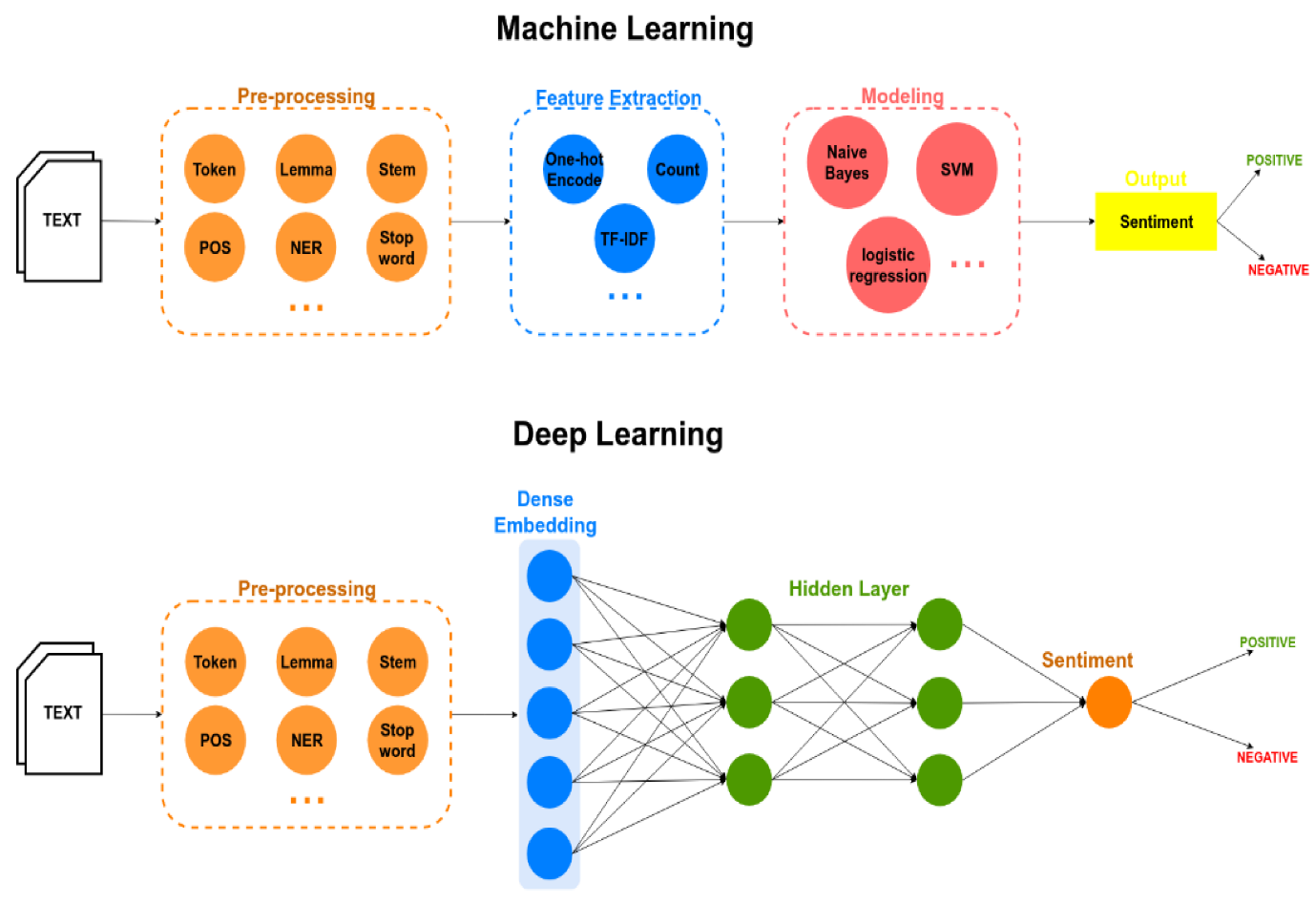
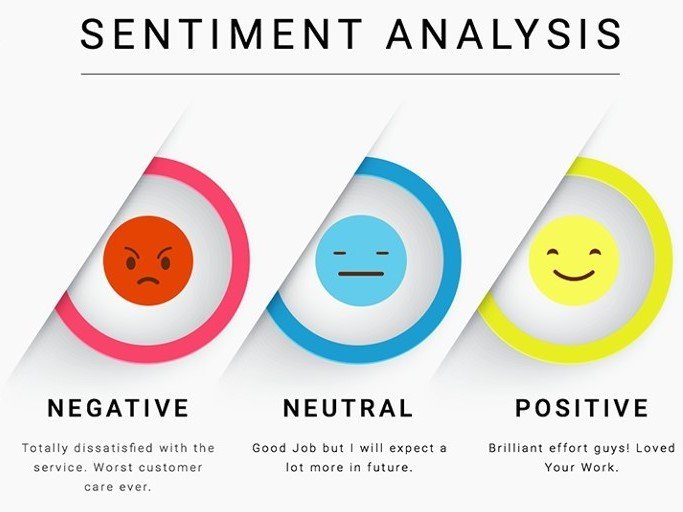
**Image-Text-Sentimental Analysis:**

