CHAPTER I

INTRODUCTION AND DESIGN OF THE STUDY

1.1 INTRODUCTION

Image classification can be defined as the task of categorizing images into one or multiple predefined classes. Although the task of categorizing an image is instinctive and habitual to humans, it is much more challenging for an automated system to recognize and classify images.

It is the process of assigning spectral classes into information classes. Spectral classes are groups of pixels that are uniform with respect to their brightness values in the different spectral channels of data. Information classes are categories of interest. In short, image classification is a process of assigning all pixels in the image to a particular class based on spectral information represented by the digital numbers (Anupam Anand, 2017).

Deep leaning, a subset of machine learning is one of the best fitted methods of implementing image classification systems. It allows computational models to learn by gathering knowledge from experience. Complex concepts can be learnt due to its hierarchical conceptualization. It has an immense influence on fields such as computer vision, object detection, object recognition, speech recognition and text processing etc.

Nowadays, sharing images to social networks (be it personal, or everyday scenes, or opinions depicted in the form of cartoons or memes) to convey emotions has increased rapidly. Analysing content like this (also known as visual emotion analysis) from social media websites and/or photo-sharing websites like Flickr, Twitter, Tumblr, etc., can give insights into the general sentiment of people about an event that is happening (for ex: presidential elections) also, it would be useful to automatically predict emotional tags on them - like violence, happiness, fear etc. (Vasavi and Aditi).

Psychological studies have provided evidence that human emotions can be aroused by visual content. e.g., images (Lang, Bradley and Cuthbert 1998, Jost et al 2011). In recent years, after the breath-taking success in traditional computer vision tasks such as object detection and image classification, researchers have gradually turned their eyes from low-level vision tasks to high-level ones, including visual reasoning, image aesthetic assessment, visual emotion analysis etc.

Among all the high-level vision tasks, Visual Emotion Analysis (VEA) is one of the most challenging tasks. It is difficult to classify images into emotions because visual sentiments are evoked by different types of information – visual and semantic information, where visual information includes colours or textures, and semantic information includes types of objects evoking emotions and/or their combinations (Takahisa yamamoto et al., 2021). But with the rapid development of convolutional neural networks, more and more researchers employed deep learning networks to VEA. Instead of designing emotion features manually, CNNs were capable of mining emotional representations automatically in an end to-end manner and consequently achieved better results (Jingyuan yang et al., 2021).

If we search for a tag "love" on google, we get a wide variety of images: roses, a mother holding her baby, images with hearts, etc. these images are very different from one another and yet depict the same emotion of “love” in them. In this project, I explored the possibility of using deep learning to predict the emotion depicted by those images. (Vasavi and Aditi)

In 1970, Paul Ekman and Friesen defined six universal basic emotions: happiness, sadness, fear, disgust, surprise, and anger, such basic affective categories are sufficient to describe most of the emotions that has been exhibited.

However instead of ‘anger’ I chose ‘violence’ as it is more suited to the applications we are looking at - more and more people are turning to social media to show their support for causes or protest against something (for example, a lot of social messages poured in after terrorist attacks in Paris in November, 2015) for the same reason, I add ‘love’ as an emotion category. To limit to 5 emotions, I dropped ‘disgust’ and ‘surprise’. So, the final chosen emotion categories are: love, happiness, violence, fear and sadness.

After finalizing the categories, I collected data (images) for these categories from google. I experimented with various deep learning classification methods and finally tried transfer learning from a convnet with millions of parameters, which is pre-trained on large-scale data (ImageNet) for object recognition.

1.2 NEED OF THE STUDY

Vast amount of research is devoted to sentiment analysis of textual data, there has been very limited work that focuses on analysing sentiment of image data (Xu can et al., 2014). Algorithms to identify sentiment can be helpful to understand user behaviours and are widely applicable to many applications such as blog recommendation, behaviour targeting and viral marketing.

Visual Emotion Analysis (VEA) has attracted great attention recently, due to the increasing tendency of expressing and understanding emotions through images on social networks (Jingyuan yang et al., 2021). Different from traditional vision tasks, VEA is inherently more challenging since it involves a much higher level of complexity and ambiguity in human cognitive process.

To perform visual emotion analysis from the images, the initial step is to classify the images based on the emotion category. This project aims to do accomplish that initial step.

1.3 OBJECTIVES OF THE STUDY

1. To manually curate a dataset that consists of diverse set of images that depicts the same emotion for a particular category.
2. To construct a deep learning classifier that classifies the images based on the emotion depicted.
3. To feed the images for neural network i.e., simple feed forward neural network, and convolution neural network.
4. To conduct experiments with different variants of the same network by fine tuning
5. To compare the performance of the model with different variants and to select a best model from the available ones.

1.4 SCOPE OF THE STUDY

Emotion classification in images has applications in automatic tagging of images with emotional categories, automatically categorizing video sequences into genres like thriller, comedy, romance, etc (Vasavi and Aditi).

To understand the general sentiment of people about an event that is happening in the society. Understanding sentiments or emotions from images has different applications including image captioning opinion mining, advertisement and recommendation (Takahisa yamamoto et al., 2021).

Inferring emotion tags from the images from social media has great significance as it can play a vital role in recommendation systems, image retrieval, human behaviour analysis and, advertisement applications (Anam Manzoor et al., 2020).

1.5 CHALLENGES OF THE STUDY

The problem of labelling images with the emotion they depict is very subjective and can differ from person to person. Also, due to cultural or geographical differences some images might invoke a different emotion in different people - like in India people light candles to celebrate a festival called “Diwali”, however in western countries candles are lit, most of the times, to mark an occasion of mourning. In addition to the above, there is another fundamental challenge involved in tackling the problem of extracting emotional category from images. A person could have tagged an image as, say “fear”, it could be because the image makes them feel that emotion when they look at it or the person in the image may experience fear. This kind of image tagging can confuse the network (Vasavi and Aditi).

1.6 CHAPTER SCHEME

This project report is organized into six chapters as follows:

Chapter I _ Introduction

The chapter deals with the introduction, need, objectives, scope, challenges of the study and the chapter scheme.

Chapter II – Review of Literature

The chapter deals the most relevant research and theories related to ‘Emotion Detection from Images Using Deep Learning’.

Chapter III – System Specification

The environment that has been used for developing the deep learning models has been summarized.

Chapter IV – Methodology

This chapter presents the process flow, concepts and techniques that are used in this project.

Chapter V – Dataset

In this chapter, details regarding the data collection, other available datasets are summarised.

Chapter VI – Experiments and Results

In this chapter, details regarding the various experiments conducted, with the dataset has been presented and summarized.

CHAPTER II

REVIEW OF LITERATURE

2.1 INTRODUCTION

In this chapter, the most relevant research and theories related to 'Emotion Detection from Images Using Deep Learning' were discussed.

2.2 RELATED RESEARCH AND THEORIES

Anam Manzoor et al., (2020) "Inferring Emotion Tags from Object Images Using Convolutional Neural Network" introduced a novel idea of inferring emotion tags from the images based on object-related features. They mentioned that emotions are a fundamental part of human behavior and can be stimulated in numerous ways. Different types of objects that we see in daily life such as cake, crab, television, trees, etc., may excite certain emotions. Likewise, object images that we see and share on different platforms are also capable of expressing or inducing human emotions. They have created an emotion-tagged dataset from the publicly available object detection dataset (i.e., "Caltech-256") using subject evaluation from 212 users. They have used CNN to automatically extract the high-level features from object images for recognizing nine emotion categories, such as amusement, awe, anger, boredom, contentment, disgust, excitement, fear, and sadness. Overall, the proposed scheme achieved an accuracy rate of approximately 85% and 79%.

Jingyuan yang, Jie Li, Xiumei Wang, Yuxuan Ding and Xinbo Gao (2021) "Stimuli-aware visual emotion analysis." stated that Visual emotion analysis (VEA) has attracted great attention recently and is more challenging as it involves a much higher level of complexity and ambiguity in human cognitive process. Most of the existing methods adopt deep learning techniques to extract general features from the whole image, disregarding the specific features evoked by various emotional stimuli. Inspired by the Stimuli-Organism Response (S-O-R) emotion model in psychological theory, they proposed a stimuli-aware VEA method consisting of three stages, namely stimuli selection (S), feature extraction (O) and emotion prediction (R). First, specific emotional stimuli (i.e., color, object, face) are selected from images by employing the off-the-shelf tools. Then, they have designed three specific networks, i.e., Global-Net, Semantic-Net and Expression-Net, to extract distinct emotional features from different stimuli.

Jufeng yang, Dongyu She, Yu-Kun Lai, Paul L. Rosin, Ming-Hsuan Yang "Weakly supervised coupled networks for visual sentiment analysis." tried solving the problem of visual sentiment analysis using the high-level abstraction in the recognition process. They presented a weakly supervised coupled convolutional network with two branches to leverage the localized information. The first branch detects a sentiment specific soft map by training a fully convolutional network with the cross spatial pooling strategy, which only requires image-level labels, thereby significantly reducing the annotation burden. The second branch utilizes both the holistic and localized information by coupling the sentiment map with deep features for robust classification. They integrate the sentiment detection and classification branches into a unified deep framework and optimize the network in an end-to-end manner. They have conducted experiments on six benchmark datasets and stated that the proposed method performs favorably against the state-of the-art methods for visual sentiment analysis.

Lailatul qadri binti zakariaa, Paul Lewisb , Wendy Hallc (2017) "Automatic image tagging by using image content analysis." presented a simple image classifier which attempts to classify building images based on their low level features which are color and edges. The classifier is developed by using Bayesian Inference method provided by Infer.net tool. Equivalently, the classifier is said to be an annotator which aims to annotate images with a specific tag when appropriate. In this example, the image classifier is developed to identify and distinguish building and non-images. The method could be extended into other classes such as mountains, beaches and forest.

