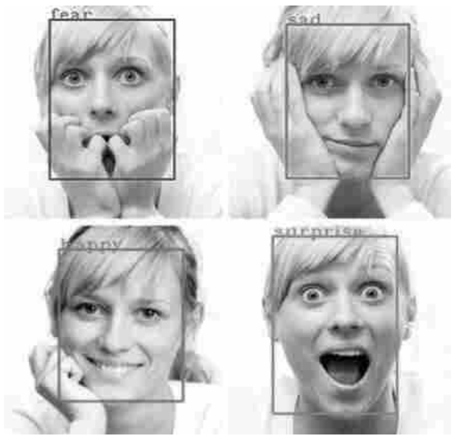
औद्योगिक प्रशिक्षण के लिए राष्ट्रीय संस्थान

National Institute for Industrial Training

One Premier Organization with Non Profit Status | Registered Under Govt. of WB
Empanelled Under Planning Commission Govt. of India
Inspired By: National Task Force on IT & SD Government of India

National Institute for Industrial Training- One Premier Organization with Non Profit Status Registered Under Govt.of West Bengal,Empanelled Under Planning Commission Govt.of India , Empanelled Under Central Social Welfare Board Govt. of India , Registered with National Career Services , Registered with National Employment Services.



Sentiment Analysis

Face Emotion Detection Project

Report I

Subham Kundu

<subhamkundu486@gmail.com>

Supervised By: Soumotanu Mazumdar

STUDENT PROFILE

Name: Subham Kundu
College: Techno India University
Stream: B. Tech CSE
Email: subhamkundu486@gmail.com
LinkedIn: https://www.linkedin.com/in/subham-kundu-10654994/
GitHub Profile: https://github.com/subhamrex
Project: Face Emotion Detection

Abstract

A huge amount of data is available to the web users with evolution of web technology. The available resources in web are used by the users and also, they involve in giving feedbacks and thus generate additional information. It is very essential to explore, analyse and organize their opinions and feedbacks in an efficient way for better decision making. So, sentiment analysis (SA) is used to find the users opinion about a particular topic, product or problem in a more efficient way. The main aim of SA is to solve the problems in relation to opinions about products, movies, politics, review sites etc. Sentiment analysis can be done with different modalities by taking inputs from text, image, audio and video. This paper proposes a system that will automatically recognise the facial expression from the image and classify emotions for final decision. The system uses a simplified method called `Viola Jones Face Detection' algorithm for face localization. Then facial features are extracted using three methods `Zernike moments', `LBP' and `DCT transform'. The different feature vectors are combined together using a subset feature selection algorithm to improve the performance of recognition and classification process. Finally, the combined features are trained and classified using SVM, Random Forest and KNN classifier. Experiments are conducted on FER-2013 database and finally performance of the system is evaluated.

Acknowledgements

The success and final outcome of this project required a lot of guidance and assistance from many people and I am extremely privileged to have got this all along the completion of my project. All that I have done is only due to such supervision and assistance and I would not forget to thank them.

I would like to express my heartfelt thanks and gratitude to my supervisor Mr. Soumotanu Mazumdar of National Institute for Industrial Training who gave me the golden opportunity to do this project and guided me in an exemplary manner. It helped me in doing a lot of research and I came to know about a lot of things related to this topic.

1. Introduction

Humans have always had the innate ability to recognize and distinguish between face's emotions. Now computers are able to do the same. This opens up tons of applications. Emotions are efficacious and self-explanatory in normal day-to-day human interactions. The most noticeable human emotion is through their facial expressions. The Facial Expression Recognition (FER) is quite complex and tedious but helps in various applications areas such as healthcare, emotionally driven robots, and human-computer interaction. Al-though the advancements in FER increases its effectiveness, achieving a high accuracy is still a challenging task. The six most generic emotions of a human are anger, happiness, sadness, disgust, fear, and surprise. Moreover, the emotion called contempt was added as one of the basic emotions. FER is a baffling task and its accuracy is completely dependent on the parameters selected, such as illumination factors, occlusion, i.e., obstruction on the face like hand, age, and sunglasses. Researchers of the field are taking these parameters into consideration while making their FER model so that the considerable accuracy can be achieved.

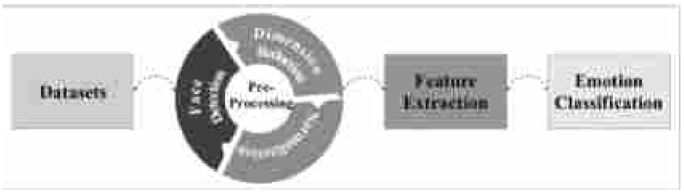
The description of some important factors for FER is as follows:

Illumination factor: The light intensity falling on the object affects the classification of the model. The textural values increase the false acceptance rate due to either by minimizing the distance between classes or by increasing the contrast.

Expression Intensity: The expression recognition is highly dependent on the intensity of the expression. The expression is recognized more accurately when the expression is less subtle. It highly affects the accuracy of the model.

Occlusion: If occlusion is present on an image, then it becomes difficult for the model to extract features from the occluded part due to inaccurate face alignment and imprecise feature location. It also introduces noise to outliers and the extracted features.

FER systems can be either static or dynamic based on image. Static FER considers only the face point location information from the feature representation of a single image, whereas, the Dynamic Image FER considers the temporal information with continuous frames. The static FER process over is exhibited in FIGURE 1 with description of steps as follows.



1.1 Motivation

Paul Ekman first coined the term FER in the mid-1980s.Since then, various machine learning techniques like random-forest classifiers, artificial neural networks, etc. were used by the researchers to recognize the seven basic emotions. They also claimed good and effective results. Automated human emotion detection is all-important in security and surveillance applications these days. To further improve its performance, the researchers are trying hard to explore further in this field. Various challenges like occlusion in datasets, over-fitting of models, etc. have to be taken care of while implementing the FER. As per the literature explored and knowledge of the authors, no survey is available, which exhaustively compares the FER approaches from the perspective of AI. Motivated from the aforementioned fact, we present a comprehensive survey on FER using Artificial Intelligence (AI) techniques in which we have explored the state-of-the-art machine learning and DL (DL) approaches with their merits and demerits.

1.2 Dataset

To avoid over-fitting we need extensive training data. A dataset must have well-defined emotion tags of facial expression is essential for testing, training, and validating the algorithms for the development offer. These datasets contain a sequence of images with distinct emotions, as mentioned above. We have re-viewed many datasets to train different models for real-world profits. We have used FER-2013 dataset for this project. The dataset has gathered from Kaggle. This dataset has a problem. Some categories of this dataset have low number of pictures. So, we have generated data

1.3 Pre-Processing

This step pre-processes the dataset by removing noise and data compression. Various steps involved in data pre-processing are:

(i) facial detections the power to detect the location of the face in any image or frame. It is often considered as a special case of object-class detection, which determines whether the face is present in an image or not,

(ii) dimension reduction is used to reduce the variables by a set of principal variables. If the number of features is more, then it gets tougher to visualize the training set and to work on it. Here, PCA and LDA can be used to handle the aforementioned situation.

(iii) normalization: It is also known as feature scaling. After the dimension reduction step, reduced features are normalized without distorting the differences in the range of values of features. There are various normalization methods, namely Z Normalization, Min-Max Normalization, Unit Vector Normalization, which improves the numerical stability and speeds up the training of the model.

1.4 Feature Extraction

It is the process of extracting features that are important for FER. It results in smaller and richer sets of attributes that contain features like face edges, corners, diagonal, and other important information such as distance between lips and eyes, the distance between two eyes, which helps in speedy learning of trained data.

1.5 Emotion Classification

It involves the algorithms to classify the emotions based on the extracted features. The classification has various methods, which classiest images into various classes. The classification of a FER image is carried out after passing through pre-processing steps of face detection and feature extraction. Various classification techniques are discussed later in the proposed survey.

The FER system has various applications such as computer-human interactions, healthcare system, and social marketing. In the proposed survey, we analyse the existing surveys pertaining to different approaches of FER proposed by the authors globally. We compare the surveys and develop taxonomy on various pre-processing, feature extraction, and emotion classification steps. We also discuss the various open issues and future research challenges related to FER.

2. SCOPE OF THE SURVEY:

Facial sentiment analysis is the most trending topics in Computer Vision area. A lot of literature has already been published by researchers across the globe in this field, but still, many researchers are trying to solve the challenges and issues in FER. Various surveys have been published in recent years [7], [15], [26], [27] on sentiment analysis. These surveys have mainly focused on traditional methods like support vector machine (SVM), decision tree classifiers, and artificial neural network (ANN). The DL methods have rarely been explored by the researchers working in the same field. So, in this paper, we analysed the surveys on facial sentiment analysis and presented a comparative analysis. For example, we surveyed various methods for facial detection, facial feature extraction and classification of FER, but not presented the proper com-parison of methods considered and the dataset used. Later, the we also presented the survey on FER, but they had not mentioned anything about datasets useful for emotions recognition. In Another survey we did on various traditional feature extraction techniques like principal component analysis (PCA), Linear Discriminant Analysis

(LDA), and Locally Linear Embedding (LLE), and thereafter they proposed an ensemble classifier. They failed to compare the advanced DL approach, which is currently the most novel approach in FER. Again, Bhaskar also lacks in explaining various DL approaches. Recently, DL-based FER approaches have been explored which gave detailed surveys without the discussion on FER. Therefore, in the proposed survey, we make a systematic survey of various databases used for FER, various methods for face detection, facial feature extraction, and emotion classification, future challenges, and current issues in facial sentiment analysis. Our aim for this survey that it would be quite beneficial for those who want to explore in this field and they will get a complete overview of all the advanced systematic approaches in facial sentiment analysis.

3. RESEARCH

CONTRIBUTIONS In this paper, we surveyed various existing literature on Facial Sentiment Analysis focusing on the DL techniques, datasets, and the methodologies used to classify emotions. Following are the crisp contributions of the paper.

We present an in-depth survey on FER methods and dataset used. Then, we highlight the advanced methods used for FER and their comparative analysis.

We present a taxonomy on FER methods based on face detection, feature extraction, and emotion classification.

Finally, we presented the open issues and research challenges in the Facial Sentiment Analysis

4. About Project:

This project has been performed to detect the task of classifying a facial emotion which will be Angry, Disgust, Fear, Happy, Neutral, Sad and Surprise.

Firstly, the data has been thoroughly pre-processed as described in section 1.3. After that a new file has been made with the formatted data. Next that data has been fed into machine learning algorithms and neural networks
to compare and contrast the model performance.

The algorithms used are as follows
1. Artificial Neural Network
2. 1 Dimensional Convolutional Neural Network

4.1 Objectives

The sole objective of the project is to predict facial emotion using machine learning.
Different techniques have been applied for this purpose and a comparative study has been made between the different accuracies shown by the various models.

The steps involved are -
1. DATA MINING- Data mining is a process of discovering patterns in large data sets involving methods at the intersection of machine learning, statistics and database systems Data mining is an interdisciplinary subfield of computer science and statistics with an overall goal to extract information (with intelligent methods) from a data set and transform the information
into a comprehensible structure for further use.
2. DATA CLEANING- Data cleaning is the process of preparing data for analysis by removing or modifying data that is incorrect, incomplete, irrelevant, duplicated, or improperly formatted.
3. DATA PREPROCESSING- Data pre-processing is a data mining technique that involves transforming raw data into an understandable format. Real-world data

is often incomplete, inconsistent, lacking in certain behaviours or trends, and is likely to contain many errors. Data pre-processing is a proven method of resolving such issues.

4. EXPLORATORY DATA ANALYSIS - In statistics, exploratory data analysis is an approach to analysing data sets to summarize their main characteristics, often using statistical graphics and other data visualization methods.

5. DIVIDING INTO TRAINING AND TESTING SET- Data splitting is the act of partitioning available data into two portions; usually for cross-validatory purposes. One portion of the data is used to develop a predictive model and the other to evaluate the model's performance.

6. APPLYING VARIOUS CLASSIFICATION MODELS- Various classification models were used to predict the probability of a particular problem like stroke in patients, after taking into consideration various other factors.

5. General Introduction to Relevant Topics

Data Science - Data science is the domain of study that deals with vast volumes of data using modern tools and techniques to find unseen patterns, derive meaningful information, and make business decisions. Data science uses complex machine learning algorithms to build predictive models.

The data used for analysis can be from multiple sources and present in various formats. Data science or data-driven science enables better decision making, predictive analysis, and pattern discovery.

Data science can add value to any business who can use their data well. From statistics and insights across workflows and hiring new candidates, to helping senior staff make better-informed decisions, data science is valuable to any company in any industry.

By extrapolating and sharing these insights, data scientists help organizations to solve vexing problems. Combining computer science, modelling, statistics, analytics, and math skills— along with sound business sense— data scientists uncover the answers to major questions that help organizations make objective decisions.

Machine Learning - Machine learning is a type of technology that aims to learn from experience. For example, as a human, you can learn how to play chess simply by observing other people playing chess. In the same way, computers are programmed by providing them with data from which they learn and are then able to predict future elements or conditions. There are various steps involved in machine learning:

1. collection of data
2. filtering of data
3. analysis of data
4. algorithm training
5. testing of the algorithm
6. using the algorithm for future predictions

Machine learning uses different kinds of algorithms to find patterns, and these Algorithms are classified into two groups:

● supervised learning

● unsupervised learning

<u>Supervised Learning</u>

Supervised learning is the science of training a computer to recognize elements by giving it sample data. The computer then learns from it and is able to predict future datasets based on the learned data.

For example, you can train a computer to filter out spam messages based

on past information. Supervised learning has been used in many applications, e.g., Facebook, to search images based on a certain description. You can now search images on Facebook with words that describe the contents of the photo. Since the social networking site already has a database of captioned images; it is able to search and match the description to features from photos with some degree of accuracy.

There are only two steps involved in supervised learning:

- training
- testing

Some of the supervised learning algorithms include:

- decision trees
- support vector machines
- naive Bayes
- k-nearest neighbour
- linear regression

Generally, most of the Neural Networks are used in supervised methods. Either to predict (regression) something or in classification. Convolutional Neural Networks (CNN) are widely employed in Image Classifications and other object recognition applications. Classification of Images based on their attributes is one of the most famous applications of CNN.

5.1 Programming Language - Python:

Python is a widely used high-level programming language for general-purpose programming, created by Guido van Rossum and first released in 1991.

Python features a dynamic type system and automatic memory management and supports multiple programming paradigms, including object- oriented, imperative, functional programming, and procedural styles. It has a large and comprehensive standard library. Two major versions of Python are currently in active use:

Python 3.x is the current version and is under active development. Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

Advantages of Python

1. Easy Syntax

2. Readability

3. High-Level Language

4. Object-oriented programming

5. It's Open source and Free

6. Cross-platform

7. Widely Supported

8. It’s Safe

9. Extensible

Easy Syntax of Python

Python’s syntax is easy to learn, so both non- programmers and programmers can start programming right away.

Very Clear Readability of Python

Python’s syntax is very clear, so it is easy to understand program code. (Python is often referred to as “executable pseudo-code” because its syntax mostly follows the conventions used by programmers to outline their ideas without the formal verbosity of code in most programming languages.

Python High-Level Language

Python looks more like a readable, human language than like a low-level language. This gives you the ability to program at a faster rate than a low-level language will allow you.

Python Is Open-Source and Free

Python is both free and open-source. The Python Software Foundation distributes pre -made binaries that are freely available for use on all major operating systems called CPython. You can get CPython’s source-code, too. Plus, we can modify the source code and distribute as allowed by CPython’s license.