**By Kumari Pooja**



## Contents

1. Declaration
2. Acknowledgment
3. Introduction
4. Literature Review
5. Methodology
   1. Image Model
      1. Image Preprocessing
      2. Model Building
      3. Training
      4. Testing
   2. Text Model for Sentiment Analysis
      1. Text Cleaning
      2. Model Building
      3. Training
      4. Testing
   3. Integration Model
6. Result and Analysis
7. Bibliography

# Introduction

Sentiment Analysis is one of the rigorous field where extensive study is happening. In this digital world and specifically in social media platforms like Facebook, twitter, LinkedIn etc. people convey their thoughts or give their reactions through posts or comments.

These posts contain sometimes only images, sometimes only texts and sometimes both. And from perspective of sentiment analysis, most of the work has been done using one form of data, like only texts or only using images but not both.

Using only one type of data and make a model upon it is known as unimodal. And when we include two types of data like images and texts and make a model then it is known as bi-modal or multi-modal.

The motivation for this project comes from the research paper by (Huang *et.al*, 2019). They proposed Deep Multimodal Attention Model (DMAF). In this research paper, a sentiment analysis model is presented on multi-modal dataset using the deep neural network. It gives the accuracy of 88% on flickr-m dataset.

This project aims to create a bi-modal using both images and text for doing sentiment analysis. We are not recreating this research paper but tried to create a model using cnn and simple Machine Learning classification tool to work on both type of dataset at one time.

# Literature Review

The paper by (HUANG *et.al,* 2020) proposed **Attention-Based Modality-Gated Networks(AMGN)**. This method tries to find the correlation between image and text and use it for the sentiment analysis. To extract the features from the images, a **visual attention model** is proposed and to extract the features from text, a **modality-gated LSTM** model is proposed. They used a visual-sentiment attention modal that extract image features for each word. Then the model was experimented on both manually annotated datasets and datasets labeled by machines. Collected and labeled the dataset Getty, the model gave the accuracy of 88.2%, 79% accuracy on Twitter dataset, 87.3% accuracy on machine labeled Flickr dataset and 89.2% accuracy on manually labeled Flickr dataset.

The paper proposed by (Setiawan *et.al,* 2021) also worked on image-text sentiment analysis. They provided the sentiment analysis on Indonesian social media post. They achieved 70.1% average accuracy with **ensemble learning using Logistic Regression** and **Image-text concept as meta classifier.** The suggested model fuses three different points of view using an ensemble method and a metaclassifier. Using Deep Convolutional Neural Networks, the text was classified. To maintain semantic meaning, the input feature for text representation uses Word2Vec. Additionally, the DeepSentiBank model is used to extract concepts from images and SenticNet 5 to analyse concepts from texts. They collected 2089 Adjective Noun Pairs and used a Multi-Layer Perceptron to classify them. Then, using ensemble learning, predicted probabilities from each classifier for each image, text, and concept was combined.

Then a meta-classifier was used to combine Image, Text, and Concept information in order to predict the final sentiment.

Another work on this area done by (Qian *et.al.,* 2019), proposed a Text-Image Sentiment Classification**(TISC)** method. They used Deep CNN to analyse the characteristics of images and tweets. Fine tuned the AlexNet pre-trained CNN for images and for initializing text features they used AffectiveSpace. The model was evaluated on twitter dataset. And by balancing their parameters of alpha and beta as *α*

+ *β ∈*(0*,* 0*.*5), it gives the accuracy of 0.805.

