Title Of Dissertation

Binary Classification of Emotions Using Hybrid Model on Spectrogram Images formed on Beta And Gamma Wave Frequency of GAMEEMO - EEG Signal Dataset

Under Dr. Manjari Gupta

Name - Priti
Course - M.Sc. Mathematics and Computing
Semester - 4 th
Roll No. -21419MAC032
DST-CIMS
Banaras Hindu University

Abstract: Electroencephalography (EEG)-based emotion recognition attempts to detect the affective states of humans directly via spontaneous EEG signals, bypassing the peripheral nervous

system. In this project, we explore various transfer learning techniques for EEG-based emotion recognition, and focus on the three research tasks outlined as follows:

1. Converting EGG signal to Spectrogram Images using stft value of EEG signal Dataset
2. Classification by using Transfer learning Model trained on these spectrogram images.
3. Using Hybrid Model which has pretrained Transfer learning model as feature extractor and a binary classifier.

We propose several novel methods in this thesis to address the three research gaps and validate our proposed methods by experiments. Extensive comparisons between our methods justify the advantages of our methods.

Electroencephalogram (EEG) based emotional analysis has been employed in medical science, security and human-computer interaction with good success. In the recent past, deep learning-based approaches have significantly improved the classification accuracy when compared to classical signal processing and machine learning based frameworks.

With the emotion recognition algorithms, EEG-enabled human-computer interfaces can be adapted to the user's internal feelings and can be driven by the user's emotions. The affective interfaces can be applied in many applications such as 1) games where the flow can be changed according to user's emotions, 2) medical applications to monitor emotions of the patient, 3) neuromarketing, 4) human factors evaluation, etc.

INDEX:

Introduction

It begins with the background introduction to and the motivation for EEG-based emotion recognition.

Background

Electroencephalogram (EEG) is the recording of the electric potential of human brain. The first effort to capture and record human EEG was by a German physiologist and psychiatrist Hans Berger in 1924 [1]. Since then, EEG was gradually adopted in clinical environment to facilitate the diagnosis of certain brain diseases, such as Epileptic Seizure, Attention Deficit Hyperactive Disorder (ADHD) and Alzheimer's Disease etc. For over decades, EEG has been limited to medical environment, as the EEG device was costly and immobile, and the acquisition of EEG required professional help. However, in recent years, the advancement of manufacturing technology has introduced to the market new EEG devices which are wearable, portable, wireless and easy to use. Such new devices greatly simplify the acquisition of EEG. Subjects can easily access their EEG even without the help of medical professional. This paves the way for the application of EEG to expand from medical use to personal entertainment use. On the other hand, technological alternatives to EEG such as Functional Magnetic Resonance Imaging (FMRI), Magnetoencephalography (MEG) and functional Near-Infrared Spectroscopy (fNIR) remains expensive and limited to medical use.
Apart from its key role as a diagnostic tool for the brain in the healthcare sector, the current applications of EEG include but are not limited to a) entertainment, b) cognitive training, c) rehabilitation, d) human factors investigation, e) marketing, and f) brain-computer interfaces. For example, EEG-driven games have been developed and distributed by companies like Emotiv and NeuroSky to entertain the users. EEG-based neurofeedback training can be used to improve the cognitive ability of healthy subjects and help to recover part of the brain functions for patients who suffer from stroke attack [4-6] or substance addiction. Researchers have also used EEG to analyze human factors for workplace optimization.

Besides, the high temporal resolution provided by EEG device makes it possible to monitor the user's emotion in real time. In the case when real-time emotion recognition is necessary, EEG-based emotion recognition may be preferable.
EEG feature extraction is one key step to successful EEG-based emotion recognition. For a long time, affective EEG features have been hand-engineered by domain experts. While pertinent to the

classification task, hand-engineered feature requires great amount of expertise. Recently, the renaissance of neural networks has reintroduced the possibility of unsupervised feature learning from raw data or relatively low-level feature. It has proven that deep neural networks can effectively learn discriminative features from raw image pixels or speech signals. We are motivated to research unsupervised EEG feature extraction given raw EEG signals or relatively low-level features.

Motivation

EEG feature extraction is one key step to successful EEG-based emotion recognition. For a long time, affective EEG features have been hand-engineered by domain experts. While pertinent to the classification task, hand-engineered feature requires great amount of expertise. Recently, the renaissance of neural networks has reintroduced the possibility of unsupervised feature learning from raw data or relatively low-level feature. It has proven that deep neural networks can effectively learn discriminative features from raw image pixels or speech signals. We are motivated to research unsupervised EEG feature extraction given raw EEG signals or relatively low-level features.

Objectives:

The aim of this project is to find the effectiveness of transfer learning models and hybrid models in predicting human emotions using images formed by Spectrogram .

The Objectives is to conduct a comprehensive review of the experiment on EEG-based emotion recognition and transfer learning techniques.To acquire and preprocess the GAMEEMO dataset, which includes EEG signals recorded from individuals during emotional experiences. To apply spectrogram analysis to the pre-processed EEG signals to extract frequency-domain features that capture the underlying patterns related to emotions.To design and implement a transfer learning framework that utilizes pre-trained models from related EEG-based emotion recognition tasks to enhance the prediction of human emotions in the GAMEEMO dataset. To evaluate the performance of the transfer learning approach by comparing it with Hybrid Model in which Transfer leaning model act as a feature extractor and a machine leaning classifier is used to predict the class. To assess the impact of different transfer learning strategies, such as fine-tuning and feature extraction, on the predictive performance of the emotion recognition model. To analyze the influence of various factors, including the size of the source dataset, the similarity of emotional contexts, and the transferability of learned representations, on the transfer learning performance.To interpret the results and provide insights into the potential advantages and limitations of transfer learning on GAMEEMO EEG signals using spectrogram analysis for emotion prediction. To discuss the implications of the findings for the field of affective computing, highlighting the potential applications and future research directions. To present recommendations for improving the accuracy and generalizability of EEG-based emotion recognition systems using transfer learning, based on the insights gained from this research.

It aims to investigte that how transfer learning techniques, combined with spectrogram analysis of EEG signals can enhance the prediction of human emotions in real-world applications.

Dataset

In this section, dataset GAMEMOO is used to evaluate the Pre trained Transfer learning models. Data processing methods and experiment results are presented.

Database for Emotion Recognition System - GAMEEMO

This dataset includes computer games-based EEG signals. They are collected from 28 different subjects with wearable and portable EEG device called 14 channel Emotiv Epoc+. Subjects played emotionally 4 different computer games (boring, calm, horror and funny) for 5 minutes and totally 20 minutes long EEG data available for each subject. Subjects rated each computer game based on the scale of arousal and valence by applying SAM form. We provided both raw and preprocessed EEG data with .csv and. mat format in our data repository. Each subject's rating score and SAM form are also available. The aim of this dataset is to provide an alternative data for emotion recognition process and show the performance of wearable EEG devices against traditional ones. In the main folder (GAMEEMO) researches will find 29 different folders (28 for subjects and 1 for gameplay). S01, S02, … represents the subjects who participated in the experiment. Gameplay folder shows the gameplay of each game. In each subjects folder, researchers will find preprocessed EEG data, raw EEG data csv and .mat format. Also SAM ratings are available with .pdf format. Games are represented as G1, G2, G3, and G4. G1 refers Game 1, G2 refers G2, and so on.

Methodology

The proposed methodology as shown in Fig. 1.

We first download the dataset GAMEEMO eeg signal.Then we preprocess the data to remove null or missing values and convert it into stft values . After that we use spectrogram to convert these eeg signals into Images . Then we train transfer learning models on these images and test the accuracy.

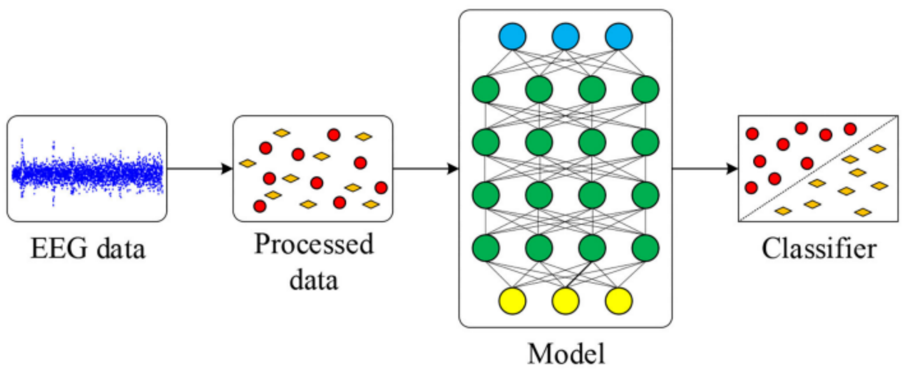


FIGURE 1 | Emotion classification procedure in this work.

