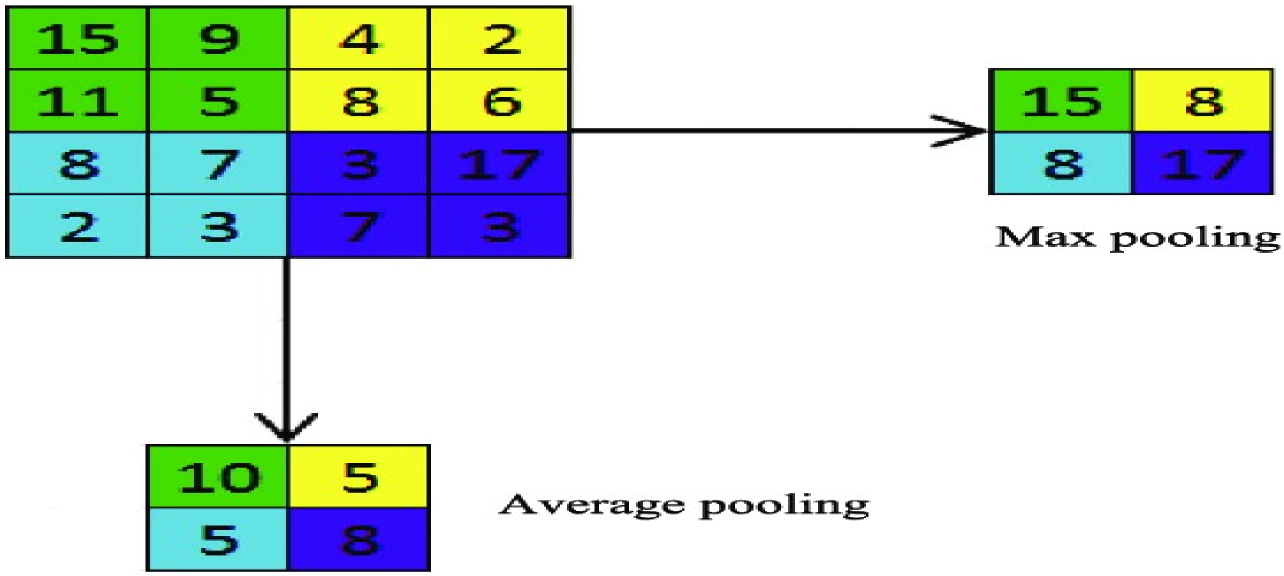
The main building block of CNN is the convolutional layer. Convolution is a mathematical operation to merge two sets of information. In our case the convolution is applied on the input data using a convolution filter to produce a feature map. There are a lot of terms being used so let’s visualize them one by 12 one. On the left side is the input to the convolution layer, for example the input image.



On the right is the convolution filter, also called the kernel, we will use these terms interchangeably. This is called a 3x3 convolution due to the shape of the filter. We perform the convolution operation by sliding this filter over the input. At every location, we do element-wise matrix multiplication and sum the result. This sum goes into the feature map.

5.4 Pooling Layer

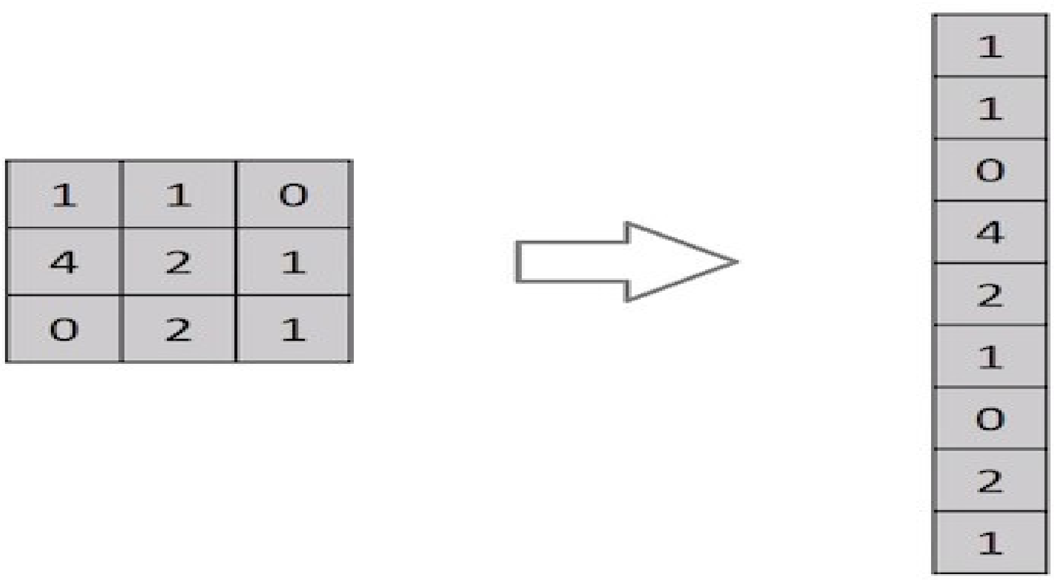
Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data through dimensionality reduction. There are two types of Pooling: Max Pooling and Average Pooling. Max Pooling returns the maximum value from the portion of the image covered by the Kernel. On the other hand, Average Pooling returns the average of all the values from the portion of the image covered by the Kernel. Max Pooling also performs as a Noise Suppressant. On the other hand, Average Pooling simply performs dimensionality reduction as a noise suppressing mechanism. Hence, we can say that Max Pooling performs a lot better than Average Pooling



