

SENTIMENTAL ANALYSIS ON TEXTUAL IMAGES USING CONVOLUTIONAL NEURAL NETWORK

a project report submitted by

JOEMON JOHNSON	(URK18EC010)
PRAVIN BOSCO J	(URK18EC060)
ALLWYN K GEEVARUGHESE	(URK18EC070)
VISHNUMOHAN R K	(URK18EC072)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

ELECTRONICS AND COMMUNICATION ENGINEERING

under the supervision of

Dr. A. AMIR ANTON JONE M.E., Ph.D.,



**DEPARTMENT OF ELECTRONICS AND COMMUNICATION
ENGINEERING**

KARUNYA INSTITUTE OF TECHNOLOGY AND SCIENCES

(Deemed to be university)

Karunya Nagar, Coimbatore - 641 114. INDIA

APRIL 2022



Karunya INSTITUTE OF TECHNOLOGY AND SCIENCES
(Deemed to be University)

(Declared as Deemed-to-be university-under Sec-3 of the UGC Act, 1956)

KARUNYA NAGAR, COIMBATORE-641114

BONAFIDE CERTIFICATE

This is to certify that the project report entitled, “**Sentimental Analysis on Textual Images using Convolutional Neural Network**” is a bonafide record of work of the following candidates who carried out the project work under my supervision during the academic year 2020-2021.

JOEMON JOHNSON	(URK18EC010)
PRAVIN BOSCO J	(URK18EC060)
ALLWYN K GEEVARUGHESE	(URK18EC070)
VISHNUMOHAN R K	(URK18EC072)

Dr. A. Amir Anton Jone M.E., Ph.D.,
SUPERVISOR

Dr. D. Nirmal M.E., Ph.D.,
Head of The Department

DEPARTMENT OF ELECTRONICS AND
COMMUNICATION ENGINEERING

Submitted for the Half Semester Viva Voce examination held on **12th APRIL 2022**

.....
(Internal Examiner)

.....
(External Examiner)

ABSTRACT

Deep learning, in addition to its success in many other application fields, has become increasingly popular in sentiment analysis in recent years. Deep learning neural networks have profoundly changed the field of natural language processing (NLP).

The views and perceptions of reality, as well as the decisions we make, are heavily influenced by how others see and interpret the world. Sentiment analysis or opinion mining refers to the computer study of people's opinions, feelings, emotions, evaluations, and attitudes about things like products, services, organizations, people, topics, events, themes, and their attributes. In the areas of speech and image recognition, deep learning has made great progress. Deep learning is a powerful machine learning technique that learns several layers of data representations or features and produces cutting-edge prediction results.

In this project, the machine learning model will extract the text from the textual images, and the sentiments of the extracted texts will be predicted using another machine learning model. From this it is possible to determine people's feelings or opinions on a certain event. This allows to determine if a certain item or service is excellent, terrible, or preferable. It may also be used to find out what people think about any event or person, as well as the polarity of text, whether positive or negative.

This paper shows a web application that predicts the sentiments of textual images using a neural network architecture that leverages Convolutional Neural Network (CNN).

ACKNOWLEDGEMENT

First and foremost, we would like to thank **Almighty God** for all the blessings He has bestowed upon us to work thus far and finish this project. We are grateful to our most respected founder (late) **Dr. D.G.S. Dhinakaran, C.A.I.I.B, Ph.D.**, and honorable chancellor **Dr. Paul Dhinakaran, M.B.A, Ph.D.**, for their grace and blessing.

We express our gratitude to the Vice Chancellor **Dr. P. Mannar Jawahar, Ph.D.**, Pro Vice Chancellor (QS) **Dr. Ridling Margarat Waller, Ph.D.**, **Dr. E. J. James**, Pro-Vice Chancellor (R&C) and the Registrar **Dr. R. Elijah Blessing, Ph.D.**, Karunya Institute of Technology and Sciences, for their enduring leadership.

We extend my thanks to our Director, Engineering and technology, **Dr. Prince Arul Raj, Ph.D.**, Karunya institute of technology and sciences, for his excellent encouragements in course of this work.

We are very thankful to **Dr. D. Nirmal, M.E., Ph.D.**, Professor & Head, Department of Electronics and Communication Engineering, Karunya institute of technology and sciences for his constant readiness in providing help and encouragement at all stages in our project.

We express our deepest gratitude to our mentors **Dr. T. Mary Neebha Ph.D.**, **Dr. N.M. Siva Mangai Ph.D.**, for giving us this opportunity and providing us with an environment to complete our project successfully.

Our sincere and special thanks to our guide, **Dr. A. Amir Anton Jone M.E., Ph.D.**, Assistant Professor, for his immense help and guidance. We would like to extend a thankful heart for her constant support through the entire project.

Finally, we would like to extend our deepest appreciation to our family and friends for all that they were to us during the project period.

CHAPTER	CONTENTS	PAGE NO
BONAFIDE CERTIFICATE		<i>(ii)</i>
ABSTRACT		<i>(iii)</i>
ACKNOWLEDGEMENT		<i>(iv)</i>
LIST OF FIGURES		<i>(viii)</i>
LIST OF TABLES		<i>(ix)</i>
ABBREVIATIONS		<i>(x)</i>
1. INTRODUCTION		1
1.1 Proposed Work		1
1.2 Related Works		
1.3 Advantage over Previous Models		11
2. LITERATURE SURVEY		12
2.1 Pre-Processing		12
2.2 Types of Sentiment Classification		12
2.2.1 Document Level Sentiment Classification		12
2.2.2 Sentence Level Sentiment Classification		13
2.2.3 Aspect Level Sentiment Classification		13
2.3 Optical Character Recognition		13
2.4 Deep Learning		14
2.5 Types of text extraction		15
2.5.1 Region based method		15
2.5.2 CC based method		15
2.5.3 Edge based method		15
2.5.4 Texture based method		16
2.5.5 Morphological based method		16
2.6 Deep Learning architectures for image extraction and text classification		16
2.7 Performance Comparison		17
2.8 Machine learning model used		18
3. PROPOSED SYSTEM		19
3.1 Proposed Block Diagram		19
3.2 Web Application		19

3.3 Convolutional Neural Networks	20
3.4 Polarity Distribution	21
4. TRAINING AND TESTING THE CNN MODELS	22
4.1 Block Diagram of Sentiment CNN Model	22
4.2 IMDB Review Dataset	22
4.3 Word Embedding	22
4.4 Pre-Trained Word Embedding Model	23
4.5 Training and Testing the CNN Model	23
4.6 Sentiment Model Output	24
4.6.1 Positive Sentiments	24
4.6.2 Negative Sentiments	24
4.7 Block diagram of CNN model for text extraction	25
4.8 Image dataset for text extraction model	26
4.9 Steps of text extraction	26
4.9.1 Pre-processing an input image	26
4.9.2 Detection of the text	26
4.9.3 Recognition of texts	26
4.10 Summary of the text extraction model	27
4.11 Training and testing of the CNN model for text extraction	28
4.12 Performance Matrices	29
4.12.1 Precision Score	29
4.12.2 Recall Score	29
4.12.3 F1 Score	30
5. TEXT EXTRACTION FROM IMAGE AND SENTIMENT ANALYSIS	31
5.1 Front End	31
5.1.1 React	31
5.1.2 Next.js	33
5.1.3 Tailwind CSS	35
5.1.4 Other developer utilities	36
5.2 Back End	38
5.2.1 Python	38
5.2.1.1 spaCy	39
5.2.1.2 PyTorch	39
5.2.1.3 Pandas	39
5.2.1.4 PIL (Python Imaging Library)	40
5.2.1.4.1 Image	40
5.2.1.5 Numpy	40
5.2.1.6 Math	40

5.2.1.7 Scikit-learn	40
5.2.1.7.1 sklearn.matrices	41
5.2.1.7.2 sklearn.matrices.precision_score	41
5.2.1.7.3 sklearn.matrices.recall_score	41
5.2.1.7.4 sklearn.matrices.f1_score	41
5.2.1.8 shutil	41
5.2.1.8.1 shutil.copyfile	42
5.2.1.9 keras	42
5.2.1.9.1 Sequential	42
5.2.1.9.2 Keras layer	42
5.2.1.9.2.1 Dense layer	42
5.2.1.9.2.2 Flatten	43
5.2.1.10 seaborn	43
5.2.1.11 sys module	43
5.2.1.12 OS Module	43
5.2.2 Amazon Web Services	43
5.2.2.1 Amazon Sagemaker	43
5.2.2.2 Working of Amazon Sagemaker	44
5.2.2.3 API gateway	45
5.2.2.4 REST API	45
5.2.2.5 AWS API Gateway	46
5.2.2.6 Senti API	46
6. RESULT AND DISCUSSION	47
6.1 User Interface	47
6.2 Uploading a textual image	47
6.3 Uploading the link of a textual image	49
7. CONCLUSION AND FUTURE SCOPE	50
7.1 Conclusion	50
7.2 Future Scope	50
7.2.1 Social Media Monitoring	50
7.2.2 Brand Monitoring	50
7.2.3 Product Analysis	51
8. REFERENCES	52