Database for Emotion Recognition System - GAMEEMO

This dataset includes computer games-based EEG signals. They are collected from 28 different subjects with wearable and portable EEG device called 14 channel Emotiv Epoc+. Subjects played emotionally 4 different computer games (boring, calm, horror and funny) for 5 minutes and totally 20 minutes long EEG data available for each subject. Subjects rated each computer game based on the scale of arousal and valence by applying SAM form. We provided both raw and preprocessed EEG data with .csv and. mat format in our data repository. Each subject's rating score and SAM form are also available. The aim of this dataset is to provide an alternative data for emotion recognition process and show the performance of wearable EEG devices against traditional ones. In the main folder (GAMEEMO) researches will find 29 different folders (28 for subjects and 1 for gameplay). S01, S02, … represents the subjects who participated in the experiment. Gameplay folder shows the gameplay of each game. In each subjects folder, researchers will find preprocessed EEG data, raw EEG data csv and .mat format. Also SAM ratings are available with .pdf format. Games are represented as G1, G2, G3, and G4. G1 refers Game 1, G2 refers G2, and so on.

1.Preprocessing of Data:

The preprocessing phase is the first step, which has two objectives: remove artifacts and filter data such as blinking, heartbeats, and other effects [4,13]. These can be implemented in the acquisition (phase) through hardware or by software-based techniques. Some well-known techniques for preprocessing EEG signals are Common Spatial Patterns (CSP), Principle Component Analysis (PCA), Common Average Referencing (CAR), Surface Laplacian(SL), Independent Component Analysis(ICA), Adaptive Filtering, Digital Filter, and others.

(a)Remove a bad channel:

You can detect bad channels even before you have finished collecting the data. For example, if you know one of the channels was not functioning properly or if you noticed that one of the electrodes lost contact with the scalp during the experiment, you can mark it to be excluded from analysis.

The most common way of detecting bad channels after the data has been collected is by visualizing the raw data. Using MNE, this can be done

(b)**Filtering:**

When looking at the frequencies of a digital signal, whether it be audio, EEG, or otherwise, a popular thing to do is to *filter* certain frequencies, such that either some frequencies are removed, or possibly that some filters remain. There are a number of types of filters:

- *Low-pass filter:* ‘Low’ frequencies below a certain value are kept (they ‘pass’), while high frequencies are removed. This is also known as a high-cut filter. It may help to think of the audio version of this, which would be something that removed all the high notes from a sound.
- *High-pass filter (a.k.a Low-cut):* The same as above, but only high frequencies remain, and only those below a certain value are removed.
- *Band-pass filter:* Combining the two, this keeps only frequencies between a lower and upper bound. The opposite is a band-cut filter, which removes all frequencies in a particular range.

(c) STFT and Features :

The second phase in EEG signal analysis is feature extraction, where features of the signal are obtained using different signal processing techniques, such as Fast Fourier Transform (FFT), Principal Component Analysis (PCA), Wavelet Transformations (WT), Auto Regressive (AR), and others [4]. Analysis in time or frequency domain offers only time/frequency and amplitude information. The aforementioned techniques are commonly used in both domains. The Time-Frequency domain allows extracting information in the two domains simultaneously; EEG analysis is based on the timefrequency image processing technique or spectrogram, a technique commonly used in Short Time Fourier Transform, which maps the signal into a two-dimensional function of frequency and time [2]. This section shows a review of the literature on extracting features of EEG signals using STFT. EEG signals in time-frequency domain are retrieved using the spectrogram, by applying a Short Time Fourier Transform to the signal. STFT is applied to partition the EEG signal into several segments of short-time signals by shifting the time window with some overlapping [15], a process called windowing. Depending on the time windowing function w[n], a spectrogram is classified as a narrowband or wideband. If the time window is short, then its Fourier transform will be a wideband and a longer time returns a narrowband spectrogram [16]. The STFT general equation of a signal S is given by equation (1):

$$S(m,k)=\sum s(n+mN')w(n)e -j\ 2\pi\ N\ N-1\ nk\ n=0\ ,\ (1)$$

where k=[0:K] is the kth Fourier coefficient.

K=N/2 is the frequency index corresponding to the Nyquist frequency. S(m,k) indicates the m-index time-frequency(frame) spectrogram. N=window segment length.

N'=the shifting step of the time window

w(n)=windowing method of an N - point sequence.

N' should be smaller than N in order to produce an overlap between the time windows. S depends on the window function; in practice, different window shapes are used, such as: Symmetric, Unimodal and Gaussian

(d) Spectrogram:

The spectrogram contains a compromise between time resolution and frequency resolution: a large window provides less localization in time and more discrimination in frequency. The window obtains a time-slice of the signal, during which the spectral characteristics are nearly constant [16]; the obtained segments shift the time window with some overlapping. The spectrogram is defined as the magnitude of S(m,k), represented as A(m,k), as show in equation (2):

$$A(m,k)= 1/N\ |S(m,k)|\text{^}2\ .$$

The spectrogram resolution can be enhanced modifying the length window; a long window provides a better frequency resolution, but poor time resolution. A short window, however, provides better time resolution but poor frequency resolution. A good visualization in the spectrogram depends the selection of an appropriate window length and overlapping. Fig. 2 shows a spectrogram of a signal, which is a time-varying spectral representation of a signal. A spectrogram layout is usually as follows: the x-axis represents time, the y-axis represents frequency, and the third dimension is amplitude (spectral content) of a frequency-time pair, which is color coded. This three dimensional data can also create a 3D plot, where the intensity is represented as height on the z-axis but a 2D chart provides a better understanding.

(e) Frequency Bands:

An EEG Signal is commonly described in terms of brain activities; these are divided into frequency bands, which have a certain biological significance and different properties. Delta (δ) has the highest amplitude and the slowest wave; it is associated with deep sleep and waking states. Theta (θ) has an amplitude greater than 20 µV and a range of 4-7 Hz, and is linked with idling, creative inspiration, unconscious material, drowsiness, and deep meditation. Alpha (α) has an amplitude of 30-50 µV and a range of 8- 13 Hz, it is usually associated with relaxation, concentration, and sometimes with attention. Mu (µ), is found in the alpha wave and is regularly related to suppression, indicating that the motor neurons are working. Beta (β) is linked to alertness, thinking, and active concentration, and falls in the range between 12 and 30 Hz. Lastly, Gamma(γ) with a frequency greater than 30 Hz, is seen during short term memory matching of recognized objects, sounds, or tactile sensations .

2. Transfer Learning Models:

Transfer Learning is an advanced form of machine learning. It is much smarter and more effective compared to conventional machine learning.

Transfer learning is a research problem in machine learning that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem. For example, knowledge gained while learning to recognize cars could be applied when trying to recognize trucks.

Commonly used Transfer Learning Models:

Now, we will discuss the popular and commonly used models in transfer learning. Most of these models that we will discuss further are used in the task of image classification.

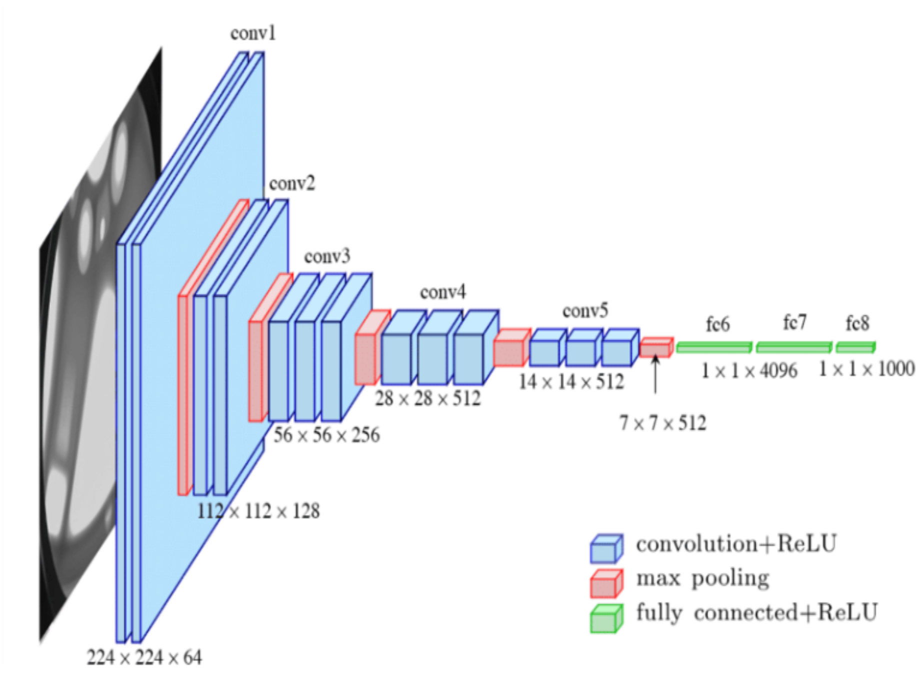
VGG16 Model for Transfer Learning:

The first VGG models were created by Karen Simonyan and Andrew Zisserman, and first presented in the paper Very Deep Convolutional Networks for Large-Scale Image Recognition in 2015. VGG16 has 16 layers with weights, and VGG19 has 19 layers with weights.

At the time, VGG models really came as a breakthrough, for a number of reasons. First, the authors were able to outperform the competition by a large amount on the Image Net Large-Scale Visual Recognition Challenge (ILSVRC). Then, they showed that, with transfer learning, their models generalize well to other image recognition tasks on smaller datasets achieving state-of-the art performance on these datasets as well. Finally, they made their best-performing networks available to the public for further research and practical applications.

The VGG architecture is quite straightforward and very similar to the original convolutional networks. The main idea behind VGG was to make the network deeper by stacking more convolutional layers. And this was made possible by restricting the size of the convolutional windows to only 3x3 pixels.

