CS6005 Deep Learning Techniques
Course Project

Bird Species Identification using Audio Signal Processing and Deep Learning

TEAM MEMBERS:

1. NANDA KUMAR R - 2019103545 (Q BATCH)
2. KIRAN GEORGE GAURDIAN - 2019103029 (Q BATCH)

Table of Contents

1. Abstract

An important task in ecology is the monitoring of animal population for a variety of reasons: biological interest, research purposes, conservation purposes or game management. Identification of animal species based on their sounds, calls or songs is one method of wildlife monitoring that helps in better understanding their breeding behaviour, biodiversity and population dynamics. In this regard, birds are particularly useful since they are acoustically active and respond quickly to changes in their environment. Thus, large scale, accurate bird recognition is needed by bird watchers, conservation organisations, ecology consultants, and ornithologists.

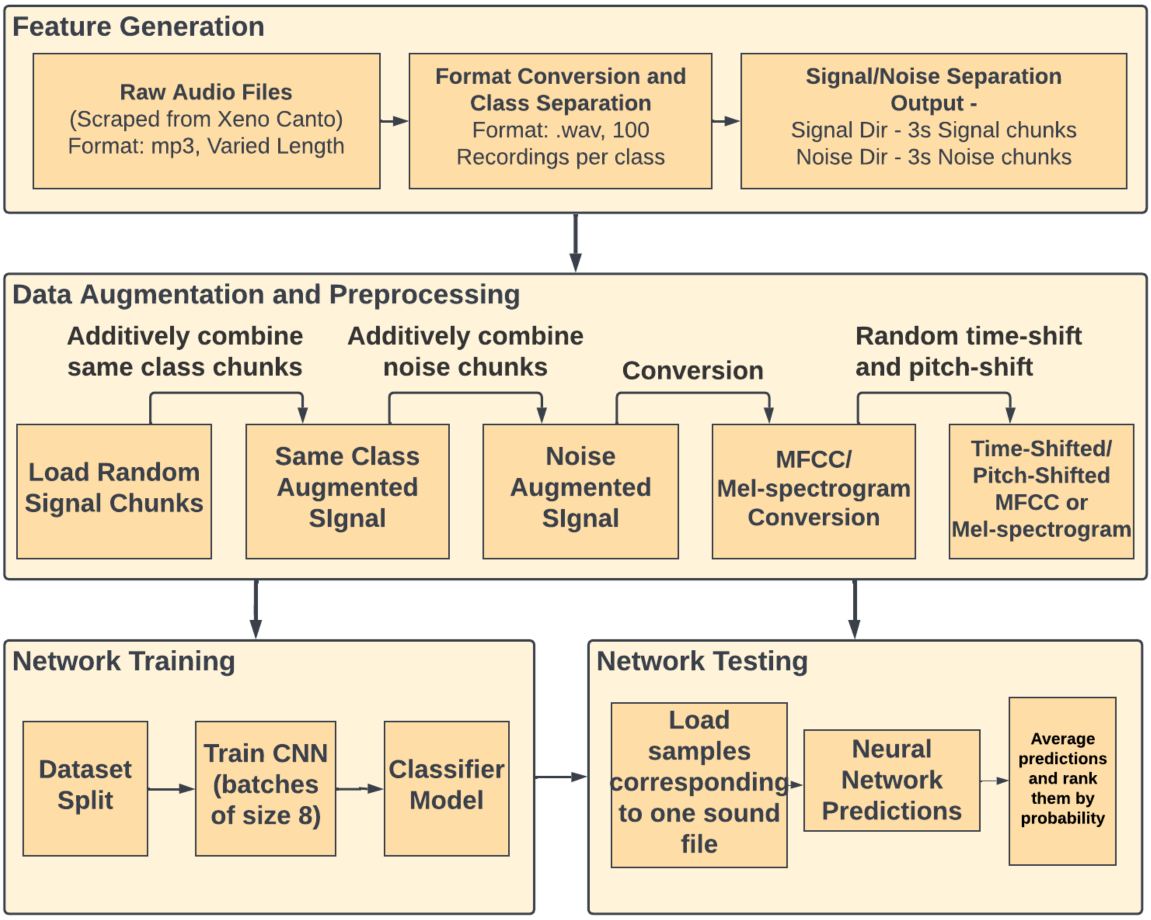
Bird species classification can be done manually by domain experts; however, with growing amounts of data, this rapidly becomes a tedious and time-consuming process. Therefore automatic tools which can aid in this process are needed. Several bird species identification challenges such as the BirdCLEF, the Neural Information Processing Scaled for Bioacoustics (NIPS4B) 2013 have been held recently, with the goal of creating and evaluating such automatic classifiers on bird song recordings taken from the field. The main challenges that have made this task difficult to tackle are background noise, multiple birds singing at the same time, variable length of sound recordings and a large number of different species.