Max pooling and Avg pooling

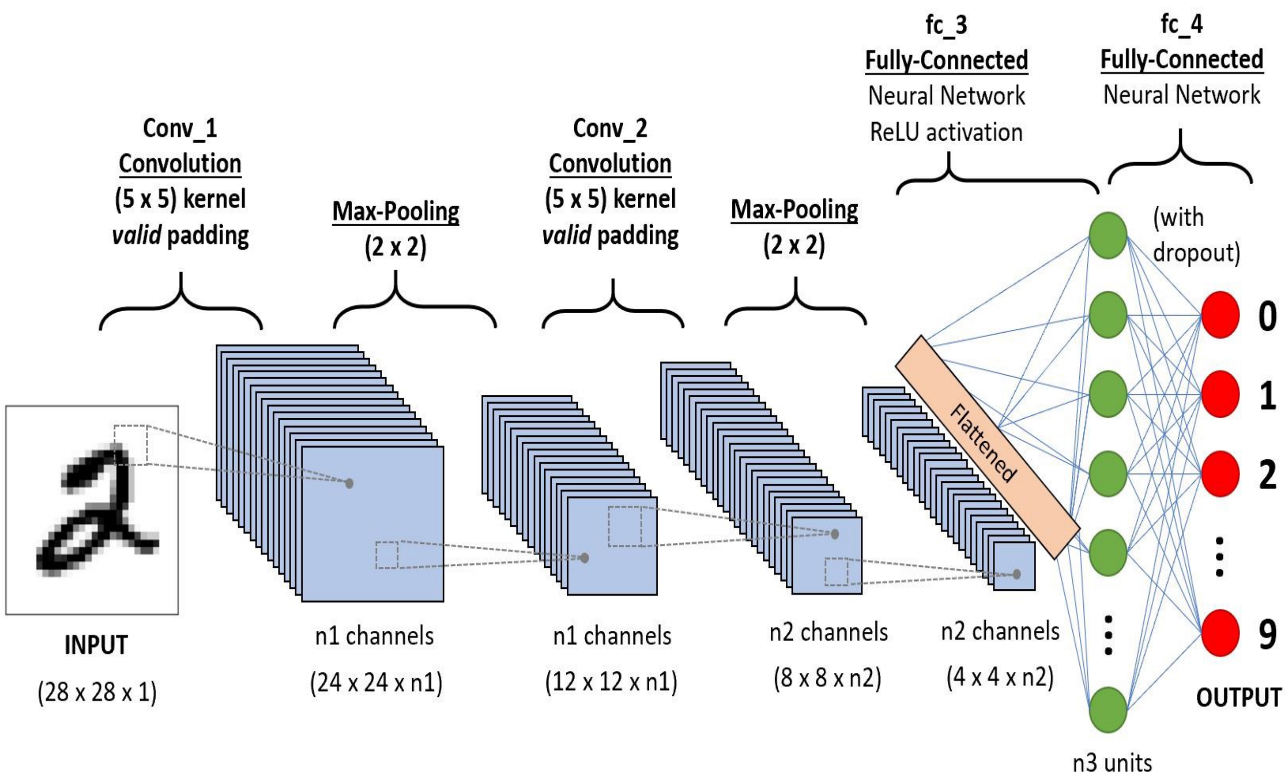
The Convolutional Layer and the Pooling Layer together form the i-th layer of a Convolutional Neural Network. Depending on the complexities in the images, the number of such layers may be increased for capturing low-levels details even further, but at the cost of more computational power.

5.5 Flattening



Flattening of a 3x3 image matrix into a 9x1 vector evious two steps, we're supposed to have a pooled feature map by now. As the name of this step implies, we literally flatten our pooled feature map into a column like in the image above.

5.6 Fully Connected Layers



Adding a Fully-Connected layer is a (usually) cheap way of learning non-linear combinations of the high-level features as represented by the output of the convolutional layer. The Fully-Connected layer is learning a possibly non-linear function in that space.

6. Source of Data

The following Data Sources can be used to get the recordings of bird sound and then the entire data can be integrated to form a dataset

1) Florida Bird Sounds
https://www.floridamuseum.ufl.edu/birds/florida-bird-sounds/

2) Daves Gammons Nature Sounds
http://org.elon.edu/naturesounds/commonName.html

3) BirdCLEF

<u>7. Implementation</u>

7.1 Preparing data set

Code Snippets

- **Bird_audio_dataset_prepare.ipynb**

This notebook consists of codes for the preparation of dataset , extraction of audio features of bird sound.

jupyter major project Last Checkpoint: a day ago (autosaved)

select directry of bird soud

In [2]:
```
#using os library to selecting bird sound directory
import os

DATASET_PATH = os.listdir()
DATASET_PATH
```