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
3.1	Proposed Block Diagram of Sentiment Analysis	19
3.2	User Interface of the Proposed Web Application	20
3.3	Polarity Distribution	21
4.1	Block Diagram of CNN Model	22
4.5.1	Training the model for sentiment analysis	23
4.5.2	Testing the model for sentiment analysis	24
4.6.1	Positive Sentiments	24
4.6.2	Negative Sentiments	25
4.7	Block diagram of CNN model for text extraction	25
4.10	Summary of The Text Extraction Model	27
4.11.1	Training the model for text extraction	28
4.11.2	Testing the model for text extraction	29
4.12.1	Precision Score	29
4.12.2	Recall Score	30
4.12.3	F1 Score	30
5.1.1	Component hierarchy of the application as seen in React Dev Tools	32
5.1.2.1	Next.js development server with fast refresh	33
5.1.2.2	Next.js build step	34
5.1.3.1	Responsive UI design with Tailwind CSS	36
5.1.4.1	VS Code Extensions	36
5.1.4.2	Prettier Config	37
5.1.4.3	NPM and Husky	37
5.2.2.1	Working process of Amazon Sagemaker	44
6.1	User Interface of Web Application	47
6.2.1	User uploading a textual image	47
6.2.2	Uploaded picture	48
6.2.3	Sentiment of extracted Text	48
6.3.1	Entering the URL of the image	49
6.3.2	Output of the image from the URL	49

LIST OF TABLES

TABLE NO.	TITLE	PAGE NO.
2.7	Comparison between different Deep Learning models	17

ABBREVIATIONS

CNN	-	Convolved Neural Network
ANN	-	Artificial Neural Network
SVM	-	Support Vector Machine
RAE	-	Recursive Auto-encoders Network
RNTN	-	Recursive Neural Tensor Network
MVRNN	-	Matrix-vector Recursive Neural Network
LSTM	-	Long Short-Term Memory
AdaRNN	-	Adaptive Recursive Neural Network
FFT	-	Fast Fourier Transform
PIL	-	Python Imaging Library
MLP	-	Multi-Layer Perception-based Deep Learning
NLP	-	Natural Language Processing
REST	-	Representative State Transfer
API	-	Application Programming Interface
AWS	-	Amazon Web Services
MIME	-	Multi-purpose Internet Mail Extension
BSD	-	Berkeley Source Distribution
WSGI	-	Web Server Gateway Interface

CHAPTER 1

INTRODUCTION

This chapter gives a brief introduction about the sentimental analysis on a textual image using deep learning technique. The aim is to deliver a user-friendly web application which predicts the sentiments of textual images through the application of natural language processing and visualising it in a web application.

1.1 PROPOSED WORK

The working of sentiment analysis on textual image project using Deep Learning, in which the machine learning models are implemented using CNN. As the "convolutional" filter (i.e., kernel) travels over the image, CNN extracts the significant aspects of the image, which has been widely employed on image datasets.

And for the sentiment classification if the input is presented as one-dimensional data, the same CNN function can be used in the text. The local information texts are retained while the filter goes over the text area, allowing the important features to be extracted. As a result, it's a good idea to use CNN for text classification. Here proposed is a fully functional web application, where a user can upload a textual image and our machine learning models in the backend will extract the text from the image and displays the sentiment of the text.

1.2 RELATED WORKS

The comparison between ANN (Artificial Neural Networks) and SVM (Support Vector Machines) done by Moraes[1] in document level sentiment classification shows that the ANN can produce better results compared to SVM.

Le and Mikolov [2] proposed an unsupervised learning algorithm, Paragraph Vector in order to overcome the weakness of Bag of Words. This algorithm learns the vector representations for sentences, documents and paragraphs.

Glorot [3] learned the domain adaptation problems for the sentiment classification. He proposed Stacked Denoising Autoencoder using sparse rectifier units, deep learning

system in which both the labelled and unlabelled data can be extracted using unsupervised text feature or representation.

A semi-supervised autoencoder which introduced by Zhai and Zhang [4] , for obtaining good document vectors considers the sentiments always in its learning stage for the sentiment classification. By deriving a discriminative loss function from its label information and by relaxing loss function to Bregman Divergence in autoencoder, the model tries to learn the task specific textual data representation.

BoW CNN , a CNN variant introduced by Johnson and Zhang [5] does the bag of words conversion in its convolutional layer itself. And a model which stores the sequential information of words through concatenating one hot vector of multiple words called Seq-CNN also introduced by them.

With the help of sentence relationships, to learn the document representations Tang [6] introduced a neural network model which learns the representations of sentences first using LSTM or CNN from the word embedding. And in document representations for sentiment classification the utilization of a GRU for adaptively encoding the semantics of the sentences and also their inherent relations.

In the review division Tang [7] used product representations and user representations. For providing better textual representations, those presentations can also store overall product qualities, individual user preferences like global trends.

At both word level and semantic level the product characteristics and global user preference taken into account of the model incorporated by Chen[8] which uses product information and user information through sentence and word attentions for the classification.

A cached LSTM model introduced by Xu[9] in order to capture overall semantic information inside a long text. And according to different forgetting rates in the model, the memory is divided into different groups.

For the document level sentiment rating and prediction of reviews, a hierarchical attention type network has been proposed by Yang[10]. In the model, it contains the

word level and the sentence level attention mechanisms. And this allows the model to provide nearly attention to individual sentences and words while making the document representation.

Yin[11] introduced a hierarchical and interactive attention-based model in which the prediction of the aspect sentiment rating is done in the document level. Moreover, the pseudo aspect questions and the documents are computed to learn the aspect aware representation of the document.

An attention based Long Short-Term Memory network designed by Zhou[12] for the classification of cross lingual sentiment at the document level consists of two LSTM for the representation of bilingual, and each of the networks of LSTM are also structured hierarchically.

An adversarial memory network proposed only for the classification of the cross-domain sentiment classification by Li[13] transfer its learning setting where the modelling is done on the data from the target domain and the source together.

For the classification in the sentence level, a semi supervised RAE(Recursive Autoencoders Network) proposed by Socher et al. [14] can be used to obtain the deducted vector representation of a sentence.

A MVRNN (Matrix-vector Recursive Neural Network) in which a tree structure matrix representation associated for each of the words is proposed by Socher [15]. And from an external parser the tree structure is obtained.

Socher [16] the publishers again introduced the RNTN (Recursive Neural Tensor Network) where, for the better and enhanced capture of element interactions, the compositional functions which are based on tensors are used.

A DCNN(Dynamic Computational Neural Network) particularly for semantic modelling of the sentences proposed by Kalchbrenner [17], uses the non-linear subsampling function technique by using the operator of K-Max pooling.

Kim[18] experimented with different variants named CNN-multichannel in which the multiple set of word embeddings are used, CNN non static in which all the word embeddings are finely tuned and pre trained, CNN static in which the word embeddings

are fixed and pre trained, CNN rand in which the word embeddings are randomly initialized in the sentence level sentiment classification.

A CharSCNN (Character to Sentence CNN) introduced by dos Santos and Gatti [19] extracts the relevant features from the sentences and words of different sizes by using two convolutional layers in order to perform sentiment analysis of the short texts.

During the compositional process by simulating the word interactions Wang [20] used LSTM for the twitter sentiment analysis. And compared to additive ones in the simple RNN (recurrent neural network) to produce accurate compositional results and to provide better flexibility, the multiplicative operations between word embeddings via structures of gates are used. And similar to the bidirectional, unidirectional Long Short-Term Memory can be extended to bidirectional Long Short-Term Memory[21] , through the hidden layer by allowing the bidirectional connections.

Wang [22] presented a regional CNN-LSTM model to predict text valence arousal ratings, which comprises of two parts: regional CNN and LSTM.

Wang [23] proposed a hybrid CNN and RNN architecture for sentiment categorization of short texts that takes use of CNN's coarse-grained local characteristics and RNN's long-distance dependencies.

Guggilla [24] introduced a deep neural network model based on LSTM and CNN that uses word2vec and linguistic embeddings for claim categorization (classifying sentences to be factual or feeling).

To improve phrase and sentence representation, Huang [25] advocated encoding syntactic information (e.g., part-of-speech tags) in a tree structured LSTM.

For fine-gained sentiment categorization of financial microblogs and news, Akhtar [26] suggested many multi-layer perceptron-based ensemble models.

Guan [27] used a CNN with poor monitoring to classify sentiment at the sentence (and aspect) level. It has a two-step learning process: initially, it learns a sentence representation that is weakly monitored by overall review scores, and then it fine-tunes it using sentence level labels.

Teng [28] provided a paradigm for sentiment categorization based on a multilingual dictionary. To identify the sentiment value of a phrase, bidirectional LSTM is used to learn the sentiment depth, strengthening, and contradiction of lexical sentiments.

Yu and Jiang [29] examined learning generic phrase embeddings for cross-domain sentence sentiment classification and created a neural network model with two different CNNs that simultaneously train two hidden feature representations from both labelled and unlabelled input.