Python is a Cross-platform

Python runs on all major operating systems like Microsoft Windows, Linux, and Mac OS X.

Python Object-oriented programming

Object-oriented programming allows you to create data structures that can be reused, which reduces the amount of repetitive work that you'll need to do.

Programming languages usually define objects with namespaces, like class or def, and objects can edit themselves by using keyword, like this or self.

Most modern programming languages are object-oriented (such as Java, C++, and C#) or have support for OOP features (such as Perl version 5 and later). Additionally, object-oriented techniques can be used in the design of almost any non-trivial software and implemented in almost any programming or scripting language.

Python Widely Supported Programming Language

Python has an active support community with many websites, mailing lists, and USENET "net news" groups that attract a large number of knowledgeable and helpful contributors.

Python is a Safe

Python doesn't have pointers like other C-based languages, making it much more reliable. Along with that, errors never pass silently unless they're explicitly silenced. This allows you to see and read why the program crashed and where to correct your error.

Python Batteries Included Language

Python is famous for being the “batteries are included” language. There are over 300 standards library modules which contain modules and classes for a wide variety of programming tasks. For example, the standard library contains modules for safely creating temporary files (named or anonymous), mapping files into memory (including use of shared and anonymous memory mappings), spawning and controlling sub-processes, compressing and decompressing files (compatible with gzip or PK-zip) and archives files (such as Unix/Linux “tar”). Accessing indexed “DBM” (database) files, interfacing to various graphical user interfaces (such as the TK toolkit and the popular WxWindows multi-platform windowing system),

parsing and maintaining CSV (comma- separated values) and “.cfg” or “.ini” configuration files (similar in syntax to the venerable WIN.INI files from MS-DOS and MS-Windows), for sending e-mail, fetching and parsing web pages, etc. It’s possible, for example, to create a custom web server in Python using less than a dozen lines of code, and one of the standard libraries, of course.

Python is Extensible

In addition to the standard libraries there are extensive collections of freely available add-on modules, libraries, frameworks, and tool-kits. These generally conform to similar standards and conventions. For example, almost all of the database adapters (to talk to almost any client-server RDBMS engine such as MySQL, Postgres, Oracle, etc) conform to the Python DBAPI and thus can mostly be accessed using the same code. So, it’s usually easy to modify a Python program to

support any database engine.

5.2.1 Future Scopes of Python

Python is one of the fastest growing languages and has undergone a successful span of more than 25 years as far as its adoption is concerned. This success also reveals a promising future scope of python programming language. In fact, it has been continuously serving as the best programming language for application development, web development, game development, system administration, scientific and numeric computing, GIS and Mapping etc.

Popularity of python

The reason behind the immense popularity of python programming language across the globe is the features it provides. Have a look at the features of python language.

(1) **Python Supports Multiple Programming Paradigms:**

Python is a multi-paradigm programming language including features such

as object-oriented, imperative, procedural, functional, reflective etc.

(2) **Python Has Large Set of Library and Tools**

Python has very extensive standard libraries and tools that enhance the overall functionality of python language and also helps python programmers to easily write codes. Some of the important python libraries and tools are listed below.

● Built-in functions, constants, types, and exceptions.

● File formats, file and directory access, multimedia

services.

● GUI development tools such as Tkinter

● Custom Python Interpreters, Internet protocols and support, data compression and

archiving, modules etc.

● Scrappy, wxPython, SciPy, matplotlib, Pygame, PyQT, PyGTK etc.

(3) **Python Has a Vast Community Support**

This is what makes python a favourable choice for development purposes. If you are

having problems writing python a program, you can post directly to python community and will get the response with the solution of your problem. You will also find many new ideas regarding python technology and change in the versions.

(4) **Python is Designed for Better Code Readability**

Python provides a much better code readability as compared to another programming language. For example, it uses whitespace indentation in place of curly brackets for delimiting the block of codes.

(5) **Python Has a Vast Community Support**

This is what makes python a favourable choice for development purposes. If you are

having problems writing python a program, you can post directly to python community and will get the response with the solution of your problem. You will also find many new ideas regarding python technology and change in the versions.

(6) **Python is Designed for Better Code Readability**

Python provides a much better code readability as compared to another programming language. For example, it uses whitespace indentation in place of curly brackets for delimiting the block of codes.

(7) **Python Contains Fewer Lines of Codes**

Codes are written in python programming language complete in fewer lines thus reducing the efforts of programmers. Let's have a look on the following "Hello World" program written in C, C++, Java, and Python.

While, C, C++, and Java take six, seven and five lines respectively for a simple "Hello World" program. Python takes only a single line which means, less coding effort and time is required for writing the same program.

Future Technologies Counting on Python

Generally, we have seen that python programming language is extensively used for web development, application development, system administration, developing games etc.

But do you know there are some future technologies that are relying on python? As a matter of fact, Python has become the core language as far as the success of these technologies is concerned. Let's dive into the technologies which use python as a core element for research, production and further developments.

(1) **Artificial Intelligence (AI)**

Python programming language is undoubtedly dominating the other languages when future technologies like Artificial Intelligence (AI) come into the play.

There are plenty of python frameworks, libraries, and tools that are specifically developed to direct Artificial Intelligence to reduce human efforts with increased accuracy and efficiency for various development purposes. It is only the Artificial Intelligence that has made it possible to develop speech recognition

system, autonomous cars, interpreting data like images, videos etc.

We have shown below some of the python libraries and tools used in various Artificial Intelligence branches.

● Machine Learning- PyML, PyBrain, scikit-learn, MDP Toolkit, GraphLab Create,

MIPy etc.

● General AI- pyDatalog, AIMA, EasyAI, SimpleAI etc.

● Neural Networks- PyAnn, pyrenn, ffnet, neurolab etc.

● Natural Language & Text Processing- Quepy, NLTK, gensim

(2) **Big Data**

The future scope of python programming language can also be predicted by the way it has helped big data technology to grow. Python has been successfully contributing in analysing a large number of data sets across computer clusters through its high-performance toolkits and libraries. Let’s have a look at the python libraries and toolkits used for Data analysis and handling other big data issues.

● Pandas

● Scikit-Learn

● NumPy

● SciPy

● GraphLab Create

● IPython

● Bokeh

● Agate

● PySpark

● Dask

(3) **Networking**

Networking is another field in which python has a brighter scope in the future. Python programming language is used to read, write and configure routers and switches and perform other networking automation tasks in a cost-effective and secure manner. For these purposes, there are many libraries and tools that are built on the top of the python language. Here we have listed some of these python libraries and tools especially used by network engineers for network automation.

● Ansible

● Netmiko

● NAPALM (Network Automation and Programmability

Abstraction Layer with Multivendor Support)

● Pyeapi

● Junos PyEZ

● PySNMP

● Paramiko SSH

Real-Life Python Success Stories

Python has seemingly contributed as a core language for increasing productivity regarding various development purposes at many of the IT organizations. We have shown below some of the real-life python success stories.

● Australia’s RMA Department D-Link has successfully implemented python

for creating DSL Firmware Recovery System.

● Python has helped Gusto.com, an online travel site, in reducing development

costs and time.

● ForecastWatch.com also uses python in rating the accuracy of weather forecast

reports provided by companies such as Accuweather, MyForecast.com and The

Weather Channel.

● Python has also benefited many product developments companies such as

Acqutek, AstraZeneca, GravityZoo, Carmanah Technologies Inc. etc in

creating autonomous devices and software.

● Test&Go uses python scripts for Data Validation.

● Industrial Light & Magic(ILM) also uses python for batch processing that

includes modeling, rendering and compositing thousands of picture frames per day.

There is a huge list of success stories of many organizations across the globe which are using python for various purposes such as software development, data mining, unit testing, product development, web development, data validation, data visualization etc. These success stories directly point towards a promising future scope of python programming language.

5.1.2 Top Competitors of Python

The future scope of python programming language also depends on its competitors in the IT market. But, due to the fact that it has become a core language for future technologies such as artificial intelligence, big data, etc., it will surely rise further and will be able to beat its competitors.

Competitors and Alternatives to Python Programming Language

- Microsoft.
- Oracle.
- IBM.
- Tableau.
- SAP.
- Alteryx.
- Blue Yonder.
- Gurobi.2

5.2 Imports

The libraries that have been imported for this project are stated as follows:

1. **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.

2. **matplotlib:** matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK. It is used to plot for data visualization.

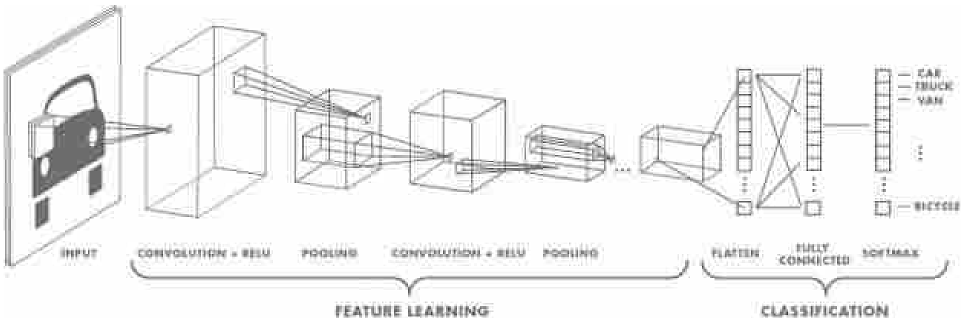
3. **TensorFlow -**TensorFlow is a free and open-source software library for machine learning. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. TensorFlow computations are expressed as stateful dataflow graphs. The name TensorFlow derives from the operations that such neural networks perform on multidimensional data arrays, which are referred to as tensors.

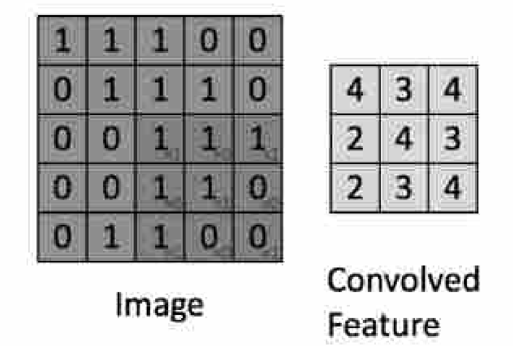
Most models are made of layers. Layers are functions with a known mathematical structure that can be reused and have trainable variables. In TensorFlow, most high-level implementations of layers and models, such as Keras. It has three layers:

I. Input layers: The input layer of a neural network is composed of artificial input neurons, and brings the initial data into the system for further processing by subsequent layers of artificial neurons.

II. Hidden layers: A hidden layer is located between the input and output of the algorithm, in which the function applies weights to the inputs and directs them through an activation function as the output.

III. Output layers: The output layer is responsible for producing the final result. The output layer takes in the inputs which are passed in from the layers before it, performs the calculations via its neurons and then the output is computed.

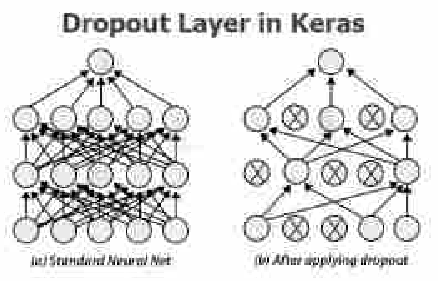




Besides, I have used Convolutional Neural Network (CNN) which is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a CNN is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, CNN have the ability to learn these filters/characteristics. The role of the CNN is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction. I did Convolution operation on a 244x244x3 image matrix with a 3x3x3 Kernel.

.

4. Keras- Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the



TensorFlow library.

5. OpenCV: OpenCV is a library of programming functions mainly aimed at real-time computer vision. Using this library, we have load face detection module and face emotion detection model in a video stream. Specially It helps us to show a proper output with a video frame. It also helps in image processing.

6. Hardware and software requirements

Software Requirements:

Operating System: Windows/Linux

Programming Language: Python 3.8

Software (IDE): PyCharm or VSCode

Hardware requirements:

Speed: 233MHz and above

Hard disk: 10GB

RAM: 512 MB

7. Formal description of the Training Models used

7.1 Artificial Neural Network (ANN)

An artificial neural network (ANN) is the piece of a computing system designed to simulate the way the human brain analyses and processes information. It is the foundation of artificial intelligence (AI) and solves problems that would prove impossible or difficult by human or statistical standards. ANNs have self-learning capabilities that enable them to produce better results as more data becomes available.

- An artificial neural network (ANN) is the component of artificial intelligence that is meant to simulate the functioning of a human brain. Processing units make up ANNs, which in turn consist of inputs and outputs. The inputs are what the ANN learns from to produce the desired output.
- Back propagation is the set of learning rules used to guide artificial neural networks.
- The practical applications for ANNs are far and wide, encompassing finance, personal communication, industry, education, and so on

Artificial neural networks are built like the human brain, with neuron nodes interconnected like a web. The human brain has hundreds of billions of cells called neurons. Each neuron is made up of a cell body that is responsible for processing information by carrying information towards (inputs) and away (outputs) from the brain.

An ANN has hundreds or thousands of artificial neurons called processing units, which are interconnected by nodes. These processing units are made up of input and output units. The input units receive various forms and structures of information based on an internal weighting system, and the neural network attempts to learn about the information presented to produce one output report. Just like humans need rules and guidelines to come up with a result or output, ANNs also use a set of learning rules called back propagation, an abbreviation for backward propagation of error, to perfect their output results.

An ANN initially goes through a training phase where it learns to recognize patterns in data, whether visually, aurally, or textually. During this supervised phase, the network compares its actual output produced with what it was meant to produce—the desired output. The difference between both outcomes is adjusted using back propagation. This means that the network works backward, going from the output unit to the input units to adjust the weight of its connections between the units until the difference between the actual and desired outcome produces the lowest possible error.

Practical Applications for Artificial Neural Networks (ANNs)

Artificial neural networks are paving the way for life-changing applications to be developed for use in all sectors of the economy. Artificial intelligence platforms that are built on ANNs are disrupting the traditional ways of doing things. From translating web pages into other languages to having a virtual assistant order grocery online to conversing with chat bots to solve problems, AI platforms are simplifying transactions and making services accessible to all at negligible costs.

Artificial neural networks have been applied in all areas of operations. Email service providers use ANNs to detect and delete spam from a user's inbox; asset managers use it to forecast the direction of a company's stock; credit rating firms use it to improve their credit scoring methods; e-commerce platforms use it to personalize recommendations to their audience; chat bots are developed with ANNs for natural language processing; deep

learning algorithms use ANN to predict the likelihood of an event; and the list of ANN incorporation goes on across multiple sectors, industries, and countries.

In a neural network, there are three essential layers –

Input Layers

The input layer is the first layer of an ANN that receives the input information in the form of NumPy array, numbers, audio files, image pixels, etc.

Hidden Layers

In the middle of the ANN model are the hidden layers. There can be a single hidden layer, as in the case of a perceptron or multiple hidden layers. These hidden layers perform various types of mathematical computation on the input data and recognize the patterns that are part of.

Output Layer

In the output layer, we obtain the result that we obtain through rigorous computations performed by the middle layer.