Minyoung huh, Pulkit Agrawal, Alexei A. Efros (2016) "What makes imagenet good for transfer learning?" did an empirical investigation into how ImageNet dataset makes the learnt features as good as they are. They concentrated on number of examples, number of classes, balance between images-per-class and classes, and the role of fine- and coarse-grained recognition. They pre-trained CNN features on various subsets of the ImageNet dataset and evaluated transfer performance on a variety of standard vision tasks. They also stated that most changes in the choice of pre-training data long thought to be critical, do not significantly affect transfer performance.

Ms. Sonali b Gaikwad, prof. S. R. Durugkar (2017) "Image sentiment analysis using different methods: a recent survey" collected a set of images from Flickr and Instagram, and then prepared their sentiment labels. They have proposed a novel image sentiment analysis method that uses the latent correlations among multiple views of training images. In the proposed method, they first extracted features from visual, textual, and sentiment views. Then, to project the features from these views, they followed the framework of multi-view CCA using explicit feature mappings. In the embedding space, a sentiment polarity classifier is trained based on the projected features.

Quanzeng you, Jiebo luo, Hailin jin, Jianchao yang (2016) "Building a large scales dataset for image emotion recognition: the fine print and the benchmark." stated that while the recent successes of many computer vision related tasks are due to the adoption of convolutional neural network (CNNs), visual emotion analysis has not achieved the same level of success, This may be primarily due to the unavailability of confidently labeled and relatively large image data sets for visual emotion analysis. And introduced a new data set called FI (Flickr Instagram) which has 3+ million weakly labeled images of eight different emotion classes namely amusement, awe, anger, contentment, disgust, excitement, fear and sad thereby ended up 30 times as large as the current largest publicly available visual emotion data set. They stated that this data set encourages further research on visual emotion analysis.

Rameswar panda, Jianming Zhang, Haoxiang Li, Joon-Young Lee, Xin Lu, and Amit K. Roy-Chowdhury "Contemplating visual emotions: understanding and overcoming dataset bias" presented a detailed information on (a) 3-level emotion hierarchy that they have used for creation of the WEBEmo dataset (b) information on different training and test datasets including motivation and the labeling protocol, (c) additional experiments including duplicate analysis, correlation analysis on all emotion categories, implementation details of the compared methods. They followed the Parrott's emotion wheel and constructed a three-level emotion hierarchy to retrieve stock images for the WEBEmo dataset with emotion categories (anger, fear, joy, love, sadness and surprise). They collected for about 150K images from both Google and Flickr using the secondary level emotion categories. They have also mentioned the keywords they used to search and download the images, e.g., "disgust", "envy", "exasperation", "irritability" and "rage" are used as search keywords to collect images of "anger" category. They used five human subjects to label the images within each emotion category and remove the rest non-emotional images from the collection. In total, they acquire a total of 8350 images (anger: 1604, fear: 1280, joy: 1964, love: 724, sadness: 2221, surprise: 557).

Shaojing fan, Zhiqi Shen, Ming Jiang, Bryan L. Koenig, Juan Xu, Mohan S. Kankanhalli, and Qi Zhao "Emotional attention: a study of image sentiment and visual attention." focused on the relation between emotional properties of an image and visual attention. They have created an EMOtional attention dataset (EMOd). It is a diverse set of emotion eliciting images, and each image has (1) eye-tracking data collected from 16 subjects, (2) intensive image context labels including object contour, object sentiment, object semantic category, and high-level perceptual attributes such as image aesthetics and elicited emotions. They have discovered an emotion prioritization effect: for the images, emotion-eliciting content attracts human attention strongly, but such advantage diminishes dramatically after initial fixation. Aiming to model the human emotion prioritization computationally, they have designed a deep neural network for saliency prediction, which includes a novel subnetwork (CASNet—Context-Adaptive Saliency Network) that learns the spatial and semantic context of the image scene. The proposed network outperforms the state-of-the art on three benchmark datasets, by effectively capturing the relative importance of human attention within an image.

Takahisa yamamoto, Shiki takeuchi, and atsushi nakazawa (2021) "Image emotion recognition using visual and semantic features reflecting emotional and similar objects." stated that One of the difficulties in classifying images into emotions is that visual sentiments are evoked by different types of information – visual and semantic information where visual information includes colors or textures, and semantic information includes types of objects evoking emotions and/or their combinations. Existing methods use only visual information, but they have introduced a novel algorithm for both visual and semantic information simultaneously. For semantic features, they introduced an object vector and a word vector. The object vector is created by an object detection method and reflects existing objects in an image. The word vector is created by transforming the names of detected objects through a word embedding model. This vector will be similar among objects that are semantically similar. These semantic features and a visual feature made by a fine-tuned convolutional neural network (CNN) are concatenated. They have performed the classification by the concatenated feature vector. Extensive evaluation experiments using emotional image datasets showed that the proposed method achieves the best accuracy. The improvement in accuracy of the method from existing methods is 4.54% at the highest.

Vasavi Gajarla, Aditi Gupta "Emotion detection and sentiment analysis of images." Explored the possibility of using deep learning to predict the emotion depicted by an image. The results look promising and indicate that neural nets are indeed capable of learning the emotion essayed by an image. We then collected data from Flickr for five categories namely love, happiness, fear, sadness and violence. They have experimented various classification methods on the data - SVM on high level features of VGG-ImageNet, fine-tuning on pretrained models like RESNET, Places205-VGG16 and VGG ImageNet. In all of the fine-tuning experiments, it has been observed that standard data augmentation techniques like mirroring and random cropping of images increase their accuracy. Having a higher learning rate for the later layers also yields a better accuracy. They used Stochastic Gradient Descent optimization for all the networks, and the base learning rate at a start value of 0.0001 gave better accuracies.

Xu can, et al (2014) "Visual sentiment prediction with deep convolutional neural networks." stated that although vast amount of research is devoted to sentiment analysis of textual data, there has been very limited work that focuses on analyzing sentiment of image data. They have proposed a novel visual sentiment prediction framework that performs image understanding transfer learning from a CNN with millions of parameters which is pre-trained on large-scale data for object recognition. Experiments conducted on two real-world datasets from Twitter and Tumblr demonstrate the effectiveness of the proposed framework. In all the experiments, they have used Area Under the receiver operating characteristic curve (AUC) as the metric for performance evaluation as it is less sensitive to imbalanced datasets and therefore it is very appropriate to the problem. They stated that Twitter dataset is a much noisier dataset that the Tumblr dataset. So, comparatively the Tumblr dataset showed a better result than the Twitter dataset.

You, quanzeng, et al (2015). "Robust image sentiment analysis using progressively trained and domain transferred deep networks." stated that researches have largely relied on textual sentiment analysis to develop systems to predict political elections, measure economic indicators and so on. Recently, social medical media users are increasingly using images and videos to express their opinion and share their experience. Sentiment analysis of such large-scale visual content can help better extract user sentiments toward events or topics, such as those in image tweets, so that prediction of sentiment from visual content is complementary to textual sentiment analysis. They have obtained half a million training samples by using a baseline sentiment algorithm to label Flickr images. To make use of such noisy machine labeled data, they employed a progressive strategy to fine-tune the deep network. The results show that the proposed CNN can achieve better performance in image sentiment analysis than competing algorithms

Yue zhang, Wanying Ding, Ran Xu and Xiaohua "Visual emotion representation learning via emotion-aware pre-training." stated that visual emotion recognition remains a challenging problem due to the ambiguity of emotion perception, diverse concepts related to visual emotion, and lack of large-scale annotated datasets. They have presented a large-scale multimodal pre-training method to learn visual emotion representation by aligning emotion, object, and attribute triplet with a contrastive loss. They conduct the pre-training on a large web dataset (Stock Emotion) with noisy tags and fine-tuned on smaller visual emotion classification datasets with class label supervision. They have utilized the scene graph trained on large-scale open domain datasets to extract a diverse set of objects and attributes from an image, together with noisy tags from the image, they used a transformer-based fusion model for the pre-training task. The method achieves state-of-the-art performance for visual emotion classification.

Zijun wei, Jianming Zhang, Zhe Lin, Joon-Young Lee, Niranjan Balasubramanian, Minh Hoai (2020) "Learning visual emotion representations from web data." They have curated a webly derived large-scale dataset, StockEmotion, which has more than a million images. StockEmotion uses 690 emotion related tags as labels giving us a fine-grained and diverse set of emotion labels, circumventing the difficulty in manually obtaining emotion annotations. They used this dataset to train a feature extraction network, EmotionNet, which they further regularized using joint text and visual embedding and text distillation. Our experimental results establish that EmotionNet trained on the StockEmotion dataset outperforms on four different visual emotion tasks.

CHAPTER III

SYSTEM SPECIFICATION

3.1 INTRODUCTION

In this chapter, the environment that I used for developing the deep learning models has been summarized.

3.2 HARDWARE SPECIFICATION

S. N	CATEGORY	SPECIFICATION
1	Hardware	Google Colab’s free GPU instance.
2	GPU Model	NVIDIA K80
3	CPU	2 x vCPU
4	RAM	12 GB
5	Memory Clock	0.82 GHz
6	Guaranteed Resources	No
7	Max execution time	12 hours
8	Max idle time	90 min
9	Release Year	2014
10	Disk Space	358 GB

Google Colaboratory, or Google “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited for deep learning. It requires no setup to use, while providing access free of charge to computing resources including GPUs. (research.google.com, n.d.)