Consider the architecture of the VGG16 convolutional network, shown below.



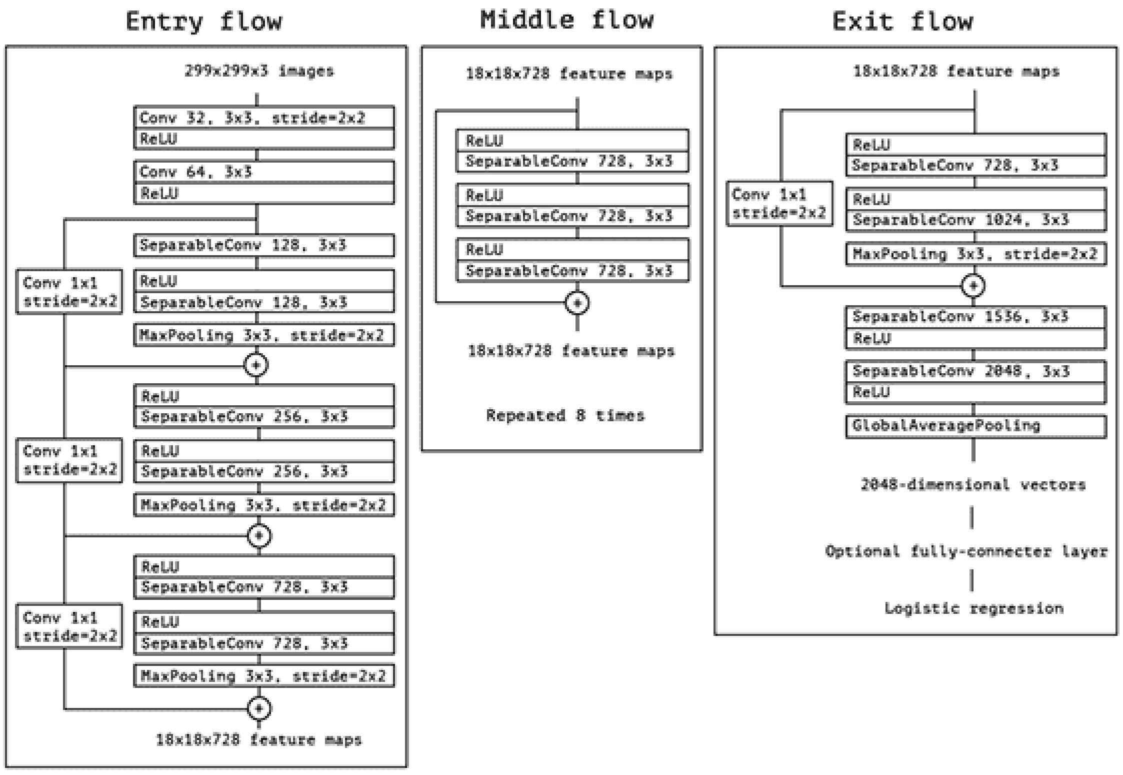
Feature extraction with VGG16 ¶

With keras, it's easy to import only the convolutional part of VGG16, by setting the include_top parameter to False :

The first important thing to note is that the predict method of our model is designed to work on several images. These images are supposed to be stored in a numpy array with shape (n,224,224,3) , where n is the number of images to be processed. So first, we have loaded an image, and converted it to a numpy array of shape (224,224,3) . To match the signature of the predict method, we then created an array of shape (1,224,224,3) with np.expand_dims .

Xception Model:

Xception was proposed by none other than **François Chollet** himself, the creator and chief maintainer of the Keras library.Xception is an extension of the Inception architecture which replaces the standard Inception modules with depthwise separable convolutions.



3.SVM Classifier:

(SVMs) are a type of supervised machine learning algorithm that can be used for classification and regression tasks. In this article, we will focus on using SVMs for image classification.

When a computer processes an image, it perceives it as a two-dimensional array of pixels. The size of the array corresponds to the resolution of the image, for example, if the image is 200 pixels wide and 200 pixels tall, the array will have the dimensions 200 x 200 x 3. The first two dimensions represent the width and height of the image, respectively, while the third dimension represents the RGB color channels. The values in the array can range from 0 to 255, which indicates the intensity of the pixel at each point.

In order to classify an image using an SVM, we first need to extract features from the image. These features can be the color values of the pixels, edge detection, or even the textures present in the image. Once the features are extracted, we can use them as input for the SVM algorithm.

The SVM algorithm works by finding the hyperplane that separates the different classes in the feature space. The key idea behind SVMs is to find the hyperplane that maximizes the margin, which is the distance between the closest points of the different classes. The points that are closest to the hyperplane are called support vectors.One of the main advantages of using SVMs for image classification is that they can effectively handle high-dimensional data, such as images. Additionally, SVMs are less prone to overfitting than other algorithms such as neural networks.

Experiment and Results:

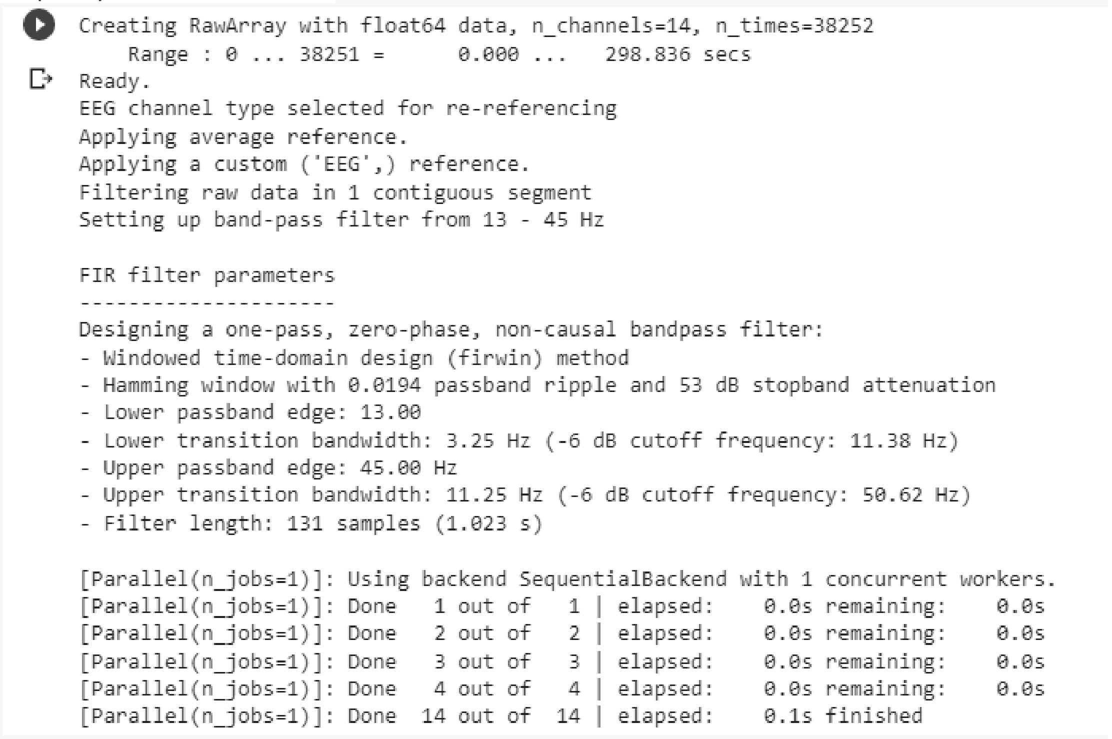
Preprocessing on the GAMEEMO Dataset:

Length of data is 20 minutes long EEG data for each subject.. Then band pass filtering is then applied. EEG data are contained in 14 channels, from which 4 are chosen for experiments. After that, EEG signals are decomposed into α (1–7 Hz), β (8–13 Hz), θ (14–30Hz), and γ bands (30–45 Hz). After band pass filtering,we just chose beta and Gamma wave frequancy (i.e.13-45 Hz) for our experimnt.Then signal windowing on frequency bands is applied. EEG signals are divided into short time frames in order to facilitate signal processing, thus time windows with diffferent overlaps are applied to EEG data in order to increase samples for training. So window size of 5 seconds are used for evaluating the proposed network. From the start of each recorded EEG signal, data are segmented by a sliding time window with an overlap of 1 second for each frequency band. For each trial of 300 s, 60 segments are obtained using an 5s time window moving 4s.

We are using 2 subjects participants from the datset, 480 (60 segments ×14 channels × 2 participants) samples are obtained for each game using time windows of 5 seconds with 1 second overlap .