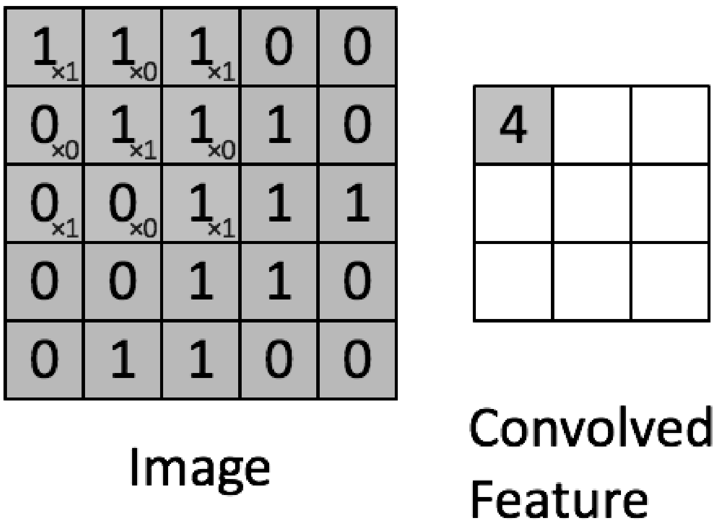
In a neural network, there are multiple parameters and hyperparameters that affect the performance of the model. The output of ANNs is mostly dependent on these parameters. Some of these parameters are weights, biases, learning rate, batch size etc. Each node in the ANN has some weight.

Each node in the network has some weights assigned to it. A transfer function is used for calculating the weighted sum of the inputs and the bias. After the transfer function has calculated the sum, the activation function obtains the result. Based on the output received, the activation functions fire the appropriate result from the node. For example, if the output received is above 0.5, the activation function fires a 1 otherwise it remains 0.

7.2 Convolutional Neural Network (CNN)

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. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

Convolution Layer — The Kernel



Convoluting a 5x5x1 image with a 3x3x1 kernel to get a 3x3x1 convolved feature

Image Dimensions = 5 (Height) x 5 (Breadth) x 1 (Number of channels, eg. RGB)

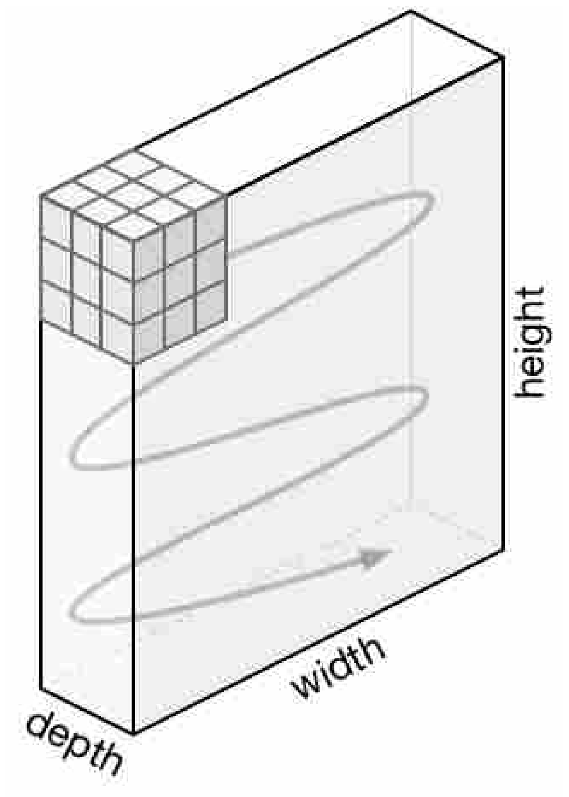
In the above demonstration, the green section resembles our **5x5x1 input image, I**. The element involved in carrying out the convolution operation in the first part of a Convolutional Layer is called the **Kernel/Filter, K**, represented in the color yellow. We have selected **K as a 3x3x1 matrix.**

Kernel/Filter, K = 1 0 1

0 1 0

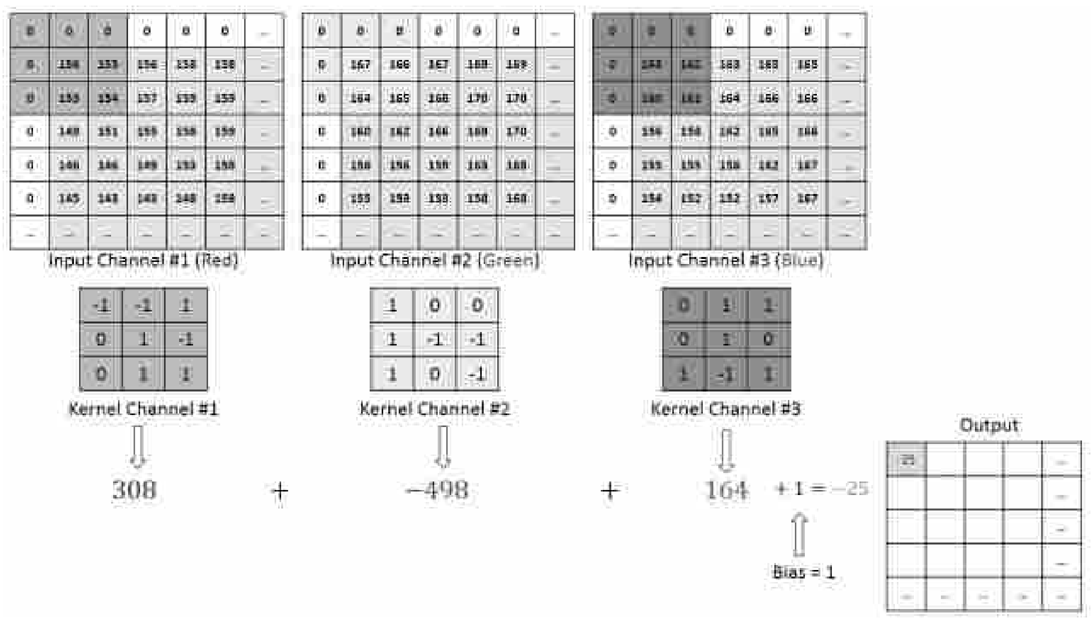
1 0 1

The Kernel shifts 9 times because of **Stride Length = 1 (Non-Strided)**, every time performing a **matrix multiplication operation between K and the portion P of the image** over which the kernel is hovering.



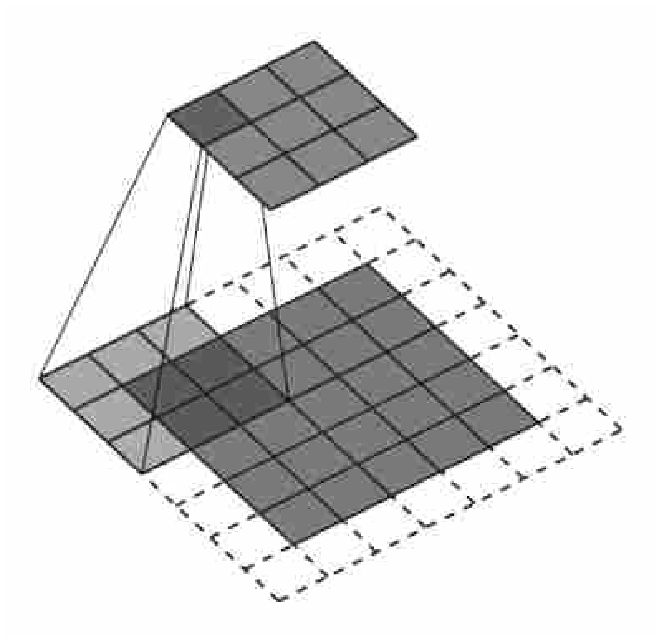
Movement of the Kernel

The filter moves to the right with a certain Stride Value till it parses the complete width. Moving on, it hops down to the beginning (left) of the image with the same Stride Value and repeats the process until the entire image is traversed.



Convolution operation on a MxNx3 image matrix with a 3x3x3 Kernel

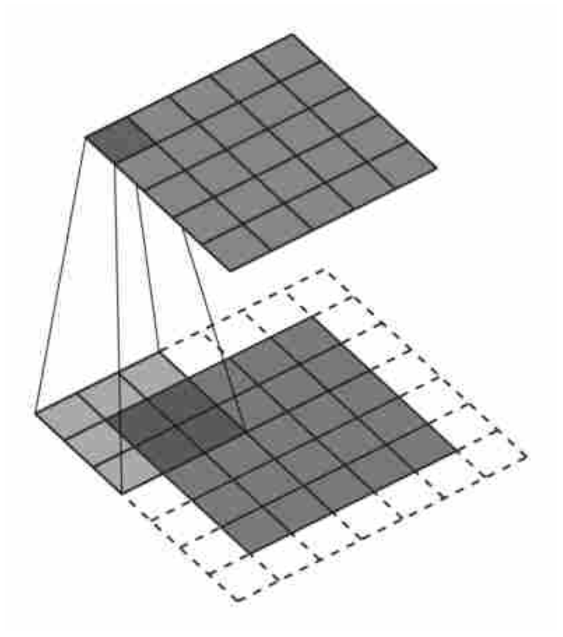
In the case of images with multiple channels (e.g. RGB), the Kernel has the same depth as that of the input image. Matrix Multiplication is performed between Kn and In stack ([K1, I1]; [K2, I2]; [K3, I3]) and all the results are summed with the bias to give us a squashed one-depth channel Convoluted Feature Output.



Convolution Operation with Stride Length = 2

The objective of the Convolution Operation is to **extract the high-level features** such as edges, from the input image. ConvNets need not be limited to only one Convolutional Layer. Conventionally, the first ConvLayer is responsible for capturing the Low-Level features such as edges, color, gradient orientation, etc. With added layers, the architecture adapts to the High-Level features as well, giving us a network, which has the wholesome understanding of images in the dataset, similar to how we would.

There are two types of results to the operation — one in which the convolved feature is reduced in dimensionality as compared to the input, and the other in which the dimensionality is either increased or remains the same. This is done by applying **Valid Padding** in case of the former, or **Same Padding** in the case of the latter.



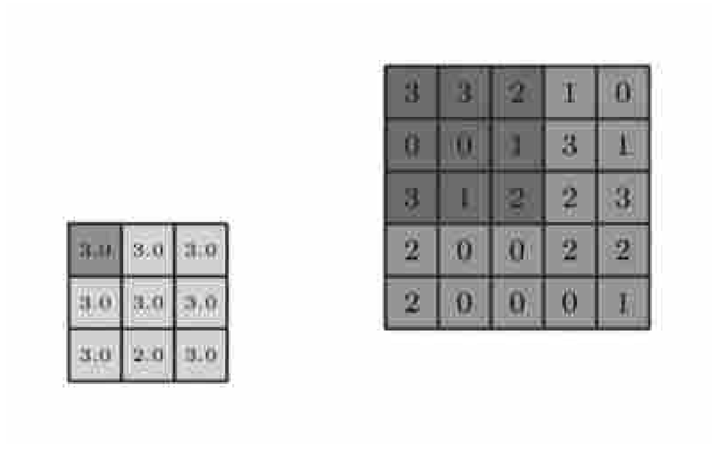
SAME padding: 5x5x1 image is padded with 0s to create a 6x6x1 image

When we augment the 5x5x1 image into a 6x6x1 image and then apply the 3x3x1 kernel over it, we find that the convolved matrix turns out to be of dimensions 5x5x1. Hence the name — **Same Padding**.

On the other hand, if we perform the same operation without padding, we are presented with a matrix which has dimensions of the Kernel (3x3x1) itself — **Valid Padding**.

The following repository houses many such GIFs which would help you get a better understanding of how Padding and Stride Length work together to achieve results relevant to our needs.

Pooling Layer

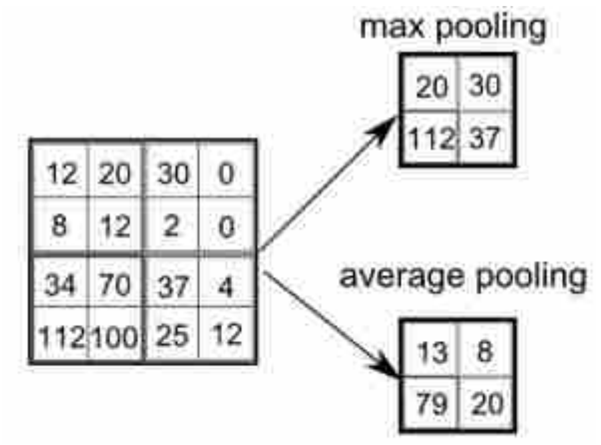


3x3 pooling over 5x5 convolved feature

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. Furthermore, it is useful for **extracting dominant features** which are rotational and positional invariant, thus maintaining the process of effectively training of the model.

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**. It discards the noisy activations altogether and also performs de-noising along with dimensionality reduction. 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**.

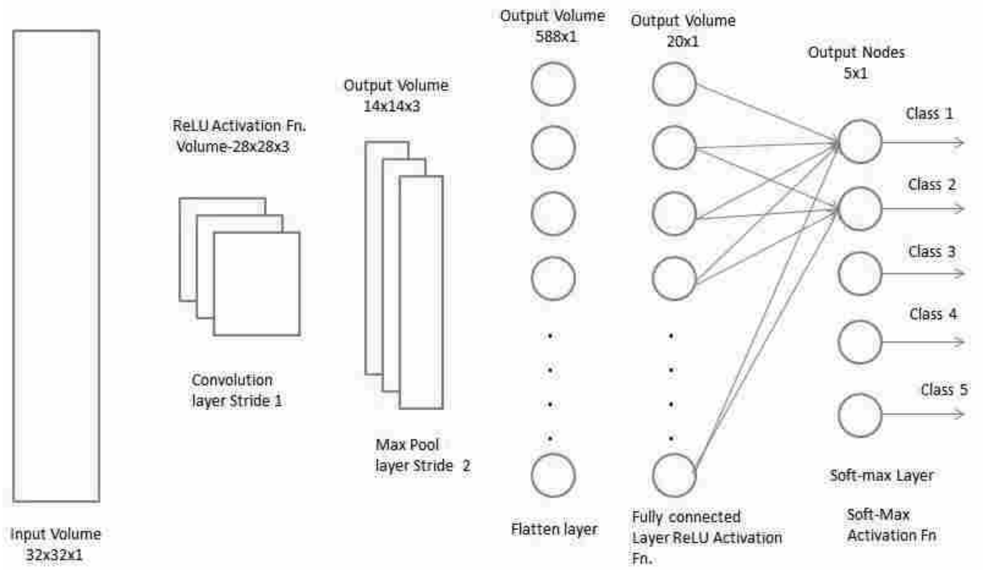


Types of Pooling

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.

After going through the above process, we have successfully enabled the model to understand the features. Moving on, we are going to flatten the final output and feed it to a regular Neural Network for classification purposes.

Classification — Fully Connected Layer (FC Layer)



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.

Now that we have converted our input image into a suitable form for our Multi-Level Perceptron, we shall flatten the image into a column vector. The flattened output is fed to a feed-forward neural network and backpropagation applied to every iteration of training. Over a series of epochs, the model is able to distinguish between dominating and certain low-level features in images and classify them using the **Softmax Classification** technique.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlaps to cover the entire visual area.

One of the main parts of Neural Networks is Convolutional neural networks (CNN). CNNs use image recognition and classification in order to detect objects, recognize faces, etc. They are made up of neurons with learnable weights and biases. Each specific neuron receives numerous inputs and then takes a weighted sum over them, where it passes it through an activation function and responds back with an output.

The first layer in a CNN network is the CONVOLUTIONAL LAYER, which is the core building block and does most of the computational heavy lifting. Data or imaged is convolved using filters or kernels.