Graphics processing unit, a specialized processor originally designed to accelerate graphics rendering. Designed for parallel processing, GPUs are becoming more popular for use in artificial intelligence (AI). Because GPUs incorporate an extraordinary amount of computational capability, they can deliver incredible acceleration in workloads. Many of today’s deep learning technologies rely on GPUs for processing. (www.intel.in, n.d.)

3.3 SOFTWARE SPECIFICATION

PROGRAMMING LANGUAGE		
S. N	**CATEGORY**	**VERSION**
1	Python	3.8.16

SOFTWARE LIBRARIES		
S. N	**CATEGORY**	**VERSION**
1	Tensorflow	2.9.2
2	Keras	2.9.0
3	numpy.	1.21.6
4	Matplotlib	3.2.2
5	OS Module	Python standard version

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. (www.python.org.doc, n.d.)

Tensorflow – Software library for Artificial Intelligence

Keras – Deep Learning API written in python running on top of Tensorflow2. This is also a cross-platform software can run on Tensorflow, Theano, CNTK, Pytorch

Numpy – Library used for working with arrays.

Matplotlib – cross platform data visualization and graphical plotting library for python

OS Module - The module contains several useful functions that help us to access, modify, and perform OS-related tasks such as access and modifying directories.

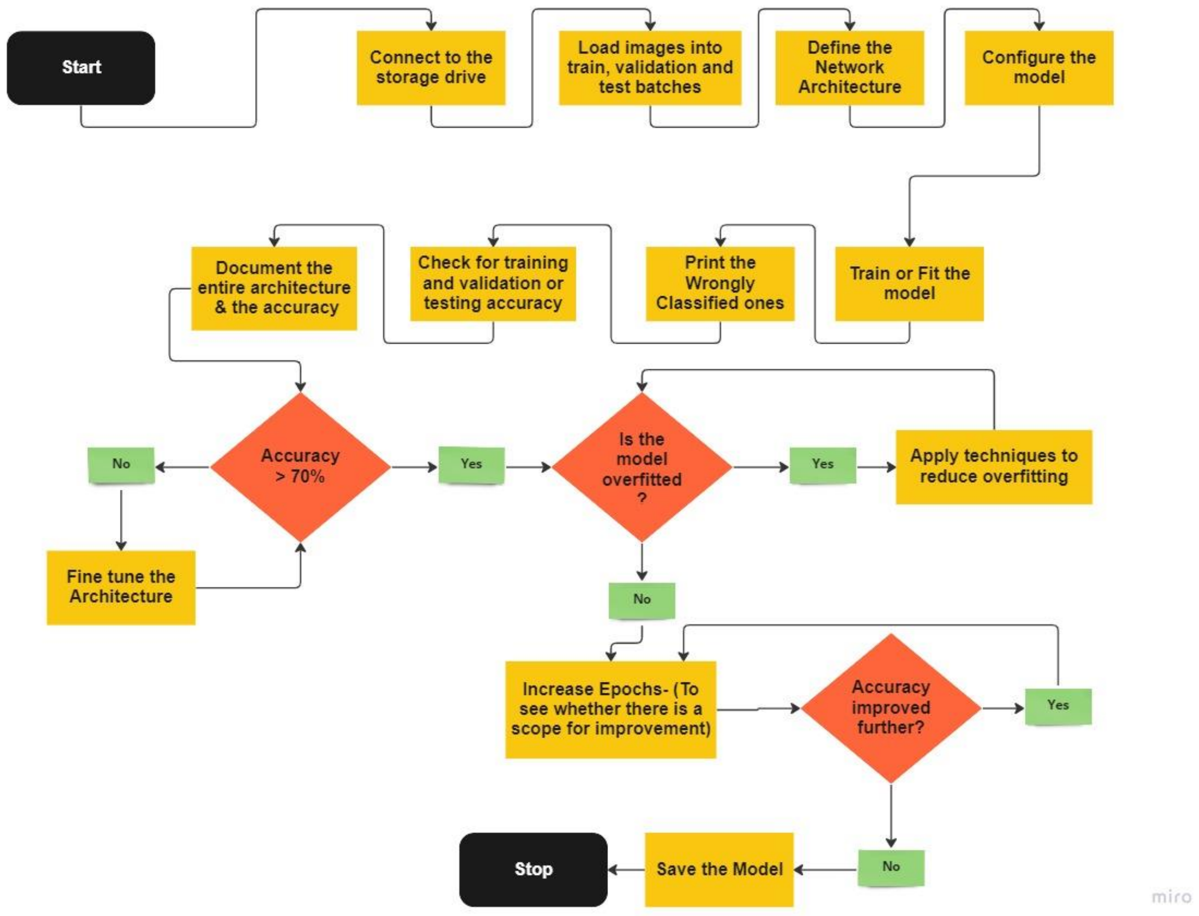
CHAPTER IV

METHODOLOGY

4.1 INTRODUCTION

In this chapter, the process flow, concepts and techniques that are used in this project are summarized.

4.2 PROCESS FLOW OF THE PROJECT



Start
Connect to the storage drive
Load images into train, validation and test batches
Define the Network Architecture
Configure the model
Train or Fit the model
Print the Wrongly Classified ones
Check for training and validation or testing accuracy
Document the entire architecture & the accuracy
Accuracy > 70%
No
Fine tune the Architecture
Yes
Is the model overfitted ?
Yes
Apply techniques to reduce overfitting
No
Increase Epochs- (To see whether there is a scope for improvement)
Accuracy improved further?
Yes
No
Save the Model
Stop
miro

Start the Process. Connect to the storage driver, where the data is stored. Load the images into Train, Validation and Test batches respectively. Define the network architecture i.e., add desired layers. Configure the model with defining the metrics for classification, optimizer and the loss function. Train or fit the model to the architecture.

Print the wrongly classified images to understand, for which category of images the model is more confused. Check for the training and validation accuracy. Document the entire architecture defined, optimizer and loss functions used, no. of epochs trained with the received accuracy. If accuracy is lesser than 70% fine tune the architecture of the model i.e., Capacity of the network (No. of Dense layers) can be reduced or increased, the optimizer used can be changed, the learning rate in the optimizer can be adjusted, no. of epochs can be increased, data can be normalized, convolutional layers can be added and so on.

Then, check if the model is underfitted or overfitted. If underfitted increasing the number of epochs is the solution. If overfitted, there are various methods to reduce overfitting such as weight regularization, dropout layer, image augmentation, reducing the learning rate, using the weights of the pretrained network and fine tuning the blocks of the pretrained network.

If the model is not overfitted or underfitted, then try increasing epochs to check whether there is a further scope for improvement. If there is an improvement in the accuracy, increase epochs or if the accuracy is stagnant, save the model and stop the process.

4.3 IMAGE PRE-PROCESSING

When it comes to training images in deep learning, we have to consider these two things:

1. Datatype:

Data should be of homogenous type i.e., dtype of train_images[0], train_images[1], train_images[2] should be same. Either all are “int32” or “float64” or “float32”.

Preferable data type for neural network is float32

Code snippet:

```
#Defining the data Generator
datagen = ImageDataGenerator(rescale=1./255,
                             featurewise_std_normalization=True,
                             samplewise_std_normalization=True,
                             validation_split=0.2,
                             dtype = "float32")
```

2. Range:

Pixel value of grayscale or rgb images ranges from 0 to 255. Working with this huge range of values may deteriorate the learning. To enhance learning, the range can be changed from 0 to 255 to 0 to 1.

Code snipet:

```
model = tf.keras.models.Sequential([
    layers.Rescaling(1./255, input_shape=(IMG_SHAPE, IMG_SHAPE, 3)),
    layers.Flatten(),
    layers.Dense(512, activation='relu', kernel_regularizer = 'l2'),
    layers.Dropout(0.5),
    layers.Dense(5, activation = 'softmax')
])
```

4.4 NETWORKS USED

- Deep Neural Network
- Convolutional Neural Network

Deep Neural Network

Perceptron is arranged in layers, with the first layer taking in inputs and the last layer producing outputs. The middle layers have no connection with the external world, and hence are called hidden layers.

Each perceptron in one layer is connected to every perceptron on the next layer. Hence information is constantly "fed forward" from one layer to the next., and this explains why these networks are called feed-forward networks. This progression of computations through the network is called forward propagation. There is no connection among perceptron in the same layer. (stanford.edu, n.d.)

Another process called backpropagation uses algorithms, like gradient descent, to calculate errors in predictions and then adjusts the weights and biases of the function by moving backwards through the layers in an effort to train the model. Together, forward propagation and backpropagation allow a neural network to make predictions and correct for any errors accordingly.

In this case, only the Dense layers are used. If there is only one dense layer, it is known as shallow learning and if there is two or more Dense layers, it is defined as Deep neural networks or Deep feed forward neural networks (www.ibm.com, n.d.)

Code snippet:

```
from tensorflow.keras import layers
from tensorflow.keras.layers import Dense
model = tf.keras.models.Sequential([
layers.Flatten(),
layers.Dense(512, activation='relu'),
layers.Dense(218, activation='relu'),
layers.Dense(126, activation='relu'),
layers.Dense(64, activation='relu'),
layers.Dense(32, activation='relu'),
layers.Dense(12, activation='relu'),
layers.Dense(5, activation = 'softmax')
])
```

Convolutional Neural Network:

Convolutional neural network (Convnet for short is a class of artificial neural network that uses convolution. It uses mathematical operation called convolution in place of general matrix multiplication in at least one of their layers. In mathematics in particular functional analysis convolution is a mathematical operation on two functions (f and g) that produces third function that expresses how the shape of one is modified by the other. The term convolution refers to both the result function and to the process of computing it.

