**AN EFFICIENT AGE INVARIANT FACE IDENTIFICATION WITH GENDER AND EMOTION IDENTIFICATION USING CNN**

**INTRODUCION**

**Overview of the project**

Automated age detection has been generally used in daily lives that come across, majorly in a person to computer interaction, visual surveillance, biometric analysis, and other applications of commercial use. Many of the interactions we once had face-to-face have quickly moved online.

Those hoping to sign up to services which require identity verification no longer have the luxury of being able to call into their local branch; the global pandemic has decreased typical human interaction meaning account opening and identity verification now has to be done remotely.

Easy, secure online shopping has made e-commerce accessible to all people and all industries, including those which would typically require identity verification. Previously ‘underserved’ industries such as e-cigarettes, alcohol, dating apps and adult-only subscriptions are being enabled online by identity and age verification software.

Almost a third of the global population buys products or services online and 86% of [millennials](https://www.acuant.com/blog/capturing-the-millennial-market-with-identity-verification/) make purchases through their mobile phones. This unprecedented access to online products and services is making it harder for age-restricted merchants to regulate their products and services. But, without the necessary checks and if underaged customers are able to access these platforms, they can be subject to serious repercussions, legal and otherwise.

 The physical sale of alcohol for example, is one area where the legislation is very clear. A legal purchasing limit is set, age verification is required and fines are actively enforced. These purchases require a face-to-face interaction solely for age verification. Recognize the face then measure the features and use the algorithm to find out the age.

This, along with the increased wider online activity – increasing the risk of online fraud – has raised the already high demand for competent identity and [age verification](https://www.acuant.com/blog/3-top-tips-online-age-verification/) software.

Analysing expressions on the person’s face plays a very vital role in identifying emotions and behaviour of a person. Recognizing these expressions automatically results in a crucial component of natural human-machine interfaces. Therefore, research in this field has a wide range of applications in biometric authentication, surveillance systems, emotions to emoticons in various social media platforms. Another application includes conducting customer satisfaction surveys.

As we know that the large corporations made huge investments to get feedback and do surveys but fail to get equitable responses . Emotion & gender recognition through facial gestures is a technology that aims to improve product and services performance by monitoring customer behaviour to specific products or service staff by their evaluation.

Emotion classification task , emotion on the person’s face classified into seven classes , which are: ”angry, disgust, fear, happy, sad, surprise and neutral”. Finally, we try to predict who are male or female among them in gender classification task.

Interpreting emotion on the person’s face with the help of machine learning(ml) techniques is very complicated due to the large variance in samples from each task. As a result, millions of parameters within the model were trained using thousands of samples. Also, the accuracy of humans in identification of facial expression is 65% ± 5%.this can be seen by manually classifying the images of fer-2013 dataset which contains the classes: “angry”, “disgust”, “fear”, “happy”, “sad”, “surprise”, “neutral”.



Fig:fer-2013 samples for emotion classification

Large amount of research has been conducted to determine gender and emotion using facial features on different public standard datasets which permit public performance comparison of the proposed methods .as a result, there has been a lot of active research, with several recent papers using the concept of convolutional neural networks (cnns) for feature extraction and inference.

Facial expressions can be recognised using nonverbal communication between humans, and also facial expression interpretation has been extensively researched. Facial expression is important in human interaction, and the facial expression recognition(fer) algorithm uses computer vision techniques to aid in applications like human-computer interaction and data analytics.

Face recognition: the base model for our facial recognition is trained on over four million images of more than 40, 000 individuals. The large variation in images of each identity make our deep model robust to common challenges in face recognition

Emotion recognition is one of the trending research fields. It is involved in several applications. Its most interesting applications include robotic vision and interactive robotic communication. Human emotions can be detected using both speech and visual modalities. Facial expressions can be considered as ideal means for detecting the persons' emotions.

In our proposed project two models are used, one for age-gender prediction and other model is trained for emotion recognition using CNN. By recognizing the emotion of a person we can improve the recommendation system.( in [deep learning](https://en.wikipedia.org/wiki/Deep_learning), a convolutional neural network is a class of [artificial neural network](https://en.wikipedia.org/wiki/Artificial_neural_network) (ann), most commonly applied to analyse visual imagery.

Cnns are also known as shift invariant or space invariant artificial neural networks (siann), based on the shared-weight architecture of the convolution kernels or filters that slide along input features and provide translation-[equivariant](https://en.wikipedia.org/wiki/Equivariant_map) responses known as feature maps. Counter-intuitively, most convolutional neural networks are not [invariant](https://en.wikipedia.org/wiki/Translation_invariant) to translation, due to the down sampling operation they apply to the input.

They have applications in [image and video recognition](https://en.wikipedia.org/wiki/Computer_vision), [recommender systems](https://en.wikipedia.org/wiki/Recommender_system), [image classification](https://en.wikipedia.org/wiki/Image_classification), [image segmentation](https://en.wikipedia.org/wiki/Image_segmentation), [medical image analysis](https://en.wikipedia.org/wiki/Medical_image_computing), [natural language processing](https://en.wikipedia.org/wiki/Natural_language_processing), [brain–computer interfaces](https://en.wikipedia.org/wiki/Brain%E2%80%93computer_interface), and financial [time series](https://en.wikipedia.org/wiki/Time_series)).

An improvement in the performance of these tasks was observed by using the convolutional neural network (CNN). The dataset was obtained from Ut face dataset for age gender classification and fer2013 dataset for emotion recognition.

The dataset was collected from Kaggle.( Kaggle, a subsidiary of [google llc](https://en.wikipedia.org/wiki/Google_LLC), is an online community of [data scientists](https://en.wikipedia.org/wiki/Data_science) and [machine learning](https://en.wikipedia.org/wiki/Machine_learning) practitioners. Kaggle allows users to find and publish data sets, explore and build models in a web-based data-science environment, work with other data scientists and machine learning engineers, and enter competitions to solve data science challenges.)

Problem definition is age invariant model (aim) for joint disentangled representation learning and photo realistic cross-age face synthesis to address the challenging problem.

According to recent studies face images of different individuals usually share common aging characteristics (e.g., wrinkles), and face images of the same individual contain intrinsic features that are relatively stable across ages. Facial representations of a person in the latent space can hence be decomposed into an age-specific component which reflects the aging effect and an identity-specific component which preserves intrinsic identity information.

The latter would be invariant to age variations and ideal for cross-age face recognition when achievable. This finding inspires us to develop a novel and unified deep neural network, termed as age invariant model (aim).

The aim jointly learns disentangled identity representations that are invariant to age, and photorealistic cross-age face image synthesis that can highlight important latent representations among the disentangled ones end-to-end. Therefore, they mutually boost each other to achieve age-invariant face recognition.

Aim takes as input face images of arbitrary ages with other potential distracting factors like various illumination, expressions, poses, and occlusion. It outputs facial representations invariant to age variations and meanwhile preserves discriminativeness across different identities. The aim can learn age-invariant representations and effectively synthesize natural age regressed/progressed faces.

We present the results given various inputs along with different challenging factors to show that aim has automatically learned to achieve the robustness to skin colour, illumination, expression, pose, and occlusion besides the age variation. In particular, aim extends from an auto-encoder based generative adversarial network (gan) and includes a disentangled representation learning sub-net (rln) and a face synthesis sub-net (fsn) for age-invariant face recognition.

Rln consists of an encoder and a discriminator that compete with each other to learn discriminative and age-invariant representations. It introduces cross-age domain adversarial training to promote encoded features that are indistinguishable w. R.t. The shift between multi-age domains, and cross-entropy regularization with a label smoothing strategy to constrain cross-age representations with ambiguous separability.

The discriminator incorporates dual agents to encourage the representations to be uniformly distributed to smooth the age transformation while preserving identity information. The representations are then concatenated with a continuous age condition code to synthesize age regressed/progressed face images, such that the learned representations are explicitly disentangled from age variations.

Fsn consists of a decoder and a local-patch based discriminator that compete with each other to synthesize photorealistic cross-age face images. Fsn uses an attention mechanism to guarantee robustness to large background complexity and illumination variance.

The discriminator incorporates dual agents to add realism to synthesized cross-age faces while forcing the generated faces to exhibit desirable rejuvenation/aging effects. Moreover, we propose a new large-scale cross-age face recognition (cafr) benchmark dataset to facilitate existing efforts and future research on age-invariant face recognition.

Cafr contains 1,446,500 face images from 25,000 subjects annotated with age, identity, gender, race and landmark labels. Extensive experiments on both our cafr and other standard cross-age datasets (morph, cacd, and fgnet) demonstrate the superiority of aim over the state of-the-arts.