In this project we present a bird species classification method that addresses the above challenges by making use of audio signal processing and deep learning techniques. The most promising classification technique has proven to be deep convolutional neural networks. We used one of the largest publicly available dataset, the Xeno Canto collaborative database for training the model. The four main steps in developing the classification model are extraction of spectrograms from audio, dataset extension through extensive augmentation, finding the best architecture with respect to number of classes and sample count and finally, training the model on the augmented dataset. Lastly, we package the model into an application that can listen for bird calls and make species predictions in real-time.

2. Block diagram

There are 4 major modules -

1. Feature Generation
2. Data Augmentation
3. CNN Model Training
4. CNN Model Testing



3. Network Architecture

The CNN model was constructed using tensorflow/keras. TensorFlow is an open-sourced end-to-end platform, for multiple machine learning tasks, while Keras is a high-level neural network library that runs on top of TensorFlow.

The model architecture consists of 4 convolution layers with max pooling layers inserted between every two convolution layers. Batch normalisation was used to normalise inputs to a layer for each mini-batch.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 13, 517, 32)       832

 conv2d_5 (Conv2D)           (None, 13, 517, 32)       25632

 max_pooling2d_2 (MaxPooling  (None, 7, 259, 32)       0
 2D)

 batch_normalization_2 (Batc  (None, 7, 259, 32)       128
 hNormalization)

 conv2d_6 (Conv2D)           (None, 7, 259, 64)        18496

 conv2d_7 (Conv2D)           (None, 7, 259, 64)        36928

 max_pooling2d_3 (MaxPooling  (None, 4, 130, 64)       0
 2D)

 batch_normalization_3 (Batc  (None, 4, 130, 64)       256
 hNormalization)

 flatten_1 (Flatten)         (None, 33280)             0

 dense_2 (Dense)             (None, 64)                2129984

 dense_3 (Dense)             (None, 5)                 325

=================================================================
Total params: 2,212,581
Trainable params: 2,212,389
Non-trainable params: 192
```

CNN Model Architecture

Model Build Function -

```
def build_model1(input_shape):

  model = Sequential()

  model.add(Conv2D(32, kernel_size=5,input_shape=input_shape, activation =
```

```
'relu',padding="same"))
  model.add(Conv2D(32, kernel_size=5, activation = 'relu',padding="same"))

  model.add(MaxPool2D(2,2, padding="same"))
  model.add(BatchNormalization())

  model.add(Conv2D(64, kernel_size=3,activation = 'relu',padding="same"))
  model.add(Conv2D(64, kernel_size=3,activation = 'relu',padding="same"))
  model.add(MaxPool2D(2,2,padding="same"))
  model.add(BatchNormalization())

  model.add(Flatten())
  model.add(Dense(64, activation = "relu"))
  model.add(Dense(5, activation = "softmax"))

  return model
```

Model Creation -

```
model = build_model1(input_shape)

early_stopping = EarlyStopping(monitor='val_loss', patience=3, min_delta =
0.1, verbose = 1)