By altering the deep semantic network representation of both users' posted tweets and their social ties, Zhao [30] proposed a persistent random walk network learning strategy for sentiment categorization of biased tweets.

Mishra [31] employed CNN to extract intellectual qualities from the eye-movement (or gaze) data of human readers reading the text and use them as augmented features for sentiment classification with literal characteristics.

For the challenge, Qian [32] provided a linguistically normalized LSTM. To effectively capture the sentiment response in sentences, the proposed model incorporates linguistic resources such as sentiment vocabulary, contradiction words, and intensity words into the LSTM.

For objective-dependent twitter sentiment categorization, Dong [33] proposed an Adaptive Recursive Neural Network (AdaRNN) that learns to propagate the sentiments of words towards the goal based on context and syntactic structure. It gives the SoftMax classifier the representation of the core node as features in order to forecast the split over classes.

Vo and Zhang [34] investigated aspect-based Twitter sentiment categorization utilising fine programmed features, which are additional characteristics discovered using unsupervised learning approaches. The research demonstrated how numerous embeddings, various pooling functions, and sentiment lexicons may be useful sources of information.

Tang [35] created Target-dependent LSTM (TD-LSTM) and Target-connection LSTM (TC-LSTM) to extend LSTM by accounting for the target, because LSTM can capture semantic interactions between the target and its context words in a more flexible manner. They treated the specified target as a feature and combined it with context information to classify aspect emotion.

For aspect level sentiment categorization, Ruder [36] advocated using a hierarchical and bidirectional LSTM model that can benefit both intra- and inter-sentence interactions. The suggested approach is language-independent due to its single reliance on sentences and their structures inside a review. Word embeddings are delivered into a bidirectional LSTM at the sentence level. The target embedding is concatenated with the final states of the forward and backward LSTMs and fed into a bidirectional overview LSTM. The forward and backward LSTM outputs are concatenated and input into a final layer, which outputs a probabilistic model across sentiments at each time step.

Zhang.[37] suggested a syllable level recognition technique to solve the problem of pooling functions, which do not explicitly describe tweet-level interpretation, based on the work of Dong [33] and Vo and Zhang [34]. This is demonstrated using two gated artificial neural networks. A bi-directional gated neural network is used to link the words in a tweet so that pooling functions can be applied across the convolution layer instead of words, which best reflects the target and its circumstances. Second, a three-way fortified neural network structure is used to analyse the relationship between the target reference and its surrounding contextual factors, overcoming limitations by using gated neural network structures to model the bounding tweet's linguistic features as well as the interaction between the surrounding contexts and the target. The bias of typical RNN towards the endpoints of a sequence has been demonstrated to be reduced by using gated neural networks.

Wang [38] suggested an attention-based LSTM technique with target embedding, which was shown to be a good way to induce the neural model to pay attention to the relevant section of a text. In reaction to a certain element, the attention mechanism is employed to push the model to pay attention to the crucial section of a sentence. To increase

performance of the classifier, Yang [39] developed two attention-based bidirectional LSTMs. The attention framework of Liu and Zhang [40] was expanded by distinguishing the attention gained from the left and right contexts of a particular target/aspect. They added numerous gates to better restrict their attention participation.

Tang [41] presented a sentiment classification at the aspect level end-to-end memory network that combines an attention mechanism with external memory to capture the relative importance of each background word to the given target aspect. This method carefully captures the importance of each surrounding word when inferring the emotion polarity of an aspect. The relevance degree and text representation are determined using multiple computational layers, each of which is a neural attention model over an external memory.

Lei [42] proposed using a neural network technique to derive pretexts (reasons) for customer reviews from input sequence. A generator and a decoder make up the model. The encoder converts any such text to a task-specific target vector, and the generator provides a distribution across plausible rationales (derived text). Each point of the target vector represents the answer or rating for the corresponding aspect in multi-aspect sentiment analysis.

Li [43] integrated the target identification task with the sentiment classification task to better characterise the aspect-sentiment interaction. They showed that sentiment recognition can be handled with an end-to-end machine learning framework that connects the two sub-tasks utilising deep neural networks. As a result, signals generated during target detection can be used in this fashion.

Ma [44] presented the Interactive Attention Network (IAN), which takes into account both target and contextual attention. That is, it employs two attention networks to identify the key words in the target articulation as well as the key words in the context.

To help enhance the mood of multifaceted settings, Chen [45] advocated using a recurrent attention network. To do so, they offer a model that employs a persistent attention structure and learns non-linear attention combinations in GRUs.

Tay [46] created a Dyadic Memory Network that uses either neural tensor compositions or holographic representations to simulate dyadic interactions between aspect and context.

Sanjeev Ahuja [47] utilised SentiWordNet to influence the overall polarization of book review tweets, based on tweet data obtained from WordNet. It takes into account word sentiment and is separated into positive, negative, and objective sentiment scores. When all of these scores are added together, the relative strength is determined. To improve the accuracy of the tweet data, a state-of-the-art technique has been implemented.

Rachana Bandana, [48] proposed the sentiment analysis with document level for the movie review classification applied by heterogeneous features. Positive and negative sentiment classification can be done by this method.

Purtata Bhoir, [49] used the subjectivity analysis with two methods as SentiWordNet and Naive Bayes. Among these Naive Bayes classifier is a probabilistic model theorem to calculate the probability with a class of individual sentences or text. Naive Bayes analyzed for extracting the feature and opinion from the public tweets and gives better solution compared to SentiWordNet.

M. Ali Fauzi, [50] shows the ensemble technique with Naive bayes for sentiment analysis of movies on Indonesian Twitter data. They came up with lexicon-based features which hold the percentage of positive and negative tweet words based on their part of speech.

The model introduces by V.K. Singh, R. Piryani, A. Uddin, [51] also used for highlighting the tweet products with a very short period of time used in different domains.

Using Naive Bayes Vasilija Uzunova, [52] introduced Macedonian film reviews sentiment analysis. With the help of sentiment polarity, they classified the film comment text and checked whether it is positive or negative based on opinion of the public reviews

In 2019 Benaddy et al. employed Deep Convolutional Neural Networks (DCNN) (see Appendix 1: Neural network terminologies) in order to improve the accuracy in

identifying Tifinagh characters [53]. This technique involves using a deep learning algorithm which can take in an input image, assign importance to various aspects/objects in the image and be able to differentiate one image from the other. This proposed system was tested on data set of approximately 25,000 and achieved the best recognition accuracy (99.10%) when compared to other established OCR methods such as the horizontal and vertical centreline or baseline of characters. However, this was an improvement of only 0.07% on the previously most accurate method of a combination of multiple classifiers with statistical features.

Roy et al. had already adopted and improved on this principle in 2017 when they employed a layer-wise technique of DCNN in order to improve the accuracy in identifying Bangla characters [54]. This technique DCNN incorporated with the layerwise training model, which is a multi-stage process that involves adding layers of convolutional and pooling processes followed by fully connected layers which contain multiple neurons, and applies the back-propagation algorithm to find the weights of importance. This layer layer-wise-trained DCNN produced results that improved upon standard DCNN nearly 10%.

In 2018 Jangid & Srivastava performed a similar study using Devanagari [55]. This layerwise-trained DCNN produced results that improved upon standard DCNN by up to 1.5% across sample sizes of up to almost 57,000 characters. However, as noted in 3.2.2, the additional resources required for the use of neural networks in conjunction with OCR can be considerable. Yet, there seems to be scarcity of research into the application of deep learning in OCR in regards to the performance costs of doing so, and whether this is even feasible on a mobile platform integrated into a mobile application that expects real-time results.

Yin et al. expanded on Jangid & Srivastava's researched further in 2019 using deep learning techniques to improve existing OCR approaches for recognising Chinese uppercase character by considering network weight, and test time [56]. Generally, the deeper the number of layers of the neural network and the more parameters, the more accurate the conclusions are, but accurate results mean more computing resources are consumed. In their study Yin et al. reduced overhead through the practice of 'pruning', or the removal of parameters that do not contribute significantly to the output. This is

achieved by identifying the most redundant neurons using an Average Percentage of Zeroes (ApoZ) algorithm and removing some of the unnecessary network neurons and retaining the weight parameters which are important to the network and reducing the parameters in order to reduce the computational complexity of the model. Using this method, accuracy decreased by 1.26% when compared with a standard Convolutional Neural Network (CNN) but achieved a network weight reduction of 96.5%.

Likewise, Valueva et al. researched two techniques in combination to reduce hardware costs in the implementation of CNN architecture [57]. In their application of a CNN, the neural network is divided into distinct hardware and software partitions to increase performance and reduce the cost of implementation resources. They combined this with novel residue number system in the hardware part of the convolutional layer of the CNN to implement the convolutional layer of the neural network. A residue numeral system is a numeral system representing integers by their values modulo several pairwise coprime integers called the moduli. This ‘multi-modular arithmetic’ system is widely used for computation with large integers, typically in linear algebra, because it provides faster computation than with the usual numeral systems, even when the time for converting between numeral systems is taken into account [58]. The implementation of these two methods in combination showed a reduction in hardware costs of 7.86% to 37.78% and reduced the average time of image recognition by 41.17%.