**Cross-modality consistent regression(CCR)** model, proposed by (You *et.al.,* 2016) has also done great work on predicting the sentiment using image-text data. They used CNN for extracting features from image data and fine tuned it and trained a paragraph vector model for extracting features from text. Then they trained multi-modality regression model on top of them. On gettty images, the CCR gives 80% accuracy and on twitter dataset the model gives 80.9% accuracy.

The paper by (Huang *et.al*, 2019) provided **(Deep Multimodal Attention Fusion)DMAF** model. It uses two unimodal attention mechanism to extract features from images and text separately. Then an intermediate fusion is used to find the correlation between image and text. Then these three models are combined using late fusion. The model gives 86.9% accuracy data collected from getty, 76.3% accuracy on twitter dataset, 85.9% accuracy on Flickrw dataset and 88% accuracy on flickr-m dataset.

#### Model Comparison Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1** | **Accuracy** |
| **AMGN** | 0.898 | 0.876 | 0.887 | 0.882 |
| **Ensemble Learning with Logistic Regression** | - | - | - | 0.701 |
| **TISC** | - | - | - | 0.805 |
| **CCR** | 0.831 | 0.805 | 0.818 | 0.809 |
| **DMAF(flickr-m)** | 0.882 | 0.870 | 0.876 | 0.880 |

**Methodology**

**Image Model**

* *Image Dataset*

For this project the dataset is taken from publicly available dataset repository, kaggle. The description of the dataset is described below:

* **ck+: Extended Cohn Kanade** dataset, introduced by Patrick Lucey et al. is a collection of facial expression dataset. The dataset is divided into seven categories as follows:

###### Anger:

**Contempt:**

**Disgust:**

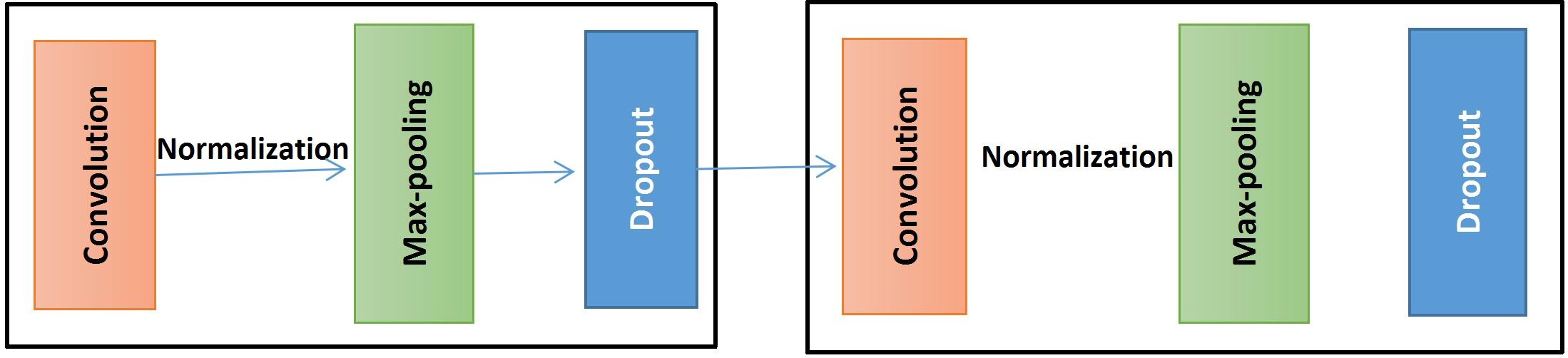
**Fear:**

**Happy:**

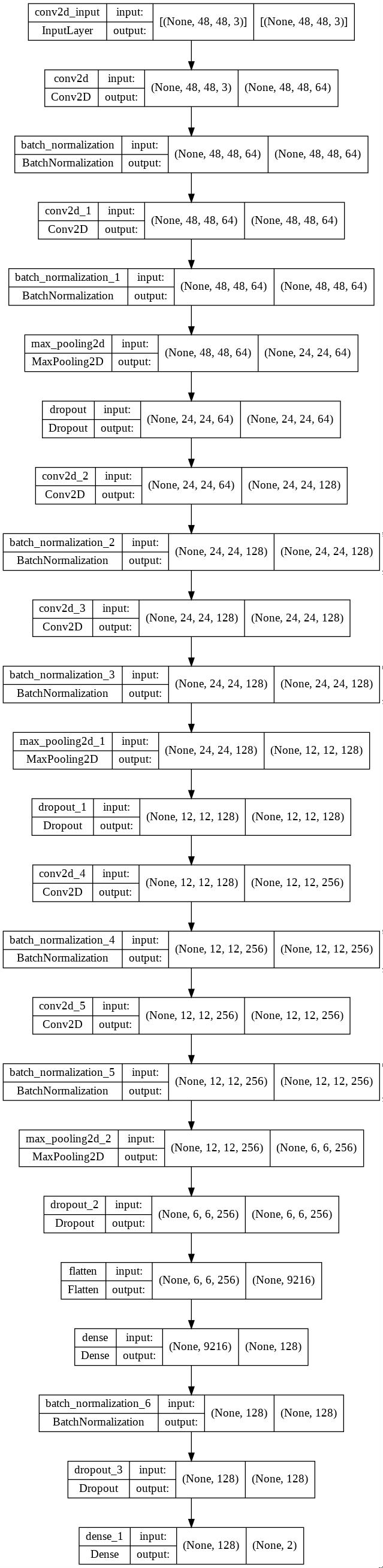
**Sadness:**

**Surprise:**

* The dataset is combined in two categories, **positive** and **negative. Positive** contains happy and surprise categories and **Negative** contains anger, fear, sadness, contempt and disgust.
* The dataset contains total 908 images, from which model is trained on 888 images and tested on 20 images. Further positive category has 457 images on training set and 431 images on negative category.
* Next created a function which reads an image from both directory and stores it pixel values in an array. We have also stored corresponding label in another array which contains binary values where 1 denotes positive and 0 denotes negative.
* Then converted each of these binary values to a vector of two as we want to predict the probability for both the categories at the output layer of the image model. I use to\_categorical() method from utils library.
* Since pixel values lies in the range of 0 to 255 and that’s a huge range of values, so divided the pixel values by 255 to bring all the values in range (0,1).
* Next divided the dataset to train and validation using sklearn ‘s train\_test\_split() method with 80-20% and stratify equals to True.
* I got 710 images for training and 178 images for validation. The shape of each input image is 48 by 48 with three channels I.e(48\*48\*3).



* Then build the network model as follows:
  + We first initialize the model with tensorflow Sequential() method to attach the layers sequentially.
  + Then the first layer is a **convolution layer** with 64 (5 \*5) filters with **padding as same**, **activation function as elu** and weight is initialized using **he\_normal** method.
  + Next the output of the convolution was **normalized** and followed by a **max-pooling** layer with (2 \* 2) matrix.
  + Then used **dropout** layer to avoid **overfitting** of the model. And this process is done three times one after the other.
  + Next flattened the output and added one dense layer which is again normalized.
  + Then added one dropout layer and then the dense layer.

The layer wise architecture is shown below.

#### CNN Model Architecture

* The images are then fed to the model architecture.
* We used loss function as **binary\_crossentropy**, as there are two categories, optimizer as **adam** and metrics as **accuracy**.
* Then the model is trained for **60 epochs** and **batch size 40**.
* To avoid the overfitting of the model **Early Stopping** and **ReduceLROnPlateau**

is used.

* Total parameters were 2,398,146 and trainable parameters is also same.
* After training and validating the model, stored the model and weights for further use.

***Text Model for Text Sentiment Analysis***

For text sentiment analysis we have used twitter dataset extracted from kaggle. The data is in the csv file format and the data consists of around 10,48,576 rows and 6 rows consisting of different features from which we need to find out the features that influence the target column and tells us about the sentiments of twitter post by performing the Exploratory Data Analysis(EDA).

The dataset contains around 1M tweets with six columns:-

* **target:** target column tells if a tweet is positive or negative.
* **id:** id is the twitter id.
* **date:** date tells when the tweet was posted.
* **Flag:** some kind of flag value while collecting the data.
* **User:** user tells who posted it.
* **Text:** text are the tweets.