CNN is developed exclusively for the computer vision and image processing A breakthrough in building models for images classification came with the discovery that a convolutional neural network (CNN) could be used to progressively extract higher-level representation of the image content.

It is based on shared weight architecture. In densely connected network we get only one pattern for an image in hand, whereas CNN extracts various local patterns from the data. CNN is also known as shift invariant or space invariant artificial neural network.

Two important properties of convnet:

1. The patterns they learn are translation invariant
2. They can learn spatial hierarchy way

Types of operations or layers:

A. Convolution Stage

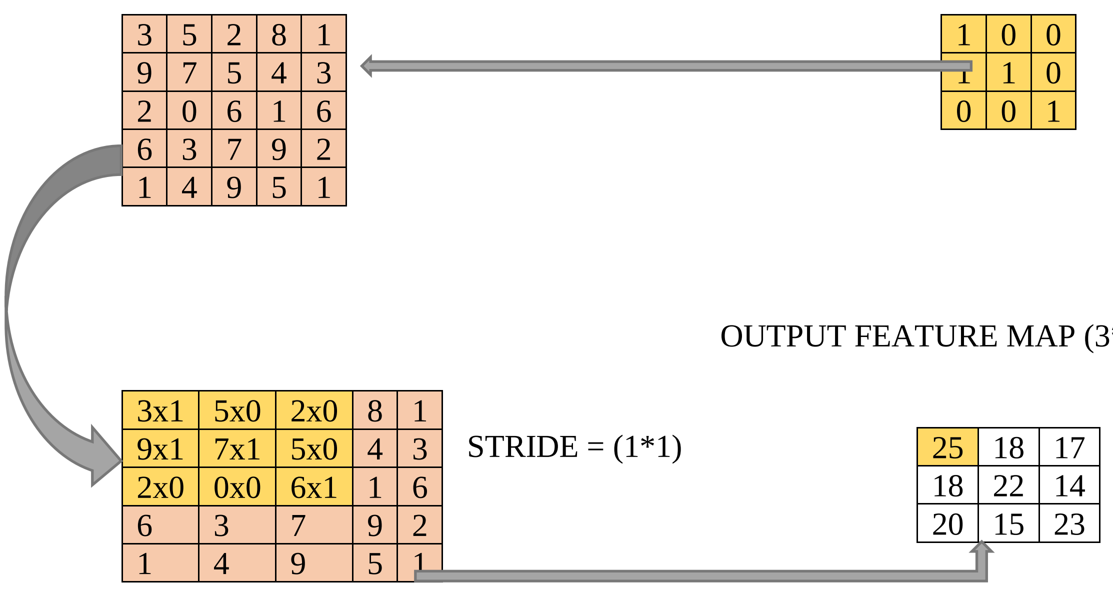
A convolution extracts tiles of the input feature maps and applies filters to them to compute new features, producing an output feature map or convolved feature (which may have a different size and depth than the input feature map).

Convolution are defined by two parameters:

1. Size of the tiles that are extracted typically 3X3 or 5X5
2. The depth of the output feature map which corresponds to the number of filters that are applied.

During convolution layer, the filters effectively slide over the input feature map’s grid horizontally and vertically one pixel at a time extracting each corresponding tile.

For Example:



INPUT FEATURE MAP (5*5)

3	5	2	8	1
9	7	5	4	3
2	0	6	1	6
6	3	7	9	2
1	4	9	5	1

KERNEL/WINDOW/FILTER (3*3)

1	0	0
1	1	0
0	0	1

3x1	5x0	2x0	8	1
9x1	7x1	5x0	4	3
2x0	0x0	6x1	1	6
6	3	7	9	2
1	4	9	5	1

STRIDE = (1*1)

OUTPUT FEATURE MAP (3*3)

25	18	17
18	22	14
20	15	23

= 3 + 0 + 0 + 9 + 7 + 0 + 0 + 0 + 6 = 25

From this it is clear that the output value in the feature map does not have to connect to each pixel value in the input image. It only needs to connect to the receptive field where the filter is being applied.

By default, our kernels are only applied where the filter fully fits on top of the input. But I can also control this behavior and the size of our output with Padding.

Padding – Pads the outside of the input 0's to allow the kernel to reach the boundary pixels. padding = 1 means I have one layer of 0's around our border

Strides – Controls how far the kernels "steps" over pixels. For a convolution or pooling operation, the stride S denotes the number of pixels by which the window moves after each operation. Strides = (2,2) means our kernel moves 2 data points to the right for each row then moves 2 points down to the next row.

Filters:

Using a filter smaller than the input is intentional as we allow the same filter (set of Weights) to be multiplied by the input array multiple times at different points on the input. If the filter is designed to detect a specific type of feature in the input, then the application of that filter systematically across the entire input image allows the filter an opportunity to discover that feature anywhere in the image. This capability is commonly referred to as translation invariance.

The Weights in the feature detector remains fixed as it moves across the image, which is also known as parameter sharing (sharing same filter or same kernel across the image). Going through convolutional layers, process features a large amount of data, which makes it hard to train the neural network, to compress the data, I need to go through pooling.

B. Pooling Stage

Helps to reduce complexity, improve efficiency and the risk of overfitting. Receives the result form a convolution layer and compress it. Filter of pooling layer is smaller than the feature map. Down sampling operation typically applied after a convolution layer which does some spatial invariance.

As a result, I get pooled feature maps that are a summarized version of the features detected in the input. It reduces the number of dimensions of the feature map, while still preserving the most critical feature information.

Two types of pooling:

1. Max pooling

Aggregating using maximum of the window

2. Average pooling

Aggregating using average of the window.

Pooling operation takes two paraments:

1. **size** – the pooling filter (typically 2x2)
2. **stride** – The distance in pixels separating each extracted tile.

Fully connected layer:

The flattened output is fed to a feed – forward neural network and back propagation is applied at every iteration of training.

Job – perform classification based on the feature extracted by the convolutions or previous layers.

General overview:

1. A convolution layer is responsible for recognizing features in pixels.
2. A pooling layer is responsible for making these features more abstract.
3. A fully connected layer is responsible for using the acquired features for prediction

4.5 TECHNIQUES USED TO OVERCOME OVERFITTING

- Optimizing the capacity of the network
- Weight Regularization
- Dropout
- Optimizing the learning rates
- Adding convolutional layers
- Image Augmentation
- Transfer Learning
- Fine tuning the pretrained network

Optimizing the capacity of the network:

1. Units in the Dense layer can be decreased or increased
2. Number of Dense layers can be added or removed
3. Number of convolution layers can be added or removed

Optimizing the learning rates:

The optimizer that has been mentioned during model configuration has learning rate as a parameter. Decreasing learning rate may enhance learning and slow up the convergence, increasing leaning rate may speed up convergence and can lead to missing the optimum value.

Code snippet:

```
tf.keras.optimizers.RMSprop(learning_rate=0.001)
```

Learning rate in keras is a floating-point value, Defaults to 0.001.

Adding convolutional layers:

For image processing the convolutional neural network works well when compared to deep neural network. So, adding convolution layer to the network architecture can improve the performance of the model.

Code Snippet:

```
#Network architecture
model = tf.keras.models.Sequential([
    layers.Conv2D(32, (3,3), padding='same', activation = 'relu'),
    layers.MaxPooling2D(2,2),
    layers.Conv2D(64, (3,3),padding='same', activation = 'relu'),
    layers.MaxPooling2D(2,2),
    layers.Flatten(),
    layers.Dense(512, activation='relu'),
    layers.Dense(5, activation = 'softmax')
])
```

Image Augmentation

Overfitting also happens when we have less data. So, increasing the number of data points can reduce overfitting. But, when we have very less chance of collecting new data, we can augment the existing images for generating additional training samples. For ex: Taking image1 and flipping it, rotating it, zooming it, increasing brightness, shearing it and generating 5 different images from the existing image.

These augmentations can be done with keras preprocessing layers:

1. `tf.keras.layers.RandomFlip`,
2. `tf.keras.layers.RandomRotation`, and
3. `tf.keras.layers.RandomZoom`

Code Snippet:

```
data_augmentation = keras.Sequential(
  [
    layers.RandomFlip("horizontal",
                      input_shape=(IMG_SHAPE,
                                  IMG_SHAPE,
                                  3)),
    layers.RandomRotation(0.2),
    layers.RandomZoom(0.2),
  ]
)
```

And these augmentations can be added to the network architecture

```
model = tf.keras.models.Sequential([
    data_augmentation,
    layers.Conv2D(32, (3,3), padding='same', activation = 'relu'),
    layers.MaxPooling2D(2,2),
    layers.Flatten(),
    layers.Dense(512, activation='relu', kernel_regularizer = 'l2'),
    layers.Dense(5, activation = 'softmax')
])
```

Transfer Learning

Extracting important features of the data and feeding it to the model improves performance of the model. Feature extraction can be done with the help of a pretrained network. Transfer Learning is the concept of using weights of the pretrained model to our existing model. A pre-trained model is a saved network that was previously trained on a large dataset, (for example – ImageNet) for classification.

The intuition behind transfer learning for image classification is that if a model is trained on a large and general enough dataset, this model will effectively serve as a generic model of the visual world. We can then take advantage of these learned feature maps without having to start from scratch by training a large model on a large dataset.

Every Pretrained network has two main things:

1. Convolution Base
2. Classifier

Convolution base contains blocks of convolution and pooling layers and the Classifier defines the classifying criteria.