Tesseract is an open-source OCR released under the Apache License, originally developed by Hewlett-Packard as proprietary software between 1985 and 1995. It was released as open source in 2005 and development has been supported by Google since 2006 [59].

Before machine learning and deep learning became ubiquitous in OCR around the 2010s, Tesseract was considered one of the most accurate open-source OCR engines available at that time [60].

1.3 ADVANTAGE OVER PREVIOUS MODELS

- User friendly web application
- Tailor-made User Interface
- Realtime sentiment analysis on a textual image

CHAPTER 2

LITERATURE SURVEY

2.1 PRE-PROCESSING

2.1.1 The main purpose of text pre-processing was to remove the unwanted tags from the most highlighted texts, and removing the unwanted URLs, removing the unwanted links, spaces, and brackets by using those features using nltk toolkit. After removing these all the movie review will show the reviews in lowercase alphabets. These characteristics were utilised to train the movie review network data. The following steps are involved for pre-processing:

- The Tokenization process will split all the reviews in a text form into word form and it will be done for any character of the data, especially on space character.
- Word2Vec is an embedding model which maps the review words into vector format, and then each and every single tweet word is converted into respective numerical vector form which will be taken from the movie reviews. This Vector format is used for training the CNN and LSTM models and it will join the dimensions into the output array of data.
- Stop words is a collection of Natural Language processing which removes the unwanted sentences from the movie reviews or removing the repeated words which will be showing from the dataset. It will print the output of most occurring text words in an array format.
- The sequential model is a crucial procedure to rotate the numeric array representing data from the word2vec which embedding into a digitalized vector format, which is provided to any sort of neural network in a binary vector format. This format will be passed to build the further classification model of movie reviews.

2.2 TYPES OF SENTIMENT CLASSIFICATIONS

2.2.1 Document Level Sentiment Classification

The purpose of document-level sentiment classification is to automate the process of detecting whether a textual review indicates a favourable or negative sentiment on a

certain topic. It's a binary classification task in this case. It can also be framed as a regression challenge, such as inferring a review's total rating score from 1 to 5 stars. At this stage, the goal is to figure out what the document's general opinion is. At the document level, sentiment analysis implies that each document communicates ideas about a particular entity.

2.2.2 Sentence Level Sentiment Classification

Sentence-level sentiment analysis is one of the main directions in sentiment analysis area. The existing work on the task concentrated on recognizing the polarity of a sentence (e.g., positive, neutral, negative), according to semantic information learned from the textual content of sentences. Sentence representation given by neural networks is critical for both sentence and document sentiment classification. Because phrases are typically shorter than documents, syntactic and semantic data (such as parse trees, opinion lexicons, and part-of-speech tags) may be beneficial. Additional information, such as review ratings, social links, and cross-domain data, can also be considered.

2.2.3 Aspect Level Sentiment Classification

The goal of aspect-level sentiment categorization is to figure out how people feel about certain things based on surrounding phrases. The sentiment polarity of a target phrase in a sentence is determined using aspect-level sentiment analysis. Existing neural network models can help you figure out how to determine polarity. However, due to the limitations of training datasets, context relative position information for the target phrases is omitted. The accuracy of sentiment categorization can be improved by incorporating location features between words into the models.

Modelling a target's semantic relatedness to its surrounding context words is complex, making aspect level sentiment classification problematic. Context words influence the polarity of a sentence's sentiment toward the target in different ways. As a result, while developing learning models with neural networks, it is critical to capture semantic links between the target word and the context words.

2.3 OPTICAL CHARACTER RECOGNITION

One facet of image recognition technology is OCR, or the electronic or mechanical conversion of images of typed, handwritten or printed text into machine-encoded text,

whether from a scanned document, a photo of a document, a scene-photo or from subtitle text superimposed on an image [7]. After text is detected at a line/word level there are many methods that can be used to convert the text which generally come from three main approaches

- Classic Computer Vision Techniques (CCVTs), typically involving applying filters to make characters stand out from the background, contour detection to recognise the characters individually and image classification to identify the characters.
- Standard Deep Learning (StanDL), using an artificial neural network that combines multiple nonlinear processing layers that uses simple elements operating in parallel.
- Specialised Deep Learning (SpecDL) that uses such technologies as convolutional-recurrent neural networks, such as or Semi-Supervised End-to-End Scene Text Recognition (SEE) and Efficient Accurate Scene Text detector (EAST).

The first recorded uses of this technology can be traced as far back as the 19th century in reading devices for the visually impaired. In 1904 Emanuel Goldberg (1881 - 1970) developed a machine that read characters and converted them into standard telegraph code (and would later develop an OCR device for searching microfilm archives that would be acquired by IBM) , while around the same time Edmund Fournier d'Albe (1868 - 1933) developed a handheld scanner that, when moved across a printed page, produced an audio tone that corresponded to specific letters or characters.

2.4 DEEP LEARNING

Deep learning, also known as deep structured learning or differential programming, is a far more recent development in technology, the concept of which was first presented as a part of a broader family of machine learning methods based on artificial neural networks that imitates the workings of the human brain in processing data and creating patterns for use in decision making in 1986 . Machine learning can be loosely described as a self-adaptive algorithm that gets increasingly improved analysis and patterns with

experience or with newly added data, and deep learning utilises a hierarchical level of artificial neural networks to carry out the process of machine learning.

The artificial neural networks are built like the human brain, with neuron nodes connected together like a web. While traditional programs build analysis with data in a linear way, the hierarchical function of deep learning systems enables machines to process data with a nonlinear approach. Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction.

Deep learning and especially OCR are by no means new technologies and thus have been the subject of a substantial amount of academic research in a wide variety of applications, both individually and in collaboration.

2.5 TYPES OF TEXT EXTRACTION

2.5.1 Region based Method:

Region-based method uses the properties of the color or gray scale in the text region or their differences to the corresponding properties of the background. They are based on the fact that there is very little variation of color within text and this color is sufficiently distinct from its immediate background. Text can be obtained by thresholding the image at intensity level between the text color and that of its immediate background. This method is not robust to complex background and is further divided into two sub-approaches: connected component (CC) and edge based.

2.5.2 CC based Method:

CC-based methods use a bottom- up approach by grouping small components into successively larger components until all regions are identified in the image. A geometrical analysis is required to merge the text components using the spatial arrangement of those components so as to filter out non-text components and the boundaries of the text regions are marked. This method locates text quickly but fails for complex background.

2.5.3 Edge based Method:

Edges are a reliable feature of text regardless of color/intensity, layout, orientations, etc. Edge based method is focused on high contrast between the text and the background.

The three distinguishing characteristics of text embedded in images that can be used for detecting text are edge strength, density and the orientation variance. Edge based text extraction algorithm is a general-purpose method, which can quickly and effectively localize and extract the text from both document and indoor/ outdoor images. This method is not robust for handling large size text.

2.5.4 Texture based Method:

This method uses the fact that text in images has discrete textural properties that distinguish them from the background. The techniques based on Gabor filters, Wavelet, Fast Fourier transform (FFT), spatial variance, etc. are used to detect the textual properties of the text region in the image. This method is able to detect the text in the complex background. The only drawback of this method is large computational complexity in texture classification stage.

2.5.5 Morphological based Method:

Mathematical morphology is a topological and geometrical based method for image analysis. Morphological feature extraction techniques have been efficiently applied to character recognition and document analysis. It is used to extract important text contrast features from the processed images. These features are invariant against various geometrical image changes like translation, rotation, and scaling. Even after the lightning condition or text color is changed, the feature still can be maintained. This method works robustly under different image alterations.

2.6 DEEP LEARNING ARCHITECTURES FOR IMAGE EXTRACTION AND TEXT CLASSIFICATION

In this research, the three main classification algorithms are addressed:

- Deep Neural Networks (DNNs)
- Convolutional Neural Networks (CNNs)
- Recurrent Neural Networks (RNNs)

2.7 PERFORMANCE COMPARISON

Table 2.7 Comparison between different Deep Learning models

Type of Machine Learning Network	Architecture of Network	Advantages	Disadvantages
Deep Neural Networks (DNNs)	Its architecture has more than two layers, which allows coping with a complex non-linear relationship. It is often used in classification and regression analysis.	It has wide application and produces high accuracy.	The training process is complex because it has more hidden layers and the error is propagated back to the previous layer and becomes very small. The learning process is also very slow.
Convolutional Neural Networks (CNNs)	They have good performance for 2-D data. They use Convolution filters to transform 2-D data into 3-D data.	Produce high efficiency and having a faster learning.	All data must be tagged in order for the categorization to work.
Recurrent Neural Networks (RNNs)	They have the capability of learning large sequences.	Sequential events may be learnt quickly, and temporal dependence can be represented.	Requires big databases and falls into issues because of gradient vanishing.