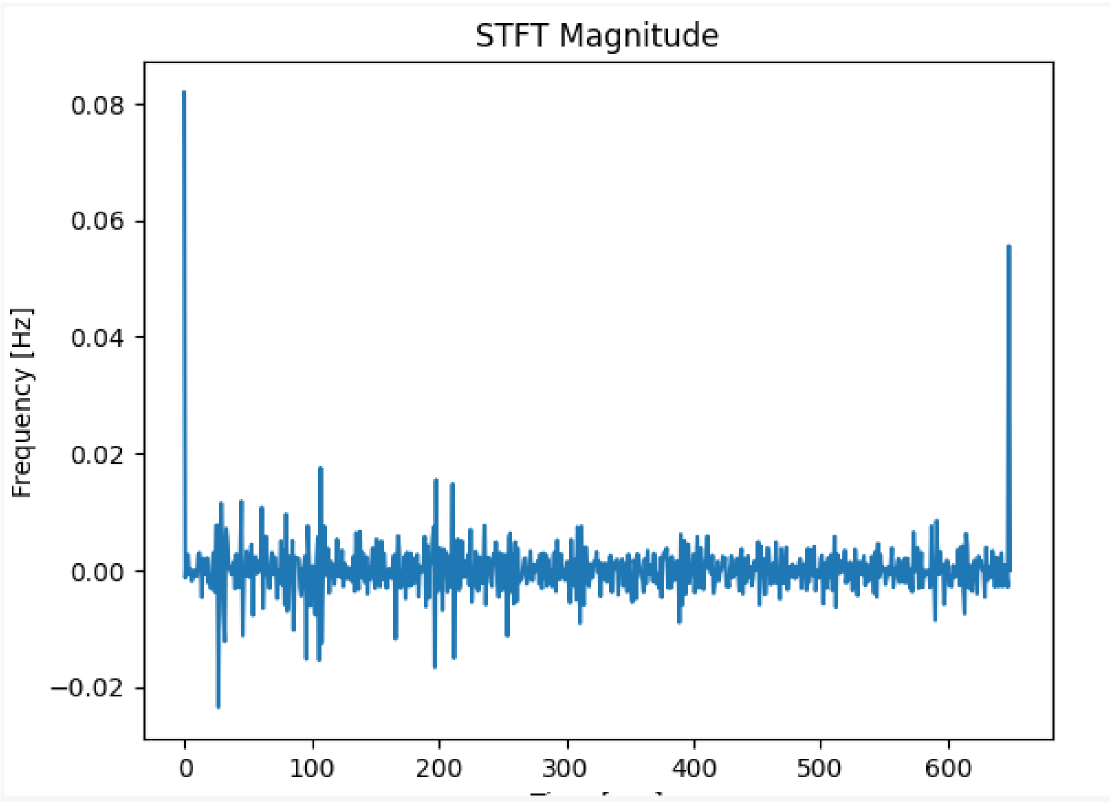
We have to download the EEG files and place in the working directory of the Python program such as Jupyter notebook or Spyder. Then we will be using the MNE Python library for the processing of the EEG signal.

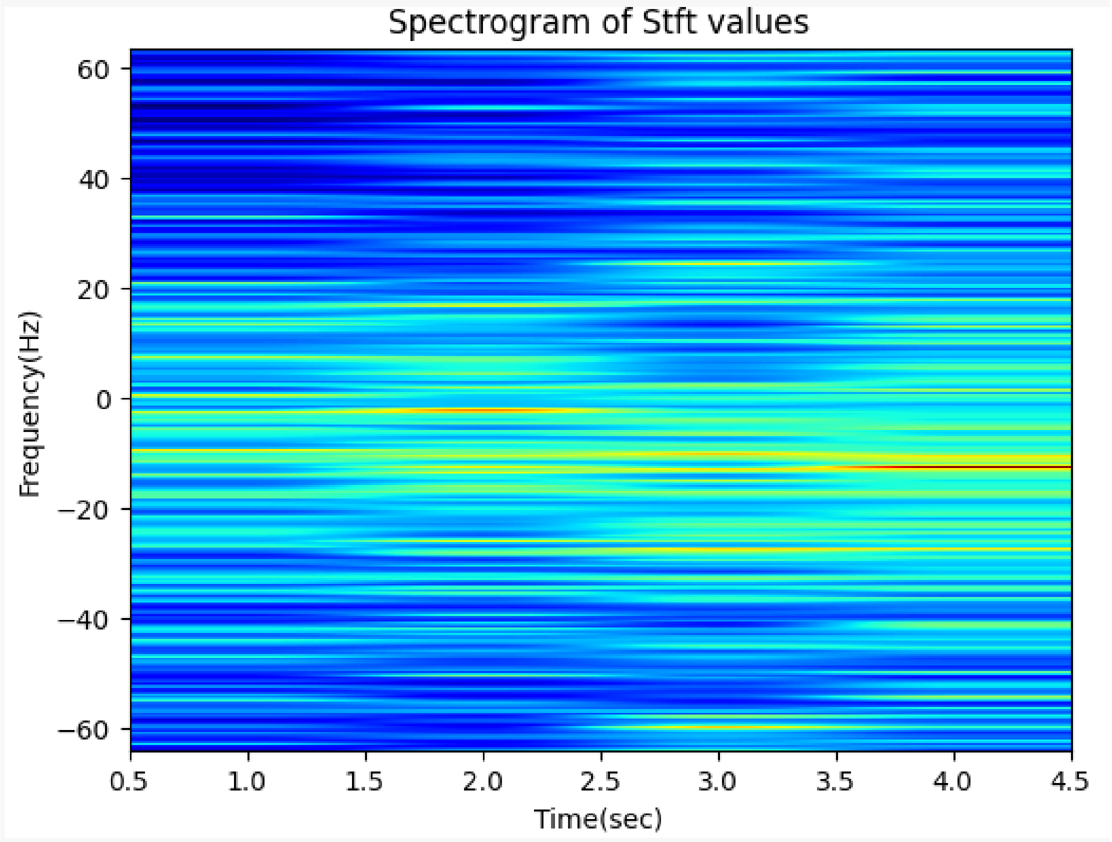
Output after using mne for preprocessing of data of subject1 and Game 2 with beta and gamma frequency and 5 s window size:



```
Creating RawArray with float64 data, n_channels=14, n_times=38252
    Range : 0 ... 38251 =      0.000 ...   298.836 secs
Ready.
EEG channel type selected for re-referencing
Applying average reference.
Applying a custom ('EEG',) reference.
Filtering raw data in 1 contiguous segment
Setting up band-pass filter from 13 - 45 Hz

FIR filter parameters
---------------------
Designing a one-pass, zero-phase, non-causal bandpass filter:
- Windowed time-domain design (firwin) method
- Hamming window with 0.0194 passband ripple and 53 dB stopband attenuation
- Lower passband edge: 13.00
- Lower transition bandwidth: 3.25 Hz (-6 dB cutoff frequency: 11.38 Hz)
- Upper passband edge: 45.00 Hz
- Upper transition bandwidth: 11.25 Hz (-6 dB cutoff frequency: 50.62 Hz)
- Filter length: 131 samples (1.023 s)

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   4 out of   4 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  14 out of  14 | elapsed:    0.1s finished
```

After preprocessing, we use STFT values of eeg signal for making spectrogram.In First image, we are just plotting the Highest magnitude of Stft data values. In the second image , we are using a spectrum of color to see the difference in frequency of stft values of dataset.