Out[2]:
```
['.ipynb_checkpoints',
 'audio dataset  preparing .ipynb',
 'bird',
 'bird sound classification through ml.ipynb',
 'bird sound data.xlsx',
 'bird sound detection through CNN.ipynb',
 'bird_1',
 'bird_sound.csv',
 'bird_sound_big_dataset.json',
 'bird_sound_data_mfccs.json',
 'class_examples.png',
 'class_examples1.png',
 'class_examples2.png',
 'data.bird',
 'data_10.json',
 'hilber transformation of bird sound.ipynb',
 'large dataset of bird sound .ipynb',
 'major project .ipynb',
 'toword data science bird sound dectection .ipynb',
 'Untitled.ipynb',
 'Untitled1.ipynb']
```

In [3]:
```
DATASET_PATH = os.listdir()[2]
DATASET_PATH
```

Out[3]:
```
'bird'
```

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

Run Code

Extracting mfccs feature from bird sound using librosa library

In [4]:

```python
import json
import math
import librosa

DATASET_PATH
JSON_PATH = "data_101.json"
SAMPLE_RATE = 22050
TRACK_DURATION = 2.9 # measured in seconds
SAMPLES_PER_TRACK = SAMPLE_RATE * TRACK_DURATION


def save_mfcc(dataset_path, json_path, num_mfcc=13, n_fft=1724, hop_length=10, num_segments=5):

    # dictionary to store mapping, labels, and MFCCs
    data = {
        "mapping": [],
        "labels": [],
        "mfcc": []
    }

    samples_per_segment = int(SAMPLES_PER_TRACK / num_segments)
    num_mfcc_vectors_per_segment = math.ceil(samples_per_segment / hop_length)

    # loop through all bird sub-folder
    for i, (dirpath, dirnames, filenames) in enumerate(os.walk(dataset_path)):

        # ensure we're processing a bird sub-folder level
        if dirpath is not dataset_path:

            # save bird label (i.e., sub-folder name) in the mapping
            semantic_label = dirpath.split("/")[-1]
            data["mapping"].append(semantic_label)
            print("\nProcessing: {}".format(semantic_label))

            # process all audio files in bird sub-dir
            for f in filenames:

                # load audio file
                file_path = os.path.join(dirpath, f)
                signal, sr = librosa.load(file_path, sr=SAMPLE_RATE)

                # process all segments of audio file
                for d in range(num_segments):

                    # calculate start and finish sample for current segment
                    start = samples_per_segment * d
                    finish = start + samples_per_segment

                    # extract mfcc
                    mfcc = librosa.feature.mfcc(signal[start:finish], sr=sr, n_mfcc=num_mfcc, n_fft=n_fft, hop_length=hop_length
                    mfcc = mfcc.T

                    # store only mfcc feature with expected number of vectors
                    if len(mfcc) == num_mfcc_vectors_per_segment:

                        data["mfcc"].append(mfcc.tolist())
                        data["labels"].append(i-1)
                        print("{}, segment:{}".format(file_path, d+1))

    # save MFCCs to json file
    with open(json_path, "w") as fp:
        json.dump(data, fp, indent=4)


if __name__ == "__main__":
    save_mfcc(DATASET_PATH, JSON_PATH, num_segments=10)
```

```
bird\7_Egretta_tricolor\hardy10 - Copy (2).wav, segment:8
bird\7_Egretta_tricolor\hardy10 - Copy (2).wav, segment:9
```