Here Convolutional Neural Networks (CNNs) are compared with the Deep Neural Networks (DNNs) and Recurrent Neural Networks (RNNs). Table 2.4 summarizes the comparison of their network architecture, advantages and disadvantages.

2.8 MACHINE LEARNING MODEL USED

Convolutional Neural Network:

CNN is a class of deep, feed-forward artificial neural networks (where connections between nodes do not form a cycle) & use a variation of multilayer perceptron designed to require minimal pre-processing. These are inspired by animal visual cortex. Convolutional neural network is used to find patterns in an image. They do that by convoluting over an image and looking for patterns. In the first few layers of CNNs the network can identify lines and corners, but can then pass these patterns down through our neural net and start recognizing more complex features as get deeper.

CHAPTER 3

PROPOSED SYSTEM

3.1 PROPOSED BLOCK DIAGRAM

Figure 3.1 depicts the suggested design's block diagram.

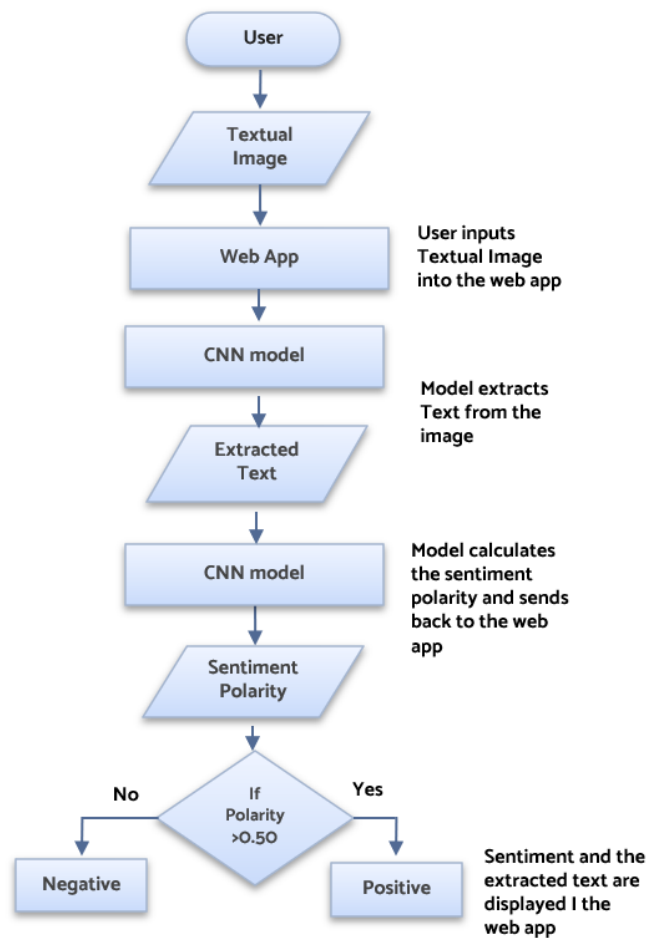


Fig 3.1 Proposed block diagram of Sentiment Analysis

3.2 WEB APPLICATION

A Web user interface, often known as a Web app, allows a user to interact with content or software on a remote server using a Web browser. The content or Web page is downloaded from the Web server, and the user interacts with it using a client, which is a Web browser. Figure 3.2 depicts the user interface of our proposed web application.

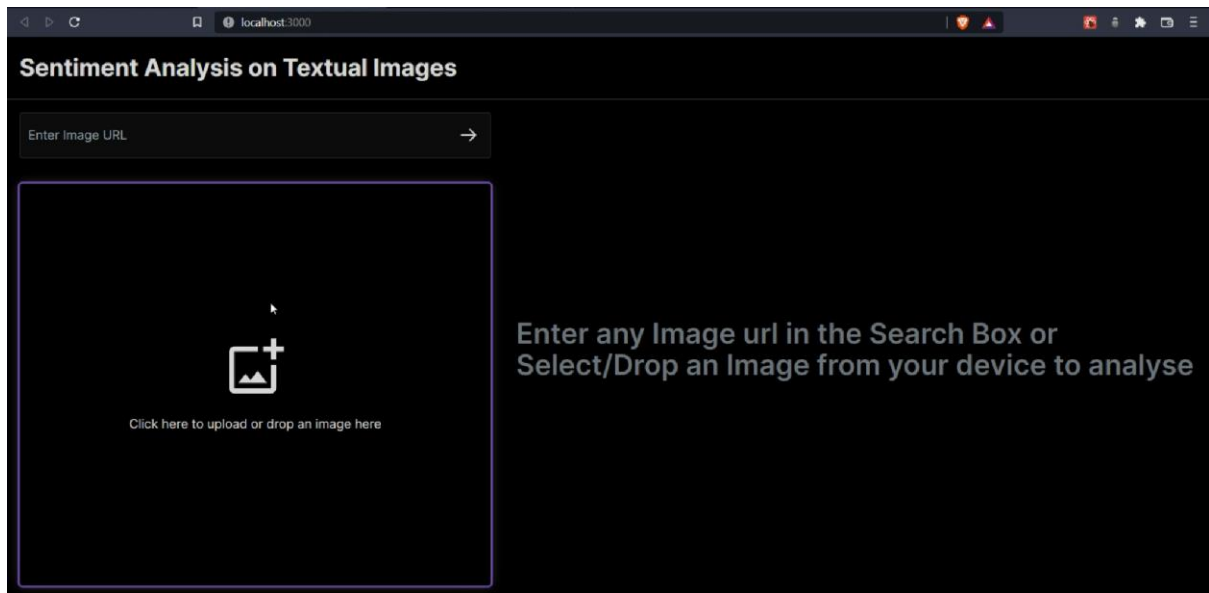


Fig. 3.2 User Interface of our Proposed Web Application

3.3 CONVOLUTIONAL NEURAL NETWORKS

Convolutional Neural Network (CNN) is a multilayer perception-based deep learning approach (MLP). CNN's three basic layers are convolution, pooling, and fully connected layer. The convolution layer, which consists of a series of independent filters, is the fundamental CNN building block. The pooling layer's goal is to simplify the representation of convolved layers. The vector from several convolutions and pooling operations on the multilayer perceptron are known as fully connected layers that are used to do a particular task such as classification.

Convolutional Neural Network is a set of neural networks that provides enough information for image processing, picture classification, and speech recognition. The Sequence of movie review data contains different lengths and keras initially gives importance for the tweet inputs where it could be in vectorized form and all the inputs to be in same length. We'll pad all of those movie review inputs afterwards, and the final product will have a review length of 100 words. For this instance, The use pad sequence function. The defining Embedding layer for training the neural networks.

The Embedding layer has a vocabulary size of 92768 and an input length of 100, and it will use a 300-dimensional embedding space can now see the Embedding layer shows the output in a 100*300 matrix form. Then there are Convolutional Neural Networks

with varying levels of dimensions such as Conv1D, Conv2D, and Conv3D. This study describes Conv1D, which is used to build one-dimensional convolutional neural networks for movie review time series classification and training the review data, which is more technically difficult and time consuming. It is very useful for calculating the text features from the in-build length of huge amount of data. From this 1D network flow, it holds the filter of 128 feature vectors and it fits the number of output filters used in the convolution network.

With 1D convolution layer, a kernel size applied to kernel weight matrix of 5 which contains 5 feature vectors and gives the size of the convolutional window. Then, for 1D temporary data, the following layer GlobalMaxpooling1D is used, which uses the most vector space in comparison to the step-by-step dimensions of the neural network.

3.4 POLARITY DISTRIBUTION

- The emotive aspect of an opinion is determined by the wording.
- The result of sentiment analysis can be derived for each entity in a sentence, document, or sentence in textual data.
- The polarity of sentiment can be classified as positive or negative.

Figure 3.5 depicts how the polarity score is distributed in the suggested model.



Fig. 3.3 Polarity Distribution

CHAPTER 4

TRAINING AND TESTING THE CNN MODELS

4.1 BLOCK DIAGRAM OF SENTIMENT CNN MODEL

Figure 4.1 depicts the suggested CNN model's block diagram.