Benchmarking aim on the popular unconstrained face recognition datasets ytf and ijb-c additionally verifies its promising generalization ability in recognizing faces in the wild. A preliminary version of this work was accepted in aaai conference on artificial intelligence (aaai) 2019.

**ORGANISATION OF THE REPORT**

Describe about the introduction.

Describe about the literature review.

Describe about the proposed system.

Describes about the architecture and modules of proposed system.

Describes about the implementation and result.

Describes about conclusion and future work.

**LITERATURE REVIEW**

**1.2.1 face verification across age progression**

**Author: Narayanan Ramanathan**

Human faces undergo considerable number of variations with aging. They develop a Bayesian age difference classifier that classifies face images of individual based on age differences and performs face verification across age progression.

Frontal face recovery methods to recover the frontal face of an individual from a non-frontal face image. Blanz and Vetter in their seminal work on 3-d morphable models for face estimate the 3-d shapes and texture of faces from a single face image and perform face recognition across varying pose and illumination.

**Advantage**

Manifestation of aging effects in human faces variation (shapes, texture) can be best understood using 3-d scans of human heads and becoming increasingly available.

**Disadvantage**

Lack of database of age progressed face image of individuals. Modelling the complex shape variation human faces undergo during one’s younger’s years or the textural variations that are observed during the later years is a very challenging task.

**1.2.2 face verification across age progression using discriminative methods**

**Author: haibin ling, Stefano soatto, Narayanan Ramanathan and David w. Jacobs**

Face verification in the presence of age progression is an important problem that has not been widely addressed. We study the problem by designing and evaluating discriminative approaches. These directly tackle verification tasks without explicit age modelling, which is a hard problem by itself. First, find that the gradient orientation (go), after discarding magnitude information, provides a simple but effective representation for this problem.

This representation is further improved when hierarchical information is used, which results in the use of the gradient orientation pyramid (gop). When combined with a support vector machine (svm) gop demonstrates excellent performance in all our experiments, in comparison with seven different approaches including two commercial systems.

Experiments are conducted on the fgnet dataset and two large passport datasets, one of them being the largest ever reported for recognition tasks.

Second, taking advantage of these datasets, empirically study how age gaps and related issues (including image quality, spectacles, and facial hair) affect recognition algorithms. We found surprisingly that the added difficulty of verification produced by age gaps becomes saturated after the gap is larger than four years, for gaps of up to ten years. In addition, find that image quality and eyewear present more of a challenge than facial hair.

The new approach demonstrated very promising results on two challenging passport databases and the fgnet dataset. In addition, being a discriminative approach, the proposed method requires no prior age knowledge and does not rely on age estimation and simulation algorithms.

The effect of the aging process on verification algorithms are studied empirically. In the experiments we observed that the difficulty of face verification algorithms saturated after the age gap is larger than four years (up to ten years) and also studied the effects of age-related issues including image quality, presence of spectacles, and facial hair.

**Process:**

First, testing on a large public dataset will be conducted for deeper understanding of the proposed approaches. We plan to work on the morph dataset for this purpose.

Second, apply other discriminative approaches (e.g., boosting) for simultaneous feature analysis and classification.

**1.2.3 face recognition and retrieval using cross-age reference coding with cross-age celebrity dataset**

**Author: bor-chun chen, chu-song chen, and Winston h. Hsu**

They use a data –driven method to address the cross-age face recognition problem, called cross-age reference coding (cacr). Carc can encode the low-level feature of a face image with an age-invariant reference space. Carc achieve high accuracy in face recognition & retrieval cross age.

They introduce a large-scale dataset called cross-age celebrity dataset (cacd) for evaluating face recognition and retrieval across age. Cacd-vs contains 4,000 image pairs across ages and it is constructed by carefully checking both web and image contents.

**Advantage**

The methods perform better than average human combing results from multiple humans can achieve higher performance.

**Disadvantage**

The performance in cross dataset setting drops considerably. The drop is probably caused by the huge difference between the appearance distributions of two datasets.

**1.2.4 on recognizing face images with weight and age variations**

**Author: maneet Singh, shruti Nagpal, Richa Singh, and Mayank vatsa**

With the increase in age, there are changes in skeletal structure, muscle mass, and body fat. For recognizing faces with age variations, researchers have generally focused on the skeletal structure and muscle mass. However, the effect of change in body fat has not been studied with respect to face recognition. In this paper, we incorporate weight information to improve the performance of face recognition with age variations.

The proposed algorithm utilizes neural network and random decision forest to encode age variations across different weight categories. The results are reported on the whoisit database prepared by the authors containing 1109 images from 110 individuals with age and weight variations. The comparison with existing state-of-the-art algorithms and commercial system on whoisit and fg-net databases shows that the proposed algorithm outperforms existing algorithms significantly.

**Advantage**

Incorporating weight information for recognizing age separated face images improves the identification performance.

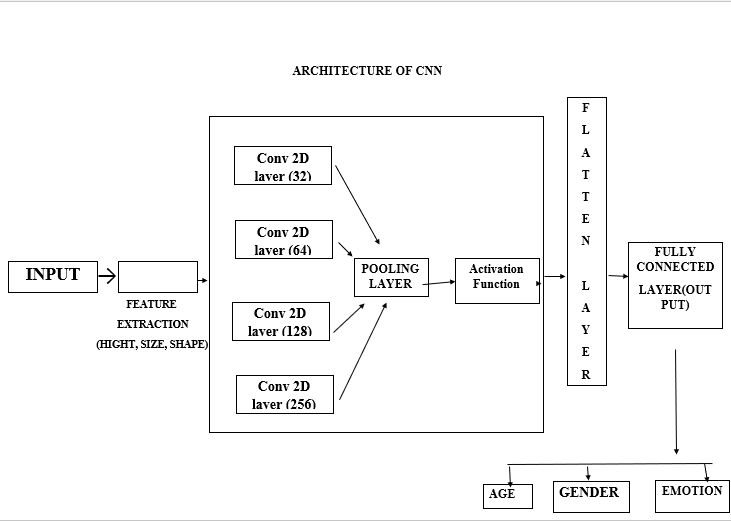
**PROPOSED SYSTEM**

Age invariant face identification with emotion and gender identification using CNN. Machine learning is used for age prediction. In this proposed system we can recognize the face then measure the features and use the algorithm to find out the age. Two models are used in this project that is one for age-gender prediction and other model is trained for emotion recognition using CNN.

By recognizing the emotion of a person, we can improve the recommendation system an improvement in the performance of these tasks was observed by using the convolutional neural network (CNN). The dataset was obtained from outface dataset for age gender classification and fer2013 dataset for emotion recognition. The dataset was collected from Kaggle.

**Advantage**

* Take less time to compiling.
* Better accuracy will be high.
* Consists of high average performance.

**Architecture of the proposed system**

**Convolution layers:**

Convolution layer extract features from the input. Multiple filters are applied to the given image. Filter moves on the input image by the stride value. Then sum all the stride and feature map represent an output value. An activation function is applied to obtained the output. All of the layers are followed by a relu activation functions. Output of each stride is known as feature map.

A convolutional have one or more feature map. Each of these feature maps is connected to the next layer. Convolution is an orderly procedure where two sources of information are intertwined; it’s an operation that changes a function into something else.

Convolutions have been used for a long time typically in image processing to blur and sharpen images, but also to perform other operations. (e.g., enhance edges and emboss) cnns enforce a local connectivity pattern between neurons of adjacent layers. Cnns make use of filters (also known as kernels), to detect what features, such as edges, are present throughout an image. There are four main operations in a CNN:

* Convolution
* Non-linearity (relu)
* Pooling or sub sampling
* Classification (fully connected layer)

  The first layer of a convolutional neural network is always a**convolutional layer.**Convolutional layers apply a convolution operation to the input, passing the result to the next layer. A convolution converts all the pixels in its receptive field into a single value. For example, if you would apply a convolution to an image, you will be decreasing the image size as well as bringing all the information in the field together into a single pixel.

The final output of the convolutional layer is a vector. Based on the type of problem we need to solve and on the kind of features we are looking to learn, we can use different kinds of convolutions.

**The 2d convolution layer**

The most common type of convolution that is used is the 2d convolution layer and is usually abbreviated as conv2d. A filter or a kernel in a conv2d layer “slides” over the 2d input data, performing an elementwise multiplication. As a result, it will be summing up the results into a single output pixel. The kernel will perform the same operation for every location it slides over, transforming a 2d matrix of features into a different 2d matrix of features.

**The dilated or atrous convolution**

This operation expands window size without increasing the number of weights by inserting zero-values into convolution kernels. Dilated or atrous convolutions can be used in real time applications and in applications where the processing power is less as the ram requirements are less intensive.