mapping	labels	mfcc__001	mfcc__002	mfcc__003	mfcc__004	mfcc__005	mfcc__006	mfcc__007	mfcc__008	mfcc__009
bird\10_Rallus_longirostris	0	-624.260986328125	92.00140380859375	18.17132568359375	33.1611328125	1.1320595741271973	16.523038864135742	14.789907455444336	3.268691301345825	-3.287925958633423
bird\1_Buteo_lineatus	0	-623.9613037109375	92.02861785888672	18.138294219970703	33.16058349609375	1.1381715536117554	16.53753662109375	14.78433609008789	3.2811431884765625	-3.2927358150482178
d\2_Haliaeetus_leucocephalus	0	-623.0792236328125	92.10421752929688	18.04083251953125	33.160396575927734	1.157281756401062	16.577327728271484	14.763258934020996	3.316338539123535	-3.30635404586792
bird\3_Pandion_haliaetus	0	-621.6622314453125	92.2115249633789	17.882274627685547	33.16387939453125	1.1917400360107422	16.63425064086914	14.717887878417969	3.3709089756011963	-3.3256025314331055
bird\4_Colinus_virginianus	0	-619.7748413085938	92.33281707763672	17.669248580932617	33.17171096801758	1.2405223846435547	16.698238372802734	14.639131546020508	3.4392940998077393	-3.3457512855529785
bird\5_Butorides_virescens	0	-617.4868774414062	92.45413208007812	17.408891677856445	33.1821403503418	1.3014147281646729	16.761693954467773	14.523655891418457	3.516831398010254	-3.3658883571624756
bird\6_Egretta_caerulea	0	-614.864013671875	92.56559753417969	17.10472869873047	33.19292449951172	1.3749357461929321	16.81945037841797	14.372047424316406	3.602875232696533	-3.385664939880371
bird\7_Egretta_tricolor	0	-611.966552734375	92.66314697265625	16.75997543334961	33.19853973388672	1.460085153579712	16.87118911743164	14.18886661529541	3.695028305053711	-3.40444016456604
bird\8_Mycteria_americana	0	-608.8475952148438	92.74568176269531	16.376928329467773	33.193687438964844	1.5573513507843018	16.920318603515625	13.980368614196777	3.7897119522094727	-3.4227712154388428
bird\9_Grus_canadensis	0	-605.55517578125	92.81458282470703	15.96061897277832	33.173744201660156	1.664536476135254	16.971195220947266	13.755878448486328	3.883894920349121	-3.4433085918426514
	1	-602.1316528320312	92.87165832519531	15.516538619995117	33.13606262207031	1.7788054943084717	17.02606964111328	13.522379875183105	3.975062847137451	-3.4672436714172363
	1	-598.6109619140625	92.91932678222656	15.04739761352539	33.07888412475586	1.8996716737747192	17.086421966552734	13.283526420593262	4.0618181228637695	-3.4927098751068115
	1	-595.0226440429688	92.95985412597656	14.557019233703613	33.00153350830078	2.025371551513672	17.15276336669922	13.043218612670898	4.143546104431152	-3.5193018913269043
	1	-591.3929443359375	92.9948959350586	14.050657272338867	32.9051513671875	2.1526060104370117	17.223119735717773	12.805130004882812	4.221736431121826	-3.546473503112793
	1	-587.7434692382812	93.02547454833984	13.532754898071289	32.79108428955078	2.2792482376098633	17.29670524597168	12.571634292602539	4.296596050262451	-3.573970079421997
	1	-584.0921020507812	93.05204010009766	13.006463050842285	32.66138458251953	2.404888391494751	17.373046875	12.343315124511719	4.367568016052246	-3.6011099815368652
	1	-580.4526977539062	93.07512664794922	12.473936080932617	32.517330169677734	2.528803825378418	17.451644897460938	12.120405197143555	4.43435001373291	-3.6270670890808105
	1	-576.813720703125	93.06205749511719	11.971010208129883	32.32716369628906	2.6822566986083984	17.498472213745117	11.937117576599121	4.464831829071045	-3.618867874145508
	1	-573.1990966796875	93.03105926513672	11.482002258300781	32.111656188964844	2.8449301719665527	17.531015396118164	11.77629280090332	4.479525089263916	-3.5951521396636963
	1	-569.62890625	92.99745178222656	10.993606567382812	31.888187408447266	2.9996888637542725	17.564208984375	11.62362289428711	4.495726108551025	-3.571979522705078
	2	-566.1090698242188	92.96112823486328	10.506807327270508	31.657405853271484	3.1458234786987305	17.598003387451172	11.480093002319336	4.514115333557129	-3.5497488975524902
	2	-562.6433715820312	92.92250061035156	10.021958351135254	31.419097900390625	3.2823128700256348	17.63217544555664	11.346366882324219	4.535425662994385	-3.5277998447418213
	2	-559.2354125976562	92.88125610351562	9.539381980895996	31.174232482910156	3.409787178039551	17.66725730895996	11.222491264343262	4.559190273284912	-3.5071730613708496
	2	-555.8870849609375	92.83724975585938	9.05849838256836	30.923065185546875	3.5293684005737305	17.70429229736328	11.108348846435547	4.584606170654297	-3.488387107849121
	2	-552.5999755859375	92.79049682617188	8.579522132873535	30.666091918945312	3.641240119934082	17.743423461914062	11.003976821899414	4.6117424964904785	-3.4716553688049316
	2	-549.3759765625	92.7403564453125	8.102975845336914	30.404468536376953	3.745666980743408	17.78441619873047	10.909684181213379	4.641350269317627	-3.4573867321014404
	2	-546.2156372070312	92.68650817871094	7.628514289855957	30.138362884521484	3.8421530723571777	17.826709747314453	10.82575798034668	4.674754619598389	-3.4449563026428223
	2	-543.11865234375	92.6285400390625	7.1550798416137695	29.867870330810547	3.931241989135742	17.87055015563965	10.751906394958496	4.711977005004883	-3.434354543685913
	2	-540.0850219726562	92.56568908691406	6.682156562805176	29.59379005432129	4.013483047485352	17.916072845458984	10.688023567199707	4.753035545349121	-3.426173686981201
	2	-537.1144409179688	92.49761962890625	6.209491729736328	29.316898345947266	4.088963508605957	17.962650299072266	10.633390426635742	4.797756671905518	-3.4207162857055664
	3	-534.2063598632812	92.42373657226562	5.736998081207275	29.038171768188477	4.157886505126953	18.00977325439453	10.587392807006836	4.846006393432617	-3.4183902740478516
	3	-531.3604125976562	92.34330749511719	5.264186859130859	28.758251190185547	4.220669746398926	18.057456970214844	10.549970626831055	4.8978095054626465	-3.4195327758789062
	3	-528.5757446289062	92.25590515136719	4.790546417236328	28.477352142333984	4.277214527130127	18.105546951293945	10.52121353149414	4.953575134277344	-3.4239754676818848
	3	-525.8516845703125	92.16087341308594	4.315433502197266	28.195852279663086	4.328129768371582	18.154497146606445	10.50114631652832	5.013031005859375	-3.4321413040161133
	3	-523.1873779296875	92.05801391601562	3.838655471801758	27.914377212524414	4.373953342437744	18.204360961914062	10.489303588867188	5.0756330490112305	-3.4443509578704834

jupyter major project Last Checkpoint: Yesterday at 7:48 PM (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run Code