There are various architectures of CNNs available which have been key in building algorithms which power and shall power AI as a whole in the foreseeable future. Some of them have been listed below:

1. LeNet
2. MobileNetv2
3. AlexNet
4. VGGNet
5. GoogLeNet
6. ResNet
7. ZFNet

Here we are using MobileNetV2 as transfer learning model.

7.3 Optimizers

Optimizers are algorithms or methods used to change the attributes of the **neural network** such as weights and learning rate to reduce the losses. **Optimizers** are used to solve optimization problems by minimizing the function.

Gradient Descent

Gradient Descent is the most basic but most used optimization algorithm. It's used heavily in linear regression and classification algorithms. Backpropagation in neural networks also uses a gradient descent algorithm.

Gradient descent is a first-order optimization algorithm which is dependent on the first order derivative of a loss function. It calculates that which way the weights should be altered so that the function can reach a minima. Through backpropagation, the loss is transferred from one layer to another and the model's parameters also known as weights are modified depending on the losses so that the loss can be minimized.

Advantages:

1. Easy computation.

2. Easy to implement.

3. Easy to understand.

Disadvantages:

1. May trap at local minima.

2. Weights are changed after calculating gradient on the whole dataset. So, if the dataset is too large than this may take years to converge to the minima.

3. Requires large memory to calculate gradient on the whole dataset.

Stochastic Gradient Descent

It's a variant of Gradient Descent. It tries to update the model's parameters more frequently. In this, the model parameters are altered after computation of loss on each training example. So, if the dataset contains 1000 rows SGD will update the model parameters 1000 times in one cycle of dataset instead of one time as in Gradient Descent.

As the model parameters are frequently updated parameters have high variance and fluctuations in loss functions at different intensities.

Advantages:

1. Frequent updates of model parameters hence, converges in less time.

2. Requires less memory as no need to store values of loss functions.

3. May get new minima.

Disadvantages:

1. High variance in model parameters.

2. May shoot even after achieving global minima.

3. To get the same convergence as gradient descent needs to slowly reduce the value of learning rate.

Mini-Batch Gradient Descent

It's best among all the variations of gradient descent algorithms. It is an improvement on both SGD and standard gradient descent. It updates the model parameters after every batch. So, the dataset is divided into various batches and after every batch, the parameters are updated.

Advantages:

1. Frequently updates the model parameters and also has less variance.

2. Requires medium amount of memory.

All types of Gradient Descent have some challenges:

1. Choosing an optimum value of the learning rate. If the learning rate is too small than gradient descent may take ages to converge.

2. Have a constant learning rate for all the parameters. There may be some parameters which we may not want to change at the same rate.

3. May get trapped at local minima.

Momentum

Momentum was invented for reducing high variance in SGD and softens the convergence. It accelerates the convergence towards the relevant direction and reduces the fluctuation to the irrelevant direction. One more hyperparameter is used in this method known as momentum symbolized by '**γ**'.

$V(t)=\gamma V(t-1) + \alpha . \nabla J(\theta)$

Now, the weights are updated by **$\theta=\theta-V(t)$.**

The momentum term **γ** is usually set to 0.9 or a similar value.

Advantages:

1. Reduces the oscillations and high variance of the parameters.

2. Converges faster than gradient descent.

Disadvantages:

1. One more hyper-parameter is added which needs to be selected manually and accurately.

Nesterov Accelerated Gradient

Momentum may be a good method but if the momentum is too high the algorithm may miss the local minima and may continue to rise up. So, to resolve this issue the NAG algorithm was developed. It is a look ahead method. We know we'll be using **$\gamma V(t-1)$** for modifying the weights so, **$\theta-\gamma V(t-1)$** approximately tells us the future location. Now, we'll calculate the cost based on this future parameter rather than the current one.

$V(t)=\gamma V(t-1)+\alpha . \nabla J(\theta-\gamma V(t-1))$ and then update the parameters using **$\theta=\theta-V(t)$.**

Adagrad

One of the disadvantages of all the optimizers explained is that the learning rate is constant for all parameters and for each cycle. This optimizer changes the learning rate. It changes the learning rate **'η'** for each parameter and at every time step **'t'.** It's a type second order optimization algorithm. It works on the derivative of an error function.

Advantages:

1. Learning rate changes for each training parameter.
2. Don't need to manually tune the learning rate.
3. Able to train on sparse data.

Disadvantages:

1. Computationally expensive as a need to calculate the second order derivative.
2. The learning rate is always decreasing results in slow training.

AdaDelta

It is an extension of **AdaGrad** which tends to remove the *decaying learning Rate* problem of it. Instead of accumulating all previously squared gradients, **Adadelta** limits the window of accumulated past gradients to some fixed size **w**. In this exponentially moving average is used rather than the sum of all the gradients.

Advantages:

1. Now the learning rate does not decay and the training does not stop.

Disadvantages:

1. Computationally expensive.

Adam

Adam (Adaptive Moment Estimation) works with momentums of first and second order. The intuition behind the Adam is that we don't want to roll so fast just because we can jump over the minimum, we want to decrease the velocity a little bit for a careful search.

Advantages:

1. The method is too fast and converges rapidly.
2. Rectifies vanishing learning rate, high variance.

Disadvantages:

1. Computationally costly.

7.4 Loss Functions

Loss functions play an important role in any statistical model - they define an objective which the performance of the model is evaluated against and the parameters learned by the model are determined by minimizing a chosen loss function.

Loss functions define what a good prediction is and isn’t. In short, choosing the right loss function dictates how well your estimator will be. This article will probe into loss functions, the role they play in validating predictions, and the various loss functions used.

Loss functions for classification

Classification problems involve predicting a discrete class output. It involves dividing the dataset into different and unique classes based on different parameters so that a new and unseen record can be put into one of the classes.

A mail can be classified as a spam or not a spam and a person’s dietary preferences can be put in one of three categories - vegetarian, non-vegetarian and vegan. Let’s take a look at loss functions that can be used for classification problems.

Binary Cross Entropy Loss

This is the most common loss function used for classification problems that have two classes. The word “entropy”, seemingly out-of-place, has a statistical interpretation.

Entropy is the measure of randomness in the information being processed, and cross entropy is a measure of the difference of the randomness between two random variables.

If the divergence of the predicted probability from the actual label increases, the cross-entropy loss increases. Going by this, predicting a probability of .011 when the actual observation label is 1 would result in a high loss value. In an ideal situation, a “perfect” model would have a log loss of 0.

Categorical Cross Entropy Loss

Categorical Cross Entropy loss is essentially Binary Cross Entropy Loss expanded to multiple classes. One requirement when categorical cross entropy loss function is used is that the labels should be one-hot encoded.
This way, only one element will be non-zero as other elements in the vector would be multiplied by zero.

Hinge Loss
Another commonly used loss function for classification is the hinge loss. Hinge loss is primarily developed for support vector machines for calculating the maximum margin from the hyperplane to the classes.

Loss functions penalize wrong predictions and does not do so for the right predictions. So, the score of the target label should be greater than the sum of all the incorrect labels by a margin of (at the least) one.

This margin is the maximum margin from the hyperplane to the data points, which is why hinge loss is preferred for SVMs.

7.5 Activation Functions

An activation function in a neural network defines how the weighted sum of the input is transformed into an output from a node or nodes in a layer of the network.

Activation functions are a critical part of the design of a neural network.

The choice of activation function in the hidden layer will control how well the network model learns the training dataset. The choice of activation function in the output layer will define the type of predictions the model can make.

Technically, the activation function is used within or after the internal processing of each node in the network, although networks are designed to use the same activation function for all nodes in a layer.

A network may have three types of layers: input layers that take raw input from the domain, hidden layers that take input from another layer and pass output to another layer, and output layers that make a prediction.

All hidden layers typically use the same activation function. The output layer will typically use a different activation function from the hidden layers and is dependent upon the type of prediction required by the model.

Most commonly used activation functions are as follows –
- Rectified Linear Activation (**ReLU**)
- Logistic (**Sigmoid**)
- Hyperbolic Tangent (**Tanh**)

ReLU Activation Function
The rectified linear activation function, or ReLU activation function, is perhaps the most common function used for hidden layers.

It is common because it is both simple to implement and effective at overcoming the limitations of other previously popular activation functions, such as Sigmoid and Tanh. Specifically, it is less susceptible to vanishing gradients that prevent deep models from being trained, although it can suffer from other problems like saturated or "*dead*" units.
The ReLU function is calculated as follows:

- max(0.0, x)

This means that if the input value (x) is negative, then a value 0.0 is returned, otherwise, the value is returned.

Sigmoid Activation Function
The sigmoid activation function is also called the logistic function.

It is the same function used in the logistic regression classification algorithm.

The function takes any real value as input and outputs values in the range 0 to 1. The larger the input (more positive), the closer the output value will be to 1.0, whereas the smaller the input (more negative), the closer the output will be to 0.0.

The sigmoid activation function is calculated as follows:

1.0 / (1.0 + e^-x)

Where e is a mathematical constant, which is the base of the natural logarithm.

Tanh Activation Function

The hyperbolic tangent activation function is also referred to simply as the Tanh (also “*tanh*” and “*TanH*“) function.

It is very similar to the sigmoid activation function and even has the same S-shape.

The function takes any real value as input and outputs values in the range -1 to 1. The larger the input (more positive), the closer the output value will be to 1.0, whereas the smaller the input (more negative), the closer the output will be to -1.0.

The Tanh activation function is calculated as follows:

- (e^x – e^-x) / (e^x + e^-x)

Where e is a mathematical constant that is the base of the natural logarithm.

7.5 Source Code Snippets

Import libraies

In [1]:
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import cv2
import os
import matplotlib.pyplot as plt
import numpy as np
from tensorflow.keras.preprocessing.image import ImageDataGenerator,load_img,img_to_arr
```

In [2]:
```
print(tf.config.list_physical_devices('GPU')) # checking tensorflow gpu is enabled or n
print(tf.__version__)
```

```
[PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
2.4.0
```

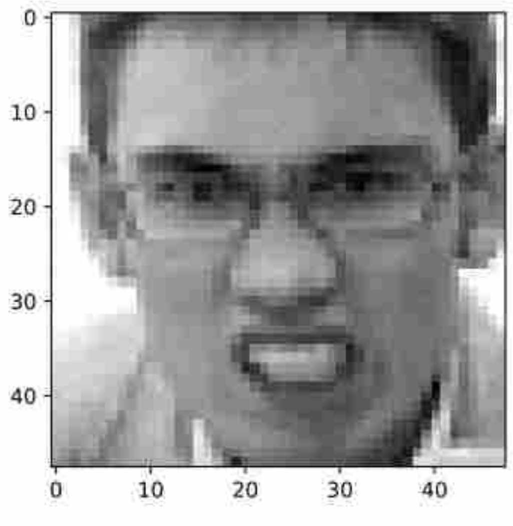
In [3]:
```
img_array = cv2.imread("dataset/train/0/Training_345125.jpg")
```

In [4]:
```
img_array.shape
```

Out[4]:
```
(48, 48, 3)
```

In [5]:
```
plt.imshow(img_array)
```

Out[5]:
```
<matplotlib.image.AxesImage at 0x1be0d8aee20>
```



In [6]:
```
data_directory = "dataset/train" # training datasets
```

In [7]:
```
classes = ["0","1","2","3","4","5","6"] # List of classes
```

data generator

In [8]:
```
# in our training data folder "1" has only 436 images, but others have more. It will ef
datagen = ImageDataGenerator(
        rotation_range=40,
        width_shift_range=0.2,
        height_shift_range=0.2,
        rescale=1./255,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        fill_mode='nearest')
```

In [9]:
```
# Add 11 images names to a list from 1 folder
imagesgen = ["Training_5420780.jpg","Training_5387344.jpg","Training_11050021.jpg","Tra
```

In [10]:
```
# create a folder named genPics1
try:
    directory = "genPic1"
    path_dir = os.path.join(data_directory,directory)
    os.mkdir(path_dir)
except Exception as e:
    print(e)
```

In [11]:
```
def imagesgenarator(img_list):
    for i in range(len(img_list)):
        pic = load_img(data_directory+"/1/"+img_list[i])
        pic_array = img_to_array(pic)
        pic_array = pic_array.reshape((1,) + pic_array.shape) # Converting into 4 dimen
        # Generate 11 images
        # batch_size: At a time, how many image should be created.
        count = 0
        for batch in datagen.flow(pic_array, batch_size=5,save_to_dir=data_directory+"/
            count += 1
            if count > 95:
                break
```

In [12]:
```
imagesgenarator(imagesgen)
```

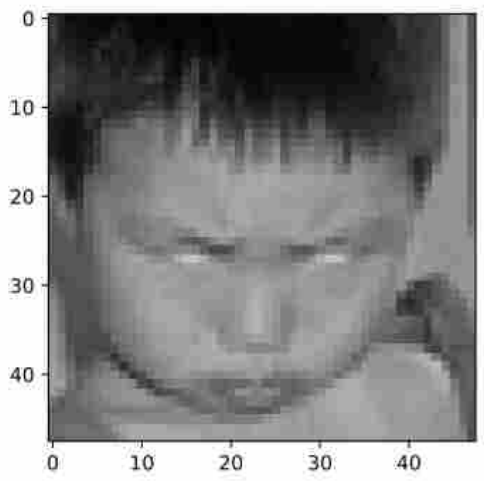
In [13]:
```
# Moved images from genPic1 to 1 folder
target_fol = r"dataset\train\1" + "\\"
source_fol = r"dataset\train\genPic1" + "\\"
def move_files(source_fol,target_fol):
    try:
        for path, dir, files in os.walk(source_fol):
            for file in files:
                if not os.path.isfile(target_fol + file):
                    os.rename(path + '\\'+ file,target_fol+file)
        print("All files are moved")
    except Exception as e:
        print(e)
```

In [14]:
```
move_files(source_fol,target_fol)
```

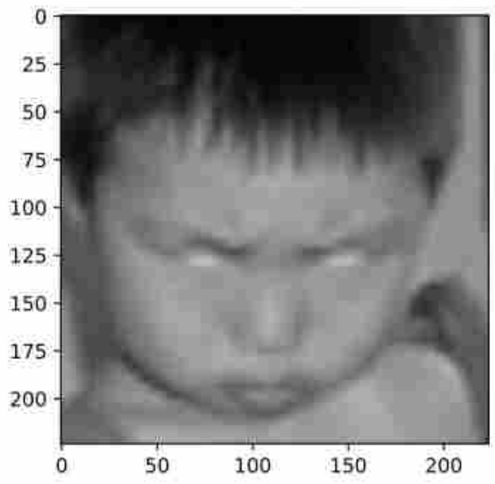
```
All files are moved
```

In [15]:
```
# Removed empty directory
try:
    os.rmdir(source_fol)
except Exception as e:
    print(e)
```

In [16]:
```
for catagory in classes:
    path = os.path.join(data_directory,catagory)
    for img in os.listdir(path):
        img_array = cv2.imread(os.path.join(path,img))
        plt.imshow(cv2.cvtColor(img_array,cv2.COLOR_BGR2RGB))
        plt.show()
        break
    break
```



In [17]:
```
img_size = (224,224) # ImageNet 224 x 224
new_img_array = cv2.resize(img_array,img_size)
plt.imshow(cv2.cvtColor(new_img_array,cv2.COLOR_BGR2RGB))
plt.show()
```



In [18]:
```
new_img_array.shape
```

Out[18]:
```
(224, 224, 3)
```

In [19]:
```
# read all the images and convert them into array
training_data = [] # data array
def create_training_data():
    for catagory in classes:
        path = os.path.join(data_directory,catagory)
        class_num = classes.index(catagory)
        for img in os.listdir(path):
            try:
                img_array = cv2.imread(os.path.join(path,img))
                new_img_array = cv2.resize(img_array,img_size)
                training_data.append([new_img_array,class_num])
            except Exception as e:
                print(f"...{e}")
```

In [20]:
```
create_training_data()
```

In [21]:
```
print(len(training_data))
```