Next to analyse the data following tasks have been performed:

##### Pre-processing/Data Cleaning

The real world data contains lots of noise which can’t be given to a machine learning model otherwise the model will also learn gibberish. To avoid that we need to carefully clean the data and remove any discrepancies.

* First we start with using value counts(), unique(),nuniques(), isnull() on the dataset and try to find the presence of any noisy data so that it can be removed from the dataset.
* These functions are useful in recognizing if we need that column or what kind of values are present in the columns or what kind of pre-processing might be required.
* And as a result we found that target variable contains only 2 unique values(4 & 0) indicating 4 as positive and 0 as negative and which need further processing.
* And each category has equal number of tweets which makes the dataset balanced. Also **Flag** variable has only one unique value “NO\_QUERY”.
* After getting an overview of the data, we concluded that we'll drop columns *"ID","Date", “User” and "Flag"* as there is not much use of those to do make our model.

Now we are left with target and text column. Now we need to clean the text column. To clean and make it in a format that our model can easily analyse following cleaning steps are taken:

* Remove all punctuation from the text. To do this a user defined function was created which takes a text and remove all punctuation marks from it and returns the punctuation removed text.
* Remove all mentions. Mentions in twitter are the twitter user name of another twitter user that the user has mentioned and it starts with “@”. With the help of user defined function and regex, removed all mentions from the text.
* Expand the contractions. Contractions are short forms like *we’re, don’t, etc.* For this we have created a dictionary "contractions" that contains collections of such contractions and it’s expanded version. Then we created a function "change" and apply the function to the text column that gives us the expansion of each contraction present in twitter text.
* Replace all 4 in target column with 1. Before the text was like:



and after cleaning we got:



Although the text is not perfectly clean but it is cleaned from lots of noise. We store the clean text in a separate column and dropped the original text column.

Now, by this we have finished the preprocessing part and thus we'll create a new CSV file and store the clean data in the new csv file so that we don’t have to run the cleaning code again and again every time we work on building the model. We can directly use this clean text data.

Next is the ***model building*** part.

* First we divided the available data into three sets: *training data*, *validation data*, and *test data*.
* The training data are those used for the learning by Machine Learning algorithms to obtain the parameters of the model with the iterative method that we have already mentioned.
* For this we created variables X and Y in which we stored "clean\_text" and "target" columns respectively.
* Now, from "sklearn.model\_selection" we used the "train\_test\_split" to split the data into train and test in the ratio of 7:3. Where the train data is 70% and test data is 30%. We further divided the train data to create train and validation data.
* We still have our data in text format and to build the machine learning model we first need to convert these text to numbers as we can’t feed text data to these Machine Learning models.
* So, with the help of CounterVectorizer of sklearn module with *max\_features* is equal to 15000 we transform our text in numeric form. What it will do is, given a train dataset, first it will extract all unique 15000 thousands words because we have specified max\_features as 15000 from the whole train dataset.
* Then for a text it will count the occurrences of each 15000 words in that text. The data that we get from this will be sparse as dataset will mostly contains zeros.
* We extract features using fit() method and count the occurrences using transform() method of CountVectorizer.
* We extract the features from train dataset and transform all train, test and validation based on features extracted from train dataset.
* After the transformation of text into vector form, the data is ready to be applied on machine learning models.
* We trained our model on Logistic Regression, MultinomialNB, Random Forest and svm. Random Forest and svm took too long to train, so we excluded these two from our ML model list.
* We also experimented with bi-gram and tri-gram model on the algorithms.
* We also used tf-idf to convert the text data to numerical form and fit our machine learning model on them.
* Then we stored the model which was giving best accuracy using pickel module.

### Integration model

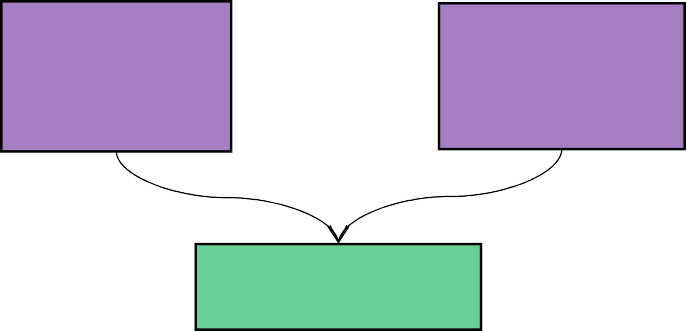


Image Sentiment

Model

Text Sentiment

Model

Average probability

score

Fig. Architecture of integration model

Above we saw, how we have done the sentiment analysis of Image and text dataset. So, basically we have a twitter dataset from which we built our text sentiment modal and we have image dataset from which we built image sentiment model. Next to get the overall prediction on image and text both we first create a test dataset. Because we have image data separately and text data separately and in order to test our model we need a dataset which has image and text both simultaneously.