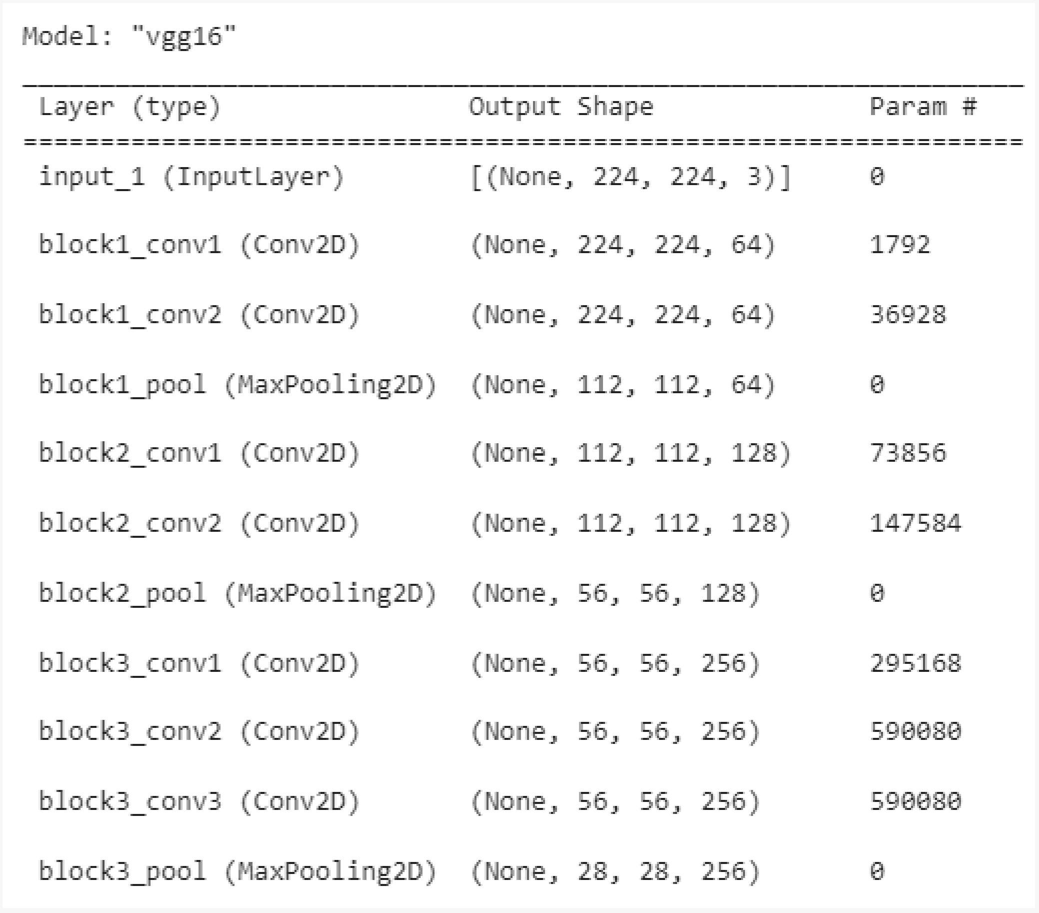




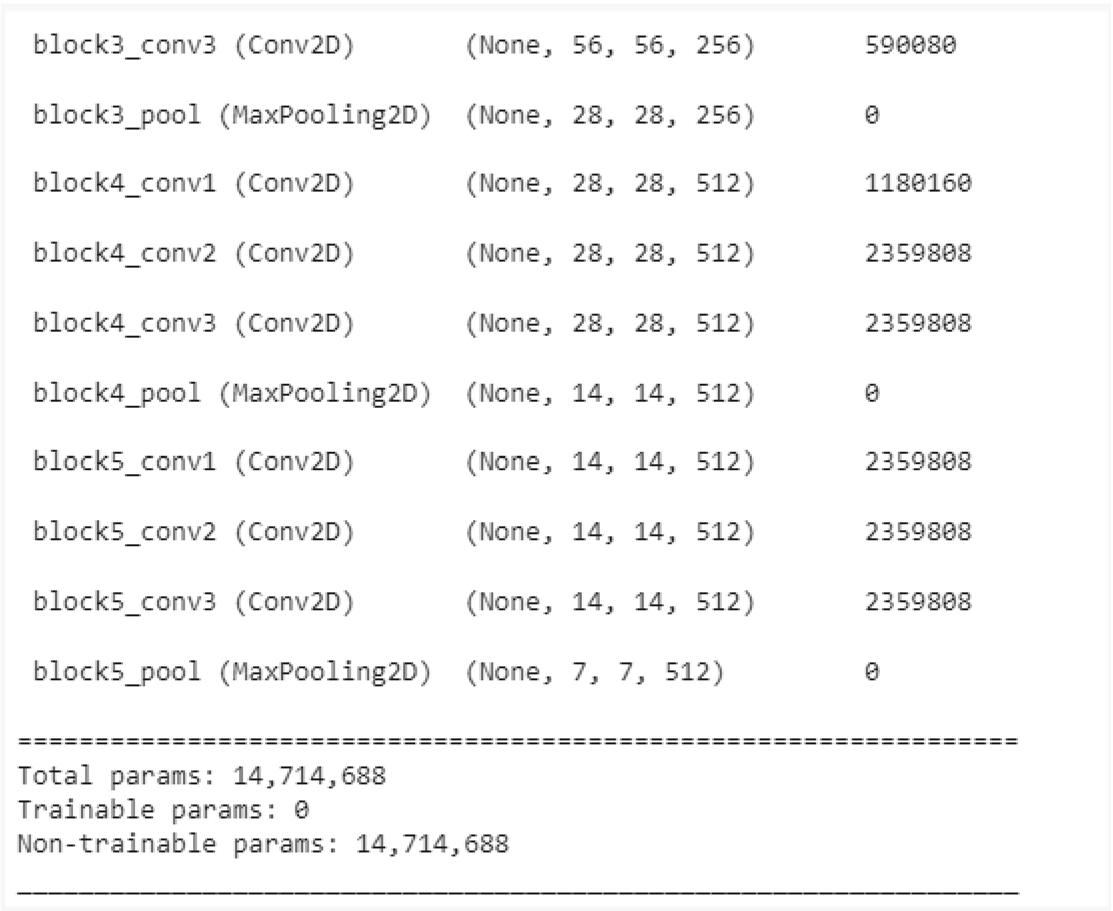
After image formaion , we store these images in a folder named boring_not_boring.
We save images of game 1 in a subfolder names “yes” and Game 2 in subfolder named “no”.We work with only 160 images because of high computation. Then, we download the pretrained transfer learning model VGG16 and Xception Model.

Then, we resize these images as (224,224) and (299,299) so that we can pass them as input to our Pretrained transfer learning models VGG16 and Xception Model respectively.
Then we label the data with 1 for game 1 and 0 for game 2.
Then we split our image datset in train and test by using train_test_split() into 80:20.

To Make loaded layers as non-trainable. This is important as we want to work with pre-trained weights.
Put base_model.trainable = False
So the architecture of VGG16 model is :



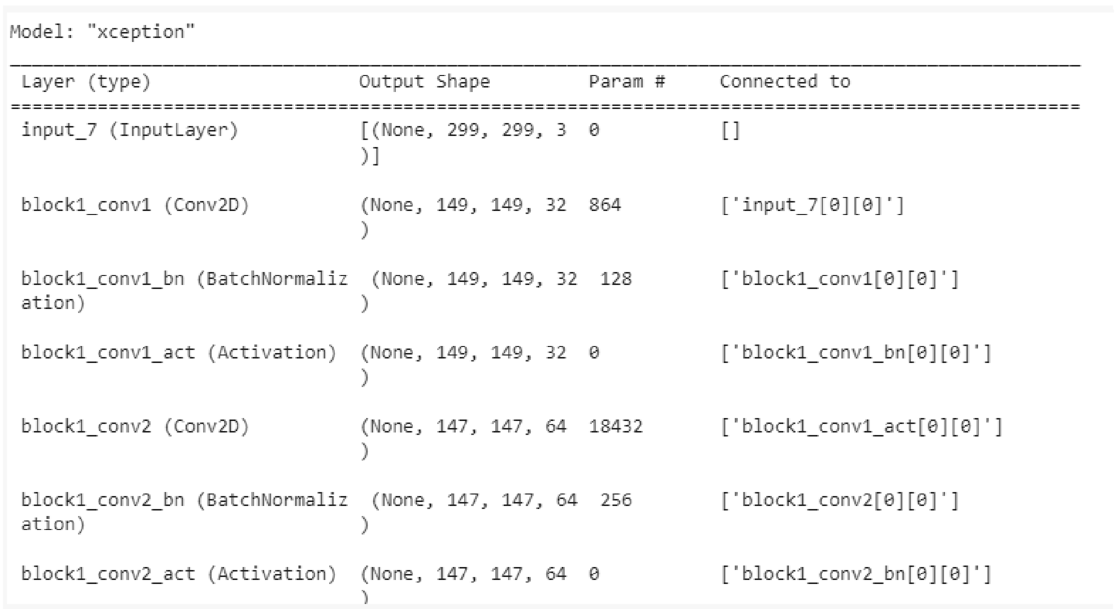
```
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0
```



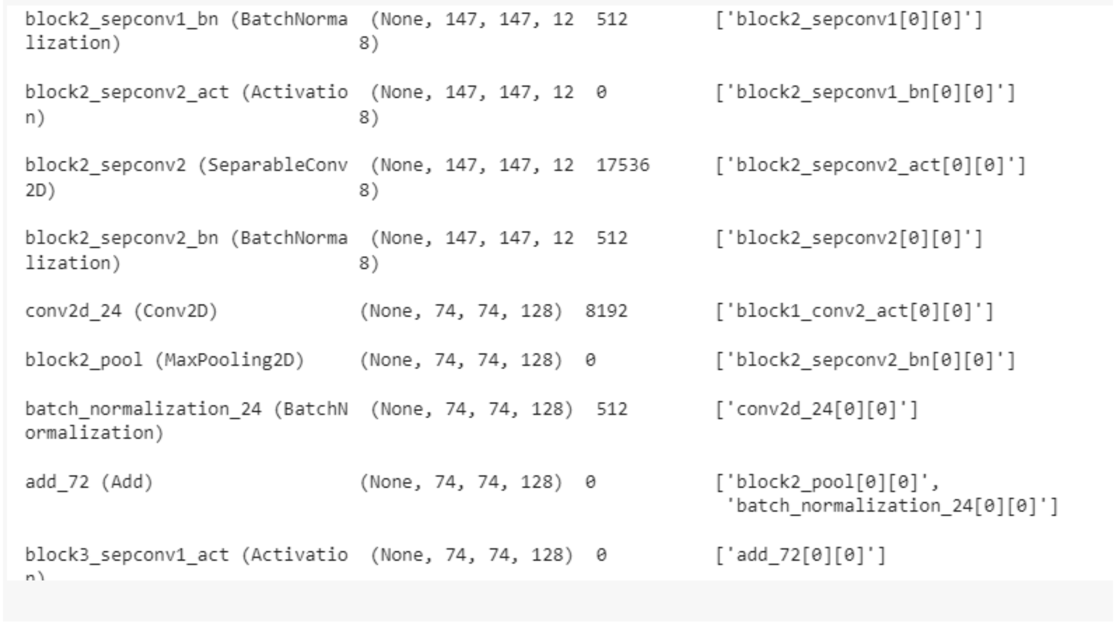
```
 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

=================================================================
Total params: 14,714,688
Trainable params: 0
Non-trainable params: 14,714,688
_________________________________________________________________
```

Then we compile the model with optimizer='adam' , loss='binary_crossentropy' ,metrics='accuracy'
 Then ,we check the class weights:
{0: 0.9552238805970149, 1: 1.0491803278688525}
Then we fit the model with epoch=10.