```
10884
```

In [22]:
```
import random
random.shuffle(training_data)
```

In [23]:
```
X = [] # features
y = [] # label
```

```
for features,label in training_data:
    X.append(features)
    y.append(label)

X = np.array(X).reshape(-1,img_size[0],img_size[1],3)   ## convert it to 4 dimenion for
```

In [24]:

```
X.shape
```

Out[24]:

```
(10884, 224, 224, 3)
```

In [25]:

```
# Normalizing the data
X = X / 255.0
```

In [26]:

```
y = np.array(y)
```

In [27]:

```
y.shape
```

Out[27]:

```
(10884,)
```

Deep learning model for training - Transfer learning

In [28]:

```
model = tf.keras.applications.MobileNetV2() # pre-trained model
```

In [29]:

```
model.summary()
```

```
Model: "mobilenetv2_1.00_224"
__________________________________________________________________________________________
__________
Layer (type)                    Output Shape         Param #     Connected to
==========================================================================================
==========
input_1 (InputLayer)            [(None, 224, 224, 3) 0
__________________________________________________________________________________________
__________
Conv1 (Conv2D)                  (None, 112, 112, 32) 864         input_1[0][0]
__________________________________________________________________________________________
__________
bn_Conv1 (BatchNormalization)   (None, 112, 112, 32) 128         Conv1[0][0]
__________________________________________________________________________________________
__________
Conv1_relu (ReLU)               (None, 112, 112, 32) 0           bn_Conv1[0][0]
__________________________________________________________________________________________
__________
expanded_conv_depthwise (Depthw (None, 112, 112, 32) 288         Conv1_relu[0][0]
__________________________________________________________________________________________
__________
expanded_conv_depthwise_BN (Bat (None, 112, 112, 32) 128         expanded_conv_depthwise
[0][0]
__________________________________________________________________________________________
__________
```

```
expanded_conv_depthwise_relu (R (None, 112, 112, 32) 0          expanded_conv_depthwise
_BN[0][0]
__________________________________________________________________________________________
__________
expanded_conv_project (Conv2D)  (None, 112, 112, 16) 512        expanded_conv_depthwise
_relu[0][0
__________________________________________________________________________________________
__________
expanded_conv_project_BN (Batch (None, 112, 112, 16) 64         expanded_conv_project
[0][0]
__________________________________________________________________________________________
__________
block_1_expand (Conv2D)         (None, 112, 112, 96) 1536       expanded_conv_project_B
N[0][0]
__________________________________________________________________________________________
__________
block_1_expand_BN (BatchNormali (None, 112, 112, 96) 384        block_1_expand[0][0]
__________________________________________________________________________________________
__________
block_1_expand_relu (ReLU)      (None, 112, 112, 96) 0          block_1_expand_BN[0][0]
__________________________________________________________________________________________
__________
block_1_pad (ZeroPadding2D)     (None, 113, 113, 96) 0          block_1_expand_relu[0]
[0]
__________________________________________________________________________________________
__________
block_1_depthwise (DepthwiseCon (None, 56, 56, 96)   864        block_1_pad[0][0]
__________________________________________________________________________________________
__________
block_1_depthwise_BN (BatchNorm (None, 56, 56, 96)   384        block_1_depthwise[0][0]
__________________________________________________________________________________________
__________
block_1_depthwise_relu (ReLU)   (None, 56, 56, 96)   0          block_1_depthwise_BN[0]
[0]
__________________________________________________________________________________________
__________
block_1_project (Conv2D)        (None, 56, 56, 24)   2304       block_1_depthwise_relu
[0][0]
__________________________________________________________________________________________
__________
block_1_project_BN (BatchNormal (None, 56, 56, 24)   96         block_1_project[0][0]
__________________________________________________________________________________________
__________
block_2_expand (Conv2D)         (None, 56, 56, 144)  3456       block_1_project_BN[0]
[0]
__________________________________________________________________________________________
__________
block_2_expand_BN (BatchNormali (None, 56, 56, 144)  576        block_2_expand[0][0]
__________________________________________________________________________________________
__________
block_2_expand_relu (ReLU)      (None, 56, 56, 144)  0          block_2_expand_BN[0][0]
__________________________________________________________________________________________
__________
block_2_depthwise (DepthwiseCon (None, 56, 56, 144)  1296       block_2_expand_relu[0]
[0]
__________________________________________________________________________________________
__________
block_2_depthwise_BN (BatchNorm (None, 56, 56, 144)  576        block_2_depthwise[0][0]
__________________________________________________________________________________________
__________
block_2_depthwise_relu (ReLU)   (None, 56, 56, 144)  0          block_2_depthwise_BN[0]
[0]
__________________________________________________________________________________________
__________
block_2_project (Conv2D)        (None, 56, 56, 24)   3456       block_2_depthwise_relu
```

```
[0][0]
__________________________________________________________________________________________________
block_2_project_BN (BatchNormal (None, 56, 56, 24)   96          block_2_project[0][0]
__________________________________________________________________________________________________
block_2_add (Add)               (None, 56, 56, 24)   0           block_1_project_BN[0]
[0]
                                                                 block_2_project_BN[0]
[0]
__________________________________________________________________________________________________
block_3_expand (Conv2D)         (None, 56, 56, 144)  3456        block_2_add[0][0]
__________________________________________________________________________________________________
block_3_expand_BN (BatchNormali (None, 56, 56, 144)  576         block_3_expand[0][0]
__________________________________________________________________________________________________
block_3_expand_relu (ReLU)      (None, 56, 56, 144)  0           block_3_expand_BN[0][0]
__________________________________________________________________________________________________
block_3_pad (ZeroPadding2D)     (None, 57, 57, 144)  0           block_3_expand_relu[0]
[0]
__________________________________________________________________________________________________
block_3_depthwise (DepthwiseCon (None, 28, 28, 144)  1296        block_3_pad[0][0]
__________________________________________________________________________________________________
block_3_depthwise_BN (BatchNorm (None, 28, 28, 144)  576         block_3_depthwise[0][0]
__________________________________________________________________________________________________
block_3_depthwise_relu (ReLU)   (None, 28, 28, 144)  0           block_3_depthwise_BN[0]
[0]
__________________________________________________________________________________________________
block_3_project (Conv2D)        (None, 28, 28, 32)   4608        block_3_depthwise_relu
[0][0]
__________________________________________________________________________________________________
block_3_project_BN (BatchNormal (None, 28, 28, 32)   128         block_3_project[0][0]
__________________________________________________________________________________________________
block_4_expand (Conv2D)         (None, 28, 28, 192)  6144        block_3_project_BN[0]
[0]
__________________________________________________________________________________________________
block_4_expand_BN (BatchNormali (None, 28, 28, 192)  768         block_4_expand[0][0]
__________________________________________________________________________________________________
block_4_expand_relu (ReLU)      (None, 28, 28, 192)  0           block_4_expand_BN[0][0]
__________________________________________________________________________________________________
block_4_depthwise (DepthwiseCon (None, 28, 28, 192)  1728        block_4_expand_relu[0]
[0]
__________________________________________________________________________________________________
block_4_depthwise_BN (BatchNorm (None, 28, 28, 192)  768         block_4_depthwise[0][0]
__________________________________________________________________________________________________
block_4_depthwise_relu (ReLU)   (None, 28, 28, 192)  0           block_4_depthwise_BN[0]
[0]
__________________________________________________________________________________________________
block_4_project (Conv2D)        (None, 28, 28, 32)   6144        block_4_depthwise_relu
[0][0]
```

```
__________________________________________________________________________________________________
block_4_project_BN (BatchNormal (None, 28, 28, 32)   128         block_4_project[0][0]
__________________________________________________________________________________________________
block_4_add (Add)               (None, 28, 28, 32)   0           block_3_project_BN[0]
[0]
                                                                 block_4_project_BN[0]
[0]
__________________________________________________________________________________________________
block_5_expand (Conv2D)         (None, 28, 28, 192)  6144        block_4_add[0][0]
__________________________________________________________________________________________________
block_5_expand_BN (BatchNormali (None, 28, 28, 192)  768         block_5_expand[0][0]
__________________________________________________________________________________________________
block_5_expand_relu (ReLU)      (None, 28, 28, 192)  0           block_5_expand_BN[0][0]
__________________________________________________________________________________________________
block_5_depthwise (DepthwiseCon (None, 28, 28, 192)  1728        block_5_expand_relu[0]
[0]
__________________________________________________________________________________________________
block_5_depthwise_BN (BatchNorm (None, 28, 28, 192)  768         block_5_depthwise[0][0]
__________________________________________________________________________________________________
block_5_depthwise_relu (ReLU)   (None, 28, 28, 192)  0           block_5_depthwise_BN[0]
[0]
__________________________________________________________________________________________________
block_5_project (Conv2D)        (None, 28, 28, 32)   6144        block_5_depthwise_relu
[0][0]
__________________________________________________________________________________________________
block_5_project_BN (BatchNormal (None, 28, 28, 32)   128         block_5_project[0][0]
__________________________________________________________________________________________________
block_5_add (Add)               (None, 28, 28, 32)   0           block_4_add[0][0]
                                                                 block_5_project_BN[0]
[0]
__________________________________________________________________________________________________
block_6_expand (Conv2D)         (None, 28, 28, 192)  6144        block_5_add[0][0]
__________________________________________________________________________________________________
block_6_expand_BN (BatchNormali (None, 28, 28, 192)  768         block_6_expand[0][0]
__________________________________________________________________________________________________
block_6_expand_relu (ReLU)      (None, 28, 28, 192)  0           block_6_expand_BN[0][0]
__________________________________________________________________________________________________
block_6_pad (ZeroPadding2D)     (None, 29, 29, 192)  0           block_6_expand_relu[0]
[0]
__________________________________________________________________________________________________
block_6_depthwise (DepthwiseCon (None, 14, 14, 192)  1728        block_6_pad[0][0]
__________________________________________________________________________________________________
block_6_depthwise_BN (BatchNorm (None, 14, 14, 192)  768         block_6_depthwise[0][0]
__________________________________________________________________________________________________
block_6_depthwise_relu (ReLU)   (None, 14, 14, 192)  0           block_6_depthwise_BN[0]
[0]
__________________________________________________________________________________________________
```

```
__________
block_6_project (Conv2D)        (None, 14, 14, 64)   12288       block_6_depthwise_relu
[0][0]
__________________________________________________________________________________________________
__________
block_6_project_BN (BatchNormal (None, 14, 14, 64)   256         block_6_project[0][0]
__________________________________________________________________________________________________
__________
block_7_expand (Conv2D)         (None, 14, 14, 384)  24576       block_6_project_BN[0]
[0]
__________________________________________________________________________________________________
__________
block_7_expand_BN (BatchNormali (None, 14, 14, 384)  1536        block_7_expand[0][0]
__________________________________________________________________________________________________
__________
block_7_expand_relu (ReLU)      (None, 14, 14, 384)  0           block_7_expand_BN[0][0]
__________________________________________________________________________________________________
__________
block_7_depthwise (DepthwiseCon (None, 14, 14, 384)  3456        block_7_expand_relu[0]
[0]
__________________________________________________________________________________________________
__________
block_7_depthwise_BN (BatchNorm (None, 14, 14, 384)  1536        block_7_depthwise[0][0]
__________________________________________________________________________________________________
__________
block_7_depthwise_relu (ReLU)   (None, 14, 14, 384)  0           block_7_depthwise_BN[0]
[0]
__________________________________________________________________________________________________
__________
block_7_project (Conv2D)        (None, 14, 14, 64)   24576       block_7_depthwise_relu
[0][0]
__________________________________________________________________________________________________
__________
block_7_project_BN (BatchNormal (None, 14, 14, 64)   256         block_7_project[0][0]
__________________________________________________________________________________________________
__________
block_7_add (Add)               (None, 14, 14, 64)   0           block_6_project_BN[0]
[0]
                                                                 block_7_project_BN[0]
[0]
__________________________________________________________________________________________________
__________
block_8_expand (Conv2D)         (None, 14, 14, 384)  24576       block_7_add[0][0]
__________________________________________________________________________________________________
__________
block_8_expand_BN (BatchNormali (None, 14, 14, 384)  1536        block_8_expand[0][0]
__________________________________________________________________________________________________
__________
block_8_expand_relu (ReLU)      (None, 14, 14, 384)  0           block_8_expand_BN[0][0]
__________________________________________________________________________________________________
__________
block_8_depthwise (DepthwiseCon (None, 14, 14, 384)  3456        block_8_expand_relu[0]
[0]
__________________________________________________________________________________________________
__________
block_8_depthwise_BN (BatchNorm (None, 14, 14, 384)  1536        block_8_depthwise[0][0]
__________________________________________________________________________________________________
__________
block_8_depthwise_relu (ReLU)   (None, 14, 14, 384)  0           block_8_depthwise_BN[0]
[0]
__________________________________________________________________________________________________
__________
block_8_project (Conv2D)        (None, 14, 14, 64)   24576       block_8_depthwise_relu
[0][0]
__________________________________________________________________________________________________
```

```
__________
block_8_project_BN (BatchNormal (None, 14, 14, 64)   256         block_8_project[0][0]
__________________________________________________________________________________________
__________
block_8_add (Add)               (None, 14, 14, 64)   0           block_7_add[0][0]
                                                                 block_8_project_BN[0]
[0]
__________________________________________________________________________________________
__________
block_9_expand (Conv2D)         (None, 14, 14, 384)  24576       block_8_add[0][0]
__________________________________________________________________________________________
__________
block_9_expand_BN (BatchNormali (None, 14, 14, 384)  1536        block_9_expand[0][0]
__________________________________________________________________________________________
__________
block_9_expand_relu (ReLU)      (None, 14, 14, 384)  0           block_9_expand_BN[0][0]
__________________________________________________________________________________________
__________
block_9_depthwise (DepthwiseCon (None, 14, 14, 384)  3456        block_9_expand_relu[0]
[0]
__________________________________________________________________________________________
__________
block_9_depthwise_BN (BatchNorm (None, 14, 14, 384)  1536        block_9_depthwise[0][0]
__________________________________________________________________________________________
__________
block_9_depthwise_relu (ReLU)   (None, 14, 14, 384)  0           block_9_depthwise_BN[0]
[0]
__________________________________________________________________________________________
__________
block_9_project (Conv2D)        (None, 14, 14, 64)   24576       block_9_depthwise_relu
[0][0]
__________________________________________________________________________________________
__________
block_9_project_BN (BatchNormal (None, 14, 14, 64)   256         block_9_project[0][0]
__________________________________________________________________________________________
__________
block_9_add (Add)               (None, 14, 14, 64)   0           block_8_add[0][0]
                                                                 block_9_project_BN[0]
[0]
__________________________________________________________________________________________
__________
block_10_expand (Conv2D)        (None, 14, 14, 384)  24576       block_9_add[0][0]
__________________________________________________________________________________________
__________
block_10_expand_BN (BatchNormal (None, 14, 14, 384)  1536        block_10_expand[0][0]
__________________________________________________________________________________________
__________
block_10_expand_relu (ReLU)     (None, 14, 14, 384)  0           block_10_expand_BN[0]
[0]
__________________________________________________________________________________________
__________
block_10_depthwise (DepthwiseCo (None, 14, 14, 384)  3456        block_10_expand_relu[0]
[0]
__________________________________________________________________________________________
__________
block_10_depthwise_BN (BatchNor (None, 14, 14, 384)  1536        block_10_depthwise[0]
[0]
__________________________________________________________________________________________
__________
block_10_depthwise_relu (ReLU)  (None, 14, 14, 384)  0           block_10_depthwise_BN
[0][0]
__________________________________________________________________________________________
__________
block_10_project (Conv2D)       (None, 14, 14, 96)   36864       block_10_depthwise_relu
[0][0]
```