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

model.summary()
```

Hyperparameters -

- Optimiser - Adam (From Keras)
- Loss - Categorical Crossentropy
- Learning rate - 0.0001
- Batch size - 16
- Epochs - 8

4. Dataset description

The Dataset was scraped from Xeno-Canto. It is an online database that provides access to sound recordings of wild birds from around the world.

Source - https://xeno-canto.org/

Sound recordings of five bird species were collected. They are -

1. Cyanistes caeruleus - Eurasian blue tit
2. Emberiza citronella - Yellowhammer
3. Fringilla coelebs - Common chaffinch
4. Parus major - Great tit
5. Passer montanus - House sparrow

Source csv file -

Src	Species	Duration	Bitrate	Sampling Rate	Channels
https://www.xeno-canto.org/sounds/uploaded/AXTFCLIYDF/1906%20mp3.mp3	Parus major	51 s	bitrate: 320000 bps	audio sampling rate: 48000 Hz	number of channels: 2
https://www.xeno-canto.org/sounds/uploaded/AXTFCLIYDF/1891%20mp3.mp3	Parus major	36 s	bitrate: 320000 bps	audio sampling rate: 48000 Hz	number of channels: 2
https://www.xeno-canto.org/sounds/uploaded/OYTNUVPRZS/LS110558_mesange_c	Parus major	39 s	bitrate: 96000 bps	audio sampling rate: 44100 Hz	number of channels: 2
https://www.xeno-canto.org/sounds/uploaded/QNKACACJYP/song1%20Erisk%2020	Parus major	44 s	bitrate: 128000 bps	audio sampling rate: 44100 Hz	number of channels: 1
https://www.xeno-canto.org/sounds/uploaded/VXZDHTKCBO/GreatTitay.mp3	Parus major	33 s	bitrate: 192000 bps	audio sampling rate: 44100 Hz	number of channels: 2
https://www.xeno-canto.org/sounds/uploaded/VXZDHTKCBO/GreattitKkoy.mp3	Parus major	23 s	bitrate: 192000 bps	audio sampling rate: 44100 Hz	number of channels: 2
https://www.xeno-canto.org/sounds/uploaded/PNYKOPBQBQ/PMAJ07h27m33s15ap	Parus major	28 s	bitrate: 128000 bps	audio sampling rate: 48000 Hz	number of channels: 2
https://www.xeno-canto.org/sounds/uploaded/PNYKOPBQBQ/PMAJ09h15m53s07a	Parus major	54 s	bitrate: 128000 bps	audio sampling rate: 48000 Hz	number of channels: 2
https://www.xeno-canto.org/sounds/uploaded/AXTFCLIYDF/DM551838.mp3	Parus major	77 s	bitrate: 256000 bps	audio sampling rate: 48000 Hz	number of channels: 2
https://www.xeno-canto.org/sounds/uploaded/WYRGBDXALK/mesange%20charbonn	Parus major	5 s	bitrate: 158204 bps	audio sampling rate: 48000 Hz	number of channels: 2