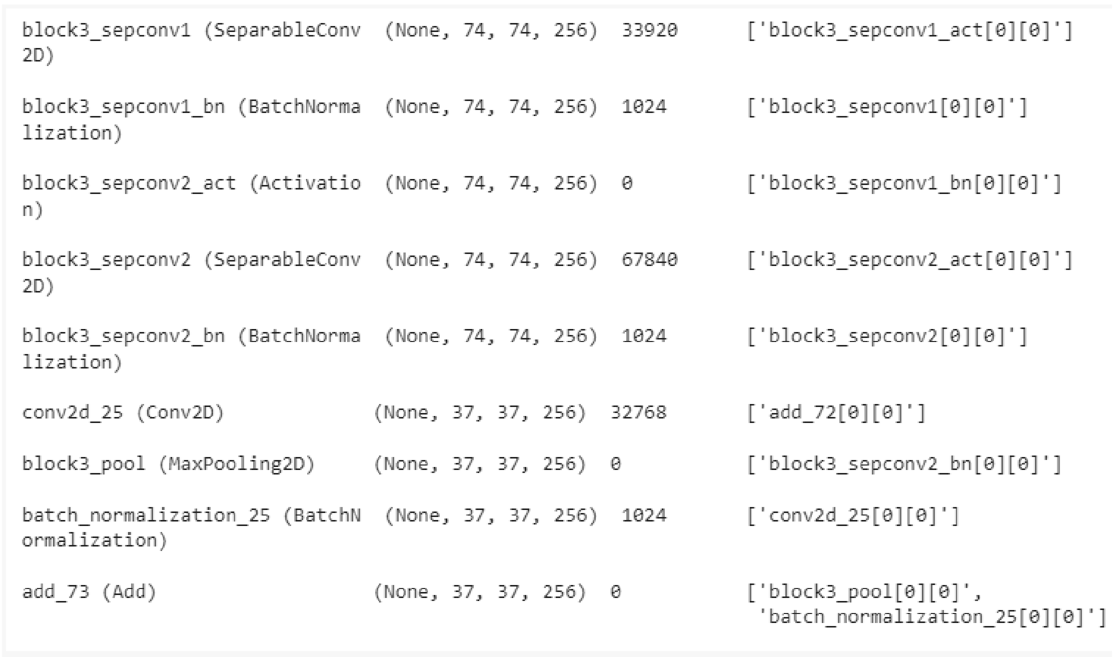
Same way , we train our xception model with same dataset and fit it.
the architecture of Xception model is :



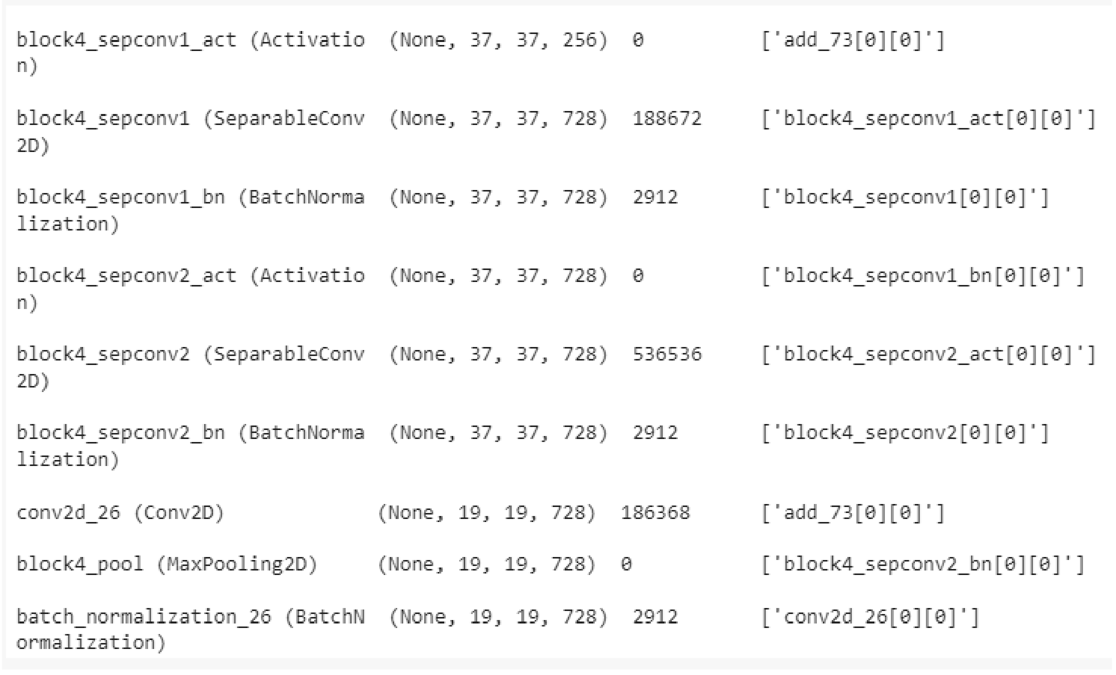
```
Model: "xception"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_7 (InputLayer)           [(None, 299, 299, 3  0           []
                                )]

 block1_conv1 (Conv2D)          (None, 149, 149, 32  864         ['input_7[0][0]']
                                )

 block1_conv1_bn (BatchNormaliz  (None, 149, 149, 32  128        ['block1_conv1[0][0]']
 ation)                         )

 block1_conv1_act (Activation)  (None, 149, 149, 32  0           ['block1_conv1_bn[0][0]']
                                )

 block1_conv2 (Conv2D)          (None, 147, 147, 64  18432       ['block1_conv1_act[0][0]']
                                )

 block1_conv2_bn (BatchNormaliz  (None, 147, 147, 64  256        ['block1_conv2[0][0]']
 ation)                         )

 block1_conv2_act (Activation)  (None, 147, 147, 64  0           ['block1_conv2_bn[0][0]']
                                )
```



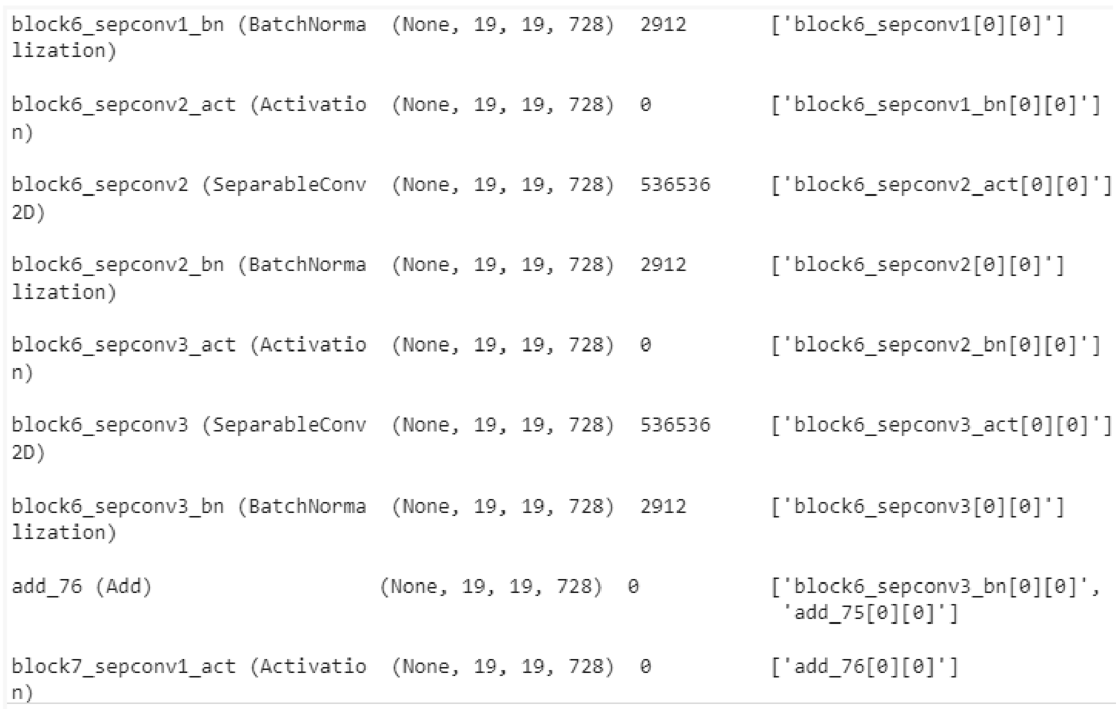
```
 block2_sepconv1_bn (BatchNorma  (None, 147, 147, 12  512        ['block2_sepconv1[0][0]']
 lization)                      8)

 block2_sepconv2_act (Activatio  (None, 147, 147, 12  0          ['block2_sepconv1_bn[0][0]']
 n)                             8)

 block2_sepconv2 (SeparableConv  (None, 147, 147, 12  17536      ['block2_sepconv2_act[0][0]']
 2D)                            8)

 block2_sepconv2_bn (BatchNorma  (None, 147, 147, 12  512        ['block2_sepconv2[0][0]']
 lization)                      8)

 conv2d_24 (Conv2D)             (None, 74, 74, 128)  8192        ['block1_conv2_act[0][0]']

 block2_pool (MaxPooling2D)     (None, 74, 74, 128)  0           ['block2_sepconv2_bn[0][0]']

 batch_normalization_24 (BatchN  (None, 74, 74, 128)  512        ['conv2d_24[0][0]']
 ormalization)

 add_72 (Add)                   (None, 74, 74, 128)  0           ['block2_pool[0][0]',
                                                                  'batch_normalization_24[0][0]']

 block3_sepconv1_act (Activatio  (None, 74, 74, 128)  0          ['add_72[0][0]']
 n)
```