```
__________________________________________________________________________________________________
block_10_project_BN (BatchNorma (None, 14, 14, 96)   384         block_10_project[0][0]
__________________________________________________________________________________________________
block_11_expand (Conv2D)        (None, 14, 14, 576)  55296       block_10_project_BN[0]
[0]
__________________________________________________________________________________________________
block_11_expand_BN (BatchNormal (None, 14, 14, 576)  2304        block_11_expand[0][0]
__________________________________________________________________________________________________
block_11_expand_relu (ReLU)     (None, 14, 14, 576)  0           block_11_expand_BN[0]
[0]
__________________________________________________________________________________________________
block_11_depthwise (DepthwiseCo (None, 14, 14, 576)  5184        block_11_expand_relu[0]
[0]
__________________________________________________________________________________________________
block_11_depthwise_BN (BatchNor (None, 14, 14, 576)  2304        block_11_depthwise[0]
[0]
__________________________________________________________________________________________________
block_11_depthwise_relu (ReLU)  (None, 14, 14, 576)  0           block_11_depthwise_BN
[0][0]
__________________________________________________________________________________________________
block_11_project (Conv2D)       (None, 14, 14, 96)   55296       block_11_depthwise_relu
[0][0]
__________________________________________________________________________________________________
block_11_project_BN (BatchNorma (None, 14, 14, 96)   384         block_11_project[0][0]
__________________________________________________________________________________________________
block_11_add (Add)              (None, 14, 14, 96)   0           block_10_project_BN[0]
[0]
                                                                 block_11_project_BN[0]
[0]
__________________________________________________________________________________________________
block_12_expand (Conv2D)        (None, 14, 14, 576)  55296       block_11_add[0][0]
__________________________________________________________________________________________________
block_12_expand_BN (BatchNormal (None, 14, 14, 576)  2304        block_12_expand[0][0]
__________________________________________________________________________________________________
block_12_expand_relu (ReLU)     (None, 14, 14, 576)  0           block_12_expand_BN[0]
[0]
__________________________________________________________________________________________________
block_12_depthwise (DepthwiseCo (None, 14, 14, 576)  5184        block_12_expand_relu[0]
[0]
__________________________________________________________________________________________________
block_12_depthwise_BN (BatchNor (None, 14, 14, 576)  2304        block_12_depthwise[0]
[0]
__________________________________________________________________________________________________
block_12_depthwise_relu (ReLU)  (None, 14, 14, 576)  0           block_12_depthwise_BN
[0][0]
__________________________________________________________________________________________________
block_12_project (Conv2D)       (None, 14, 14, 96)   55296       block_12_depthwise_relu
[0][0]
```

```
__________________________________________________________________________________________
__________
block_12_project_BN (BatchNorma (None, 14, 14, 96)   384         block_12_project[0][0]
__________________________________________________________________________________________
__________
block_12_add (Add)              (None, 14, 14, 96)   0           block_11_add[0][0]
                                                                 block_12_project_BN[0]
[0]
__________________________________________________________________________________________
__________
block_13_expand (Conv2D)        (None, 14, 14, 576)  55296       block_12_add[0][0]
__________________________________________________________________________________________
__________
block_13_expand_BN (BatchNormal (None, 14, 14, 576)  2304        block_13_expand[0][0]
__________________________________________________________________________________________
__________
block_13_expand_relu (ReLU)     (None, 14, 14, 576)  0           block_13_expand_BN[0]
[0]
__________________________________________________________________________________________
__________
block_13_pad (ZeroPadding2D)    (None, 15, 15, 576)  0           block_13_expand_relu[0]
[0]
__________________________________________________________________________________________
__________
block_13_depthwise (DepthwiseCo (None, 7, 7, 576)    5184        block_13_pad[0][0]
__________________________________________________________________________________________
__________
block_13_depthwise_BN (BatchNor (None, 7, 7, 576)    2304        block_13_depthwise[0]
[0]
__________________________________________________________________________________________
__________
block_13_depthwise_relu (ReLU)  (None, 7, 7, 576)    0           block_13_depthwise_BN
[0][0]
__________________________________________________________________________________________
__________
block_13_project (Conv2D)       (None, 7, 7, 160)    92160       block_13_depthwise_relu
[0][0]
__________________________________________________________________________________________
__________
block_13_project_BN (BatchNorma (None, 7, 7, 160)    640         block_13_project[0][0]
__________________________________________________________________________________________
__________
block_14_expand (Conv2D)        (None, 7, 7, 960)    153600      block_13_project_BN[0]
[0]
__________________________________________________________________________________________
__________
block_14_expand_BN (BatchNormal (None, 7, 7, 960)    3840        block_14_expand[0][0]
__________________________________________________________________________________________
__________
block_14_expand_relu (ReLU)     (None, 7, 7, 960)    0           block_14_expand_BN[0]
[0]
__________________________________________________________________________________________
__________
block_14_depthwise (DepthwiseCo (None, 7, 7, 960)    8640        block_14_expand_relu[0]
[0]
__________________________________________________________________________________________
__________
block_14_depthwise_BN (BatchNor (None, 7, 7, 960)    3840        block_14_depthwise[0]
[0]
__________________________________________________________________________________________
__________
block_14_depthwise_relu (ReLU)  (None, 7, 7, 960)    0           block_14_depthwise_BN
[0][0]
__________________________________________________________________________________________
__________
```

```
block_14_project (Conv2D)      (None, 7, 7, 160)    153600      block_14_depthwise_relu
[0][0]
__________________________________________________________________________________________
__________
block_14_project_BN (BatchNorma (None, 7, 7, 160)    640         block_14_project[0][0]
__________________________________________________________________________________________
__________
block_14_add (Add)              (None, 7, 7, 160)    0           block_13_project_BN[0]
[0]
                                                                 block_14_project_BN[0]
[0]
__________________________________________________________________________________________
__________
block_15_expand (Conv2D)        (None, 7, 7, 960)    153600      block_14_add[0][0]
__________________________________________________________________________________________
__________
block_15_expand_BN (BatchNormal (None, 7, 7, 960)    3840        block_15_expand[0][0]
__________________________________________________________________________________________
__________
block_15_expand_relu (ReLU)     (None, 7, 7, 960)    0           block_15_expand_BN[0]
[0]
__________________________________________________________________________________________
__________
block_15_depthwise (DepthwiseCo (None, 7, 7, 960)    8640        block_15_expand_relu[0]
[0]
__________________________________________________________________________________________
__________
block_15_depthwise_BN (BatchNor (None, 7, 7, 960)    3840        block_15_depthwise[0]
[0]
__________________________________________________________________________________________
__________
block_15_depthwise_relu (ReLU)  (None, 7, 7, 960)    0           block_15_depthwise_BN
[0][0]
__________________________________________________________________________________________
__________
block_15_project (Conv2D)       (None, 7, 7, 160)    153600      block_15_depthwise_relu
[0][0]
__________________________________________________________________________________________
__________
block_15_project_BN (BatchNorma (None, 7, 7, 160)    640         block_15_project[0][0]
__________________________________________________________________________________________
__________
block_15_add (Add)              (None, 7, 7, 160)    0           block_14_add[0][0]
                                                                 block_15_project_BN[0]
[0]
__________________________________________________________________________________________
__________
block_16_expand (Conv2D)        (None, 7, 7, 960)    153600      block_15_add[0][0]
__________________________________________________________________________________________
__________
block_16_expand_BN (BatchNormal (None, 7, 7, 960)    3840        block_16_expand[0][0]
__________________________________________________________________________________________
__________
block_16_expand_relu (ReLU)     (None, 7, 7, 960)    0           block_16_expand_BN[0]
[0]
__________________________________________________________________________________________
__________
block_16_depthwise (DepthwiseCo (None, 7, 7, 960)    8640        block_16_expand_relu[0]
[0]
__________________________________________________________________________________________
__________
block_16_depthwise_BN (BatchNor (None, 7, 7, 960)    3840        block_16_depthwise[0]
[0]
__________________________________________________________________________________________
__________
```

```
block_16_depthwise_relu (ReLU)  (None, 7, 7, 960)    0           block_16_depthwise_BN
[0][0]
__________________________________________________________________________________________________
block_16_project (Conv2D)       (None, 7, 7, 320)    307200      block_16_depthwise_relu
[0][0]
__________________________________________________________________________________________________
block_16_project_BN (BatchNorma (None, 7, 7, 320)    1280        block_16_project[0][0]
__________________________________________________________________________________________________
Conv_1 (Conv2D)                 (None, 7, 7, 1280)   409600      block_16_project_BN[0]
[0]
__________________________________________________________________________________________________
Conv_1_bn (BatchNormalization)  (None, 7, 7, 1280)   5120        Conv_1[0][0]
__________________________________________________________________________________________________
out_relu (ReLU)                 (None, 7, 7, 1280)   0           Conv_1_bn[0][0]
__________________________________________________________________________________________________
global_average_pooling2d (Globa (None, 1280)         0           out_relu[0][0]
__________________________________________________________________________________________________
predictions (Dense)             (None, 1000)         1281000     global_average_pooling2
d[0][0]
==================================================================================================
Total params: 3,538,984
Trainable params: 3,504,872
Non-trainable params: 34,112
__________________________________________________________________________________________________
```

Transfer Learning

Tuning , weights will start from last check point

In [30]:
```
base_input = model.layers[0].input
```

In [31]:
```
base_output = model.layers[-2].output
```

In [32]:
```
base_output
```

Out[32]:
```
<KerasTensor: shape=(None, 1280) dtype=float32 (created by layer 'global_average_pooling
2d')>
```

In [33]:
```
# Adding Layers
final_output = layers.Dense(128)(base_output) # Adding Layer after the output of thr gl
final_output = layers.Activation('relu')(final_output) # Activation function
final_output = layers.Dense(64)(final_output)
final_output = layers.Activation('relu')(final_output)
final_output = layers.Dense(7,activation='softmax')(final_output) # Output classes are
```

In [34]:
```
final_output # output
```

Out[34]:
```
<KerasTensor: shape=(None, 7) dtype=float32 (created by layer 'dense_2')>
```

In [35]:
```
new_model = keras.Model(inputs=base_input,outputs= final_output)
```

In [36]:
```
new_model.summary()
```

```
Model: "model"
__________________________________________________________________________________________
__________
Layer (type)                    Output Shape         Param #     Connected to
==========================================================================================
==========
input_1 (InputLayer)            [(None, 224, 224, 3) 0
__________________________________________________________________________________________
__________
Conv1 (Conv2D)                  (None, 112, 112, 32) 864         input_1[0][0]
__________________________________________________________________________________________
__________
bn_Conv1 (BatchNormalization)   (None, 112, 112, 32) 128         Conv1[0][0]
__________________________________________________________________________________________
__________
Conv1_relu (ReLU)               (None, 112, 112, 32) 0           bn_Conv1[0][0]
__________________________________________________________________________________________
__________
expanded_conv_depthwise (Depthw (None, 112, 112, 32) 288         Conv1_relu[0][0]
__________________________________________________________________________________________
__________
expanded_conv_depthwise_BN (Bat (None, 112, 112, 32) 128         expanded_conv_depthwise
[0][0]
__________________________________________________________________________________________
__________
expanded_conv_depthwise_relu (R (None, 112, 112, 32) 0           expanded_conv_depthwise
_BN[0][0]
__________________________________________________________________________________________
__________
expanded_conv_project (Conv2D)  (None, 112, 112, 16) 512         expanded_conv_depthwise
_relu[0][0
__________________________________________________________________________________________
__________
expanded_conv_project_BN (Batch (None, 112, 112, 16) 64          expanded_conv_project
[0][0]
__________________________________________________________________________________________
__________
block_1_expand (Conv2D)         (None, 112, 112, 96) 1536        expanded_conv_project_B
N[0][0]
__________________________________________________________________________________________
__________
block_1_expand_BN (BatchNormali (None, 112, 112, 96) 384         block_1_expand[0][0]
__________________________________________________________________________________________
__________
block_1_expand_relu (ReLU)      (None, 112, 112, 96) 0           block_1_expand_BN[0][0]
__________________________________________________________________________________________
__________
block_1_pad (ZeroPadding2D)     (None, 113, 113, 96) 0           block_1_expand_relu[0]
[0]
__________________________________________________________________________________________
__________
block_1_depthwise (DepthwiseCon (None, 56, 56, 96)   864         block_1_pad[0][0]
__________________________________________________________________________________________
```

```
__________
block_1_depthwise_BN (BatchNorm (None, 56, 56, 96)  384         block_1_depthwise[0][0]
________________________________________________________________________________________
__________
block_1_depthwise_relu (ReLU)   (None, 56, 56, 96)  0           block_1_depthwise_BN[0]
[0]
________________________________________________________________________________________
__________
block_1_project (Conv2D)        (None, 56, 56, 24)  2304        block_1_depthwise_relu
[0][0]
________________________________________________________________________________________
__________
block_1_project_BN (BatchNormal (None, 56, 56, 24)  96          block_1_project[0][0]
________________________________________________________________________________________
__________
block_2_expand (Conv2D)         (None, 56, 56, 144) 3456        block_1_project_BN[0]
[0]
________________________________________________________________________________________
__________
block_2_expand_BN (BatchNormali (None, 56, 56, 144) 576         block_2_expand[0][0]
________________________________________________________________________________________
__________
block_2_expand_relu (ReLU)      (None, 56, 56, 144) 0           block_2_expand_BN[0][0]
________________________________________________________________________________________
__________
block_2_depthwise (DepthwiseCon (None, 56, 56, 144) 1296        block_2_expand_relu[0]
[0]
________________________________________________________________________________________
__________
block_2_depthwise_BN (BatchNorm (None, 56, 56, 144) 576         block_2_depthwise[0][0]
________________________________________________________________________________________
__________
block_2_depthwise_relu (ReLU)   (None, 56, 56, 144) 0           block_2_depthwise_BN[0]
[0]
________________________________________________________________________________________
__________
block_2_project (Conv2D)        (None, 56, 56, 24)  3456        block_2_depthwise_relu
[0][0]
________________________________________________________________________________________
__________
block_2_project_BN (BatchNormal (None, 56, 56, 24)  96          block_2_project[0][0]
________________________________________________________________________________________
__________
block_2_add (Add)               (None, 56, 56, 24)  0           block_1_project_BN[0]
[0]
                                                                block_2_project_BN[0]
[0]
________________________________________________________________________________________
__________
block_3_expand (Conv2D)         (None, 56, 56, 144) 3456        block_2_add[0][0]
________________________________________________________________________________________
__________
block_3_expand_BN (BatchNormali (None, 56, 56, 144) 576         block_3_expand[0][0]
________________________________________________________________________________________
__________
block_3_expand_relu (ReLU)      (None, 56, 56, 144) 0           block_3_expand_BN[0][0]
________________________________________________________________________________________
__________
block_3_pad (ZeroPadding2D)     (None, 57, 57, 144) 0           block_3_expand_relu[0]
[0]
________________________________________________________________________________________
__________
block_3_depthwise (DepthwiseCon (None, 28, 28, 144) 1296        block_3_pad[0][0]
________________________________________________________________________________________
__________
```