https://www.xeno-canto.org/sounds/uploaded/SEIQHUBHAF/Great%20Tit%20%282%	Parus major	14 s	bitrate: 256000 bps	audio sampling rate: 48000 Hz	number of channels: 2
https://www.xeno-canto.org/sounds/uploaded/SEIQHUBHAF/Great%20Tit.mp3	Parus major	24 s	bitrate: 256000 bps	audio sampling rate: 48000 Hz	number of channels: 2
https://www.xeno-canto.org/sounds/uploaded/TLPLNAINFU/kjottmeis220312GOD-02	Parus major	51 s	bitrate: 128000 bps	audio sampling rate: 48000 Hz	number of channels: 2
https://www.xeno-canto.org/sounds/uploaded/AXTFCLIYDF/1737%20mp3.mp3	Parus major	63 s	bitrate: 320000 bps	audio sampling rate: 48000 Hz	number of channels: 2
https://www.xeno-canto.org/sounds/uploaded/HBPYQXTJEV/2012_03_10_Parus_ma	Parus major	107 s	bitrate: 128000 bps	audio sampling rate: 44100 Hz	number of channels: 2
https://www.xeno-canto.org/sounds/uploaded/HBPYQXTJEV/2012_03_10_Parus_ma	Parus major	75 s	bitrate: 128000 bps	audio sampling rate: 44100 Hz	number of channels: 2
https://www.xeno-canto.org/sounds/uploaded/HBPYQXTJEV/2012_03_10_Parus_ma	Parus major	23 s	bitrate: 128000 bps	audio sampling rate: 44100 Hz	number of channels: 2
https://www.xeno-canto.org/sounds/uploaded/HBPYQXTJEV/2012_03_10_Parus_ma	Parus major	25 s	bitrate: 128000 bps	audio sampling rate: 44100 Hz	number of channels: 2
https://www.xeno-canto.org/sounds/uploaded/RKAYEOOLQW/120306_08_GreatTit_c	Parus major	93 s	bitrate: 192000 bps	audio sampling rate: 44100 Hz	number of channels: 2
https://www.xeno-canto.org/sounds/uploaded/RKAYEOOLQW/120306_04_GreatTit_c	Parus major	23 s	bitrate: 320000 bps	audio sampling rate: 44100 Hz	number of channels: 2
https://www.xeno-canto.org/sounds/uploaded/TLPLNAINFU/kjottmeissang110312-02	Parus major	19 s	bitrate: 128000 bps	audio sampling rate: 44100 Hz	number of channels: 2
https://www.xeno-canto.org/sounds/uploaded/TLPLNAINFU/kjottmeis_cal110312l-02	Parus major	22 s	bitrate: 128000 bps	audio sampling rate: 44100 Hz	number of channels: 2

For each of the listed bird species, 100 recordings of varied length were scraped as mp3 files. They were then converted into wav files which is an uncompressed audio format suitable for audio processing.

5. Implementation

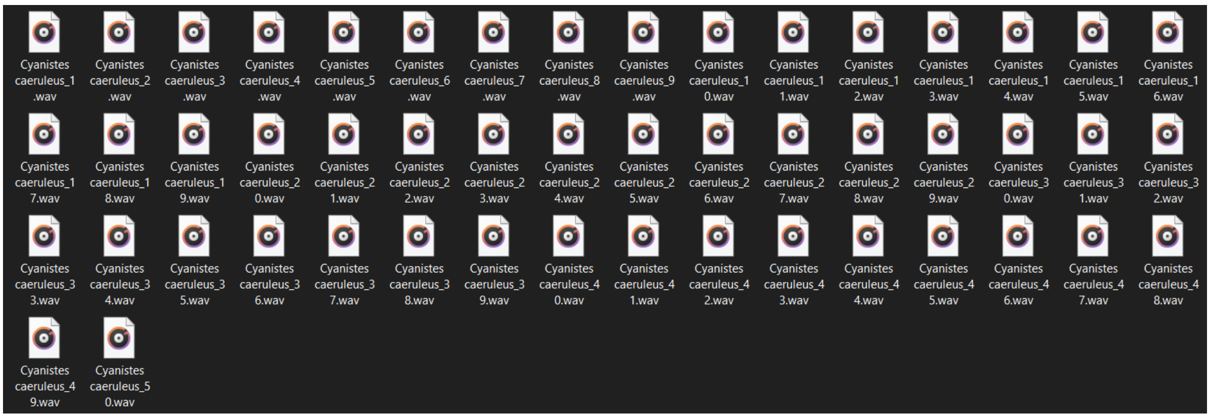
5.1. Feature Generation

The sound recordings were scraped from Xeno-canto. The scraped files are in mp3 format and need to be converted to wav format.