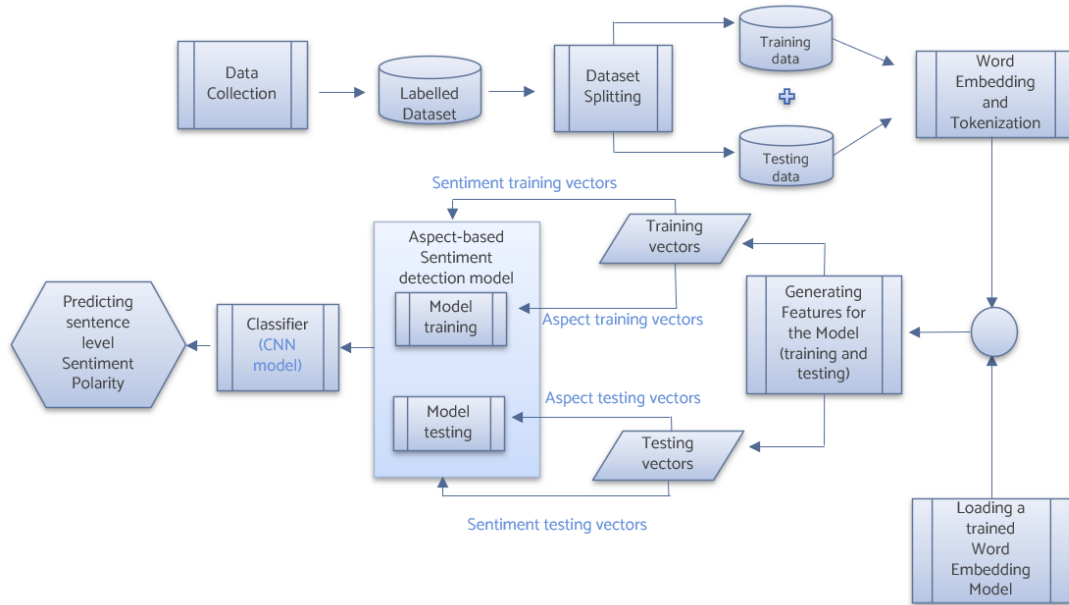


Fig. 4.1 Block diagram of CNN model

4.2 IMDB REVIEW DATASET

For the model dataset, the IMDB movie reviews dataset was used, which consists of 50,000 movie reviews. In this case the dataset is divided into two. For the training and validation 25,000 data and for the testing purpose used 25,000 data. The evaluated the classifier performance by cross-validating classifier parameters on the training set, using CNN throughout all the cases.

4.3 WORD EMBEDDING

In a word embedding, each word in the lexicon is represented as a real valued vector in a high-dimensional space. The vectors are learned in such a way that words with comparable meanings in the vector space have similar representations (close in the

vector space). This is a more expressive representation of text than more traditional methods such as bag-of-words, which ignores relationships between words or tokens, or bigram and trigram approaches, which force associations between words or tokens.

4.4 PRE-TRAINED WORD EMBEDDING MODEL

Embeddings that has been trained and they are learned in one task and then used to solve a comparable challenge. These embeddings are created by training them on big datasets, saving them, and then using them to solve other problems. The goal is to produce a representation of words that encapsulates their meanings, semantic links, and the various situations in which they are employed. The numerical representation of a text is what word embeddings are.

Pretrained word embeddings are a critical element in today's Natural Language Processing (NLP) discipline. In this proposed model, Stanford's GloVe, and the primary idea behind the GloVe word embedding is to use Global Statistics to derive the link between the words.

4.5 TRAINING AND TESTING THE CNN MODEL

The embedding vectors come from various sources, such as GloVe, and from the training process itself. During the training phase, the embedding layer was created and utilised to transform all word tokens, including the token for unseen words, to numeric values. Using the 25,000 IMDB movie review dataset, The trained and validated the CNN model.

Epoch: 01	Train Loss: 0.007	Train Acc: 99.85%	Val. Loss: 0.489	Val. Acc: 87.87%
Epoch: 02	Train Loss: 0.007	Train Acc: 99.87%	Val. Loss: 0.509	Val. Acc: 87.84%
Epoch: 03	Train Loss: 0.007	Train Acc: 99.86%	Val. Loss: 0.559	Val. Acc: 87.54%
Epoch: 04	Train Loss: 0.005	Train Acc: 99.86%	Val. Loss: 0.573	Val. Acc: 87.66%
Epoch: 05	Train Loss: 0.004	Train Acc: 99.93%	Val. Loss: 0.616	Val. Acc: 87.27%
Epoch: 06	Train Loss: 0.004	Train Acc: 99.91%	Val. Loss: 0.609	Val. Acc: 87.79%
Epoch: 07	Train Loss: 0.003	Train Acc: 99.95%	Val. Loss: 0.641	Val. Acc: 87.63%
Epoch: 08	Train Loss: 0.004	Train Acc: 99.91%	Val. Loss: 0.691	Val. Acc: 87.47%
Epoch: 09	Train Loss: 0.004	Train Acc: 99.91%	Val. Loss: 0.687	Val. Acc: 87.31%
Epoch: 10	Train Loss: 0.003	Train Acc: 99.92%	Val. Loss: 0.713	Val. Acc: 87.42%

Fig. 4.5.1 Training the model for sentiment analysis

As shown in the figure 4.5 by using ten epochs, The got 99.91% and 87.42% accuracy for the training and the validation respectively.

```
| Test Loss: 0.781 | Test Acc: 85.94% |
```

Fig. 4.5.2 Testing the model for sentiment analysis

The tested various combinations with our datasets and discovered that the CNN model with two-dimensional convolution layer was the best enhanced model for our task. The tested the CNN model using the 25,000 IMDB movie review dataset and obtained a testing accuracy of 85.94 percent with a loss percentage of 0.781, as shown in figure 4.5.2.

4.6 SENTIMENT MODEL OUTPUT

4.6.1 Positive Sentiments

The emotion underlying social media and other online remarks is measured using social sentiment. A good sentiment is when customers are enthusiastic, pleased, or delighted, as seen in Figure 4.6.1, which is one of the situations that the model is evaluated on.

```
Example for Positive Sentiments

[ ] predict_sentiment("You did a good job ,but I expect a lot more from you in the Future")
0.7446607947349548

[ ] predict_sentiment("Your love for dogs is Overwhelming")
0.6072326898574829

[ ] predict_sentiment("They are feeling happy after the win")
0.6497109532356262
```

Fig. 4.6.1 Positive sentiments

4.6.2 Negative Sentiments

A negative sentiment is when customers are angry, annoyed, or frustrated, as seen in Figure 4.6.2, which is one of the situations that the model is evaluated on.

```
Example for Negative Sentiments

✓ [37] predict_sentiment("The Pricing structure is Frustratingly Expensive")
      0.4676802158355713

✓ [78] predict_sentiment("I regret my mistake everyday")
      0.4158170223236084

✓ [86] predict_sentiment("They hated the movie")
      0.42607027292251587
```

Fig. 4.6.2 Negative sentiments

4.7 BLOCK DIAGRAM OF CNN MODEL FOR TEXT EXTRACTION

Figure 4.7 depicts the suggested CNN model's block diagram.

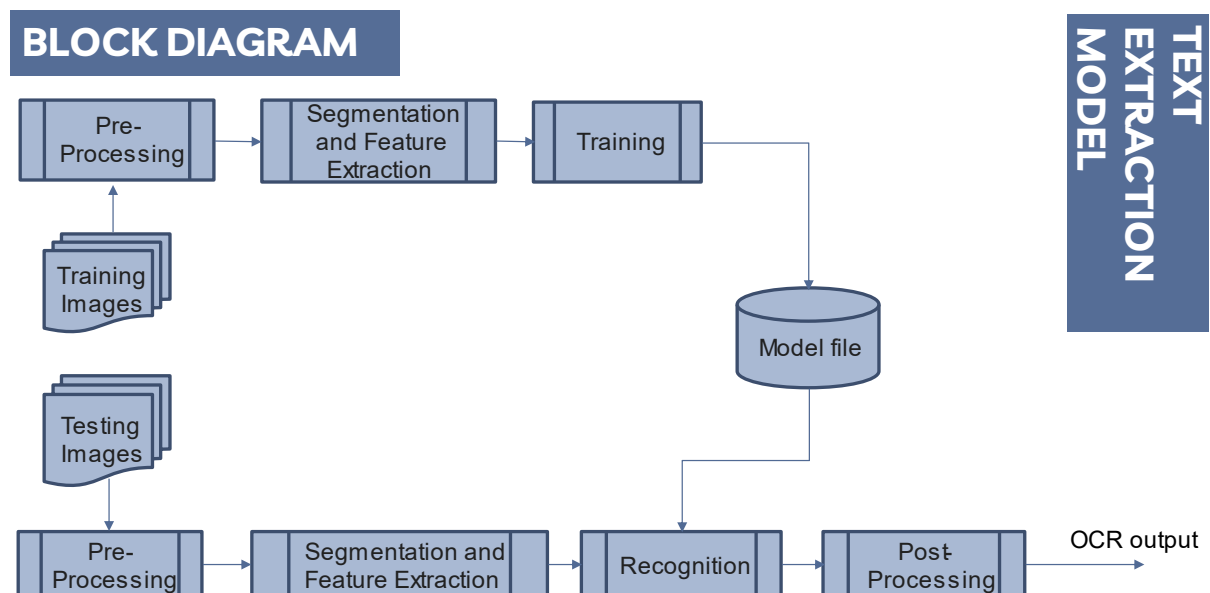


Fig. 4.7 Block diagram of CNN model for text extraction

4.8 IMAGE DATASET FOR TEXT EXTRACTION MODEL

Dataset Name: Chars74k – EnglishFnt

Characters from computer fonts with 4 variations (combinations of italic, bold and normal). 62 classes (A-Z,a-z,0-9) contain 1016 images on each class and a total of 62992 images. For training The used 50220 images (810 images from each class) and for testing The used 12772 images (206 images form each class).