```
block_3_depthwise_BN (BatchNorm (None, 28, 28, 144)  576         block_3_depthwise[0][0]
__________________________________________________________________________________________________
block_3_depthwise_relu (ReLU)   (None, 28, 28, 144)  0           block_3_depthwise_BN[0]
[0]
__________________________________________________________________________________________________
block_3_project (Conv2D)        (None, 28, 28, 32)   4608        block_3_depthwise_relu
[0][0]
__________________________________________________________________________________________________
block_3_project_BN (BatchNormal (None, 28, 28, 32)   128         block_3_project[0][0]
__________________________________________________________________________________________________
block_4_expand (Conv2D)         (None, 28, 28, 192)  6144        block_3_project_BN[0]
[0]
__________________________________________________________________________________________________
block_4_expand_BN (BatchNormali (None, 28, 28, 192)  768         block_4_expand[0][0]
__________________________________________________________________________________________________
block_4_expand_relu (ReLU)      (None, 28, 28, 192)  0           block_4_expand_BN[0][0]
__________________________________________________________________________________________________
block_4_depthwise (DepthwiseCon (None, 28, 28, 192)  1728        block_4_expand_relu[0]
[0]
__________________________________________________________________________________________________
block_4_depthwise_BN (BatchNorm (None, 28, 28, 192)  768         block_4_depthwise[0][0]
__________________________________________________________________________________________________
block_4_depthwise_relu (ReLU)   (None, 28, 28, 192)  0           block_4_depthwise_BN[0]
[0]
__________________________________________________________________________________________________
block_4_project (Conv2D)        (None, 28, 28, 32)   6144        block_4_depthwise_relu
[0][0]
__________________________________________________________________________________________________
block_4_project_BN (BatchNormal (None, 28, 28, 32)   128         block_4_project[0][0]
__________________________________________________________________________________________________
block_4_add (Add)               (None, 28, 28, 32)   0           block_3_project_BN[0]
[0]
                                                                 block_4_project_BN[0]
[0]
__________________________________________________________________________________________________
block_5_expand (Conv2D)         (None, 28, 28, 192)  6144        block_4_add[0][0]
__________________________________________________________________________________________________
block_5_expand_BN (BatchNormali (None, 28, 28, 192)  768         block_5_expand[0][0]
__________________________________________________________________________________________________
block_5_expand_relu (ReLU)      (None, 28, 28, 192)  0           block_5_expand_BN[0][0]
__________________________________________________________________________________________________
block_5_depthwise (DepthwiseCon (None, 28, 28, 192)  1728        block_5_expand_relu[0]
[0]
__________________________________________________________________________________________________
block_5_depthwise_BN (BatchNorm (None, 28, 28, 192)  768         block_5_depthwise[0][0]
__________________________________________________________________________________________________
block_5_depthwise_relu (ReLU)   (None, 28, 28, 192)  0           block_5_depthwise_BN[0]
```

```
[0]
__________________________________________________________________________________________________
block_5_project (Conv2D)        (None, 28, 28, 32)   6144        block_5_depthwise_relu
[0][0]
__________________________________________________________________________________________________
block_5_project_BN (BatchNormal (None, 28, 28, 32)   128         block_5_project[0][0]
__________________________________________________________________________________________________
block_5_add (Add)               (None, 28, 28, 32)   0           block_4_add[0][0]
                                                                 block_5_project_BN[0]
[0]
__________________________________________________________________________________________________
block_6_expand (Conv2D)         (None, 28, 28, 192)  6144        block_5_add[0][0]
__________________________________________________________________________________________________
block_6_expand_BN (BatchNormali (None, 28, 28, 192)  768         block_6_expand[0][0]
__________________________________________________________________________________________________
block_6_expand_relu (ReLU)      (None, 28, 28, 192)  0           block_6_expand_BN[0][0]
__________________________________________________________________________________________________
block_6_pad (ZeroPadding2D)     (None, 29, 29, 192)  0           block_6_expand_relu[0]
[0]
__________________________________________________________________________________________________
block_6_depthwise (DepthwiseCon (None, 14, 14, 192)  1728        block_6_pad[0][0]
__________________________________________________________________________________________________
block_6_depthwise_BN (BatchNorm (None, 14, 14, 192)  768         block_6_depthwise[0][0]
__________________________________________________________________________________________________
block_6_depthwise_relu (ReLU)   (None, 14, 14, 192)  0           block_6_depthwise_BN[0]
[0]
__________________________________________________________________________________________________
block_6_project (Conv2D)        (None, 14, 14, 64)   12288       block_6_depthwise_relu
[0][0]
__________________________________________________________________________________________________
block_6_project_BN (BatchNormal (None, 14, 14, 64)   256         block_6_project[0][0]
__________________________________________________________________________________________________
block_7_expand (Conv2D)         (None, 14, 14, 384)  24576       block_6_project_BN[0]
[0]
__________________________________________________________________________________________________
block_7_expand_BN (BatchNormali (None, 14, 14, 384)  1536        block_7_expand[0][0]
__________________________________________________________________________________________________
block_7_expand_relu (ReLU)      (None, 14, 14, 384)  0           block_7_expand_BN[0][0]
__________________________________________________________________________________________________
block_7_depthwise (DepthwiseCon (None, 14, 14, 384)  3456        block_7_expand_relu[0]
[0]
__________________________________________________________________________________________________
block_7_depthwise_BN (BatchNorm (None, 14, 14, 384)  1536        block_7_depthwise[0][0]
__________________________________________________________________________________________________
block_7_depthwise_relu (ReLU)   (None, 14, 14, 384)  0           block_7_depthwise_BN[0]
[0]
__________________________________________________________________________________________________
```

```
__________
block_7_project (Conv2D)        (None, 14, 14, 64)   24576       block_7_depthwise_relu
[0][0]
________________________________________________________________________________________
__________
block_7_project_BN (BatchNormal (None, 14, 14, 64)   256         block_7_project[0][0]
________________________________________________________________________________________
__________
block_7_add (Add)               (None, 14, 14, 64)   0           block_6_project_BN[0]
[0]
                                                                 block_7_project_BN[0]
[0]
________________________________________________________________________________________
__________
block_8_expand (Conv2D)         (None, 14, 14, 384)  24576       block_7_add[0][0]
________________________________________________________________________________________
__________
block_8_expand_BN (BatchNormali (None, 14, 14, 384)  1536        block_8_expand[0][0]
________________________________________________________________________________________
__________
block_8_expand_relu (ReLU)      (None, 14, 14, 384)  0           block_8_expand_BN[0][0]
________________________________________________________________________________________
__________
block_8_depthwise (DepthwiseCon (None, 14, 14, 384)  3456        block_8_expand_relu[0]
[0]
________________________________________________________________________________________
__________
block_8_depthwise_BN (BatchNorm (None, 14, 14, 384)  1536        block_8_depthwise[0][0]
________________________________________________________________________________________
__________
block_8_depthwise_relu (ReLU)   (None, 14, 14, 384)  0           block_8_depthwise_BN[0]
[0]
________________________________________________________________________________________
__________
block_8_project (Conv2D)        (None, 14, 14, 64)   24576       block_8_depthwise_relu
[0][0]
________________________________________________________________________________________
__________
block_8_project_BN (BatchNormal (None, 14, 14, 64)   256         block_8_project[0][0]
________________________________________________________________________________________
__________
block_8_add (Add)               (None, 14, 14, 64)   0           block_7_add[0][0]
                                                                 block_8_project_BN[0]
[0]
________________________________________________________________________________________
__________
block_9_expand (Conv2D)         (None, 14, 14, 384)  24576       block_8_add[0][0]
________________________________________________________________________________________
__________
block_9_expand_BN (BatchNormali (None, 14, 14, 384)  1536        block_9_expand[0][0]
________________________________________________________________________________________
__________
block_9_expand_relu (ReLU)      (None, 14, 14, 384)  0           block_9_expand_BN[0][0]
________________________________________________________________________________________
__________
block_9_depthwise (DepthwiseCon (None, 14, 14, 384)  3456        block_9_expand_relu[0]
[0]
________________________________________________________________________________________
__________
block_9_depthwise_BN (BatchNorm (None, 14, 14, 384)  1536        block_9_depthwise[0][0]
________________________________________________________________________________________
__________
block_9_depthwise_relu (ReLU)   (None, 14, 14, 384)  0           block_9_depthwise_BN[0]
[0]
________________________________________________________________________________________
```

```
__________
block_9_project (Conv2D)        (None, 14, 14, 64)   24576       block_9_depthwise_relu
[0][0]
__________________________________________________________________________________________
__________
block_9_project_BN (BatchNormal (None, 14, 14, 64)   256         block_9_project[0][0]
__________________________________________________________________________________________
__________
block_9_add (Add)               (None, 14, 14, 64)   0           block_8_add[0][0]
                                                                 block_9_project_BN[0]
[0]
__________________________________________________________________________________________
__________
block_10_expand (Conv2D)        (None, 14, 14, 384)  24576       block_9_add[0][0]
__________________________________________________________________________________________
__________
block_10_expand_BN (BatchNormal (None, 14, 14, 384)  1536        block_10_expand[0][0]
__________________________________________________________________________________________
__________
block_10_expand_relu (ReLU)     (None, 14, 14, 384)  0           block_10_expand_BN[0]
[0]
__________________________________________________________________________________________
__________
block_10_depthwise (DepthwiseCo (None, 14, 14, 384)  3456        block_10_expand_relu[0]
[0]
__________________________________________________________________________________________
__________
block_10_depthwise_BN (BatchNor (None, 14, 14, 384)  1536        block_10_depthwise[0]
[0]
__________________________________________________________________________________________
__________
block_10_depthwise_relu (ReLU)  (None, 14, 14, 384)  0           block_10_depthwise_BN
[0][0]
__________________________________________________________________________________________
__________
block_10_project (Conv2D)       (None, 14, 14, 96)   36864       block_10_depthwise_relu
[0][0]
__________________________________________________________________________________________
__________
block_10_project_BN (BatchNorma (None, 14, 14, 96)   384         block_10_project[0][0]
__________________________________________________________________________________________
__________
block_11_expand (Conv2D)        (None, 14, 14, 576)  55296       block_10_project_BN[0]
[0]
__________________________________________________________________________________________
__________
block_11_expand_BN (BatchNormal (None, 14, 14, 576)  2304        block_11_expand[0][0]
__________________________________________________________________________________________
__________
block_11_expand_relu (ReLU)     (None, 14, 14, 576)  0           block_11_expand_BN[0]
[0]
__________________________________________________________________________________________
__________
block_11_depthwise (DepthwiseCo (None, 14, 14, 576)  5184        block_11_expand_relu[0]
[0]
__________________________________________________________________________________________
__________
block_11_depthwise_BN (BatchNor (None, 14, 14, 576)  2304        block_11_depthwise[0]
[0]
__________________________________________________________________________________________
__________
block_11_depthwise_relu (ReLU)  (None, 14, 14, 576)  0           block_11_depthwise_BN
[0][0]
__________________________________________________________________________________________
__________
```

```
block_11_project (Conv2D)       (None, 14, 14, 96)   55296       block_11_depthwise_relu
[0][0]
__________________________________________________________________________________________
__________
block_11_project_BN (BatchNorma (None, 14, 14, 96)   384         block_11_project[0][0]
__________________________________________________________________________________________
__________
block_11_add (Add)              (None, 14, 14, 96)   0           block_10_project_BN[0]
[0]
                                                                 block_11_project_BN[0]
[0]
__________________________________________________________________________________________
__________
block_12_expand (Conv2D)        (None, 14, 14, 576)  55296       block_11_add[0][0]
__________________________________________________________________________________________
__________
block_12_expand_BN (BatchNormal (None, 14, 14, 576)  2304        block_12_expand[0][0]
__________________________________________________________________________________________
__________
block_12_expand_relu (ReLU)     (None, 14, 14, 576)  0           block_12_expand_BN[0]
[0]
__________________________________________________________________________________________
__________
block_12_depthwise (DepthwiseCo (None, 14, 14, 576)  5184        block_12_expand_relu[0]
[0]
__________________________________________________________________________________________
__________
block_12_depthwise_BN (BatchNor (None, 14, 14, 576)  2304        block_12_depthwise[0]
[0]
__________________________________________________________________________________________
__________
block_12_depthwise_relu (ReLU)  (None, 14, 14, 576)  0           block_12_depthwise_BN
[0][0]
__________________________________________________________________________________________
__________
block_12_project (Conv2D)       (None, 14, 14, 96)   55296       block_12_depthwise_relu
[0][0]
__________________________________________________________________________________________
__________
block_12_project_BN (BatchNorma (None, 14, 14, 96)   384         block_12_project[0][0]
__________________________________________________________________________________________
__________
block_12_add (Add)              (None, 14, 14, 96)   0           block_11_add[0][0]
                                                                 block_12_project_BN[0]
[0]
__________________________________________________________________________________________
__________
block_13_expand (Conv2D)        (None, 14, 14, 576)  55296       block_12_add[0][0]
__________________________________________________________________________________________
__________
block_13_expand_BN (BatchNormal (None, 14, 14, 576)  2304        block_13_expand[0][0]
__________________________________________________________________________________________
__________
block_13_expand_relu (ReLU)     (None, 14, 14, 576)  0           block_13_expand_BN[0]
[0]
__________________________________________________________________________________________
__________
block_13_pad (ZeroPadding2D)    (None, 15, 15, 576)  0           block_13_expand_relu[0]
[0]
__________________________________________________________________________________________
__________
block_13_depthwise (DepthwiseCo (None, 7, 7, 576)    5184        block_13_pad[0][0]
__________________________________________________________________________________________
__________
block_13_depthwise_BN (BatchNor (None, 7, 7, 576)    2304        block_13_depthwise[0]
[0]
__________________________________________________________________________________________________
block_13_depthwise_relu (ReLU)  (None, 7, 7, 576)    0           block_13_depthwise_BN
[0][0]
__________________________________________________________________________________________________
block_13_project (Conv2D)       (None, 7, 7, 160)    92160       block_13_depthwise_relu
[0][0]
__________________________________________________________________________________________________
block_13_project_BN (BatchNorma (None, 7, 7, 160)    640         block_13_project[0][0]
__________________________________________________________________________________________________
block_14_expand (Conv2D)        (None, 7, 7, 960)    153600      block_13_project_BN[0]
[0]
__________________________________________________________________________________________________
block_14_expand_BN (BatchNormal (None, 7, 7, 960)    3840        block_14_expand[0][0]
__________________________________________________________________________________________________
block_14_expand_relu (ReLU)     (None, 7, 7, 960)    0           block_14_expand_BN[0]
[0]
__________________________________________________________________________________________________
block_14_depthwise (DepthwiseCo (None, 7, 7, 960)    8640        block_14_expand_relu[0]
[0]
__________________________________________________________________________________________________
block_14_depthwise_BN (BatchNor (None, 7, 7, 960)    3840        block_14_depthwise[0]
[0]
__________________________________________________________________________________________________
block_14_depthwise_relu (ReLU)  (None, 7, 7, 960)    0           block_14_depthwise_BN
[0][0]
__________________________________________________________________________________________________
block_14_project (Conv2D)       (None, 7, 7, 160)    153600      block_14_depthwise_relu
[0][0]
__________________________________________________________________________________________________
block_14_project_BN (BatchNorma (None, 7, 7, 160)    640         block_14_project[0][0]
__________________________________________________________________________________________________
block_14_add (Add)              (None, 7, 7, 160)    0           block_13_project_BN[0]
[0]
                                                                 block_14_project_BN[0]
[0]
__________________________________________________________________________________________________
block_15_expand (Conv2D)        (None, 7, 7, 960)    153600      block_14_add[0][0]
__________________________________________________________________________________________________
block_15_expand_BN (BatchNormal (None, 7, 7, 960)    3840        block_15_expand[0][0]
__________________________________________________________________________________________________
block_15_expand_relu (ReLU)     (None, 7, 7, 960)    0           block_15_expand_BN[0]
[0]
__________________________________________________________________________________________________
block_15_depthwise (DepthwiseCo (None, 7, 7, 960)    8640        block_15_expand_relu[0]
[0]
__________________________________________________________________________________________________
block_15_depthwise_BN (BatchNor (None, 7, 7, 960)    3840        block_15_depthwise[0]
[0]
__________________________________________________________________________________________________
__________
block_15_depthwise_relu (ReLU)  (None, 7, 7, 960)    0           block_15_depthwise_BN
[0][0]
__________________________________________________________________________________________________
__________
block_15_project (Conv2D)       (None, 7, 7, 160)    153600      block_15_depthwise_relu
[0][0]
__________________________________________________________________________________________________
__________
block_15_project_BN (BatchNorma (None, 7, 7, 160)    640         block_15_project[0][0]
__________________________________________________________________________________________________
__________
block_15_add (Add)              (None, 7, 7, 160)    0           block_14_add[0][0]
                                                                 block_15_project_BN[0]
[0]
__________________________________________________________________________________________________
__________
block_16_expand (Conv2D)        (None, 7, 7, 960)    153600      block_15_add[0][0]
__________________________________________________________________________________________________
__________
block_16_expand_BN (BatchNormal (None, 7, 7, 960)    3840        block_16_expand[0][0]
__________________________________________________________________________________________________
__________
block_16_expand_relu (ReLU)     (None, 7, 7, 960)    0           block_16_expand_BN[0]
[0]
__________________________________________________________________________________________________
__________
block_16_depthwise (DepthwiseCo (None, 7, 7, 960)    8640        block_16_expand_relu[0]
[0]
__________________________________________________________________________________________________
__________
block_16_depthwise_BN (BatchNor (None, 7, 7, 960)    3840        block_16_depthwise[0]
[0]
__________________________________________________________________________________________________
__________
block_16_depthwise_relu (ReLU)  (None, 7, 7, 960)    0           block_16_depthwise_BN
[0][0]
__________________________________________________________________________________________________
__________
block_16_project (Conv2D)       (None, 7, 7, 320)    307200      block_16_depthwise_relu
[0][0]
__________________________________________________________________________________________________
__________
block_16_project_BN (BatchNorma (None, 7, 7, 320)    1280        block_16_project[0][0]
__________________________________________________________________________________________________
__________
Conv_1 (Conv2D)                 (None, 7, 7, 1280)   409600      block_16_project_BN[0]
[0]
__________________________________________________________________________________________________
__________
Conv_1_bn (BatchNormalization)  (None, 7, 7, 1280)   5120        Conv_1[0][0]
__________________________________________________________________________________________________
__________
out_relu (ReLU)                 (None, 7, 7, 1280)   0           Conv_1_bn[0][0]
__________________________________________________________________________________________________
__________
global_average_pooling2d (Globa (None, 1280)         0           out_relu[0][0]
__________________________________________________________________________________________________
__________
dense (Dense)                   (None, 128)          163968      global_average_pooling2
d[0][0]
__________________________________________________________________________________________________
__________
```