**Separable convolutions**

There are two main types of separable convolutions: spatial separable convolutions, and depth wise separable convolutions. The spatial separable convolution deals primarily with the spatial dimensions of an image and kernel: the width and the height. Compared to spatial separable convolutions, depth wise separable convolutions work with kernels that cannot be “factored” into two smaller kernels. As a result, it is more frequently used.

**Transposed convolutions**

These types of convolutions are also known as deconvolutions or fractionally strided convolutions. A transposed convolutional layer carries out a regular convolution but reverts its spatial transformation.

**Pooling layer:**

The main function of the pooling layer is to reduce the spatial size of the input. It helps to make the input representation smaller and convenient. Max pooling operation are used to take out only maximum from a pool and the rest is dropped. Activation functions to produce the output that feature map is connected to the next convolutional layer.

A problem with the output feature maps is that they are sensitive to the location of the features in the input. One approach to address this sensitivity is to down sample the feature maps. This has the effect of making the resulting down sampled feature maps more robust to changes in the position of the feature in the image, referred to by the technical phrase “local translation invariance.”

Pooling layers provide an approach to down sampling feature maps by summarizing the presence of features in patches of the feature map. Two common pooling methods are average pooling and max pooling that summarize the average presence of a feature and the most activated presence of a feature respectively.

Operation works and how to implement it in convolutional neural networks. Pooling is required to down sample the detection of features in feature maps. How to calculate and implement average and maximum pooling in a convolutional neural network. How to use global pooling in a convolutional neural network.

A pooling layer is a new layer added after the convolutional layer. Specifically, after a nonlinearity (e.g., relu) has been applied to the feature maps output by a convolutional layer; for example, the layers in a model may look as follows:

* Input image
* Convolutional layer
* Nonlinearity
* Pooling layer

The addition of a pooling layer after the convolutional layer is a common pattern used for ordering layers within a convolutional neural network that may be repeated one or more times in a given model.

The pooling layer operates upon each feature map separately to create a new set of the same number of pooled feature maps.

Pooling involves selecting a pooling operation, much like a filter to be applied to feature maps. The size of the pooling operation or filter is smaller than the size of the feature map; specifically, it is almost always 2×2 pixels applied with a stride of 2 pixels.

This means that the pooling layer will always reduce the size of each feature map by a factor of 2, e.g., each dimension is halved, reducing the number of pixels or values in each feature map to one quarter the size. For example, a pooling layer applied to a feature map of 6×6 (36 pixels) will result in an output pooled feature map of 3×3 (9 pixels).

The pooling operation is specified, rather than learned. Two common functions used in the pooling operation are:

**Average pooling**: calculate the average value for each patch on the feature map.

**Maximum pooling (or max pooling)**: calculate the maximum value for each patch of the feature map.

The result of using a pooling layer and creating down sampled or pooled feature maps is a summarized version of the features detected in the input. They are useful as small changes in the location of the feature in the input detected by the convolutional layer will result in a pooled feature map with the feature in the same location. This capability added by pooling is called the model’s invariance to local translation.

Mathematically the pooling operation works by sliding a two-dimensional filter across the three-dimensional feature map and summarizes the features that come in the way of filters. So, if a feature map of dimension h \* w \* c is presented then the output obtained by the pooling will be.

(h – f+ 1)/ s \* (w – f + 1) \* c

Where

H and we are the height and width of the feature map respectively

C is the channel presented in the feature map

F is the size of the filter

S is stride length

  Pooling layers reduces the dimension of the feature maps, so if in any condition where the structure or the dimensions of any data is high, we can use the pooling layers with the convolutional layer so the feature map generated by the convolutional layer is high dimensional can be reduced in the low dimensional and rest computational work will cost low amount of efforts.

Pooling layers summarizes the featured map so that the model will not need to be trained on precisely positioned features. This makes a model more reliable and robust.

Pooling layers into three categories.

Max pooling layer

Min pooling layer

Average pooling layer

Global pooling layer

**Max pooling**

In max-pooling, the layer operates with the most prominent feature of the feature map provided by the convolutional layer. More basically we can say it selects the maximum valued element from the region captured by the filter in any feature map.

**Min pooling**

In min pooling, the layer operates with the most non-prominent feature of the feature map provided by the convolutional layer. More basically we can say it selects the minimum valued element from the region captured by the filter in any feature map.

**Average pooling**

In average pooling, the layer operates by selecting the average values of the elements available in the patch of the feature map. Basically, the whole feature map gets down sampled to the average value captured by the region of the feature map. So, the max-pooling gives the most prominent feature of any patch where the average pooling gives the average of the covered area.

**Global pooling layer**

The global pooling layer takes the average or max of the feature map and the resulting vector can directly feed into the SoftMax layer which prohibits the chances of overfitting so basically, we can divide the global pooling layer into two types.

* Global average pooling.
* Global max pooling

**Global average pooling**

The global average pooling layer takes the average of each feature map then sends the average value to the activation layer.

**Global max pooling**

The global max-pooling layer takes the maximum of each feature map and sends it directly to the activation layer in a fully connected layer.

**Flatten layer:**

Once the pooled feature map is obtained the next step is flatten it. Flatten layer involves transforming entire pooled feature map matrix into a single dimension(1d). This layer is connected to fully connected layer.

Flattening is converting the data into a 1-dimensional array for inputting it to the next layer. We flatten the output of the convolutional layers to create a single long feature vector. And it is connected to the final classification model, which is called a fully-connected layer.

In other words, we put all the pixel data in one line and make connections with the final layer. After a series of convolution and pooling operations on the feature representation of the image, we then flatten the output of the final pooling layers into a single long continuous linear array or a vector.

The process of converting all the resultant 2-d arrays into a vector is called flattening.

Flatten output is fed as input to the fully connected neural network having varying numbers of hidden layers to learn the non-linear complexities present with the feature representation.

**Output layer:**

The output layer is to identify the multiple set of outputs and choose one best case of image using soft max activation function. The convolution and pooling layer in combination are used for feature extraction while full connected layers are used for classification. The output layer in an artificial neural network is the last layer of neurons that produces given outputs for the program.

Though they are made much like other artificial neurons in the neural network, output layer neurons may be built or observed in a different way, given that they are the last “actor” nodes on the network.

A typical traditional neural network has three types of layers:

One or more input layers, one or more hidden layers, and one or more output layers. Simple feedforward neural networks with three individual layers provide basic easy-to-understand models. More sophisticated, innovative neural networks may have more than one of any type of layer – and as mentioned, each type of layer may be built differently.

A traditional artificial neuron is composed of some weighted inputs, a transformation function and activation function corresponding to the biological neuron’s axon. However, output layer neurons may be designed differently in order to streamline and improve the end results of the iterative process.

In a sense, the output layer coalesces and concretely produces the end result. However, to understand the neural network better, it is important to look at the input layer, hidden layers and output layer together as a whole.

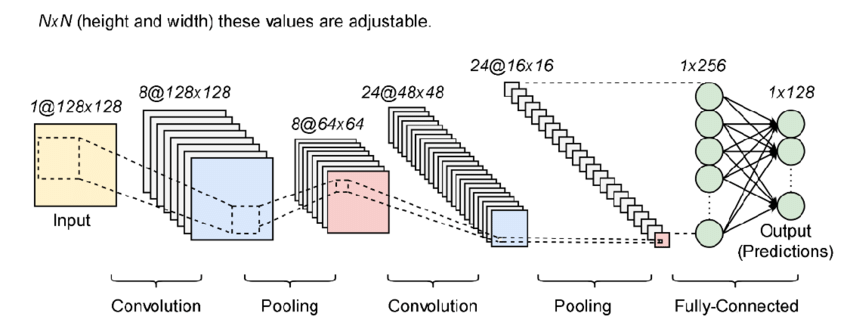
**MODULES**

Fig: architecture of a standard convolutional neural network

* Importing the libraries
* Loading the dataset (input image)
* Training the dataset using CNN
* Output prediction

**Importing the libraries**

a) Pandas’ library is used for loading the dataset. It eases data analysis, data manipulation and cleaning the data. Software library written for the python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause bsd license.

**Advantages**

* Fast and efficient for manipulating and analysing data.
* Data from different file objects can be loaded.
* Easy handling of missing data (represented as nan) in floating point as well as non-floating-point data.
* Size mutability: columns can be inserted and deleted from data frame and higher dimensional objects.
* Data set merging and joining.
* Flexible reshaping and pivoting of data sets
* Provides time-series functionality.
* Powerful group by functionality for performing split-apply-combine operations on data sets.

b) NumPy (NumPy stands for numerical python.) Library is used for mathematical operation. This library supports large matrices and multidimensional data. It also has functions for working in domain of linear algebra, Fourier transform, and matrices.

In python we have lists that serve the purpose of arrays, but they are slow to process. NumPy aims to provide an array object that is up to 50x faster than traditional python lists. The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.