Feature Extraction:

Use the representations learned by a previous network to extract meaningful features from new samples. If we use our own classifier and use the convolution base of the pretrained network, we are actually using the representations learned by the previous network to extract meaningful features from new samples. You do not need to (re)train the entire model. The base convolutional network already contains features that are generically useful for classifying pictures.

Code snippet:

```
# Defining the VGG16
conv_base = tf.keras.applications.Xception(weights = 'imagenet',
                include_top = False,
                input_shape = (256, 256, 3))
#Freezing the convolution base
conv_base.trainable = False
#Network architecture
model = tf.keras.models.Sequential([
    layers.Rescaling(1./255, input_shape=(IMG_SHAPE, IMG_SHAPE, 3)),
    data_augmentation,
    conv_base,
    layers.Flatten(),
    layers.Dense(512, activation='relu', kernel_regularizer = 'l2'),
    layers.Dropout(0.5),
    layers.Dense(5, activation = 'softmax')
])
```

Fine tuning the pretrained network:

Unfreeze a few of the top layers of a frozen model base and jointly train both the newly-added classifier layers and the last layers of the base model. This allows us to "fine-tune" the higher-order feature representations in the base model in order to make them more relevant for the specific task.

Code snippet:

```
# Defining the VGG16
conv_base = tf.keras.applications.Xception(weights = 'imagenet',

                include_top = False,
                input_shape = (256, 256, 3))

# Enabling only the block 5
conv_base.trainable = True
set_trainable = False
for layer in conv_base.layers:
  if layer.name == 'block5_conv1':
    set_trainable = True
  if set_trainable:
    layer.trainable = True
  else:
    layer.trainable = False

#Network architecture
model = tf.keras.models.Sequential([
    data_augmentation,
    conv_base,
    layers.Flatten(),
    layers.Dense(512, activation='relu', kernel_regularizer = 'l2'),
    layers.Dense(5, activation = 'softmax')
])
```

4.6 PRE-TRAINED NETWORKS USED

1. VGG16

VGG stands for Visual Geometry Group; it is a standard deep Convolutional Neural Network (CNN) architecture with 16 convolutional layers. The VGG16 model achieves almost 92.7% top-5 test accuracy in ImageNet. ImageNet is a dataset consisting of more than 14 million images belonging to nearly 1000 classes. The network has a default image input size of 224 * 224

2. RESNET 50

ResNet stands for Residual Network. Trained on ImageNet. ResNet50 is used to denote the variant that can work with 50 neural network layers. Residual blocks make it considerably easier for the layers to learn identity functions. As a result, ResNet improves the efficiency of DNN with more neural layers while minimizing the percentage of errors. The network has a default image input size of 224 * 224

3. Xception

Xception is a convolutional neural network that is 71 layers deep. It involves separable convolutions. Trained on ImageNet. The network has a default image input size of 299 * 299.

CHAPTER V

DATASET

5.1 INTRODUCTION

In this chapter, details regarding the data collection, other available datasets are summarised.

5.2 DATA COLLECTION

I have collected 5000 images in total with 1000 images in every category for the selected 5 emotional categories - Love, Happiness, Violence, Fear, and Sadness from Google. I split the data in each category such that 73% of the data (3630) is used for training, and 27% of the data (270) is used for testing in DNN experiments and 80% for training and 20% for validation in CNN experiments.

Steps involved:

Step 1: Searching for the tags in google

Step 2: Downloading the images with the help of “Fatkun batch download” chrome extension

Search Keywords Used:

The thousand images in every class have been collected with the search term:

1. **“fear”** – scary, scary places, ghosts, haunted places, haunting scenarios, anxious, fright, panic, terrific.
2. **“happiness”** – laughter, happy employee, party, happy emojis, kids laughing, delighted, smiling.
3. **“love”** - Love, mom and kid, pet’s love, dad and kid, marriage, heart in, love emojis, affection.
4. **“sadness”** – sad, sorrow, depressed, stressed, upset, disappointment, heartbroken, pessimistic.
5. **“violence”** – violence, protest, right to freedom, crime, kill, fight, dispute.

Data cleaning:

Images that are irrelevant or not exactly depicts the emotion are ignored

The images that contain entirely texts or quotes are ignored

Images that are not in the .jpg format are ignored.

5.3 AVAILABLE DATASETS

S. N	Name	Total images	No. of Categories
1	Flickr & Instagram	23000	8
2	EmotionROI	1980	6
3	Art Photo	806	8
4	IAPSa	395	8
5	Stock Emotion	1.17 million	8
6	Abstract	228	8
7	Twitter I	1269	2
8	Twitter II	603	2
9	Flickr	60745	2
10	Instagram	42,856	2

The datasets that have eight categories mostly has Amusement, Awe, Anger, Contentment, Disgust, Excitement, Fear and Sad as categories

The datasets that have six categories mostly has Anger, Disgust, Fear, Joy, Sad, Surprise as categories

CHAPTER VI

EXPERIMENTS AND RESULTS

6.1 INTRODUCTION

In this chapter, details regarding the various experiments conducted with the dataset, has been presented and summarized.

6.2 EXPERIMENTS WITH DEEP NEURAL NETWORK

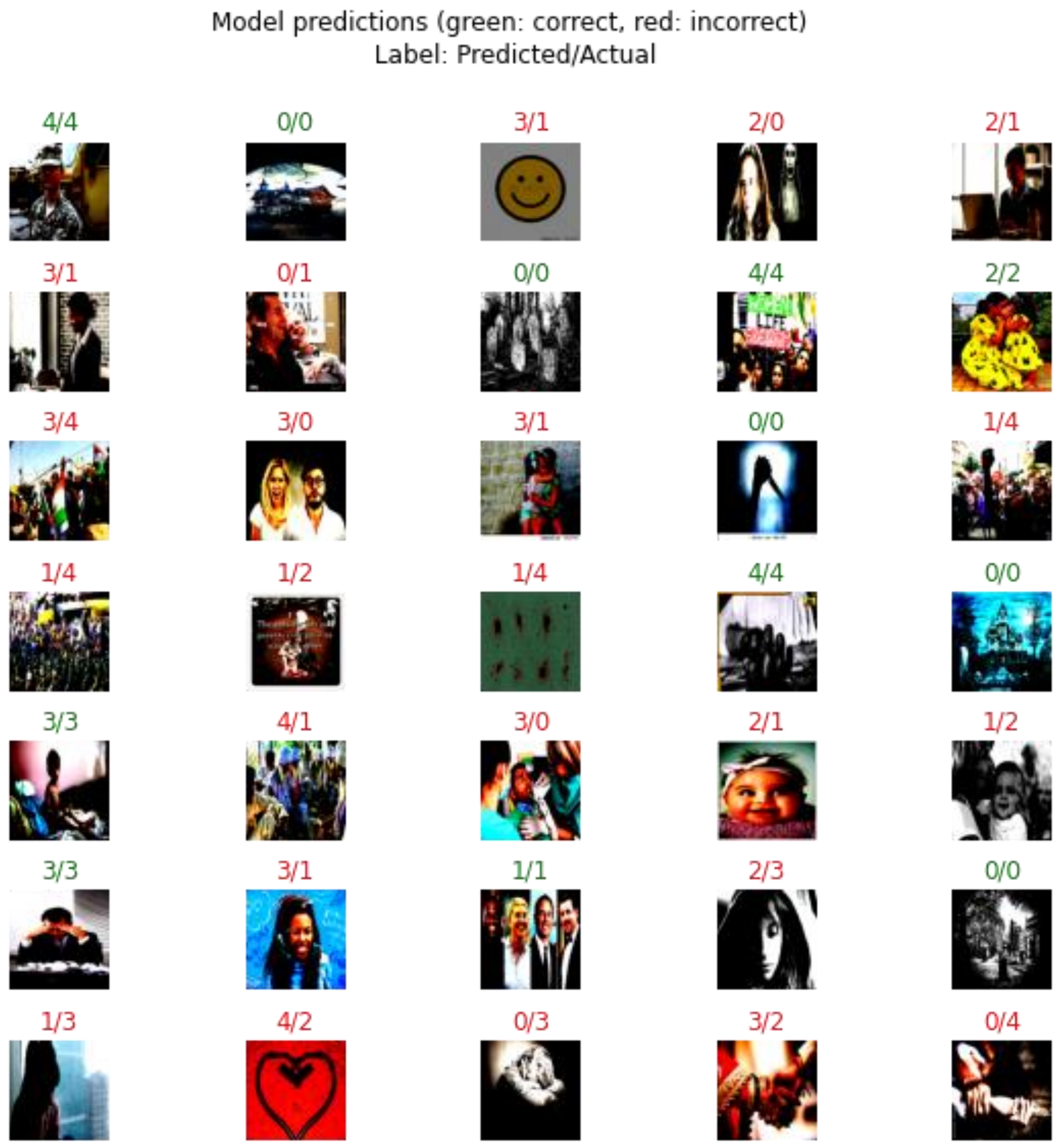
Universally, the model that has been used in all the experiments is Sequential, the metrics is accuracy, optimizer is RMSProp and trained for twenty Epochs.

6.2.1 Experiment-1 Model and Accuracy

No. of Dense layer	Units in dense layer	Optimizer	Pixel	Train acc	Test acc	Train loss	Test loss
3	512, 128, 64	RMSProp	0 to 255	0.1934	0.203	1.6029	1.6092

The accuracy was very less. I got only 20% accuracy in testing and 19% accuracy in training. The model is not overfitted but the performance of the model was extremely low. Let’s consider this as a baseline model and try to improve further.