Bird Sound Classification Through Artificial Nurale Network

In [10]:

```
import numpy as np
from sklearn.model_selection import train_test_split
import tensorflow.keras as keras
import matplotlib.pyplot as plt

DATA_PATH = "bird_sound_data_mfccs.json"


def load_data(data_path):


    with open(data_path, "r") as fp:
        data = json.load(fp)

    X = np.array(data["mfcc"])
    y = np.array(data["labels"])
    return X, y


def plot_history(history):


    fig, axs = plt.subplots(2)

    # create accuracy sublpot
    axs[0].plot(history.history["accuracy"], label="train accuracy")
    axs[0].plot(history.history["val_accuracy"], label="test accuracy")
    axs[0].set_ylabel("Accuracy")
    axs[0].legend(loc="lower right")
    axs[0].set_title("Accuracy eval")

    # create error sublpot
    axs[1].plot(history.history["loss"], label="train error")
    axs[1].plot(history.history["val_loss"], label="test error")
    axs[1].set_ylabel("Error")
    axs[1].set_xlabel("Epoch")
    axs[1].legend(loc="upper right")
    axs[1].set_title("Error eval")

    plt.show()


def prepare_datasets(test_size, validation_size):

    # load data
    X, y = load_data(DATA_PATH)

    # create train, validation and test split
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size)
    X_train, X_validation, y_train, y_validation = train_test_split(X_train, y_train, test_size=validation_size)

    # add an axis to input sets
    X_train = X_train[..., np.newaxis]
    X_validation = X_validation[..., np.newaxis]
    X_test = X_test[..., np.newaxis]

    return X_train, X_validation, X_test, y_train, y_validation, y_test


def build_model(input_shape):


    # build network topology
    model = keras.Sequential()

    # 1st conv layer
    model.add(keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=input_shape))
    model.add(keras.layers.MaxPooling2D((3, 3), strides=(2, 2), padding='same'))
    model.add(keras.layers.BatchNormalization())

    # 2nd conv layer
    model.add(keras.layers.Conv2D(32, (3, 3), activation='relu'))
    model.add(keras.layers.MaxPooling2D((3, 3), strides=(2, 2), padding='same'))
    model.add(keras.layers.BatchNormalization())

    # 3rd conv layer
    model.add(keras.layers.Conv2D(32, (2, 2), activation='relu'))
    model.add(keras.layers.MaxPooling2D((2, 2), strides=(2, 2), padding='same'))
    model.add(keras.layers.BatchNormalization())

    # flatten output and feed it into dense layer
    model.add(keras.layers.Flatten())
    model.add(keras.layers.Dense(64, activation='relu'))
    model.add(keras.layers.Dropout(0.5))

    # output layer
    model.add(keras.layers.Dense(10, activation='softmax'))
```

jupyter major project Last Checkpoint: Yesterday at 7:48 PM (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Removing Over Fitting

In [13]:

```python
# path to json file that stores MFCCs and bird labels for each processed segment
DATA_PATH

def load_data(data_path):


    with open(data_path, "r") as fp:
        data = json.load(fp)

    # convert lists to numpy arrays
    X = np.array(data["mfcc"])
    y = np.array(data["labels"])

    print("Data succesfully loaded!")

    return X, y


def plot_history(history):

    fig, axs = plt.subplots(2)

    # create accuracy sublpot
    axs[0].plot(history.history["accuracy"], label="train accuracy")
    axs[0].plot(history.history["val_accuracy"], label="test accuracy")
    axs[0].set_ylabel("Accuracy")
    axs[0].legend(loc="lower right")
    axs[0].set_title("Accuracy eval")

    # create error sublpot
    axs[1].plot(history.history["loss"], label="train error")
    axs[1].plot(history.history["val_loss"], label="test error")
    axs[1].set_ylabel("Error")
    axs[1].set_xlabel("Epoch")
    axs[1].legend(loc="upper right")
    axs[1].set_title("Error eval")

    plt.show()


if __name__ == "__main__":

    # load data
    X, y = load_data(DATA_PATH)

    # create train/test split
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

    # build network topology
    model = keras.Sequential([

        # input layer
        keras.layers.Flatten(input_shape=(X.shape[1], X.shape[2])),

        # 1st dense layer
        keras.layers.Dense(512, activation='relu', kernel_regularizer=keras.regularizers.l2(0.001)),
        keras.layers.Dropout(0.2),

        # 2nd dense layer
        keras.layers.Dense(256, activation='relu', kernel_regularizer=keras.regularizers.l2(0.001)),
        keras.layers.Dropout(0.2),

        # 3rd dense layer
        keras.layers.Dense(64, activation='relu', kernel_regularizer=keras.regularizers.l2(0.001)),
        keras.layers.Dropout(0.2),

        # output layer
        keras.layers.Dense(10, activation='softmax')
    ])

    # compile model
    optimiser = keras.optimizers.Adam(learning_rate=0.0001)
    model.compile(optimizer=optimiser,
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])

    model.summary()

    # train model
    history = model.fit(X_train, y_train, validation_data=(X_test, y_test), batch_size=32, epochs=100)

    # plot accuracy and error as a function of the epochs
    plot_history(history)

    # evaluate model on test set
    test_loss, test_acc = model.evaluate(X_test, y_test, verbose=2)
```

jupyter audio dataset preparing Last Checkpoint: 03/26/2021 (autosaved)