Arrays are very frequently used in data science, where speed and resources are very important. NumPy is a python library and is written partially in python, but most of the parts that require fast computation are written in c or c++.

**Advantages**

* NumPy uses much less memory to store data.
* The NumPy arrays takes significantly less amount of memory as compared to python lists. It also provides a mechanism of specifying the data types of the contents, which allows further optimisation of the code.

c) Matplotlib is used for plotting a numerical data. It performs high defined figures like graphs, histogram, scatterplot etc. Matplotlib is a low-level graph plotting library in python that serves as a visualization utility. Matplotlib is mostly written in python, a few segments are written in c, objective-c and JavaScript for platform compatibility.

Matplotlib is an amazing visualization library in python for 2d plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader scipy stack. One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.  Python 3 support started with matplotlib

**Advantages**

* Matplotlib supports various types of graphical representations like bar graphs, histograms, line graph, scatter plot, stem plots, etc.
* Matplotlib can be used in multiple ways including python scripts, the python and ipython shells, Jupiter notebooks. Matplotlib is a 2-d plotting library.

d) Keras is used for creating a deep learning model (CNN). Keras is a deep learning api written in python, running on top of the machine learning platform [TensorFlow](https://github.com/tensorflow/tensorflow). It was developed with a focus on enabling fast experimentation. Being able to go from idea to result as fast as possible is key to doing good research.

Keras runs on top of open-source machine libraries like TensorFlow, theano or cognitive toolkit (cntk). Theano is a python library used for fast numerical computation tasks. TensorFlow is the most famous symbolic math library used for creating neural networks and deep learning models.

TensorFlow is very flexible and the primary benefit is distributed computing. Cntk is deep learning framework developed by Microsoft. It uses libraries such as python, c#, c++ or standalone machine learning toolkits. Theano and TensorFlow are very powerful libraries but difficult to understand for creating neural networks.

Keras is based on minimal structure that provides a clean and easy way to create deep learning models based on TensorFlow or theano. Keras is designed to quickly define deep learning models. Well, keras is an optimal choice for deep learning applications.

**Features**

* Keras leverages various optimization techniques to make high level neural network api easier and more performant. It supports the following features :
* Consistent, simple and extensible api.
* Minimal structure - easy to achieve the result without any frills.
* It supports multiple platforms and backends.
* It is user friendly framework which runs on both cpu and gpu.
* Highly scalability of computation.

**Benefits**

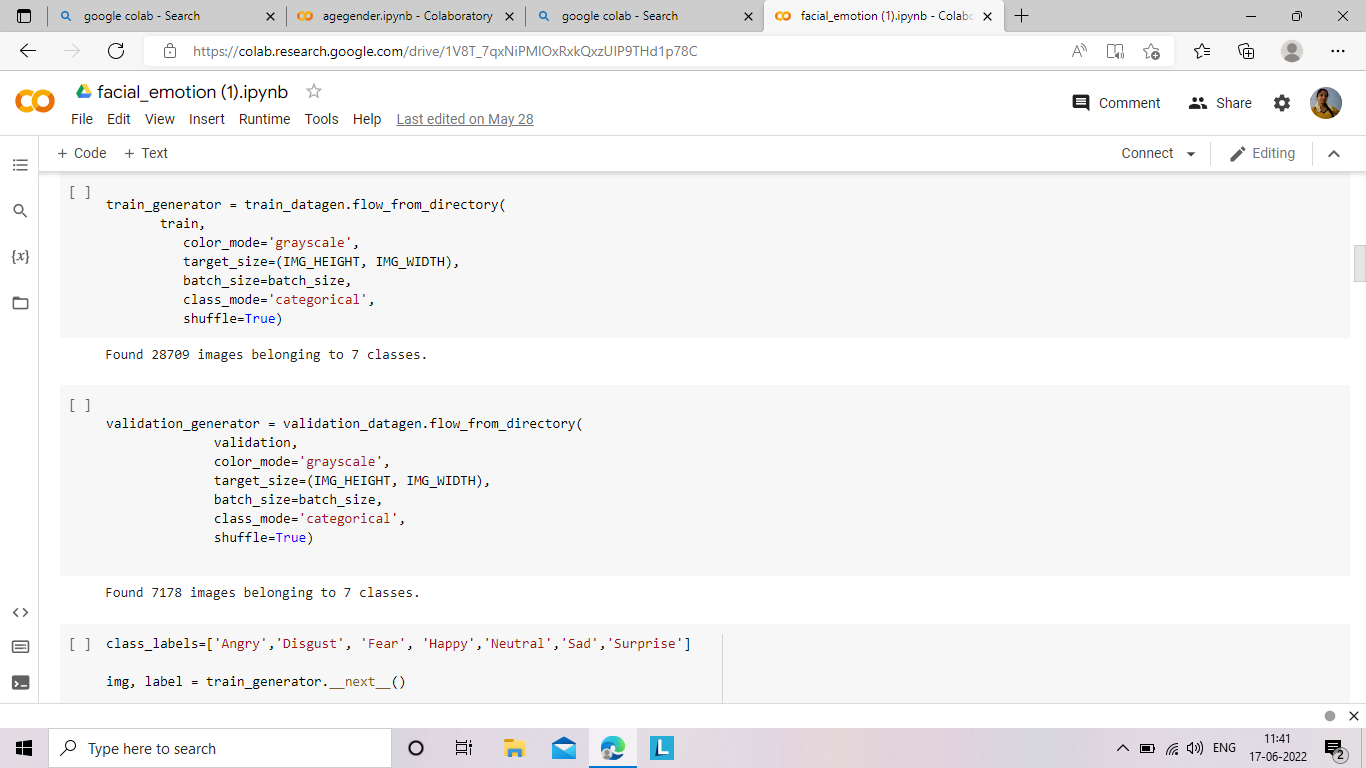
Keras is highly powerful and dynamic framework and comes up with the following advantages −

* Larger community support.
* Easy to test.
* Keras neural networks are written in python which makes things simpler.
* Keras supports both convolution and recurrent networks.
* Deep learning models are discrete components, so that, you can combine into many ways.

**Advantages**

* Simple -- but not simplistic. Keras reduces developer cognitive load to free you to focus on the parts of the problem that really matter.
* Flexible -- keras adopts the principle of progressive disclosure of complexity: simple workflows should be quick and easy, while arbitrarily advanced workflows should be possible via a clear path that builds upon what you've already learned.
* Powerful -- keras provides industry-strength performance and scalability: it is used by organizations and companies including nasa, YouTube, or waymo.

**Training the dataset**

**Preprocessing** (pre-processing refers to the transformations applied to our data before feeding it to the algorithm. Data pre-processing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis.) The image to correct unwanted color, reshape, zoom, reshape, flip, rotation, cropping using image data generator.

**Need of data pre-processing**

For achieving better results from the applied model in machine learning projects the format of the data has to be in a proper manner.

Some specified machine learning model needs information in a specified format, for example, random forest algorithm does not support null values, therefore to execute random forest algorithm null values have to be managed from the original raw data set.

Another aspect is that the data set should be formatted in such a way that more than one machine learning and deep learning algorithm are executed in one data set, and best out of them is chosen.

**1. Rescale data**

When our data is comprised of attributes with varying scales, many machine learning algorithms can benefit from rescaling the attributes to all have the same scale.

This is useful for optimization algorithms used in the core of machine learning algorithms like gradient descent.

It is also useful for algorithms that weight inputs like regression and neural networks and algorithms that use distance measures like k-nearest neighbours.

**2. Binarize data (make binary)**

We can transform our data using a binary threshold. All values above the threshold are marked 1 and all equal to or below are marked as 0.

This is called binarizing your data or threshold your data. It can be useful when you have probabilities that you want to make crisp values. It is also useful when feature engineering and you want to add new features that indicate something meaningful.

**3. Standardize data**

Standardization is a useful technique to transform attributes with a gaussian distribution and differing means and standard deviations to a standard gaussian distribution with a mean of 0 and a standard deviation of 1. To correct illumination variation due to shadow and underexposure. Next, convert rgb image to gray scale image.

**Training data** is an extremely large dataset that is used to teach a [machine learning](https://www.techopedia.com/definition/8181/machine-learning-ml) model. Training data is used to teach [prediction models](https://www.techopedia.com/definition/14004/predictive-modeling) that use machine learning algorithms how to extract features that are relevant to specific business goals. For [supervised ml](https://www.techopedia.com/definition/30389/supervised-learning) models, the training data is labelled. The data used to train [unsupervised ml](https://www.techopedia.com/definition/30390/unsupervised-learning)models is not labelled.