IMAGES OF BOTH CORRECTLY AND WRONGLY CLASSIFIED



From the above image it is clear that the model got major confusions with happiness and love as both has positive faces. Sad has been predicted as Happiness because of the yellow colour sad and happy emojis. The Heart-in image of love was misunderstood as blood and predicted as violence because both of them are red in colour.

6.2.2 Experiment-2 Model and Accuracy

No. of Dense layer	**Units in dense layer**	**Optimizer**	**Pixel**	**Train acc**	**Test acc**	**Train loss**	**Test loss**
5	512, 218, 128, 64, 32	RMSProp	0 to 255	0.2433	0.263	1.7563	2.312

The number of dense layers has been increased from three to five. The accuracy of the model slightly improved by increasing the capacity of the network.

6.2.3 Experiment-3 Model and Accuracy

No. of Dense layer	Units in dense layer	Optimizer	Pixel	Train acc	Test acc	Train loss	Test loss
5	512, 218, 128, 64, 32	RMSProp	0 to 1	0.4699	0.3911	1.2565	1.4979

It is clear that when the data range has been reduced from 0-255 to 0-1 i.e., when the data is normalised, the accuracy of the model improved from 20% to 40%. But the model seems to be overfitted. So, let’s try to reduce in further experiments.

6.2.4 Experiment-4 Model and Accuracy

No. of Dense layer	**Units in dense layer**	**Weight Regularisation**	**Pixel**	**Train acc**	**Test acc**	**Train loss**	**Test loss**
5	512, 218, 128, 64, 32	L1	0 to 1	0.1934	0.203	1.6029	1.6092

In this experiment, to reduce overfitting, L1 weight regularisation has been added to the layer. But the accuracy of the model decreased from 40% to 20%.

6.2.5 Experiment-5 Model and Accuracy

No. of Dense layer	**Units in dense layer**	**Weight Regularisation**	**Pixel**	**Train acc**	**Test acc**	**Train loss**	**Test loss**
5	512, 218, 128, 64, 32	L2	0 to 1	0.1945	0.2	1.6096	1.6094

In this experiment, I tried with L2 weight regularisation. But the accuracy of the model did not improve further. Both the regularisation gives nearly same result. Deep Neural Networks shows poor performance for the dataset. So, let's try adding convolution layers.

6.3 EXPERIMENTS WITH CNN

Universally, the model that has been used in all the experiments is Sequential, the metrics is accuracy, pixel range is normalised to 0-1, number of convolution and pooling layers are 4 (expect for the first experiment), dropout layer with (0.5) (expect for the first experiment) and trained for twenty Epochs.

6.3.1 Experiment-1 Model and Accuracy

Dense layer	**Units in dense**	**Optim iser**	**Conv layers**	**Pooling layers**	**Train acc**	**Val acc**	**Train loss**	**Val loss**
5	512, 218, 128, 64, 32	RMSPr op	3	2	0.9905	0.3914	0.04	8.391

When adding CNN layers to the network, the training accuracy of the model improved from 20% to 99% but the testing accuracy was 40%. It is an extreme overfitting scenario. The model started memorizing the data. So, let's try to reduce it in further experiments. The CNN model predicted the happy smiley emojis as sad emojis. Violence was predicted as happiness and happiness were predicted as violence because both happiness and violence images have group of people in them.

6.3.2 Experiment-2 Model and Accuracy

No. of Dense layer	**Units in dense layer**	**optimiser**	**Weight Regulari -sation**	**Train acc**	**Val acc**	**Train loss**	**Val loss**
1	512	Rmsprop	L2	0.9638	0.463	0.1581	12.4095

Even after adding both L2 regularisation and Dropout to the layer the model seems to be overfitted. But the testing accuracy improved from 39% to 46%. Too many dense layers will also make the model overfit, So the capacity of the network is reduced from five to one.

6.3.3 Experiment-3 Model and Accuracy

No. of Dense layer	**Units in dense layer**	**optimiser**	**Train acc**	**Val acc**	**Train loss**	**Val loss**
2	512, 218	Rmsprop	0.966	0.443	0.1493	6.4199

In this experiment, I tried increased the dense layer and removed the regularisation factor. But still the accuracy remains similar to the previous experiment-2.

6.3.4 Experiment - 4 Model and Accuracy

No. of Dense layer	Units in dense layer	optimiser	Weight Regulari -sation	Train acc	Val acc	Train loss	Val Loss
3	512, 218, 128	Rmsprop	L1 & L2	0.2042	0.179	2577.148	2544.151

After adding both L1 and L2 regularisation to the layers, it decreased the accuracy of the model from 96% to 20%. It is clear that, adding any one type of weight regularisation is sufficient for the data. Adding both decreases the performance.

6.3.5 Experiment - 5 Model and Accuracy

No. of Dense layer	Units in dense layer	optimiser	Weight Regulari -sation	Train acc	Val acc	Train loss	Val loss
3	512, 218, 128	Rmsprop (0.03)	L2	0.2042	0.179	39.6498	39.6531

Increasing the learning rate of the optimiser RMSProp from default 0.001 to 0.03 decreased the performance of the model. Faster learning rates may converge faster but mostly miss the optimum value. After increasing the learning rate and using L2 regularisation the performance of the model remains the same with the previous experiment – 4

6.3.6 Experiment - 6 Model and Accuracy

No. of Dense layer	Units in dense layer	optimiser	Weight Regulari -sation	Train acc	Val acc	Train loss	Val loss
2	512, 218	Adam	L2	0.4229	0.419	1.4098	1.4074

In this experiment, Adam optimiser was used instead of RMSProp. The performance of the model improved from 20% to 42%. There was no overfitting. Adam optimiser works fine for the dataset. But 40% accuracy of the model should be further improved.

6.3.7 Experiment - 7 Model and Accuracy

No. of Dense layer	Units in dense layer	optimiser	Train acc	Val acc	Train loss	Val loss
1	1000	Adam	0.4971	0.425	1.2385	1.4029

In this experiment, I increased the units in the dense layer to1000, removed the regularisation factor and the accuracy improved from 42% to 49% in training but the model overfitted.

6.3.8 Experiment - 8 Model and Accuracy

No. of Dense layer	Units in dense layer	optimiser	Train acc	Val acc	Train loss	Val loss
1	512	Adam	0.966	0.443	0.1493	3.4199

In this experiment, I reduced the units in the dense layer from 1000 to 512 and it worked well. The train accuracy improved from 49% to 96%. From this, it is clear that, too many units in the dense layer decreases the performance of the model for this particular dataset. So, one dense layer with 512 units works fine comparatively.

6.3.9 Experiment - 9 Model and Accuracy

No. of Dense layer	Units in dense layer	optimiser	Train acc	Val acc	Train loss	Val loss
1	1000	Adadelta	0.3217	0.291	1.5481	1.5456

In this experiment, I increased the units in the dense layer to 1000, removed the regularisation factor and used Adadelta optimiser. The performance of the model decreased rapidly. Hence, Adadelta optimiser is not well suited for this dataset.

6.3.10 Experiment - 10 Model and Accuracy

No. of Dense layer	Units in dense layer	optimiser	Regulari sation	Train acc	Val acc	Train loss	Val Loss
1	512	SGD	L2	0.2427	0.243	11.5337	11.4988

In this experiment, I decreased the units in the dense layer to 512, added L2 regularisation factor and used SGD optimiser. The performance of the further decreased, but there was neither overfitting nor underfitting because of adding the L2 regularisation.

6.3.11 Experiment - 11 Model and Accuracy

No. of Dense layer	Units in dense layer	optimiser	Regularisation	Image Augmentation	Train acc	Val acc	Train loss	Val loss
1	512	Adam	L2	Yes	0.5115	0.475	1.2986	1.3224

"Too few images in the dataset" is one of the reasons for overfitting. So, to reduce overfitting and improving the performance of the model, image augmentation can be done. In this experiment, I have made image augmentation (Horizontal Random Flip, Random Rotation and Random Zoom) and added it along with the network architecture. And the performance of the model really improved to 51% in training and 47% in testing. Compared to previous experiments, the result of this experiment is good. Overfitting was reduced and accuracy was improved.

6.4 EXPERIMENTS WITH PRETRAINED CONVNET

Universally, the model that has been used in all the experiments is Sequential, the metrics is accuracy, pixel range was normalised to 0-1, number of convolution and pooling layers are 4, image augmentation is applied, dropout layer with (0.5), one dense layer with unit 512 and trained for twenty Epochs.

6.4.1 Experiment - 1 Model and Accuracy

optimiser	Pretrained Convnet	Train acc	Val acc	Train loss	Val loss
Adam	VGG16	0.4733	0.551	1.3212	1.2477

Performance of the model can be increased with transfer learning. Here, I have used VGG16 pretrained Convnet for feature extraction. But the model seems to be underfitting.

6.4.2 Experiment - 2 Model and Accuracy

optimiser	Regularisation	Pretrained Convnet	Unfreezed	Train acc	Val acc	Train loss	Val loss
RMSProp	L2	VGG16	Block 5	0.2077	0.179	1.6505	1.6518

In this experiment, I fine tuned the pre-trained convolution base and updated the weights of the block 5. But my accuracy decreased.

6.4.3 Experiment - 3 Model and Accuracy

optimiser	Pretrained Convnet	Unfreezed	Train acc	Val acc	Train loss	Val loss
Adam	VGG16	All five blocks	0.4041	0.406	0.406	1.3613

In this experiment, I fine-tuned the pre-trained convolution base and updated the weights of the all the five. My accuracy improved. Adam optimiser is better than the RMSProp optimiser. There was neither overfitting nor underfitting. But the accuracy of the model has to be further increased.