Logout

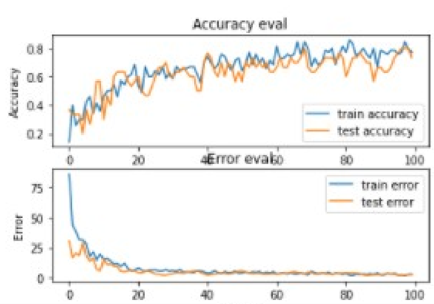
File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run Code

```python
    # create error sublpot
    axs[1].plot(history.history["loss"], label="train error")
    axs[1].plot(history.history["val_loss"], label="test error")
    axs[1].set_ylabel("Error")
    axs[1].set_xlabel("Epoch")
    axs[1].legend(loc="upper right")
    axs[1].set_title("Error eval")

    plt.show()


if __name__ == "__main__":

    # load data
    X, y = load_data(DATA_PATH)

    # create train/test split
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

    # build network topology
    model = keras.Sequential([

        # input layer
        keras.layers.Flatten(input_shape=(X.shape[1], X.shape[2])),

        # 1st dense layer
        keras.layers.Dense(512, activation='relu', kernel_regularizer=keras.regularizers.l2(0.001)),
        keras.layers.Dropout(0.1),

        # 2nd dense layer
        keras.layers.Dense(256, activation='relu', kernel_regularizer=keras.regularizers.l2(0.001)),
        keras.layers.Dropout(0.1),

        # 3rd dense layer
        keras.layers.Dense(64, activation='relu', kernel_regularizer=keras.regularizers.l2(0.001)),
        keras.layers.Dropout(0.1),

        # output layer
        keras.layers.Dense(10, activation='softmax')
    ])

    # compile model
    optimiser = keras.optimizers.Adam(learning_rate=0.0001)
    model.compile(optimizer=optimiser,
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])

    model.summary()

    # train model
    history = model.fit(X_train, y_train, validation_data=(X_test, y_test), batch_size=32, epochs=100)

    # plot accuracy and error as a function of the epochs
    plot_history(history)
```

```
Epoch 100/100
3/3 [==============================] - 0s 81ms/step - loss: 2.9832 - accuracy: 0.7802 - val_loss: 2.7347 - val_accuracy: 0.7
333
```



File Edit View Insert Cell Kernel Widgets Help   Not Trusted   Python 3

Run   Code

CNN model for bird sound classification

In [14]:
```
DATA_PATH

def load_data(data_path):

    with open(data_path, "r") as fp:
        data = json.load(fp)

    X = np.array(data["mfcc"])
    y = np.array(data["labels"])
    return X, y


def plot_history(history):

    fig, axs = plt.subplots(2)

    # create accuracy sublpot
    axs[0].plot(history.history["accuracy"], label="train accuracy")
    axs[0].plot(history.history["val_accuracy"], label="test accuracy")
    axs[0].set_ylabel("Accuracy")
    axs[0].legend(loc="lower right")
    axs[0].set_title("Accuracy eval")

    # create error sublpot
    axs[1].plot(history.history["loss"], label="train error")
    axs[1].plot(history.history["val_loss"], label="test error")
    axs[1].set_ylabel("Error")
    axs[1].set_xlabel("Epoch")
    axs[1].legend(loc="upper right")
    axs[1].set_title("Error eval")

    plt.show()


def prepare_datasets(test_size, validation_size):


    # load data
    X, y = load_data(DATA_PATH)

    # create train, validation and test split
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size)
    X_train, X_validation, y_train, y_validation = train_test_split(X_train, y_train, test_size=validation_size)

    # add an axis to input sets
    X_train = X_train[..., np.newaxis]
    X_validation = X_validation[..., np.newaxis]
    X_test = X_test[..., np.newaxis]

    return X_train, X_validation, X_test, y_train, y_validation, y_test


def build_model(input_shape):


    # build network topology
    model = keras.Sequential()

    # 1st conv layer
    model.add(keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=input_shape))
    model.add(keras.layers.MaxPooling2D((3, 3), strides=(2, 2), padding='same'))
    model.add(keras.layers.BatchNormalization())

    # 2nd conv layer
    model.add(keras.layers.Conv2D(32, (3, 3), activation='relu'))
    model.add(keras.layers.MaxPooling2D((3, 3), strides=(2, 2), padding='same'))
    model.add(keras.layers.BatchNormalization())

    # 3rd conv layer
    model.add(keras.layers.Conv2D(32, (2, 2), activation='relu'))
    model.add(keras.layers.MaxPooling2D((2, 2), strides=(2, 2), padding='same'))
    model.add(keras.layers.BatchNormalization())

    # flatten output and feed it into dense layer
    model.add(keras.layers.Flatten())
    model.add(keras.layers.Dense(64, activation='relu'))
    model.add(keras.layers.Dropout(0.0))

    # output layer
    model.add(keras.layers.Dense(10, activation='softmax'))

    return model
```

jupyter major project Last Checkpoint: Yesterday at 7:48 PM (autosaved)