The idea of using training data in machine learning programs is a simple concept, but it is also very foundational to the way that these technologies work. The training data is an initial set of data used to help a program understand how to apply technologies like neural networks to learn and produce sophisticated results. It may be complemented by subsequent sets of data called validation and testing sets.

Training data is also known as a training set, training dataset or learning set. Training data is the data you use to train an algorithm or machine learning model to predict the outcome you design your model to predict. Test data is used to measure the performance, such as accuracy or efficiency, of the algorithm you are using to train the machine.

The training data is an initial set of data used to help a program understand how to apply technologies like neural networks to learn and produce sophisticated results. It may be complemented by subsequent sets of data called validation and testing sets.

Traditional programming algorithms follow a set of instructions to transform data into a desired output with no deviations.

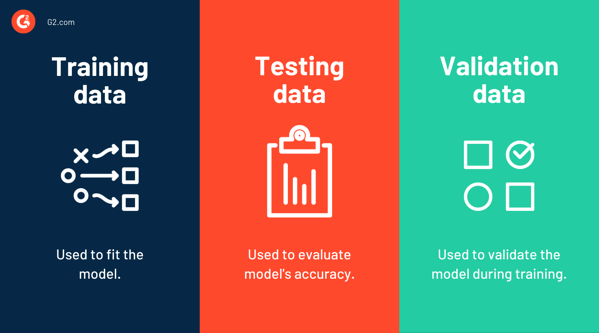
[Machine learning algorithms](https://monkeylearn.com/blog/machine-learning-algorithms/), on the other hand, enable machines to solve problems based on past observations. The great thing about machine learning models is that they improve over time, as they’re exposed to relevant training data.

Let’s break the data training process down into three steps:

**1. Feed** a machine learning model training input data

**2. Tag** training data with a desired output. The model transforms the training data into [text vectors](https://monkeylearn.com/blog/beginners-guide-text-vectorization/) – numbers that represent data features.

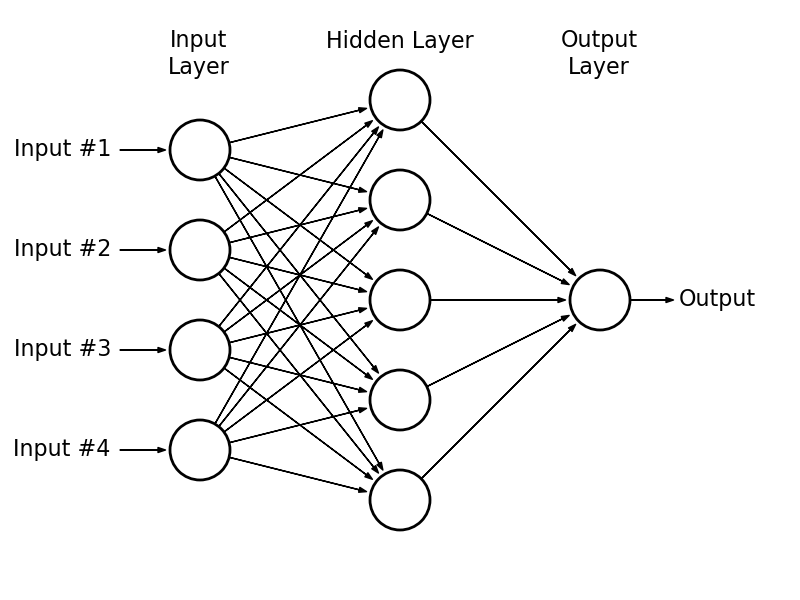
**3. Test** your model by feeding it testing (or unseen) data. Algorithms are trained to associate feature vectors with tags based on manually tagged samples, then learn to make predictions when processing unseen data.



Using a dataset to training the model and also testing the data.

Testing data can predict the accurate output. Here we are using the **convolutional neural network** model to trained the dataset. Different convolutions to extract different features like edges, height, highlighted patterns from the images.

In multilayer perceptron’s (mlp), the vanilla neural networks, each layer’s neurons connect to **all** the neurons in the next layer. We call this type of layers **fully connected**.



A convolutional neural network is different: they have convolutional layers. On a fully connected layer, each neuron’s output will be a linear transformation of the previous layer, composed with a non-linear activation function (e.g., relu or sigmoid).

Conversely, the output of each neuron in a **convolutional layer** is only a function of a (typically small) **subset** of the previous layer’s neurons.

For each classification task , the results are reported in the form of confusion matrix( in the field of [machine learning](https://en.wikipedia.org/wiki/Machine_learning) and specifically the problem of [statistical classification](https://en.wikipedia.org/wiki/Statistical_classification), a confusion matrix, also known as an error matrix,[[9]](https://en.wikipedia.org/wiki/Confusion_matrix#cite_note-9) is a specific table layout that allows visualization of the performance of an algorithm, typically a [supervised learning](https://en.wikipedia.org/wiki/Supervised_learning) one (in [unsupervised learning](https://en.wikipedia.org/wiki/Unsupervised_learning) it is usually called a matching matrix).

Each row of the [matrix](https://en.wikipedia.org/wiki/Matrix_(mathematics)) represents the instances in an actual class while each column represents the instances in a predicted class, or vice versa – both variants are found in the literature.[[10]](https://en.wikipedia.org/wiki/Confusion_matrix#cite_note-Powers2011-10) the name stems from the fact that it makes it easy to see whether the system is confusing two classes (i.e. Commonly mislabelling one as another).

It is a special kind of [contingency table](https://en.wikipedia.org/wiki/Contingency_table), with two dimensions ("actual" and "predicted"), and identical sets of "classes" in both dimensions (each combination of dimension and class is a variable in the contingency table).)

**Output prediction**

The output of the CNN is 4d matrix where batch size would be the same as input batch size but the other 3 dimension(3d) of the image might change depending upon the values of filter, kernel size and padding we use. Capturing an image and predicting the output.

**IMPLEMENTATION**

The images are stored in the file directory which the keras library can work with. The CNN has implemented in python using the keras. The convolutional neural network, or CNN for short, is a specialized type of neural network model designed for working with two-dimensional image data, although they can be used with one-dimensional and three-dimensional data.

Central to the convolutional neural network is the convolutional layer that gives the network its name. This layer performs an operation called a “convolution “.

In the context of a convolutional neural network, a convolution is a linear operation that involves the multiplication of a set of weights with the input, much like a traditional neural network. Given that the technique was designed for two-dimensional input, the multiplication is performed between an array of input data and a two-dimensional array of weights, called a filter or a kernel.

The filter is smaller than the input data and the type of multiplication applied between a filter-sized patch of the input and the filter is a dot product. A [dot product](https://en.wikipedia.org/wiki/Dot_product) is the element-wise multiplication between the filter-sized patch of the input and filter, which is then summed, always resulting in a single value. Because it results in a single value, the operation is often referred to as the “scalar product “.

In a convolution layer, first multiply the 3\*3(1 stride) in the matrix input image with apply a filter. Each element is multiplied with an element in the corresponding location. Then sum all the results, which is one output value. Output of each stride is known as feature map.

Feature output = [size of input - kernel] +1

An activation function is applied to obtained the output. All of the layers are followed by a rectified linear unit (relu) activation function. Relu convert all negative inputs to zero which decrease the ability of the model to fit or train from the data properly.

**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.

As such, a careful choice of activation function must be made for each deep learning neural network project. In this tutorial, you will discover how to choose activation functions for neural network models.

After completing this tutorial, you will know:

* Activation functions are a key part of neural network design.
* The modern default activation function for hidden layers is the relu function.
* The activation function for output layers depends on the type of prediction problem.

Divided into three parts; they are:

* Activation functions
* Activation for hidden layers
* Activation for output layers

The rectified linear activation function or relu is a **non-linear** function or **piecewise linear** function that will output the input directly if it is positive, otherwise, it will output zero.  
It is the most commonly used activation function in neural networks, especially in convolutional neural networks (cnns) & multilayer perceptron’s.

It is simple yet it is more effective than its predecessors like sigmoid or tanh.We can implement a simple relu function with python code using an if-else statement as,

Def relu(x):

If x>0:

Return x

Else:

Return 0

The rectified linear activation function or relu for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance.

The rectified linear activation function for deep learning neural networks.

The sigmoid and hyperbolic tangent activation functions cannot be used in networks with many layers due to the vanishing gradient problem.

The rectified linear activation function overcomes the vanishing gradient problem, allowing models to learn faster and perform better.

The rectified linear activation is the default activation when developing multilayer perceptron and convolutional neural networks.

At the first glance after plotting relu it seems to be a linear function. But in fact, it is a non-linear function and it is required so as to pick up & learn complex relationships from the training data.

It acts as a linear function for positive values and as a non-linear activation function for negative values.