```
#url downloading
path = "mp3Data/"
def downloader(url, species, count):
    r = requests.get(url, allow_redirects=True)
    open(path + '{0}/{1}_{2}.mp3'.format(species, species, count),
'wb').write(r.content)
```

```
#Species download
def specDwnld():
    i = 1 #Initialise starting count
    for row in df.itertuples():
        if not os.path.exists(os.path.join(path,row[2])):
            os.makedirs(os.path.join(path,row[2]))
            i = 1
        downloader(row[1], row[2], i)
        i+=1

#convert mp3 to wav files
def wavConverter():
    audio_folds = os.listdir(path)
    for folder in audio_folds:
        path1 = os.path.join(path,folder)
        audio_files = os.listdir(path1)

        if not os.path.exists(os.path.join('AudioFiles_wav',folder)):
            os.makedirs(os.path.join('AudioFiles_wav',folder))

        for file in audio_files:
            name, ext = os.path.splitext(file)
            mp3_sound = AudioSegment.from_file(os.path.join(path1,file))
            mp3_sound.export('AudioFiles_wav/{0}/{1}.wav'.format(folder,
name), format="wav")
    shutil.rmtree(path)
```

Output folder for Cyanistes caeruleus -

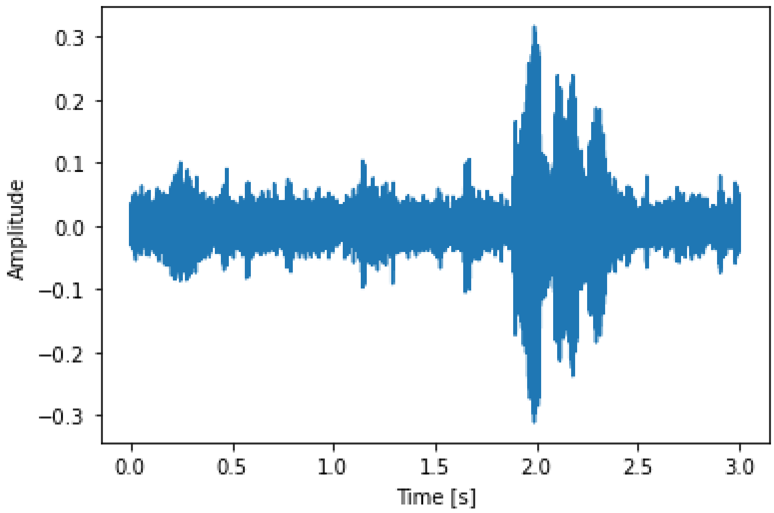


5.2. Data Preprocessing

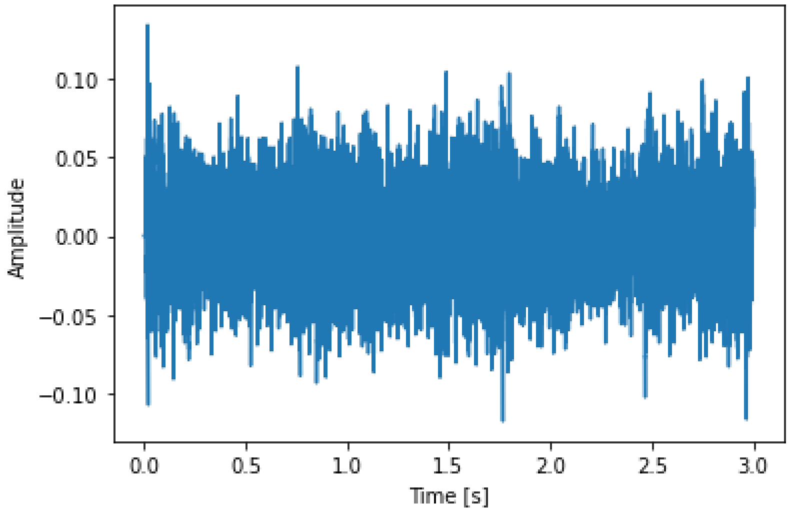
The wav files need to be separated into signal and noise parts and then equally split into segments of 3 seconds.

```
def preprocess_wave(wave, fs):
    """ Preprocess a signal by computing the noise and signal mask of the
    signal, and extracting each part from the signal
    """


sig_stft=librosa.stft(wave,n_fft=512,hop_length=128,window="hann",win_length
=512)
    Sxx=np.abs(sig_stft)**2

    #computing mask
    n_mask = compute_noise_mask(Sxx)
    s_mask = compute_signal_mask(Sxx)

    #reshape the masks
    n_mask_scaled = reshape_binary_mask(n_mask, wave.shape[0])
    s_mask_scaled = reshape_binary_mask(s_mask, wave.shape[0])

    #apply mask and extract respective time series
    signal_wave = extract_masked_part_from_wave(s_mask_scaled, wave)
    noise_wave = extract_masked_part_from_wave(n_mask_scaled, wave)