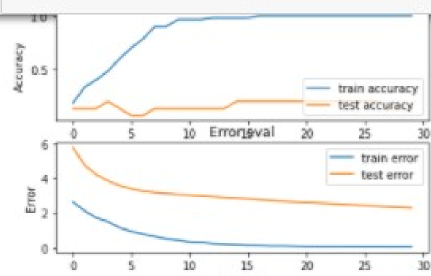
Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run Code

```python
    model.add(keras.layers.Conv2D(32, (2, 2), activation='relu'))
    model.add(keras.layers.MaxPooling2D((2, 2), strides=(2, 2), padding='same'))
    model.add(keras.layers.BatchNormalization())

    # flatten output and feed it into dense layer
    model.add(keras.layers.Flatten())
    model.add(keras.layers.Dense(64, activation='relu'))
    model.add(keras.layers.Dropout(0.0))

    # output layer
    model.add(keras.layers.Dense(10, activation='softmax'))

    return model


def predict(model, X, y):


    # add a dimension to input data for sample - model.predict() expects a 4d array in this case
    X = X[np.newaxis, ...] # array shape (1, 130, 13, 1)

    # perform prediction
    prediction = model.predict(X)

    # get index with max value
    predicted_index = np.argmax(prediction, axis=1)

    print("Target: {}, Predicted label: {}".format(y, predicted_index))


if __name__ == "__main__":

    # get train, validation, test splits
    X_train, X_validation, X_test, y_train, y_validation, y_test = prepare_datasets(0.25, 0.2)

    # create network
    input_shape = (X_train.shape[1], X_train.shape[2], 1)
    model = build_model(input_shape)

    # compile model
    optimiser = keras.optimizers.Adam(learning_rate=0.0001)
    model.compile(optimizer=optimiser,
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])

    model.summary()

    # train model
    history = model.fit(X_train, y_train, validation_data=(X_validation, y_validation), batch_size=32, epochs=30)

    # plot accuracy/error for training and validation
    plot_history(history)

    # evaluate model on test set
    test_loss, test_acc = model.evaluate(X_test, y_test, verbose=2)
    print('\nTest accuracy:', test_acc)

    # pick a sample to predict from the test set
    X_to_predict = X_test[10]
    y_to_predict = y_test[10]

    # predict sample
    predict(model, X_to_predict, y_to_predict)
```




```
1/1 - 0s - loss: 2.3101 - accuracy: 0.2000

Test accuracy: 0.20000000298023224
Target: 8, Predicted label: [9]
```

jupyter major project Last Checkpoint: Yesterday at 7:48 PM (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

Run Code

CNN model for bird sound classification using big data

In [15]:

```python
DATA_PATH ="bird_sound_big_dataset.json"


def load_data(data_path):

    with open(data_path, "r") as fp:
        data = json.load(fp)

    X = np.array(data["mfcc"])
    y = np.array(data["labels"])
    return X, y


def plot_history(history):

    fig, axs = plt.subplots(2)

    # create accuracy sublpot
    axs[0].plot(history.history["accuracy"], label="train accuracy")
    axs[0].plot(history.history["val_accuracy"], label="test accuracy")
    axs[0].set_ylabel("Accuracy")
    axs[0].legend(loc="lower right")
    axs[0].set_title("Accuracy eval")

    # create error sublpot
    axs[1].plot(history.history["loss"], label="train error")
    axs[1].plot(history.history["val_loss"], label="test error")
    axs[1].set_ylabel("Error")
    axs[1].set_xlabel("Epoch")
    axs[1].legend(loc="upper right")
    axs[1].set_title("Error eval")

    plt.show()


def prepare_datasets(test_size, validation_size):


    # load data
    X, y = load_data(DATA_PATH)

    # create train, validation and test split
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size)
    X_train, X_validation, y_train, y_validation = train_test_split(X_train, y_train, test_size=validation_size)

    # add an axis to input sets
    X_train = X_train[..., np.newaxis]
    X_validation = X_validation[..., np.newaxis]
    X_test = X_test[..., np.newaxis]

    return X_train, X_validation, X_test, y_train, y_validation, y_test


def build_model(input_shape):


    # build network topology
    model = keras.Sequential()

    # 1st conv layer
    model.add(keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=input_shape))
    model.add(keras.layers.MaxPooling2D((3, 3), strides=(2, 2), padding='same'))
    model.add(keras.layers.BatchNormalization())

    # 2nd conv layer
    model.add(keras.layers.Conv2D(32, (3, 3), activation='relu'))
    model.add(keras.layers.MaxPooling2D((3, 3), strides=(2, 2), padding='same'))
    model.add(keras.layers.BatchNormalization())

    # 3rd conv layer
    model.add(keras.layers.Conv2D(32, (2, 2), activation='relu'))
    model.add(keras.layers.MaxPooling2D((2, 2), strides=(2, 2), padding='same'))
    model.add(keras.layers.BatchNormalization())

    # flatten output and feed it into dense layer
    model.add(keras.layers.Flatten())
    model.add(keras.layers.Dense(64, activation='relu'))
    model.add(keras.layers.Dropout(0.0))

    # output layer
```

jupyter major project Last Checkpoint: Yesterday at 7:48 PM (autosaved)