```
activation (Activation)        (None, 128)         0           dense[0][0]
________________________________________________________________________________________
__________
dense_1 (Dense)                (None, 64)          8256        activation[0][0]
________________________________________________________________________________________
__________
activation_1 (Activation)      (None, 64)          0           dense_1[0][0]
________________________________________________________________________________________
__________
dense_2 (Dense)                (None, 7)           455         activation_1[0][0]
========================================================================================
==========
Total params: 2,430,663
Trainable params: 2,396,551
Non-trainable params: 34,112
________________________________________________________________________________________
__________
```

In [37]:

```
new_model.compile(loss="sparse_categorical_crossentropy",optimizer="adam",metrics=["acc
```

In [38]:

```
os.environ['TF_FORCE_GPU_ALLOW_GROWTH'] = 'true'
```

In [39]:

```
new_model.fit(X,y,epochs=15)
```

```
Epoch 1/15
341/341 [==============================] - 140s 316ms/step - loss: 1.5225 - accuracy: 0.
4249
Epoch 2/15
341/341 [==============================] - 195s 571ms/step - loss: 1.1612 - accuracy: 0.
5643
Epoch 3/15
341/341 [==============================] - 267s 784ms/step - loss: 1.0166 - accuracy: 0.
6216
Epoch 4/15
341/341 [==============================] - 314s 922ms/step - loss: 0.9492 - accuracy: 0.
6399
Epoch 5/15
341/341 [==============================] - 340s 998ms/step - loss: 0.8931 - accuracy: 0.
6758
Epoch 6/15
341/341 [==============================] - 376s 1s/step - loss: 0.8185 - accuracy: 0.696
7
Epoch 7/15
341/341 [==============================] - 381s 1s/step - loss: 0.7236 - accuracy: 0.735
3
Epoch 8/15
341/341 [==============================] - 372s 1s/step - loss: 0.7158 - accuracy: 0.742
4
Epoch 9/15
341/341 [==============================] - 398s 1s/step - loss: 0.6513 - accuracy: 0.758
1
Epoch 10/15
341/341 [==============================] - 400s 1s/step - loss: 0.5846 - accuracy: 0.793
4
Epoch 11/15
341/341 [==============================] - 394s 1s/step - loss: 0.5702 - accuracy: 0.793
3
Epoch 12/15
341/341 [==============================] - 383s 1s/step - loss: 0.4770 - accuracy: 0.829
```

```
0
Epoch 13/15
341/341 [==============================] - 377s 1s/step - loss: 0.4161 - accuracy: 0.851
4
Epoch 14/15
341/341 [==============================] - 404s 1s/step - loss: 0.3777 - accuracy: 0.868
0
Epoch 15/15
341/341 [==============================] - 404s 1s/step - loss: 0.3283 - accuracy: 0.889
4
```

Out[39]: `<tensorflow.python.keras.callbacks.History at 0x1bed8439220>`

In [40]:
```
new_model.save('face_emotion_rec_v2.h5')
```

In [41]:
```
# new_model = tf.keras.models.load_model("face_emotion_rec_v2.h5")
```

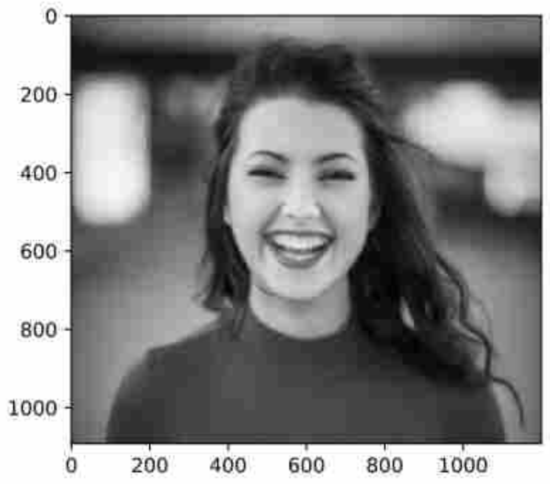
In [42]:
```
frame = cv2.imread("smiley_girl.jpg")
```

In [43]:
```
frame.shape
```

Out[43]: `(1092, 1200, 3)`

In [44]:
```
plt.imshow(cv2.cvtColor(frame,cv2.COLOR_BGR2RGB))
```

Out[44]: `<matplotlib.image.AxesImage at 0x1c4b73b4ee0>`

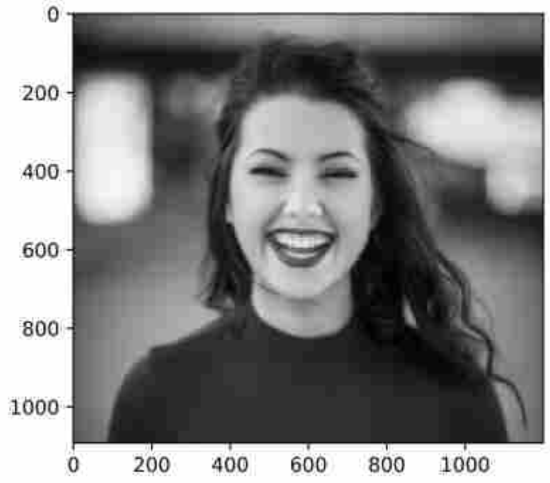


In [45]:
```
# Convert it into gray image and use face detection algorithm to detect face

face_detect = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_de

gray_img = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
gray_img.shape
```

Out[45]:
```
(1092, 1200)
```

In [46]:
```
plt.imshow(cv2.cvtColor(gray_img,cv2.COLOR_BGR2RGB))
```

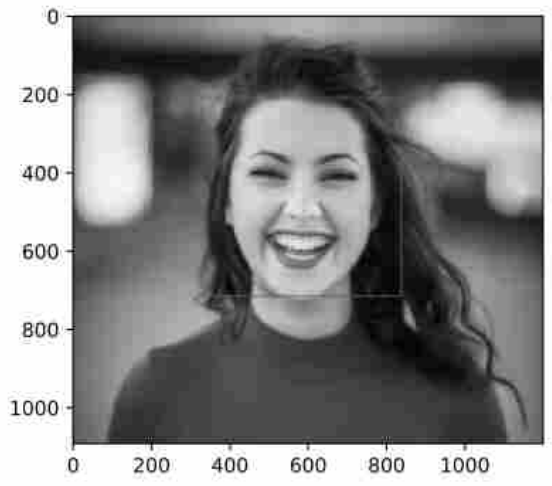
Out[46]:
```
<matplotlib.image.AxesImage at 0x1c4bef9b730>
```



In [47]:
```
faces = face_detect.detectMultiScale(gray_img,1.1,4)
for x,y,w,h in faces:
    roi_gray_img = gray_img[y:y+h,x:x+w]
    roi_color = frame[y:y+h,x:x+w]
    cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2)
    facess = face_detect.detectMultiScale(roi_gray_img)
    if len(facess) == 0:
        print("Face not detected")
    else:
        for (ex,ey,ew,eh) in facess:
            face_roi = roi_color[ey: ey+eh,ex:ex +ew]
```

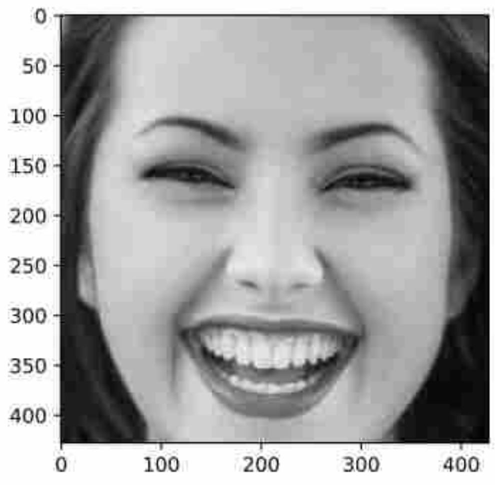
In [48]:
```
plt.imshow(cv2.cvtColor(frame,cv2.COLOR_BGR2RGB))
```

Out[48]:
```
<matplotlib.image.AxesImage at 0x1c4c255ce20>
```



In [49]:
```
# cropped image
plt.imshow(cv2.cvtColor(face_roi,cv2.COLOR_BGR2RGB))
```

Out[49]:
```
<matplotlib.image.AxesImage at 0x1c4c2ac2be0>
```



In [50]:
```
final_img = cv2.resize(face_roi,(224,224))
final_img = np.expand_dims(final_img,axis=0) # need 4th dimension
final_img = final_img/255 # normalizing
```

In [51]:
```
prediction = new_model.predict(final_img)
pred = np.argmax(prediction[0])
pred
```

```
Out[51]: 3
```

```
In [53]:
classNames= []
classFile = 'label.names'
with open(classFile,'rt') as f:
    classNames = f.read().rstrip('\n').split('\n')
```

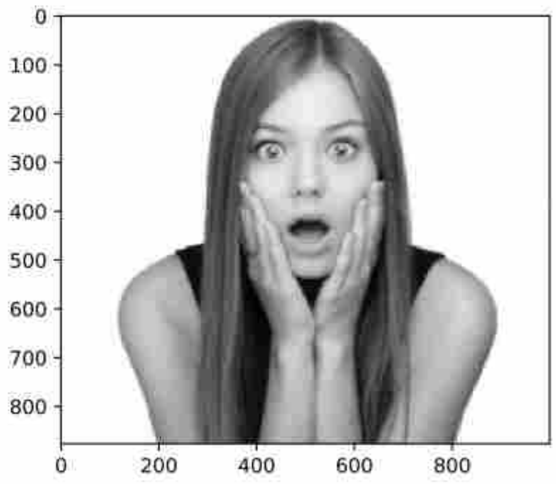
```
In [54]:
classNames[pred]
```

```
Out[54]: 'Happy'
```

Make a predict function

```
In [55]:
def prediction_img(img,detect_model,classNames,draw=False):
    frame = cv2.imread(img)
    if draw:
        plt.imshow(cv2.cvtColor(frame,cv2.COLOR_BGR2RGB))
    face_detect = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalfac

    gray_img = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
    faces = face_detect.detectMultiScale(gray_img,1.1,4)
    for x,y,w,h in faces:
        roi_gray_img = gray_img[y:y+h,x:x+w]
        roi_color = frame[y:y+h,x:x+w]
        cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2)
        facess = face_detect.detectMultiScale(roi_gray_img)
        if len(facess) == 0:
            print("Face not detected")
        else:
            for (ex,ey,ew,eh) in facess:
                face_roi = roi_color[ey: ey+eh,ex:ex +ew]

    final_img = cv2.resize(face_roi,(224,224))
    final_img = np.expand_dims(final_img,axis=0) # need 4th dimension
    final_img = final_img/255 # normalizing

    prediction = detect_model.predict(final_img)
    pred = np.argmax(prediction[0])

    return classNames[pred]

if __name__ =="__main__":
    classNames= []
    classFile = 'label.names'
    with open(classFile,'rt') as f:
        classNames = f.read().rstrip('\n').split('\n')

    detect_model = tf.keras.models.load_model("face_emotion_rec_v2.h5")
    img = "surprise_girl.jpg"
    print(prediction_img(img,detect_model,classNames,draw=True))
```

```
Surprise
```


0
100
200
300
400
500
600
700
800
0
200
400
600
800

7.6 Result

I got 88 % accuracy using CNN.

8. Conclusion

Sentiment analysis was performed on the FER- 2013 Dataset to classify the facial emotion. according to user facial emotion.

Using CNN with transfer learning model MobileNetV2 we got 88% accuracy which is really good.

9. Bibliography

The contents have been gathered from the following:

1.Google Search(https://www.google.com/)

2. Self-performed

3.Youtube Tutorials (https://www.youtube.com/)