* To create the test dataset, we first collect the list of all the image names from both test/positive and test/negative directory and a label list which stores 0 or 1 values, 0 if the image name taken from negative directory and 1 if from positive directory.
* For this we took the help of os.list() function which returns the list of all files in a path. Then using these two list, I.e. image name list and image label list, we create a dataframe with three columns. Two for the two list and one for storing the serial no I.e. 0 to 83. Total test images data are 167, where 84 images are positive and 83 are negative.
* Next to collect the text data for testing, we used the cleaned text from text model. Since, there are 84 positive images, we extracted 84 text from this dataset where target is 1 and stored in another dataframe and added one more column of serial no I.e. 0 to 83.
* Then merged the image dataframe previously created with new text dataframe on the basis of serial no. And we got a dataframe which has a positive sentiment texts and positive sentiment image names and the target column which contains 1.
* Similarly created another dataframe to store negative image names and text and at last merged the positive image\_text dataframe and negative image\_text dataframe row wise to get our final test dataset.
* Now, to predict image and text simultaneously, we first load the saved image model and weights using json and saved text model using pickle.
* Then we have created a function predict() which will do the prediction. There are two more functions which are defined inside this function I.e. predict\_img\_sentiment() and predict\_text\_sentiment().
* predict\_img\_sentiment() will give prediction on images and predict\_text\_sentiment() will give prediction on text.
* Image sentiment prediction function takes the path of the image, where it reads the images through opencv module, resize and reshape it to (48\*48\*3) and does the prediction using saved image model.
* Similarly, text sentiment prediction function takes a text as an argument, where it first transform the text in bi-gram using countvectorizer and does the prediction using saved text model and return the prediction score using *predict\_proba* which returns probability score for each class.
* After these two functions we use a for loop which iterate through each row and extract the image name and corresponding text.
* Then using image name we define the image path and pass it to image semtiment prediction function. Similarly, we pass the text to text sentiment prediction function. From both function we get the prediction score.
* From predict\_img\_sentiment() we get a score in a list which contains two values, first value tells the probability score for negative class and second value tells the probability score for positive class.
* From predict\_text\_sentiment(), similar to image prediction, we get list with two values. Next we take the negative probability score of both image and text and do the average and store it in a variable.
* The same we do to get average probability score. Then using if else, we give the overall prediction. So, if negative average probability is greater than positive probability then our prediction is negative means 0,or Else,our prediction is positive i.e 1 and appended all the prediction to a list.
* The countvectorizer used for transformation is trained on the same training dataset which was used in building text model.

# Result and Analysis

***Image Model***

On training data CNN model gives the accuracy of 98% and it was able to reduce loss to 0.02 and on validation data it gives the accuracy of 97% and on test data it gives the accuracy of 97%.

Below epoch graph on accuracy and loss data shows the performance of our model.