Logout

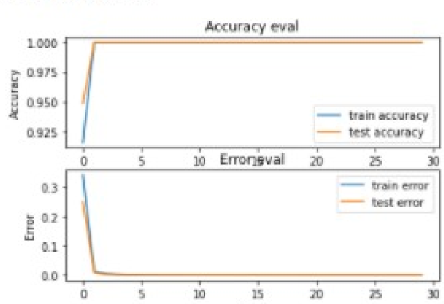
File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run Code

```
    model.add(keras.layers.MaxPooling2D((2, 2), strides=(2, 2), padding='same'))
    model.add(keras.layers.BatchNormalization())

    # flatten output and feed it into dense layer
    model.add(keras.layers.Flatten())
    model.add(keras.layers.Dense(64, activation='relu'))
    model.add(keras.layers.Dropout(0.0))

    # output layer
    model.add(keras.layers.Dense(10, activation='softmax'))

    return model


def predict(model, X, y):


    # add a dimension to input data for sample - model.predict() expects a 4d array in this case
    X = X[np.newaxis, ...] # array shape (1, 130, 13, 1)

    # perform prediction
    prediction = model.predict(X)

    # get index with max value
    predicted_index = np.argmax(prediction, axis=1)

    print("Target: {}, Predicted label: {}".format(y, predicted_index))


if __name__ == "__main__":

    # get train, validation, test splits
    X_train, X_validation, X_test, y_train, y_validation, y_test = prepare_datasets(0.25, 0.2)

    # create network
    input_shape = (X_train.shape[1], X_train.shape[2], 1)
    model = build_model(input_shape)

    # compile model
    optimiser = keras.optimizers.Adam(learning_rate=0.0001)
    model.compile(optimizer=optimiser,
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])

    model.summary()

    # train model
    history = model.fit(X_train, y_train, validation_data=(X_validation, y_validation), batch_size=32, epochs=30)

    # plot accuracy/error for training and validation
    plot_history(history)

    # evaluate model on test set
    test_loss, test_acc = model.evaluate(X_test, y_test, verbose=2)
    print('\nTest accuracy:', test_acc)

    # pick a sample to predict from the test set
    X_to_predict = X_test[10]
    y_to_predict = y_test[10]

    # predict sample
    predict(model, X_to_predict, y_to_predict)
```

```
Epoch 30/30
186/186 [==============================] - 51s 272ms/step - loss: 3.2488e-05 - accuracy: 1.0000 - val_loss: 6.7902e-06 - val
_accuracy: 1.0000
```



In [ ]:

8 Results and Outcomes

We calculated accuracies and draw the graph for each of the algorithms and the following insights were obtained.

Algorithm	Accuracy
ANN (small dataset)	78.63 %
ANN (big dataset)	63.47%
CNN (small dataset)	58.43%
CNN (big datasset)	100%

The following insights are gained from the above results:

- while ANN give less accuracy because of overfitting of dataset .
- after removing the overfitting ANN give 78-83% accuracy .
- CNN give the very less accuracy because of overffing of dataset and we increase the data set .
- After increasing the data set CNN give the bet accuracy which is 100%.

9 Tools and technologies used

10.1Software requirement

The various tools and technologies to be used are as follows:

I) Python Libraries to implement Machine Learning Models -

- Librosa - librosa is a software library written for the Puython progarmming language for audio feature extaction .

• Pandas - pandas is a software library written for the Python programming language for data manipulation and analysis.

• NumPy - NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

• Scikit learn - Scikit is an open source Python library that implements a range of machine learning, pre-processing, cross-validation and visualization algorithms using a unified interface.

• To build the machine learning models we used The Jupyter Notebook. Jupyter Notebook is an open-source web application that can be used to implement statistical modelling, data visualization, machine learning etc.

10.2Hardware requirement

Working Computer system
Processor Requirement: Intel I3 core and above.
Memory Requirement: 4GB and above.

10 Conclusion

In this projecte presented a method of classifying real time recordings of bird sound. We have compared machine learning techniques and the number of recordings for improving the performance of the classification tasks. Performance of the proposed bird identification system still leaves scope for improvement. Based on the results of this study, bird's species may be identified with an accuracy of 78% and 100%using ANN and CNN respectively. For future works, this approach can be extended to large number of bird species. Research can be carried out to investigate the audio recordings in noisy environments. This can also be extended to study other important animal species in understanding complex and sensitive ecosystems.

11. References

1. Bioacoustic detection with wavelet-conditioned convolutional neural networks, Ivan Kiskin.
2. Large-Scale Bird Sound Classification using Convolutional Neural Networks by Stefan Kahl , Technische Universität Chemnitz, Straße der Nationen 62, 09111 Chemnitz, Germany.
3. Audio Classification of Bird Species: a Statistical Manifold Approach by Forrest Briggs, Raviv Raich, and Xiaoli Z. Fern School of EECS, Oregon State University, Corvallis.
4. https://towardsdatascience.com/sound-based-bird-classification-965d0ecacb2b
5. https://towardsdatascience.com/how-to-apply-machine-learning-and-deep?learning-methods-to-audio-analysis-615e286fcbb