    return signal_wave, noise_wave
```

Generated signal file -



Generated noise file -

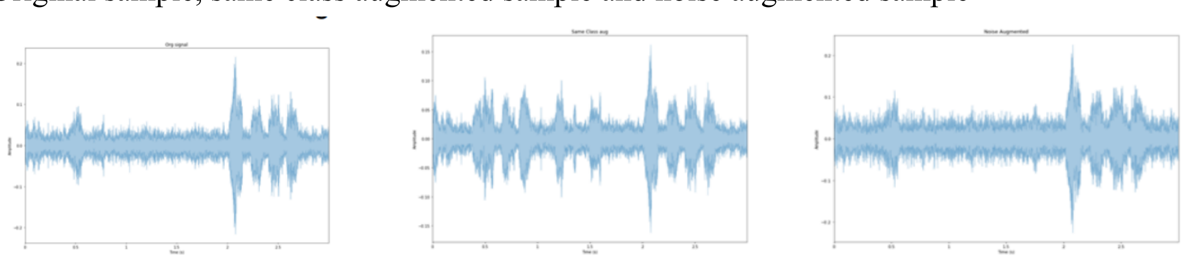


5.3. Data Augmentation

Data augmentation is a way of increasing the number of training samples in a data set by augmenting the training data. In order to improve the convergence rate of the neural network, the training samples are augmented by additively combining each sample with another same class sample, which lets the network see more relevant data at once. The samples are also additively combined with three random noise segments, to make the network more noise invariant, and therefore generalise better.

The time shift augmentation is done by splitting a spectrogram sample into two parts, along the time axis, and then placing the second part before the first. That is, a wrap-around shift in the time domain. The pitch shift augmentation is done in a similar way, but in the frequency domain (vertically), and the shift is only around 5%.

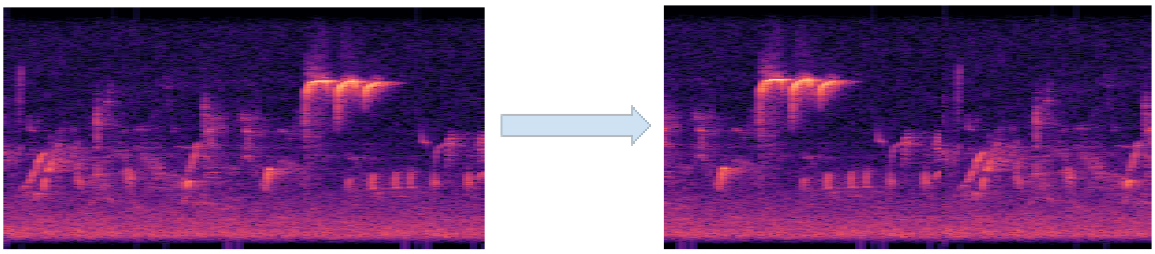
Original sample, same class augmented sample and noise augmented sample -



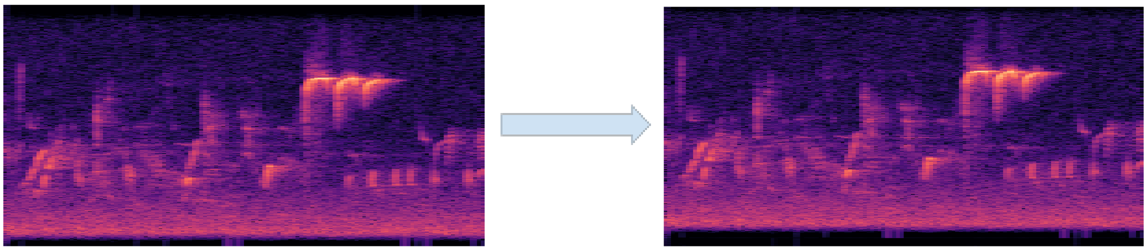
Augmentation code -

```python
def time_shift_spectrogram(spectrogram):

    nb_cols = spectrogram.shape[1]
    nb_shifts = np.random.randint(0, nb_cols)

    return np.roll(spectrogram, nb_shifts, axis=1)

def pitch_shift_spectrogram(spectrogram):

    nb_cols = spectrogram.shape[0]
    max_shifts = nb_cols//20
    nb_shifts = np.random.randint(-max_shifts, max_shifts)

    return np.roll(spectrogram, nb_shifts, axis=0)

def same_class_augmentation(wave, class_dir):

    sig_paths = glob.glob(os.path.join(class_dir, "*.wav"))
    aug_sig_path = random.choice(sig_paths)
    aug_sig, sr = load(aug_sig_path)

    alpha = np.random.rand()
    wave = (1.0-alpha)*wave + alpha*aug_sig

    return wave

def noise_augmentation(wave, noise_files):

    nb_noise_segments = 3
    aug_noise_files = []
    for i in range(nb_noise_segments):
        aug_noise_files.append(random.choice(noise_files))

    dampening_factor = 0.4
    for aug_noise_path in aug_noise_files:
        aug_noise, sr = load(aug_noise_path)
        wave = wave + aug_noise*dampening_factor

    return wave
```

Time shifted spectrogram -



Pitch shifted spectrogram -



5.4. CNN Training

Loading dataset and splitting it into train and valid set -