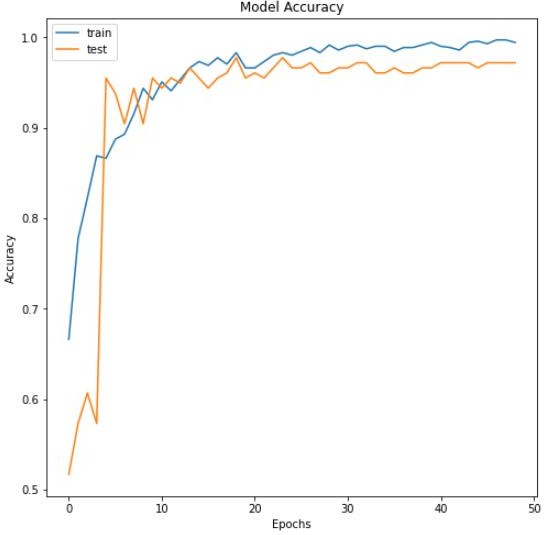


Fig. Model accuracy on epochs

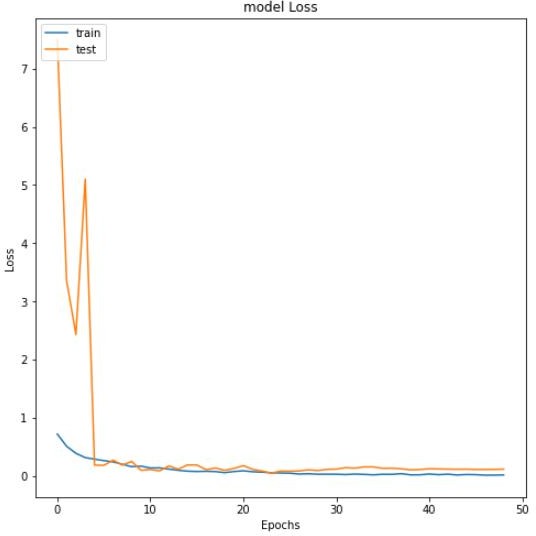


Fig. Model loss on epochs

All the result is shown in below table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Accuracy** | **Precision** | **Recall** | **F1-Score** |
| **Training Data** | 98% | 98% | 98% | 99% |
| **Validation Data** | 97% | 96% | 96% | 98% |
| **Test Data1** | 97% | 98% | 97% | 97% |
| **Test Data2(Fer Mix)** | 75% | 76% | 75% | 75% |

To test the model we also created a small dataset where we mixed some images in each category from Fer-2013 dataset. Fer2013 is also similar type of dataset as of ck+ but it contains more varieties of facial expression dataset. That’s why the accuracy dropped to 75% in test data2 where it is 97% on ck+ data.

### Text Model

On training data, text sentiment model gave 76% accuracy on Multinomial naive bayes algorithm with alpha value 5, it also gives similar score on testing and validation data, 76% on both. Logistic regression does a little better than Multinomial

naive bayes. On training data, it gives accuracy of 77% and on validation and test data, it gives accuracy of 77% on both. Then we used tf-idf vectorization method to transform the features. Below are the accuracy scores with various vectorization method:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **train** | **CV** | **test** |
| **count\_multi** | 0.766185 | 0.761163 | 0.760303 |
| **count\_LR** | 0.778457 | 0.767119 | 0.767372 |
| **tfidf\_multi** | 0.781971 | 0.777558 | 0.776987 |
| **tfidf\_LR** | 0.805893 | 0.796760 | 0.796445 |
| **bigram\_multi** | 0.787636 | 0.785527 | 0.785066 |
| **bigram\_LR** | 0.816248 | 0.806164 | 0.805999 |
| **trigram\_multi** | 0.785416 | 0.783747 | 0.782605 |
| **trigram\_LR** | 0.816196 | 0.806038 | 0.806024 |

Among these bigram and trigram model gives the best accuracy of 80.5% and 80.6 % respectively with Logistic Regression model.

##### Integration model

Till now we saw the result of individual models. Now let’s see the result on combined model. The integration model is tested on 167 data points. The image set contains

fer-dataset also. It gives the accuracy of 77%.

|  |  |  |  |
| --- | --- | --- | --- |
| **Text** | **Image** | **Actual** | **Prediction** |
| **'i love u guys r t he best'** |  | 1(positive) | 1(positive) |
| **did not place in th e peeps contest but**  **thanks for voting anyways** |  | 1(positive) | 0(negative) |
| **our duck and chic ken are taking way yy too**  **long to hatch** |  | 0(negative) | 0(negative) |
| **falling asleep just heard about that t racy girls body bei ng found how sad my heart breaks fo r that family** |  | 0(negative) | 1(positive) |

**CONCLUSION**

The Image-text sentiment analysis system is one of the essential systems today and it has very important as for an image dataset it is useful for reading/analyzing the reaction(happy, sad or neutral) and text data is for analyzing reviews about any new product in market or any kind of new application reviews.

Sentiment analysis of image-text is to understand people’s position, attitude, and opinion toward a certain product. By doing this project, we are able to work on recent development in the machine learning field. I am able to apply deep learning algorithms in building real time applications.