4.9 STEPS OF TEXT EXTRACTION

4.9.1 Pre-processing an input image

This OCR step includes simplification, detection of meaningful edges, and defining the outline of the text characters. This is a common step for any task that has an image recognition component in it. If you're interested, we've discussed a similar approach in more detail in our article on image recognition.

4.9.2 Detection of the text

This step of an OCR project requires drawing a bounding box around the pieces of text found on the image. A few of the legacy techniques used for this step include SSD, real-time (YOLO) and region-based detectors, sliding window technique, Mask R-CNN, EAST detector, etc. You can read more on some of them in this article. (On a side note, machine learning models designed specifically for image recognition and detection usually do not perform as well for OCR tasks in terms of accuracy and data loss due to the specific nature of the text and its basic features.)

4.9.3 Recognition of the text

The final OCR step is to recognize the text that was put in the bounding boxes. For this task, one or a combination of convolutional and recurrent neural networks and attention mechanisms is frequently used. Sometimes this step may also include the interpretation step, which is characteristic for more complex OCR tasks like handwriting recognition and IDC.

CNNs are one of the best techniques to use for deep learning OCR for the step of text detection. Convolution layers are commonly used for image classification tasks due to their efficiency in feature extraction. They allow detecting the meaningful edges in an image and (on a higher level) shapes and complex objects. Compared to fully-connected layers, for example, convolutional layers decrease the complexity of a machine learning algorithm by reusing the pattern-detection filters throughout an image.

4.10 SUMMARY OF THE TEXT EXTRACTION MODEL

```
#view model summary
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 125, 125, 32)	544
max_pooling2d (MaxPooling2D)	(None, 61, 61, 32)	0
conv2d_1 (Conv2D)	(None, 58, 58, 64)	32832
max_pooling2d_1 (MaxPooling2D)	(None, 28, 28, 64)	0
flatten (Flatten)	(None, 50176)	0
dense (Dense)	(None, 310)	15554870
dense_1 (Dense)	(None, 62)	19282
Total params: 15,607,528		
Trainable params: 15,607,528		
Non-trainable params: 0		

Fig. 4.10 Summery of the model

Figure 4.10 shows the summery of the CNN model used for the text extraction. The have used the sequential model for the model making, as it is having the linear stack of layers. It can be first initialized and then The add layers using add method or The can add all layers at init stage.

The layers added are as follows:

- Conv2D
- MaxPooling2D
- Flatten
- Dense

4.11 TRAINING AND TESTING THE CNN MODEL FOR TEXT EXTRACTION

```
Epoch 1/20
272/272 [=====] - ETA: 0s - loss: 0.9530 - accuracy: 0.7512WARNING:
272/272 [=====] - 1127s 4s/step - loss: 0.9530 - accuracy: 0.7512 -
Epoch 2/20
272/272 [=====] - 1037s 4s/step - loss: 0.3704 - accuracy: 0.8765
Epoch 3/20
272/272 [=====] - 1037s 4s/step - loss: 0.2508 - accuracy: 0.9108
Epoch 4/20
272/272 [=====] - 1038s 4s/step - loss: 0.2012 - accuracy: 0.9263
Epoch 5/20
272/272 [=====] - 1035s 4s/step - loss: 0.1680 - accuracy: 0.9374
Epoch 6/20
272/272 [=====] - 1026s 4s/step - loss: 0.1487 - accuracy: 0.9437
Epoch 7/20
272/272 [=====] - 1030s 4s/step - loss: 0.1284 - accuracy: 0.9507
Epoch 8/20
272/272 [=====] - 1015s 4s/step - loss: 0.1187 - accuracy: 0.9570
Epoch 9/20
272/272 [=====] - 1023s 4s/step - loss: 0.1088 - accuracy: 0.9592
Epoch 10/20
272/272 [=====] - 1043s 4s/step - loss: 0.1008 - accuracy: 0.9622
Epoch 11/20
272/272 [=====] - 1011s 4s/step - loss: 0.0885 - accuracy: 0.9670
Epoch 12/20
272/272 [=====] - 1014s 4s/step - loss: 0.0803 - accuracy: 0.9697
Epoch 13/20
272/272 [=====] - 1017s 4s/step - loss: 0.0766 - accuracy: 0.9716
Epoch 14/20
272/272 [=====] - 1011s 4s/step - loss: 0.0702 - accuracy: 0.9737
Epoch 15/20
272/272 [=====] - 1010s 4s/step - loss: 0.0695 - accuracy: 0.9744
Epoch 15/20
272/272 [=====] - 1010s 4s/step - loss: 0.0695 - accuracy: 0.9744
Epoch 16/20
272/272 [=====] - 1012s 4s/step - loss: 0.0617 - accuracy: 0.9767
Epoch 17/20
272/272 [=====] - 1005s 4s/step - loss: 0.0596 - accuracy: 0.9778
Epoch 18/20
272/272 [=====] - 1020s 4s/step - loss: 0.0538 - accuracy: 0.9792
Epoch 19/20
272/272 [=====] - 1016s 4s/step - loss: 0.0465 - accuracy: 0.9812
Epoch 20/20
272/272 [=====] - 1016s 4s/step - loss: 0.0486 - accuracy: 0.9814
```

Fig. 4.11.1 Training the model for text extraction

As shown in the figure 4.11.1 by using 20 epochs, The got 98.1% and 92.8% accuracy for the training and the validation respectively(fig.4.11.2).

```
97/97 [=====] - 25s 254ms/step - loss: 0.2442 - accuracy: 0.9290
Model performance:
loss for test dataset is : 0.24417369067668915
accuracy for test dataset is : 0.9289883375167847
```

Fig. 4.11.2 Testing the model for text extraction

4.12 PERFORMANCE MATRICES

Every machine learning task can be broken down to either *Regression* or *Classification*, just like the performance metrics. There are dozens of metrics for both problems, but we're going to discuss popular ones along with what information they provide about model performance. It's important to know how your model sees your data!

4.12.1 Precision Score

The precision is the ratio $tp / (tp + fp)$ where tp is the number of true positives and fp the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative.

The best value is 1 and the worst value is 0. Figure 4.6.1 shows the precision score of our model.

```
precision_score(y_train_2, y_train_pred) # This is my precision score
0.7669421487603306
```

Fig. 4.12.1 Precision Score

4.12.2 Recall Score

The recall is the ratio $tp / (tp + fn)$ where tp is the number of true positives and fn the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.

The best value is 1 and the worst value is 0. Figure 4.6.2 shows the recall score of our model.

```
recall_score(y_train_2, y_train_pred) # This is my recall score  
0.7986230636833046
```

Fig. 4.12.2 Recall Score

4.12.3 F1 Score

The F1 score can be interpreted as a harmonic mean of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal. The formula for the F1 score is:

$$F1 = 2 * (\text{precision} * \text{recall}) / (\text{precision} + \text{recall})$$

In the multi-class and multi-label case, this is the average of the F1 score of each class with weighting depending on the average parameter. Figure 4.6.3 shows the F1 score of our model.