6.4.4 Experiment - 4 Model and Accuracy

optimiser	Regularisation	Pretrained Convnet	Unfreezed	Train acc	Val acc	Train loss	Val loss
SGD	L2	RESNET 50	All five blocks	0.6746	0.461	10.8503	11.4582

In this experiment, I have used Resnet 50 pretrained Convnet trained on ImageNet dataset. It has 50 convolution blocks and highly dense. I updated the weights of all the five blocks and used SGD optimiser with L2 regularisation. But the model seemed to be overfit, SGD optimiser is better than both Adam and RMSProp optimisers and the accuracy improved compared to VGG16 pretrained Convnet.

6.4.5 Experiment - 5 Model and Accuracy

optimiser	Pretrained Convnet	Unfreezed	Train acc	Val acc	Train loss	Val loss
SGD	RESNET 50	Block 5	0.9015	0.716	0.2905	1.3121

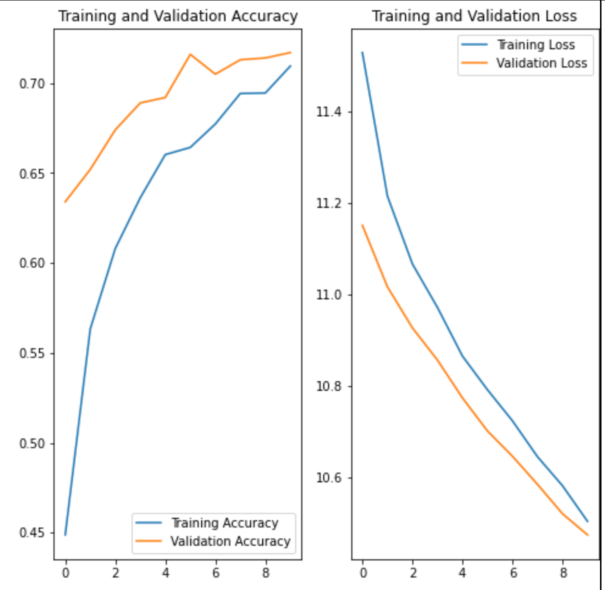
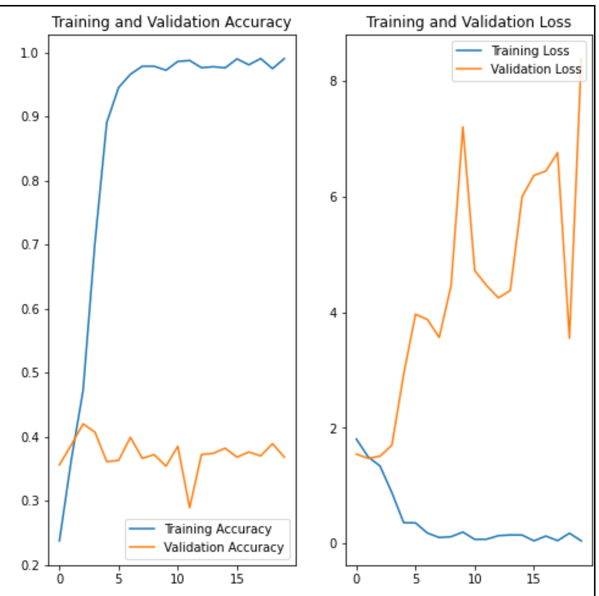
In this experiment, I updated the weights of the fifth block and used SGD optimiser without regularising factor. Accuracy improved compared to previous experiments but seemed to be overfit as there were no regularisation factors.

6.4.6 Experiment - 6 Model and Accuracy

optimiser	Regularisation	Pretrained Convnet	Train acc	Val acc	Test acc	Train loss	Val loss
SGD	L2	Xception	0.7645	0.72	0.7468	9.9039	9.9962

In this experiment, I have used the pretrained convnet “Xception”, which gave us very nice results comparatively. So far, the model that I have experimented either had very less accuracy or has extreme overfitting. But this model seems to be good comparatively. Both the training and the validation accuracy is closer to 72%. This is the first time when compared with all the above experiments, the validation accuracy is 72%. So, far I have got only 40% to 50% accuracy. As I felt this model performs well, I saved the model and tested the model with another set of images which is not there in training and validation and got a testing accuracy of 74.68%.

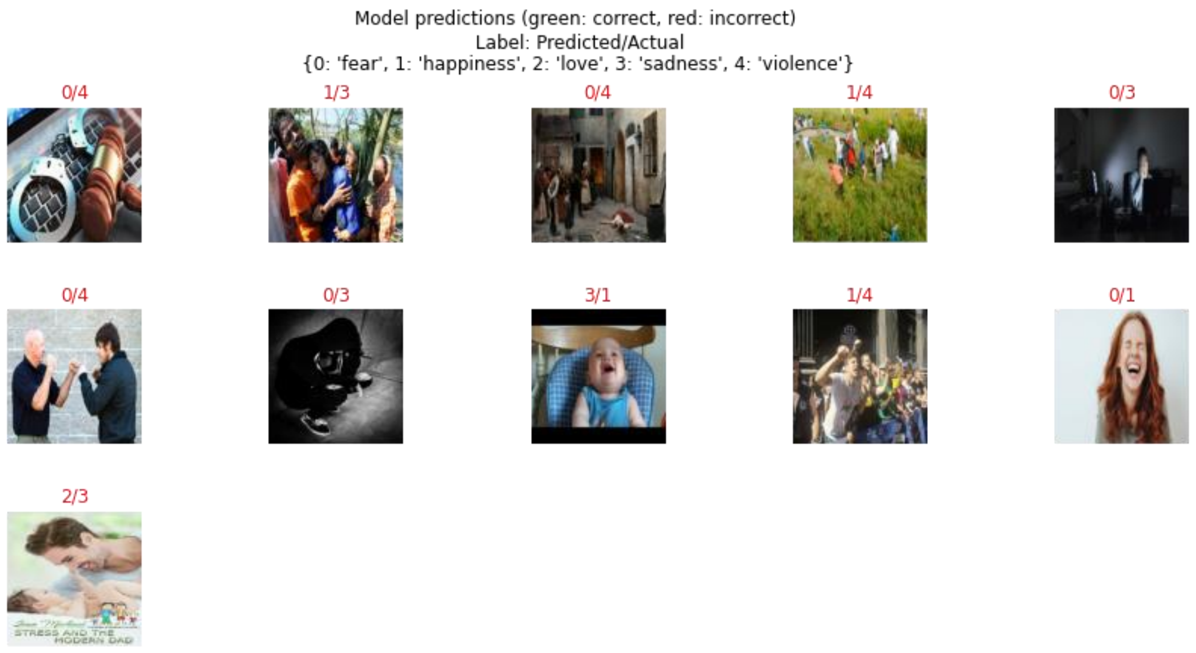
Accuracy and loss of the model.

 *Figure 1 Accuracy & loss of Xception. Less overfitting.*	 *Figure 2 Accuracy & Loss of CNN (6.3.1). More overfitting.*

From the above figure, the figure 1 and figure 2 shows the Accuracy and loss of the model in the Xception pretrained network and in normal convolution neural network (6.3.1 Experiment 1).

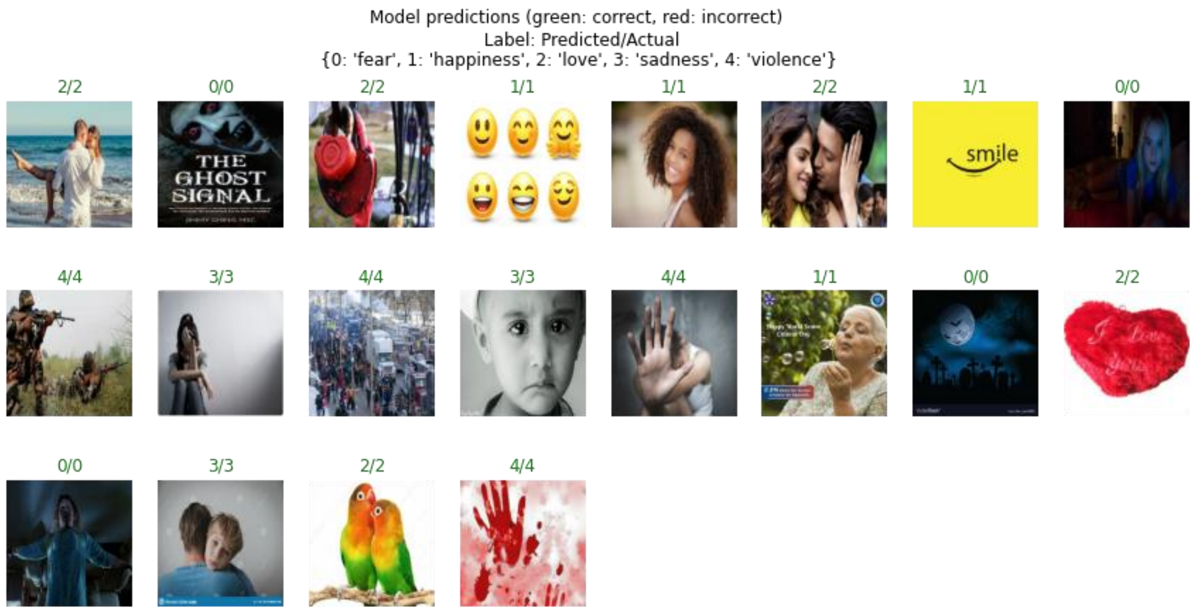
The Xception model did not overfit. Both the training and validation accuracy are nearly the same. Whereas in the CNN model there was extreme overfitting. The training accuracy is nearer to 99% and the validation accuracy is around 40%.