```
data =  np.load('/content/gdrive/My Drive/Bird Species 
Classifier/Train2.npz')
data = data['arr_0']

labels =  np.load('/content/gdrive/My Drive/Bird Species 
Classifier/Train_label2.npz')
labels = labels['arr_0']

X_train, X_test, y_train, y_test = train_test_split(data, labels, 
test_size=.2, random_state=0, stratify=labels)
```

Model building -

```
def build_model1(input_shape):

  model = Sequential()

  model.add(Conv2D(32, kernel_size=5,input_shape=input_shape, activation = 
'relu',padding="same"))
  model.add(Conv2D(32, kernel_size=5, activation = 'relu',padding="same"))

  model.add(MaxPool2D(2,2, padding="same"))
  model.add(BatchNormalization())

  model.add(Conv2D(64, kernel_size=3,activation = 'relu',padding="same"))
  model.add(Conv2D(64, kernel_size=3,activation = 'relu',padding="same"))
  model.add(MaxPool2D(2,2,padding="same"))
  model.add(BatchNormalization())

  model.add(Flatten())
  model.add(Dense(64, activation = "relu"))
  model.add(Dense(5, activation = "softmax"))

  return model
```

Model training -

```
model = build_model1(input_shape)

early_stopping = EarlyStopping(monitor='val_loss', patience=3, min_delta =
0.1, verbose = 1)

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

model.summary()
```

```
history = model.fit(X_train, y_train, validation_data=(X_test, y_test), batch_size=16, epochs=8)
```

```
Epoch 1/8
195/195 [==============================] - 5s 18ms/step - loss: 1.2262 - accuracy: 0.5238 - val_loss: 1.1179 - val_accuracy: 0.6152
Epoch 2/8
195/195 [==============================] - 3s 15ms/step - loss: 0.8316 - accuracy: 0.6821 - val_loss: 0.9037 - val_accuracy: 0.6744
Epoch 3/8
195/195 [==============================] - 3s 15ms/step - loss: 0.6214 - accuracy: 0.7735 - val_loss: 0.7398 - val_accuracy: 0.7297
Epoch 4/8
195/195 [==============================] - 3s 15ms/step - loss: 0.4630 - accuracy: 0.8414 - val_loss: 0.6940 - val_accuracy: 0.7439
Epoch 5/8
195/195 [==============================] - 3s 15ms/step - loss: 0.3291 - accuracy: 0.8893 - val_loss: 0.6400 - val_accuracy: 0.7658
Epoch 6/8
195/195 [==============================] - 3s 15ms/step - loss: 0.2159 - accuracy: 0.9373 - val_loss: 0.6311 - val_accuracy: 0.7812
Epoch 7/8
195/195 [==============================] - 3s 15ms/step - loss: 0.1224 - accuracy: 0.9717 - val_loss: 0.5947 - val_accuracy: 0.7748
Epoch 8/8
195/195 [==============================] - 3s 15ms/step - loss: 0.0670 - accuracy: 0.9900 - val_loss: 0.6087 - val_accuracy: 0.7889
```

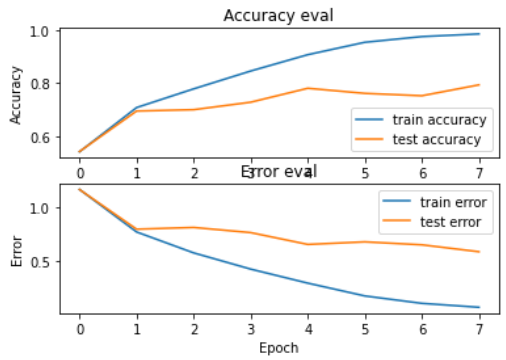
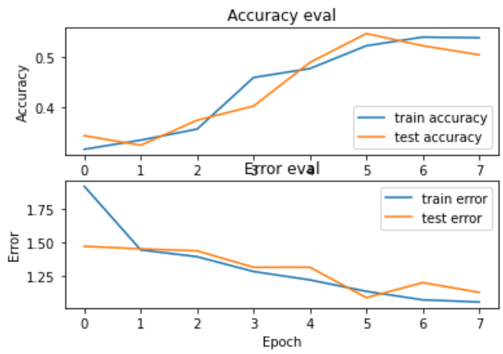
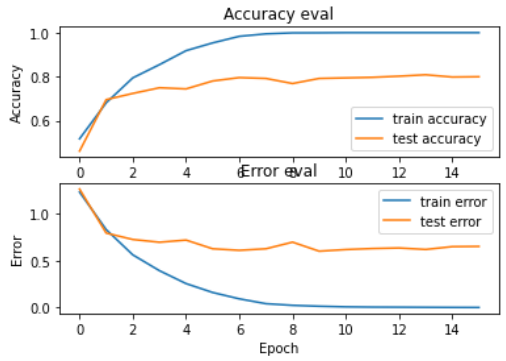
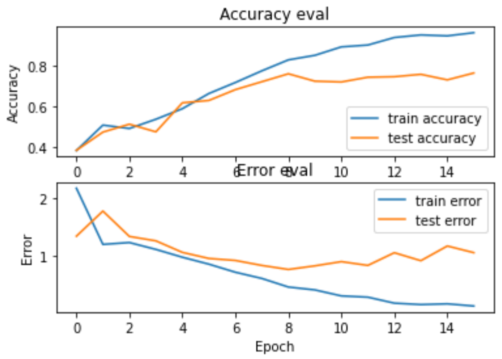
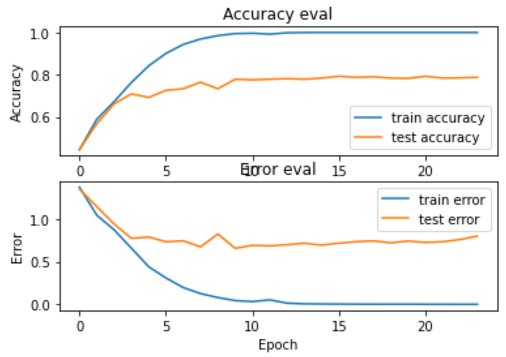
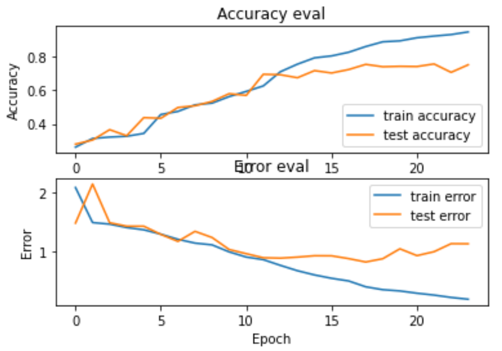
6. Result analysis

Model evaluation -