When we're using an optimizer such as sgd (stochastic gradient descent) during backpropagation, it acts like a linear function for positive values and thus it becomes a lot easier when computing the gradient. This near linearity allows to preserve properties and makes linear models easy to be optimized with gradient based algorithms.

Also, relu adds more sensitivity to weighted sum and thus this avoids neurons from getting saturated (i.e., when there is little or no variation in the output).

Pooling layer is nonlinear sampling layer. Pooling layer reduce the dimension, reducing the spatial size of the network. The pooling operation involves sliding a two-dimensional filter over each channel of feature map and summarising the features lying within the region covered by the filter.

 pooling layers are used to reduce the dimensions of the feature maps. Thus, it reduces the number of parameters to learn and the amount of computation performed in the network.

 the pooling layer summarises the features present in a region of the feature map generated by a convolution layer. So, further operations are performed on summarised features instead of precisely positioned features generated by the convolution layer. This makes the model more robust to variations in the position of the features in the input image.

**Types of pooling layers**:  
   
**Max Pooling**

Max pooling is a pooling operation that selects the maximum element from the region of the feature map covered by the filter. Thus, the output after max-pooling layer would be a feature map containing the most prominent features of the previous feature map.

**Average pooling**

Average pooling computes the average of the elements present in the region of feature map covered by the filter. Thus, while max pooling gives the most prominent feature in a particular patch of the feature map, average pooling gives the average of features present in a patch.

**Global pooling**

Global pooling reduces each channel in the feature map to a single value. Thus, a **nh x nw x nc** feature map is reduced to **1 x 1 x nc** feature map. This is equivalent to using a filter of dimensions **nh x nw** i.e., the dimensions of the feature map.   
Further, it can be either global max pooling or global average pooling.

Max pooling operation are used to take out y maximum from a pool and the rest is dropped. Max pool size is 3.

**Compiling:**

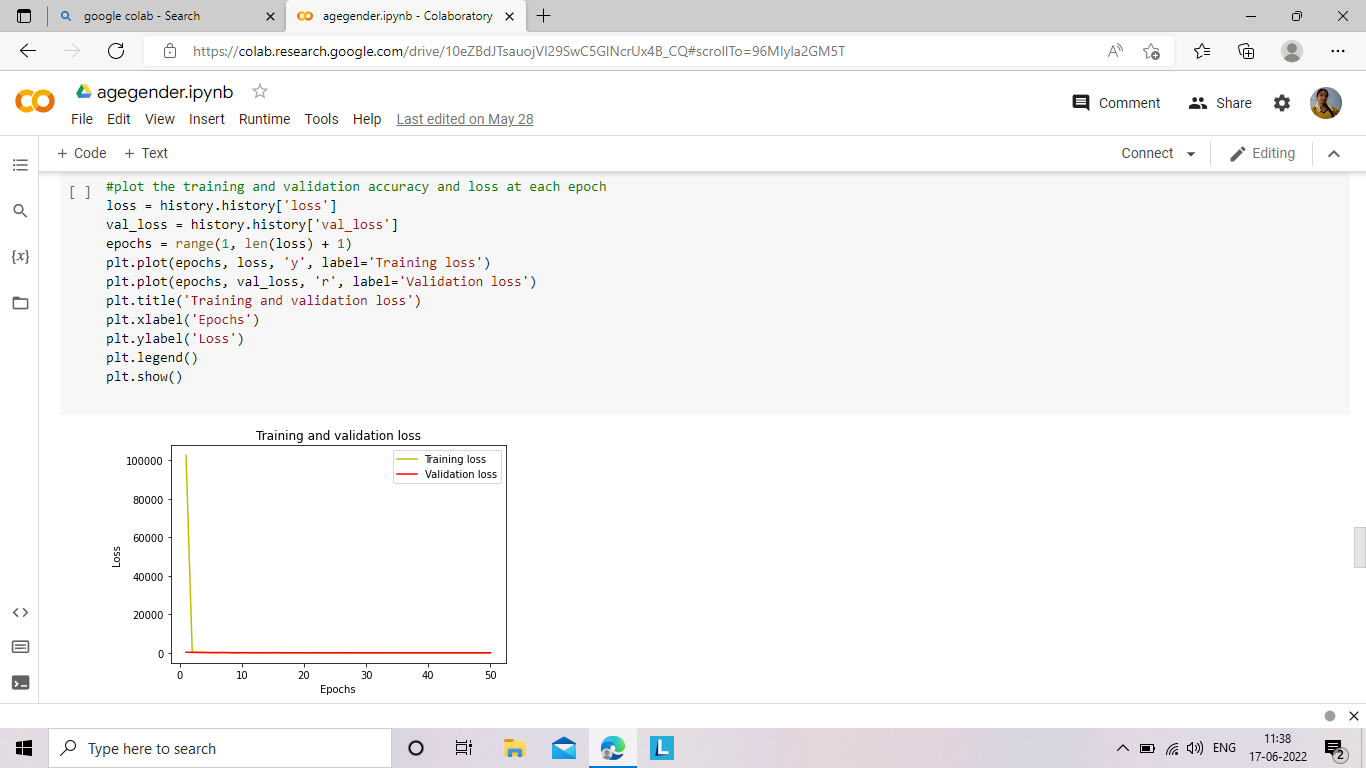
Adam optimizer is one of the widely used powerful and give best accuracy. Adaptive moment estimation is an algorithm for optimization technique for gradient descent. The method is really efficient when working with large problem involving a lot of data or parameters. It requires less memory and is efficient. Intuitively, it is a combination of the ‘gradient descent with momentum’ algorithm and the ‘rmsp’ algorithm.

Optimization is to reduce the time taken to train the model. It learning rate to reduce the losses and improve the accuracy. Optimizer solve the optimization problem by minimizing the function.

Loss function is used to calculate the gradients. The gradients are update the weights.

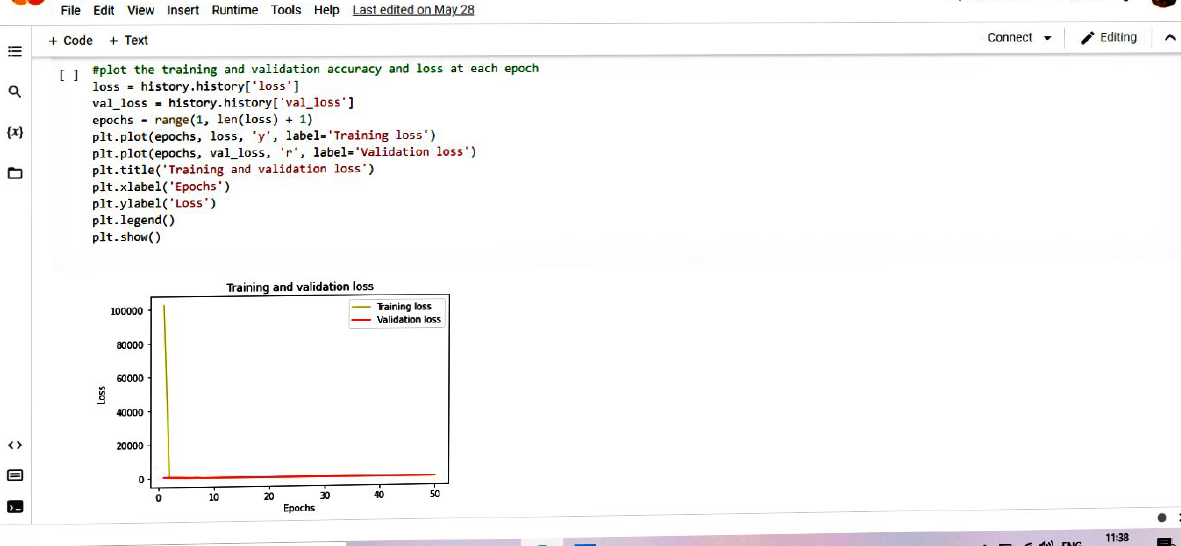
The loss function in a neural network quantifies the difference between the expected outcome and the outcome produced by the machine learning model. From the loss function, we can derive the gradients which are used to update the weights. The average over all losses constitutes the cost.