# References

1. Feiran Huang, Kaimin Wei, Jian Weng, and Zhoujun Li. 2020. Attention-Based Modality-Gated Networks for Image-Text Sentiment Analysis. ACM Trans. Multimedia Comput. Commun. Appl. 16, 3, Article 79 (August 2020), 19 pages.<https://doi.org/10.1145/3388861>
2. You, Q., Luo, J., Jin, H., & Yang, J. (2016, February). Cross-modality consistent regression for joint visual-textual sentiment analysis of social multimedia.

In *Proceedings of the Ninth ACM international conference on Web search and data mining* (pp. 13-22).

1. Qian, C., Ragusa, E., Chaturvedi, I., Cambria, E., & Zunino, R. (2019). Text-image sentiment analysis.
2. Setiawan, E. I., Juwiantho, H., Santoso, J., Sumpeno, S., Fujisawa, K., & Purnomo, M. H. Multiview Sentiment Analysis with Image-Text-Concept Features of Indonesian Social Media Posts.
3. Huang, F., Zhang, X., Zhao, Z., Xu, J., & Li, Z. (2019). Image–text sentiment analysis via deep multimodal attentive fusion. *Knowledge-Based Systems*, *167*, 26-37.
4. [https://towardsdatascience.com/convolutional-neural-network-feature-map-and-fi](https://towardsdatascience.com/convolutional-neural-network-feature-map-and-filter-visualization-f75012a5a49c) lter-visualization-f75012a5a49c
5. <https://machinelearningmastery.com/reproducible-results-neural-networks-keras/>
6. <https://www.kaggle.com/datasets/abhi8923shriv/sentiment-analysis-dataset>
7. nlp.stanford.edu/courses/cs224n/2011/reports/patlai.pdf
8. National Daily, Economic Times: articles.economictimes.indiatimes.com › Collections › Facebook
9. K. Dave, S. Lawrence, and D.M. Pennock. “Mining the peanut gallery: Opinion extraction and semantic classification of product reviews”. In Proceedings of the 12th International Conference on World Wide Web (WWW), 2003, pp. 519–528.
10. A. Pak and P. Paroubek. “Twitter as a Corpus for Sentiment Analysis and Opinion Mining”. In Proceedings of the Seventh Conference on International Language Resources and Evaluation, 2010, pp.1320–1326.
11. R. Parikh and M. Movassate, “Sentiment Analysis of User-Generated Twitter Updates using Various Classiﬁcation Techniques”, CS224N Final Report, 2009
12. A. Go, R. Bhayani, L.Huang. “Twitter Sentiment Classification Using Distant Supervision”. Stanford University, Technical Paper ,2009
13. J. Read. “Using emoticons to reduce dependency in machine learning techniques for sentiment classification”. In Proceedings of ACL-05, 43nd Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, 2005
14. L. Barbosa, J. Feng. “Robust Sentiment Detection on Twitter from Biased and Noisy Data”. COLING 2010: Poster Volume, pp. 36-44.
15. S. Batra and D. Rao, ”Entity Based Sentiment Analysis on Twitter”, Stanford University,2010
16. A. Bifet and E. Frank, ”Sentiment Knowledge Discovery in Twitter Streaming Data”, In Proceedings of the 13th International Conference on Discovery Science, Berlin, Germany: Springer,2010, pp. 1–15.
17. A. Agarwal, B. Xie, I. Vovsha, O. Rambow, R. Passonneau, “Sentiment Analysis of Twitter Data”, In Proceedings of the ACL 2011 Workshop on Languages in Social Media,2011 , pp. 30–38 [13] A. Kumar. and T. M. Sebastian, “Sentiment Analysis:Perspective on its Past, Present and Future”, International Journal of Intelligent Systems and Applications (IJISA), MECS Publisher, 2012 (Accepted to be published)
18. A. Kumar and T. M. Sebastian, “Machine learning assisted Sentiment Analysis”. Proceedings of International Conference on Computer Science & Engineering (ICCSE’2012), 2012, pp. 123-130.
19. POS Tagger[: http://www.infogistics.com/textanalysis.html](http://www.infogistics.com/textanalysis.html)
20. V. Hatzivassiloglou and K. McKeown, “Predicting the semantic orientation of adjectives”. In Proceedings of the Joint ACL/EACL Conference,2004, pp. 174–181