Wrongly classified images


Model predictions (green: correct, red: incorrect)
Label: Predicted/Actual
{0: 'fear', 1: 'happiness', 2: 'love', 3: 'sadness', 4: 'violence'}
0/4
1/3
0/4
1/4
0/3
0/4
0/3
3/1
1/4
0/1
2/3

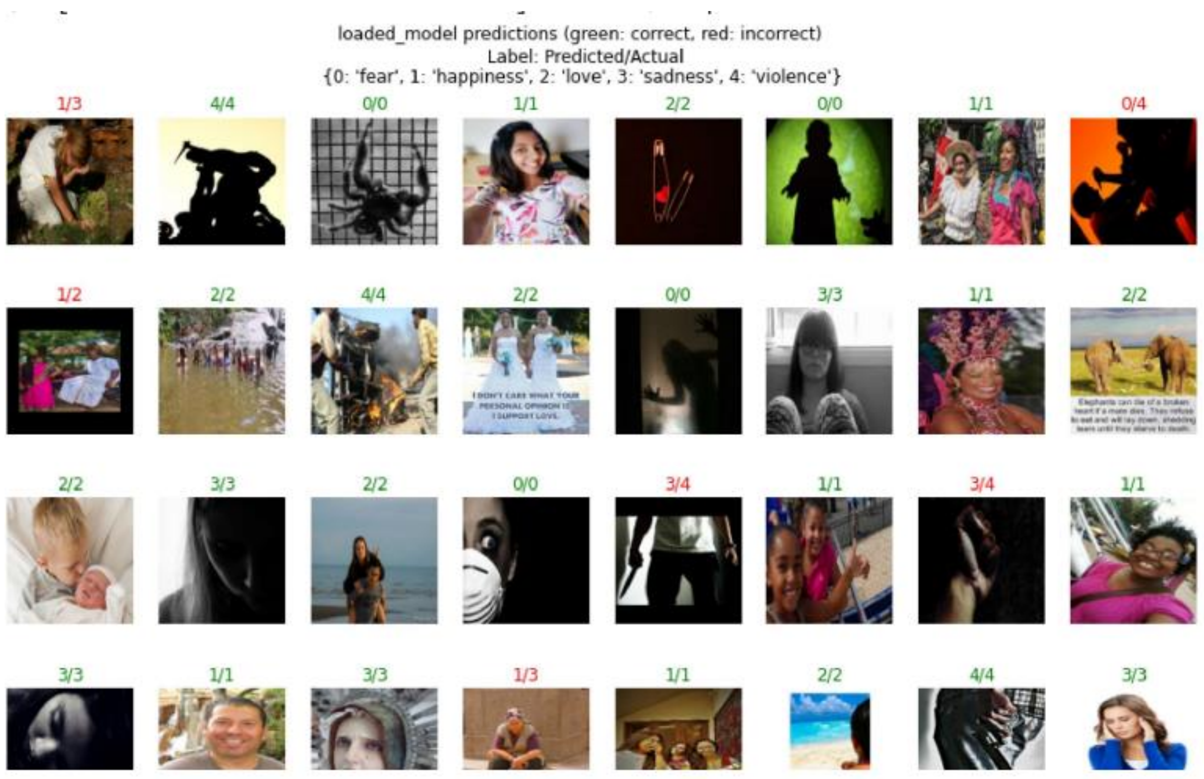
Images that have group of people are classified as happiness instead of violence. As both happiness and violence have images with group of people the model is confused within these two classes.

Images that are correctly predicted by the model.


Model predictions (green: correct, red: incorrect)
Label: Predicted/Actual
{0: 'fear', 1: 'happiness', 2: 'love', 3: 'sadness', 4: 'violence'}
2/2
0/0
2/2
1/1
1/1
2/2
1/1
0/0
4/4
3/3
4/4
3/3
4/4
1/1
0/0
2/2
0/0
3/3
2/2
4/4

From the above picture it is clear that the Xception model learnt well to classify happy emojis.

Both wrongly and correctly classified images from the test set.


loaded_model predictions (green: correct, red: incorrect)
Label: Predicted/Actual
{0: 'fear', 1: 'happiness', 2: 'love', 3: 'sadness', 4: 'violence'}
1/3
4/4
0/0
1/1
2/2
0/0
1/1
0/4
1/2
2/2
4/4
2/2
0/0
3/3
1/1
2/2
2/2
3/3
2/2
0/0
3/4
1/1
3/4
1/1
3/3
1/1
3/3
1/3
1/1
2/2
4/4
3/3

From this picture it is clear that the model learnt well about heart-in. For example: fifth picture in the first row is actually "two safety pins" and as there is a heart-in in-between, the image is correctly classified as "Love". And there is confusion between violence and sad images.

I drew following conclusions about what the model learnt for each category:

Love: The model seems to be learning image with two people showing love, hearts, red colour, flowers and serene landscapes for this category. A lot of images which are tagged as "Happiness" get classified as love.

Happiness: The model seems to learning laughing and smiling faces, bright colours and group of people. Hence, an image that has lot of people of violence get classified as Happiness. I also observe that 80% of the images that are tagged as happiness are face images. Hence this category is biased towards faces.

Violence: The model seems to be learning images of protest, images of crime, guns and no violence banners in this category. But lot of people in protest and group of people enjoying in the happiness category is getting confused.

Fear: This category has the best accuracy. The model seems to be learning dark, haunting images and spiders.

Sadness: The model seems to be learning faces that are sad, bowed down heads, dark surrounded loneliness images.

6.5 CONCLUSION:

The results show that deep learning does provide promising results on the emotion classification task. Emotion classification in images has applications in automatic tagging, automatically categorizing video sequences into genres like thriller, comedy, romance. While using Deep Neural Network alone, the accuracy of the model was very low. After normalising the data, the accuracy improved. After adding convolution layers the accuracy of the model improved drastically. After augmenting images, the model further improved. Increasing learning rate diminishes learning and decreasing learning rate enhances learning of the network. Adding any one type of weight regularisation is sufficient for the data, adding both decreases the performance. Too many units in the dense layer decreases the performance of the model for this particular dataset. Adadelta optimiser is not well suited for this dataset. SGD optimiser is better than both Adam and RMSProp optimisers for this dataset. After using "Xception pretrained network" for feature extraction, the model performed comparatively well with an accuracy of 74.68%. Compared with the baseline model (6.2.1 Experiment 1) the accuracy of model improved from 20% to 74%.

REFERENCES

1. Anam Manzoor, Waqar Ahmad, Muhammad Ehatisham-ul-Haq, Abdul Hannan, Muhammad Asif Khan, M. Usman Ashraf, Ahmed M. Alghamdi and Ahmed S. Alfakeeh (2020) "Inferring Emotion Tags from Object Images Using Convolutional Neural Network"
2. Jingyuan yang, Jie Li, Xiumei Wang, Yuxuan Ding and Xinbo Gao (2021) "Stimuli-aware visual emotion analysis."
3. Jufeng yang, Dongyu She, Yu-Kun Lai, Paul L. Rosin, Ming-Hsuan Yang "Weakly supervised coupled networks for visual sentiment analysis."
4. Lailatul qadri binti zakariaa, Paul Lewis, Wendy Hallc (2017) "Automatic image tagging by using image content analysis."
5. Minyoung huh, Pulkit Agrawal, Alexei A. Efros (2016) "What makes imagenet good for transfer learning?"
6. Ms. Sonali b Gaikwad, prof. S. R. Durugkar (2017) "Image sentiment analysis using different methods: a recent survey"
7. Quanzeng you, Jiebo luo, Hailin jin, Jianchao yang (2016) "Building a large scales dataset for image emotion recognition: the fine print and the benchmark."
8. Rameswar panda, Jianming Zhang, Haoxiang Li, Joon-Young Lee, Xin Lu, and Amit K. Roy-Chowdhury "Contemplating visual emotions: understanding and overcoming dataset bias (supplementary material)."
9. Shaojing fan, Zhiqi Shen, Ming Jiang, Bryan L. Koenig, Juan Xu, Mohan S. Kankanhalli, and Qi Zhao "Emotional attention: a study of image sentiment and visual attention."
10. Takahisa yamamoto, Shiki takeuchi, and atsushi nakazawa (2021) "Image emotion recognition using visual and semantic features reflecting emotional and similar objects."
11. Vasavi gajarla, Aditi gupta "Emotion detection and sentiment analysis of images."
12. Xu can, et al (2014) "Visual sentiment prediction with deep convolutional neural networks."
13. You, quanzeng, et al (2015). "Robust image sentiment analysis using progressively trained and domain transferred deep networks."
14. Yue zhang, Wanying Ding, Ran Xu and Xiaohua "Visual emotion representation learning via emotion-aware pre-training."

15. Zijun wei, Jianming Zhang, Zhe Lin, Joon-Young Lee, Niranjan Balasubramanian, Minh Hoai (2020) “Learning visual emotion representations from web data.”
16. https://www.intel.in/content/www/in/en/products/docs/processors/what-is-a-gpu.html#:~:text=GPUs%20may%20be%20integrated%20into,as%20a%20discrete%20hardware%20unit.
17. https://research.google.com/colaboratory/faq.html#:~:text=Colaboratory%2C%20or%20%E2%80%9CColab%E2%80%9D%20for,learning%2C%20data%20analysis%20and%20education.
18. https://www.python.org/doc/essays/blurb/
19. https://www.ibm.com/in-en/cloud/learn/deep-learning
20. https://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/Architecture/feedforward.html