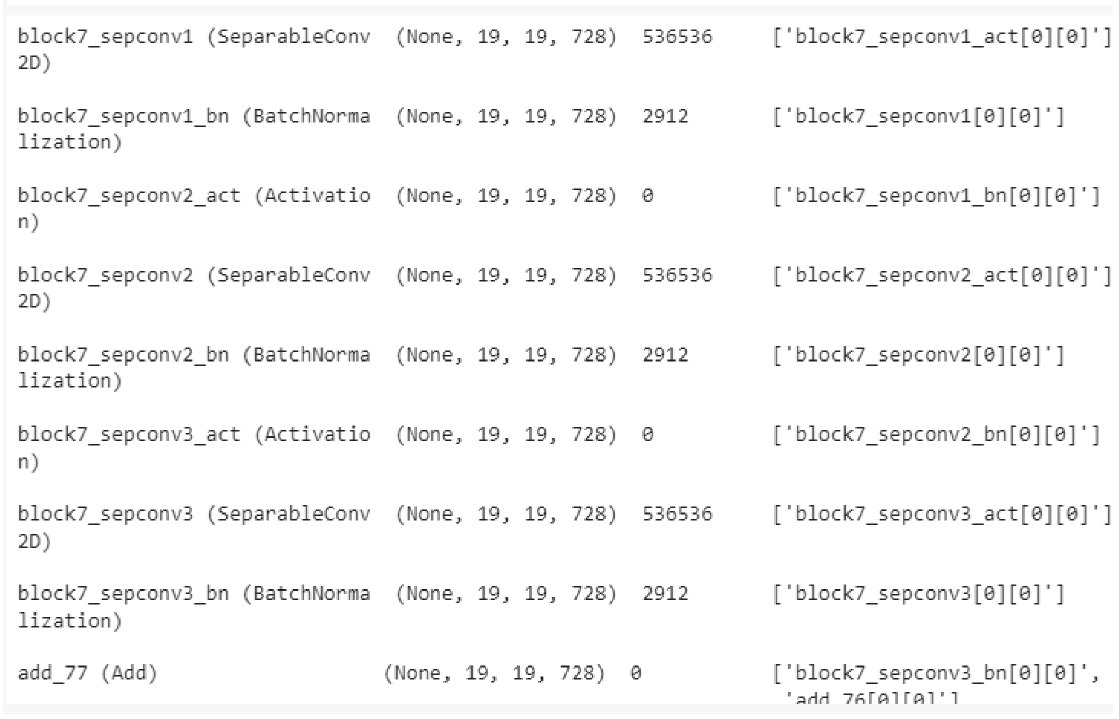
```
block3_sepconv1 (SeparableConv  (None, 74, 74, 256)  33920      ['block3_sepconv1_act[0][0]']
2D)

block3_sepconv1_bn (BatchNorma  (None, 74, 74, 256)  1024       ['block3_sepconv1[0][0]']
lization)

block3_sepconv2_act (Activatio  (None, 74, 74, 256)  0          ['block3_sepconv1_bn[0][0]']
n)

block3_sepconv2 (SeparableConv  (None, 74, 74, 256)  67840      ['block3_sepconv2_act[0][0]']
2D)

block3_sepconv2_bn (BatchNorma  (None, 74, 74, 256)  1024       ['block3_sepconv2[0][0]']
lization)

conv2d_25 (Conv2D)             (None, 37, 37, 256)  32768      ['add_72[0][0]']

block3_pool (MaxPooling2D)     (None, 37, 37, 256)  0          ['block3_sepconv2_bn[0][0]']

batch_normalization_25 (BatchN  (None, 37, 37, 256)  1024       ['conv2d_25[0][0]']
ormalization)

add_73 (Add)                   (None, 37, 37, 256)  0          ['block3_pool[0][0]',
                                                                 'batch_normalization_25[0][0]']
```



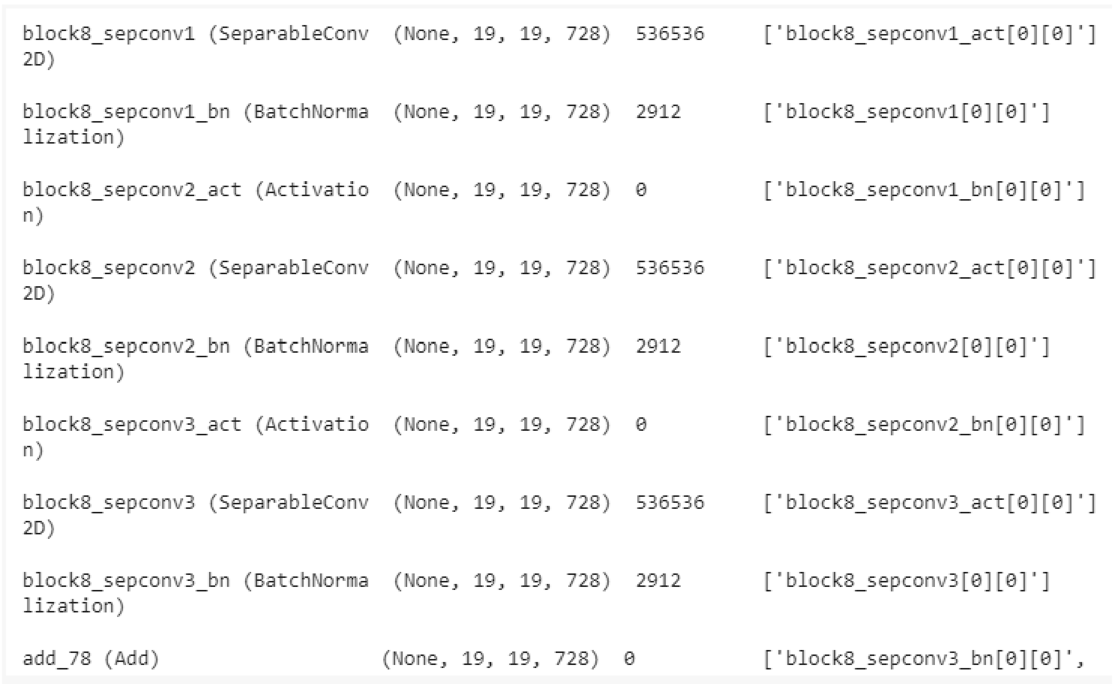
```
block4_sepconv1_act (Activatio  (None, 37, 37, 256)  0          ['add_73[0][0]']
n)

block4_sepconv1 (SeparableConv  (None, 37, 37, 728)  188672     ['block4_sepconv1_act[0][0]']
2D)

block4_sepconv1_bn (BatchNorma  (None, 37, 37, 728)  2912       ['block4_sepconv1[0][0]']
lization)

block4_sepconv2_act (Activatio  (None, 37, 37, 728)  0          ['block4_sepconv1_bn[0][0]']
n)

block4_sepconv2 (SeparableConv  (None, 37, 37, 728)  536536     ['block4_sepconv2_act[0][0]']
2D)

block4_sepconv2_bn (BatchNorma  (None, 37, 37, 728)  2912       ['block4_sepconv2[0][0]']
lization)

conv2d_26 (Conv2D)             (None, 19, 19, 728)  186368     ['add_73[0][0]']

block4_pool (MaxPooling2D)     (None, 19, 19, 728)  0          ['block4_sepconv2_bn[0][0]']

batch_normalization_26 (BatchN  (None, 19, 19, 728)  2912       ['conv2d_26[0][0]']
ormalization)
```