```
test_error,test_accuracy=model.evaluate(X_test,y_test,verbose=2)
print("Accuracy on original test set is:{}".format(test_accuracy))

pred = np.argmax(model.predict(X_test), axis=-1)
```

```
25/25 - 8s - loss: 0.5945 - accuracy: 0.7954 - 8s/epoch - 301ms/step
Accuracy on original test set is:0.7953668236732483
```

Accuracy and Error Plot -

<table>
<tr><th></th><th>Learning rate = 0.0001</th><th>Learning rate = 0.001</th></tr>
<tr><td>Epochs = 8</td><td>
</td><td>
</td></tr>
<tr><td>Epochs = 16</td><td>
</td><td>
</td></tr>
<tr><td>Epochs = 24</td><td>
</td><td>
</td></tr>
</table>

Confusion matrix - a summary of prediction results

	0	1	2	3	4
0	86	2	16	18	3
1	5	87	8	8	7
2	0	0	187	6	11
3	12	5	15	105	11
4	5	2	22	3	153

Classification metrics -

	precision	recall	f1-score	support
0	0.80	0.69	0.74	125
1	0.91	0.76	0.82	115
2	0.75	0.92	0.83	204
3	0.75	0.71	0.73	148
4	0.83	0.83	0.83	185
accuracy			0.80	777
macro avg	0.81	0.78	0.79	777
weighted avg	0.80	0.80	0.79	777

7. References

1. Martinsson, John. “Bird Species Identification using Convolutional Neural Networks.” (2017).
2. Elias Sprengel, Martin Jaggi, Yannic Kilcher and Thomas Hofmann. “Audio Based Bird Species Identification using Deep Learning Techniques.” (2016).
3. https://xeno-canto.org/
4. https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator
5. Yann LeCun, Yoshua Bengio and Hinton Geoffrey.” Deep learning”.(Nature Methods 2015)