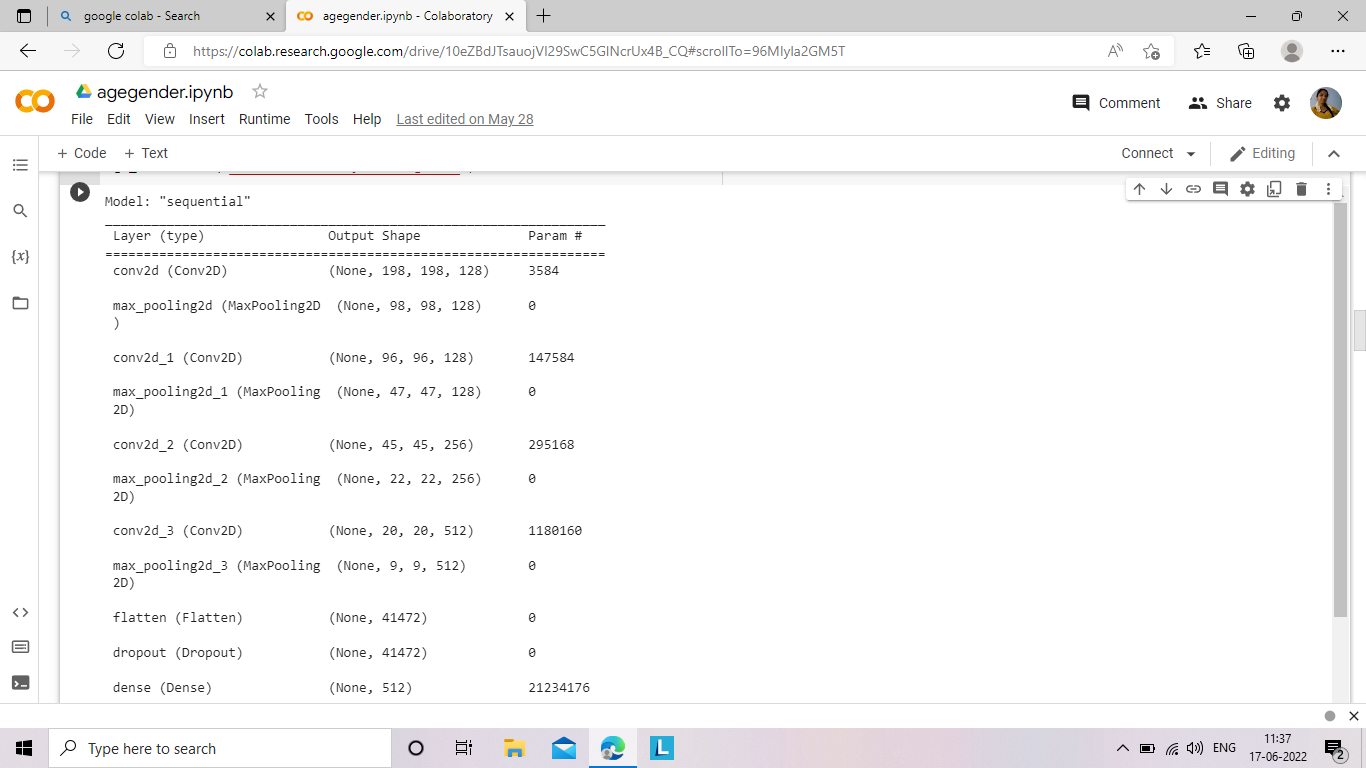
Plot the training and validation. Using a confusing metrics, plot the loss heat map.

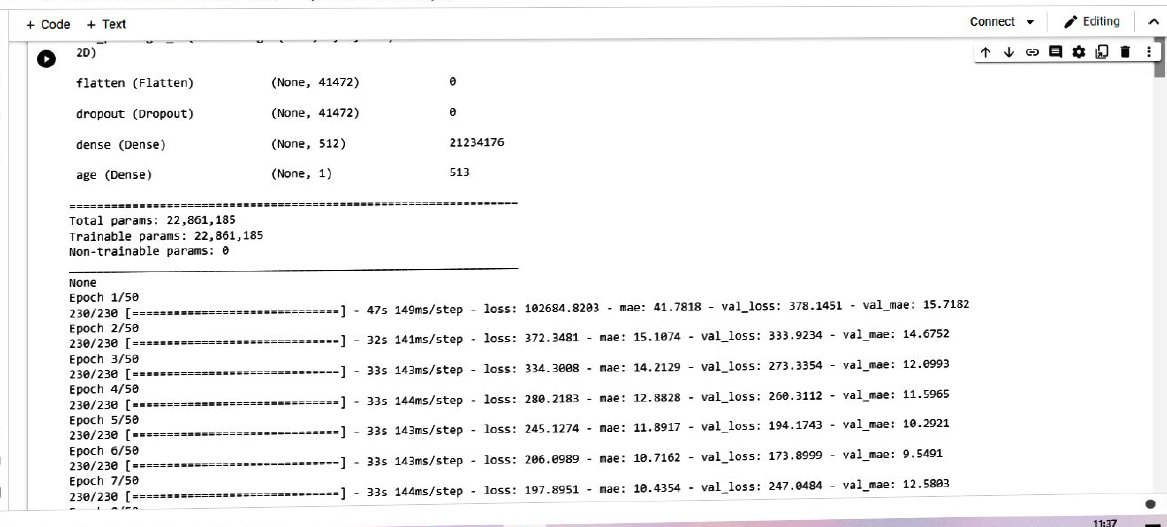
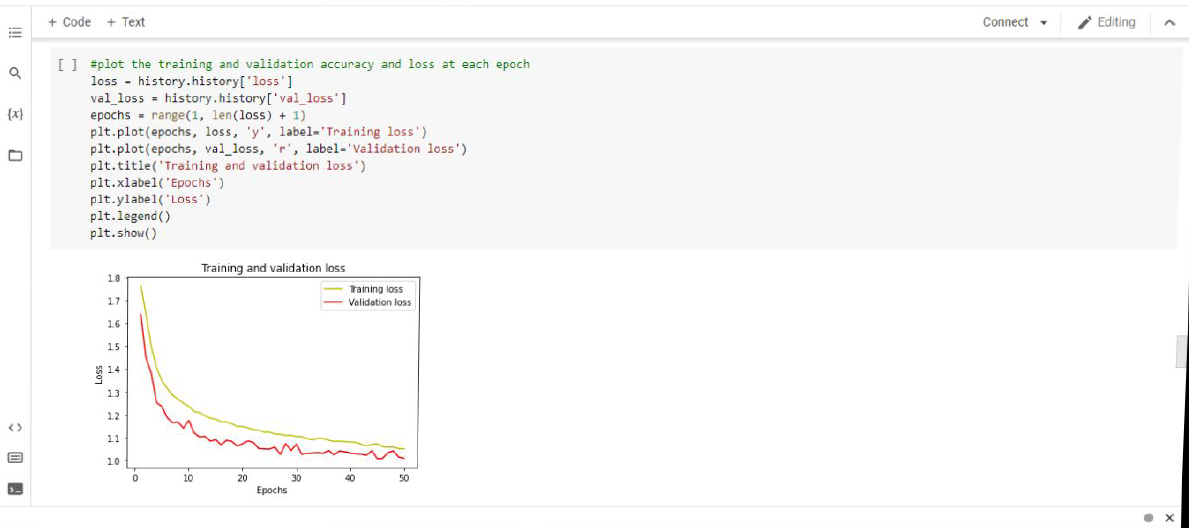


The model derived has been verified using accuracy and loss metrics to see how well the model has fit the data.

Finally, the model has been tested by using an image from a holdout set. The below table contains a detailed architecture model of the implementation.







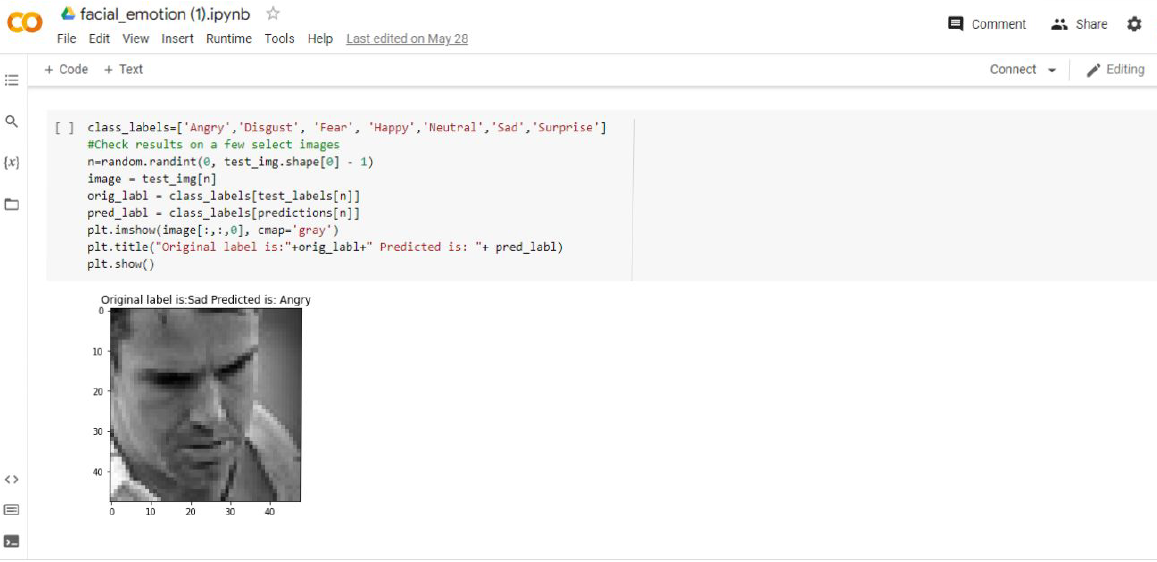
**Training and validation loss in emotion**



Training and validation accuracy in emotion



In the confusion\_ matrix verify the accuracy of each class.