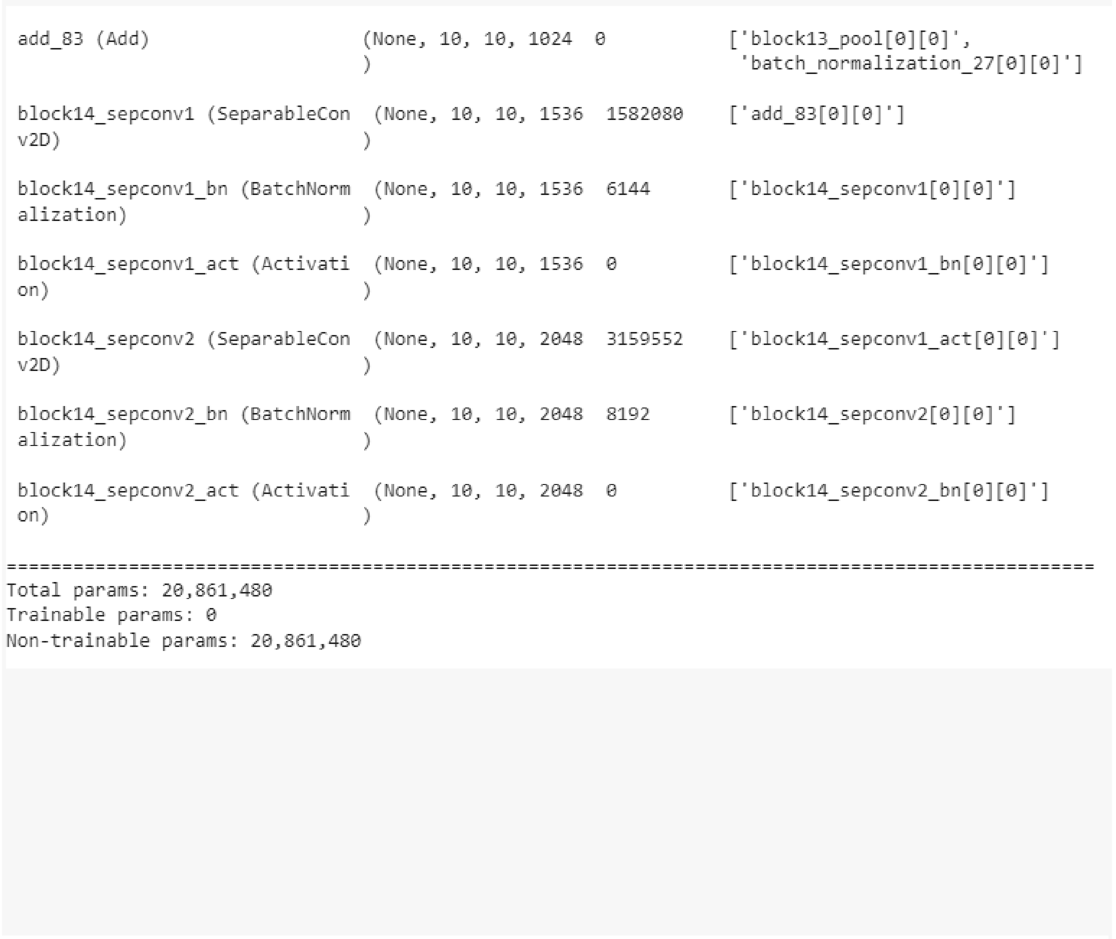
```
block6_sepconv1_bn (BatchNorma  (None, 19, 19, 728)  2912       ['block6_sepconv1[0][0]']
lization)

block6_sepconv2_act (Activatio  (None, 19, 19, 728)  0          ['block6_sepconv1_bn[0][0]']
n)

block6_sepconv2 (SeparableConv  (None, 19, 19, 728)  536536     ['block6_sepconv2_act[0][0]']
2D)

block6_sepconv2_bn (BatchNorma  (None, 19, 19, 728)  2912       ['block6_sepconv2[0][0]']
lization)

block6_sepconv3_act (Activatio  (None, 19, 19, 728)  0          ['block6_sepconv2_bn[0][0]']
n)

block6_sepconv3 (SeparableConv  (None, 19, 19, 728)  536536     ['block6_sepconv3_act[0][0]']
2D)

block6_sepconv3_bn (BatchNorma  (None, 19, 19, 728)  2912       ['block6_sepconv3[0][0]']
lization)

add_76 (Add)                   (None, 19, 19, 728)  0          ['block6_sepconv3_bn[0][0]',
                                                                 'add_75[0][0]']

block7_sepconv1_act (Activatio  (None, 19, 19, 728)  0          ['add_76[0][0]']
n)
```



```
block7_sepconv1 (SeparableConv  (None, 19, 19, 728)  536536     ['block7_sepconv1_act[0][0]']
2D)

block7_sepconv1_bn (BatchNorma  (None, 19, 19, 728)  2912       ['block7_sepconv1[0][0]']
lization)

block7_sepconv2_act (Activatio  (None, 19, 19, 728)  0          ['block7_sepconv1_bn[0][0]']
n)

block7_sepconv2 (SeparableConv  (None, 19, 19, 728)  536536     ['block7_sepconv2_act[0][0]']
2D)

block7_sepconv2_bn (BatchNorma  (None, 19, 19, 728)  2912       ['block7_sepconv2[0][0]']
lization)

block7_sepconv3_act (Activatio  (None, 19, 19, 728)  0          ['block7_sepconv2_bn[0][0]']
n)

block7_sepconv3 (SeparableConv  (None, 19, 19, 728)  536536     ['block7_sepconv3_act[0][0]']
2D)

block7_sepconv3_bn (BatchNorma  (None, 19, 19, 728)  2912       ['block7_sepconv3[0][0]']
lization)

add_77 (Add)                   (None, 19, 19, 728)  0          ['block7_sepconv3_bn[0][0]',
                                                                 'add_76[0][0]']
```



```
 block8_sepconv1 (SeparableConv  (None, 19, 19, 728)  536536     ['block8_sepconv1_act[0][0]']
 2D)

 block8_sepconv1_bn (BatchNorma  (None, 19, 19, 728)  2912       ['block8_sepconv1[0][0]']
 lization)

 block8_sepconv2_act (Activatio  (None, 19, 19, 728)  0          ['block8_sepconv1_bn[0][0]']
 n)

 block8_sepconv2 (SeparableConv  (None, 19, 19, 728)  536536     ['block8_sepconv2_act[0][0]']
 2D)

 block8_sepconv2_bn (BatchNorma  (None, 19, 19, 728)  2912       ['block8_sepconv2[0][0]']
 lization)

 block8_sepconv3_act (Activatio  (None, 19, 19, 728)  0          ['block8_sepconv2_bn[0][0]']
 n)

 block8_sepconv3 (SeparableConv  (None, 19, 19, 728)  536536     ['block8_sepconv3_act[0][0]']
 2D)

 block8_sepconv3_bn (BatchNorma  (None, 19, 19, 728)  2912       ['block8_sepconv3[0][0]']
 lization)

 add_78 (Add)                   (None, 19, 19, 728)  0          ['block8_sepconv3_bn[0][0]',
```



```
 add_83 (Add)                   (None, 10, 10, 1024  0          ['block13_pool[0][0]',
                                )                                 'batch_normalization_27[0][0]']

 block14_sepconv1 (SeparableCon  (None, 10, 10, 1536  1582080    ['add_83[0][0]']
 v2D)                           )

 block14_sepconv1_bn (BatchNorm  (None, 10, 10, 1536  6144       ['block14_sepconv1[0][0]']
 alization)                     )

 block14_sepconv1_act (Activati  (None, 10, 10, 1536  0          ['block14_sepconv1_bn[0][0]']
 on)                            )

 block14_sepconv2 (SeparableCon  (None, 10, 10, 2048  3159552    ['block14_sepconv1_act[0][0]']
 v2D)                           )

 block14_sepconv2_bn (BatchNorm  (None, 10, 10, 2048  8192       ['block14_sepconv2[0][0]']
 alization)                     )

 block14_sepconv2_act (Activati  (None, 10, 10, 2048  0          ['block14_sepconv2_bn[0][0]']
 on)                            )

==================================================================================================
Total params: 20,861,480
Trainable params: 0
Non-trainable params: 20,861,480
```

Then we use SVM classifier on our both VGG16 model and Xception Model to compare the accuracy of all four .

So,we Load the VGG16 and xception model without the top layers. Create a new model with the VGG16 base and global average pooling layer.then we Split the data into training and testing sets And Extract VGG16 features for the training and testing sets.

Then , Train an SVM classifier on the VGG16 features and Evaluate the SVM classifier on the testing set. So we get the accuracy of this classification.

Conclusions:

The accuracy of VGG16 model is : 59.375 %   . Accuracry of VGG16 and SVM classifier is: 65.625%
The accuracy of Xception model is : 56.25%   . Accuracry of Xception and SVM classifier is: 50%

Discussion:

Due to high computation of this project, we could only use very small subset of dataset so the number of images for the training and testing purpose is very less . In future , we will try to use a large amount of data for images to get better accuracy so that we could use those new insights for solving the problems.

Summary

Human emotions are complex states of feelings that result in physical and psychological changes, which can be reflected by facial expressions, gestures, intonation in speech etc. Electroencephalogram (EEG) directly measures the changes of brain activities, and emotion recognition from EEG has the potential to assess the true inner feelings of the user. Now, EEG-based emotion recognition draws attention because it is desirable that a machine can recognize human emotions during task performance and interact with us in a more humanized way. EEG can be added as an additional input to a computer during the human-machine interaction. The state-of-the-art EEG-based emotion recognition algorithms are subject-dependent and require a training session prior to real-time emotion recognition. During the training session, stimuli (audio/video) are presented to the subject to induce certain targeted emotions and meanwhile, the EEG of the subject is recorded. The recorded EEG data are subject to feature extraction to extract numerical feature parameters, and the extracted features are fed into a classifier to learn the association with their labels. However, it was found that even for the same subject, the affective neural patterns could vary over time, hence degrading the recognition accuracy in the long run. This phenomenon is termed “intra-subject variance”. Due to the existence of intra-subject variance, an EEG-based emotion recognition algorithm needs frequent re-calibration, as frequent as almost every time before running the recognition algorithm. Therefore, stable features are desired, so that re-calibration could possibly be reduced. A stable EEG feature should ideally give consistent measurements of the same emotion on the same subject over the course of time. An affective EEG database that contains multiple EEG recordings on the same subject is needed for such investigation, preferably recordings on different days. In order to establish an affective EEG database