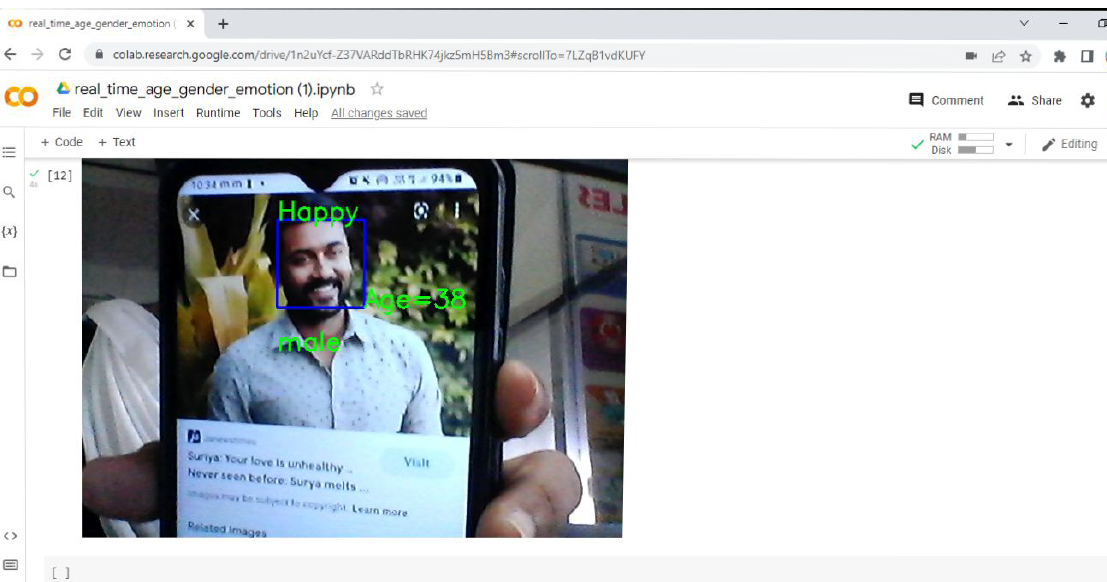
Real time age, gender, emotion set.







**RESULT WITH DEMO**





Results for our real-time emotion and gender classification tasks in hidden faces are noticed. The whole real-time pipeline involves: detection of face , classification of emotion and classification of gender in one single step .this implementation can work on both group as well as single images and also live input given through webcam.

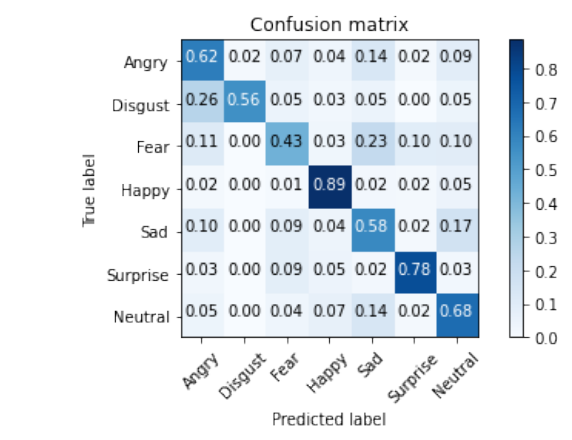


Fig: normalized confusion matrix

**COMPARISON OF RESULT**

**Existing system**

* In an existing system contain less accuracy.
* Low average performance.
* Existing system, take lot of time to compiling.

**Proposed system**

* In a proposed system contain better accuracy
* High average performance
* Take less time to compiling

**CONCLUSION & FUTURE ENHANCEMENT:**

* In a proposed system has been performed on two dataset utkface dataset for age gender classification and fer2013 dataset for emotion recognition are used for training and testing.
* For the demonstrated, capture our image and prediction is done by using that image (gender, age and emotion)
* Different lighting and quality of camera may affect the identification.
* The proposed methods give improved recognition accuracy for the image having ageing variation.
* Illumination is the challenging issues when capturing an image in future will overcome this problem.

**REFERENCE**

*[1]* [*https://www.analyticsvidhya.com/blog/2017/06/architecture-of-convolutional-neural-networks-simplified-demystified/#:~:text=the%20output%20layer%20in%20a,as%20desired%20by%20the%20network*](https://www.analyticsvidhya.com/blog/2017/06/architecture-of-convolutional-neural-networks-simplified-demystified/#:~:text=The%20output%20layer%20in%20a,as%20desired%20by%20the%20network)*.*

*[2]* [*https://www.techopedia.com/definition/33263/output-layer-neural-networks*](https://www.techopedia.com/definition/33263/output-layer-neural-networks)

*[3]* [*https://www.geeksforgeeks.org/introduction-convolution-neural-network/*](https://www.geeksforgeeks.org/introduction-convolution-neural-network/)

*[4]* [*https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/*](https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/)

*[5]* [*https://www.simplilearn.com/tutorials/deep-learning-tutorial/convolutional-neuralnetwork#:~:text=flattening%20is%20used%20to%20convert,layer%20to%20classify%20the%20image*](https://www.simplilearn.com/tutorials/deep-learning-tutorial/convolutional-neuralnetwork#:~:text=Flattening%20is%20used%20to%20convert,layer%20to%20classify%20the%20image)*.*

*[6]* [*https://towardsdatascience.com/the-most-intuitive-and-easiest-guide-for-convolutional-neural-network-3607be47480*](https://towardsdatascience.com/the-most-intuitive-and-easiest-guide-for-convolutional-neural-network-3607be47480)

*[7]* [*https://nickmccullum.com/python-deep-learning/flattening-full-connection/*](https://nickmccullum.com/python-deep-learning/flattening-full-connection/)

*[8]* [*https://discuss.boardinfinity.com/t/explain-the-role-of-the-flattening-layer-in-cnn/8546*](https://discuss.boardinfinity.com/t/explain-the-role-of-the-flattening-layer-in-cnn/8546)

*[9]* [*https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939*](https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939)

*[10] towards age-invariant face recognition Jian zhao , member, ieee, shuicheng yan, fellow, ieee, and jiashi feng , member, ieee*

*[11]* [*https://analyticsindiamag.com/comprehensive-guide-to-different-pooling-layers-in-deep-learning/*](https://analyticsindiamag.com/comprehensive-guide-to-different-pooling-layers-in-deep-learning/)

*[12] z. Li, d. Gong, q. Li, d. Tao, and x. Li, “mutual component analysis for heterogeneous face recognition,” acm trans. Intell. Syst. Technol., vol. 7, no. 3, pp. 1–23, 2016.*

*[13] z. Li, d. Gong, y. Qiao, and d. Tao, “common feature discriminant analysis for matching infrared face images to optical face images,” ieee trans. Image process., vol. 23, no. 6, pp. 2436–2445, jun. 2014.*

*[14]* [*https://keras.io/api/layers/pooling\_layers/*](https://keras.io/api/layers/pooling_layers/)

*[15]*[*https://www.researchgate.net/publication/318415028\_age\_invariant\_face\_recognition\_using\_convolutional\_neural\_networks\_and\_set\_distances*](https://www.researchgate.net/publication/318415028_Age_Invariant_Face_Recognition_Using_Convolutional_Neural_Networks_and_Set_Distances)

*[16]* [*https://ai.plainenglish.io/pooling-layer-beginner-to-intermediate-fa0dbdce80eb*](https://ai.plainenglish.io/pooling-layer-beginner-to-intermediate-fa0dbdce80eb)

*[17] w. Liu, y. Wen, z. Yu, m. Li, b. Raj, and l. Song, “sphereface: deep hypersphere embedding for face recognition,” in proc. Ieee conf. Comput. Vis. Pattern recognit., 2017, pp. 212–220.*

*[18] a. Montillo and h. Ling, “age regression from faces using random forests,” in proc. Ieee int. Conf. Image process., nov. 2009, pp. 2465–2468.*

*[19] d. Chen, x. Cao, f. Wen, and j. Sun, “blessing of dimensionality: high-dimensional feature and its efficient compression for face verification,” in proc. Ieee conf. Comput. Vis. Pattern recog., jun. 2013, pp. 3025–3032.*