```
f1_score(y_train_2, y_train_pred)  
0.7824620573355817
```

Fig. 4.12.3 F1 Score

CHAPTER 5

TEXT EXTRACTION FROM IMAGE AND ITS SENTIMENT ANALYSIS

In this proposed project the analyzed sentiments from the textual images with the help of deep learning algorithm, and the sentiments are visualized using web application. Our proposed project consists of backed and front end i.e., a full stack application. The user interface is referred to as "front-end," while the server, program, and database that operate behind the scenes to give information to the user are referred to as "back-end".

5.1 FRONT END

Everything you view and interact with while using a web application is called Front End. Front End part of any application is crucial because it is what ultimately your users will be presented with. There are several frameworks/libraries to build front end apps. The have used the most popular React library along with its meta-framework Next.js.

5.1.1 React

React is a JavaScript Library for building user interfaces. It helps us to build a complex UI by splitting the final desired design into smaller reusable components. These components have their own state and purpose. UI views and their event handlers are kept together. UI views are written in JSX, which is a syntax extension to JavaScript. At build time JSX templates are compiled to normal JavaScript objects.

React uses Virtual Document Object Model (Virtual DOM). It creates an in-memory data-structure cache, computes the resulting differences, and then updates the browser's displayed DOM efficiently. This process is called reconciliation. This selective rendering provides a major performance boost. It saves the effort of recalculating the CSS style, layout for the page and rendering for the entire page.

Hooks in React are functions that allow us to "hook into" React state and lifecycle features from function components. React has a few built-in Hooks such as useState, useEffect, and so on. The may even write our own Hooks to reuse stateful behaviour across components. This makes it simple to share Hooks across several components.

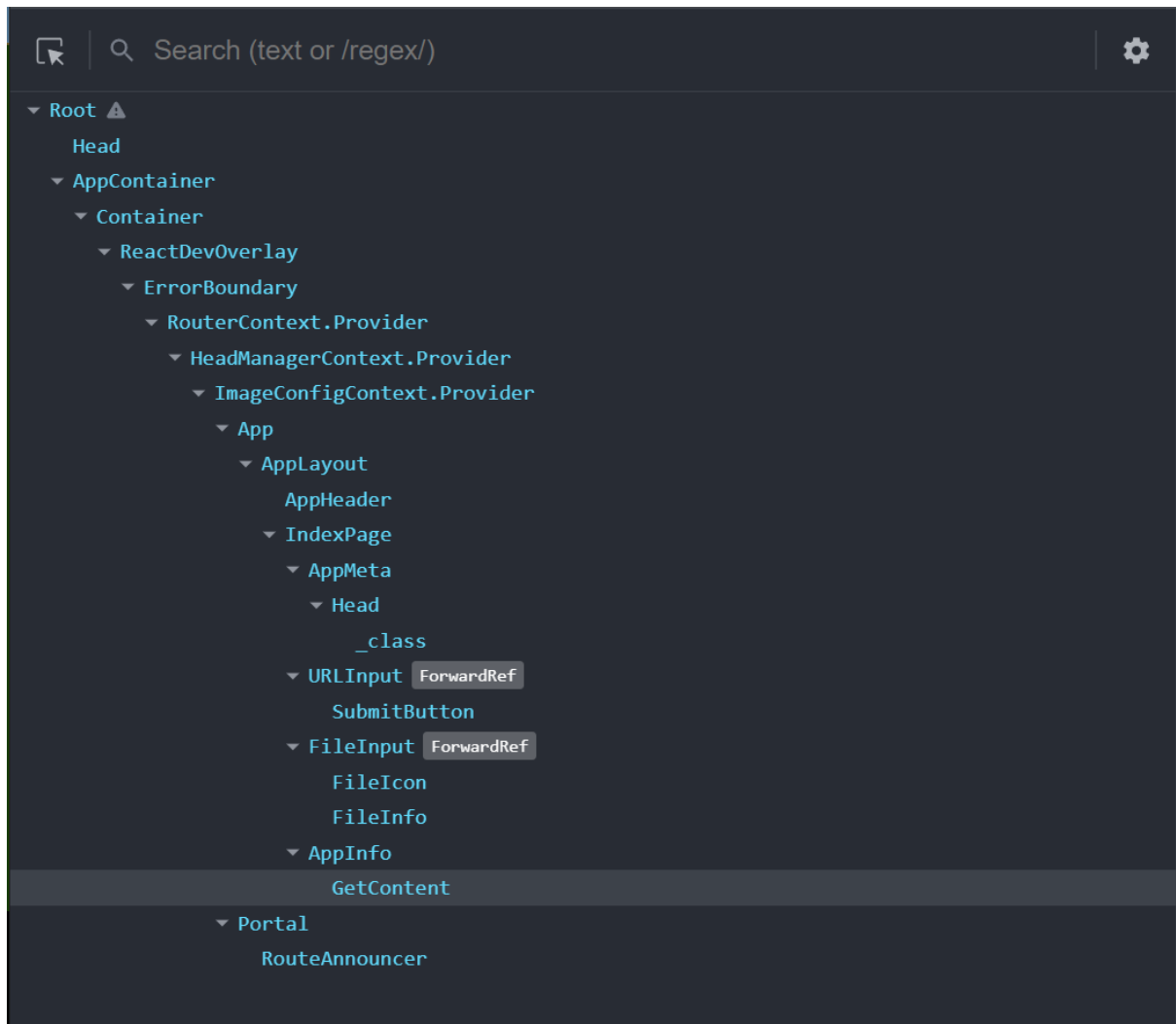


figure 5.1.1 Component hierarchy of our application as seen in React Dev Tools

As shown in the figure 5.1.1, The have split our app into small components which have their own state and purpose. The use React to write our UI logic and use ReactDOM to render it to the browser screen. React Hooks like useState and useEffect, takes care of user interactivity and fetching the data from the backend. React takes care of composing the JSX templates and ReactDOM takes care of rendering the result to the screen.

5.1.2 Next.js

Next.js is a meta-framework for React. React, like most other Single Page Application (SPA) frameworks/libraries, intentionally has a minimal API surface. So, meta-frameworks provide us with all the extra features needed to build a full web application.

Next.js gives us the best developer experience with features like, Server-Side Rendering, Static Site Generation, TypeScript support, smart bundling, routing, route prefetching, API routes, fast refresh etc. Next.js provides all of these features and more out of the box without any lower-level configuration needed from the developer side. Next.js provides us with first class support for serverless functions and is designed for modern serverless platforms.

```
vishn@vishnu:~/codes/sentiment-analysis-textual-images$ npm run dev

> my-app@0.1.0 dev
> next dev

ready - started server on 0.0.0.0:3000, url: http://localhost:3000
event - compiled client and server successfully in 448 ms (169 modules)
wait - compiling...
event - compiled successfully in 725 ms (137 modules)
wait - compiling / (client and server)...
event - compiled client and server successfully in 228 ms (225 modules)
wait - compiling /api/ving...
event - compiled client and server successfully in 134 ms (235 modules)
wait - compiling...
event - compiled client and server successfully in 227 ms (235 modules)
wait - compiling...
event - compiled client and server successfully in 289 ms (222 modules)
wait - compiling...
event - compiled successfully in 136 ms (180 modules)
█
```

figure 5.1.2.1 Next.js development server with fast refresh

Next.js provides us with a performant and accessible router, which instead of reloading the full page, renders the DOM with the appropriate component matching the route, which ultimately gives a fast and snappy user experience, thus our web app feels like a true native app. Application routes are configured by creating JavaScript/TypeScript

files under the pages folder and exporting a default React component from the file can access the route details in other components using the useRouter react hook.

Since, in single page application client-side JavaScript takes care of all stuff like routing, state management, data fetching etc, it takes a heavy toll in bundle size that was delivered to our end users. Fortunately, Next.js by default statically generates our pages at build time and serves them as Static HTML along with JavaScript assets, React will then hydrate the application into a full SPA on the client side. So, users will be presented with the application without staring at spinners and blank pages. This gives a superior user experience. Next.js also provides us a way to lazy load our assets. Next.js bundles all our CSS and JavaScript code with Webpack 5, with tree shaking support.

```
vishn@vishnu:~/codes/sentiment-analysis-textual-images$ npm run build

> my-app@0.1.0 build
> next build

info - Checking validity of types
info - Creating an optimized production build
info - Compiled successfully
info - Collecting page data
info - Generating static pages (3/3)
info - Finalizing page optimization

Page                                Size      First Load JS
┌ ○ /                               9.23 kB   96.3 kB
├   /_app                           0 B       87.1 kB
├ ○ /404                             194 B     87.3 kB
├ λ /api/sentiment                   0 B       87.1 kB
├ λ /api/uimg                        0 B       87.1 kB
├ + First Load JS shared by all    87.1 kB
├   ├── chunks/framework-5f4595e5518b5600.js 42 kB
├   ├── chunks/main-a054bbf31fb90f6a.js    27.6 kB
├   ├── chunks/pages/_app-08e59e3a25cbba30.js 16.6 kB
├   ├── chunks/webpack-5752944655d749a0.js  840 B
├   └── css/92cb6310c702bcf0.css           2.83 kB

λ (Server)  server-side renders at runtime (uses getInitialProps or getServerSideProps)
○ (Static)  automatically rendered as static HTML (uses no initial props)
```

figure 5.1.2.2 Next.js build step

Next.js also helps us to write serverless functions for our app. Similar to page components, these functions are created simply by exporting a default JavaScript/TypeScript function from the API folder which is present inside the pages

folder. This code is not shipped to the client, instead deployed on the CDN, and our application can make calls to these API routes and do the necessary.

The use Next.js to build our entire Front End. So, Next.js takes care of compiling and bundling our code. Our app has one main route, which provides the visitor with a form to enter image URL or select an image from the user device to analyse the text in the image. This image or image URL depending on which one the user has provided in the form is sent to the function that we have deployed as Next.js API route, which responds back with the sentiment.

5.1.3 Tailwind CSS

Tailwind CSS is a utility-first CSS framework, which provides low-level utility CSS classes to style our applications without leaving our view templates, in our case, JSX. These utility classes are named after their purpose. Tailwind CSS' utility classes also follow a well-engineered design system for spacing, font-sizes, shadows, colors and all other properties needed to style our application, so that our final component styles are consistent with one another.

Tailwind CSS makes it extremely easy to customize the included design system to adapt to our custom design. In our case, instead of using the provided color palette, we used a different one. Tailwind CSS makes it super simple and intuitive to build responsive design, that is adapting the application layout and styles according to the size of the user device. For this, it uses the same approach as that of tailwind variants which are used to style elements based on their CSS pseudo state - hover, focus, active, checked etc.

The latest version of Tailwind CSS ships with a new and improved Just in Time engine which crawls through our view templates (in our case JSX) and generates only the utility classes used in our project. This ensures that the final CSS files are smaller in size and only the classes that are used in the application are shipped to the browser. This also allows us to use arbitrary values for certain properties that are only used for a few times in our application, without having to add them and bloat our whole app configuration.

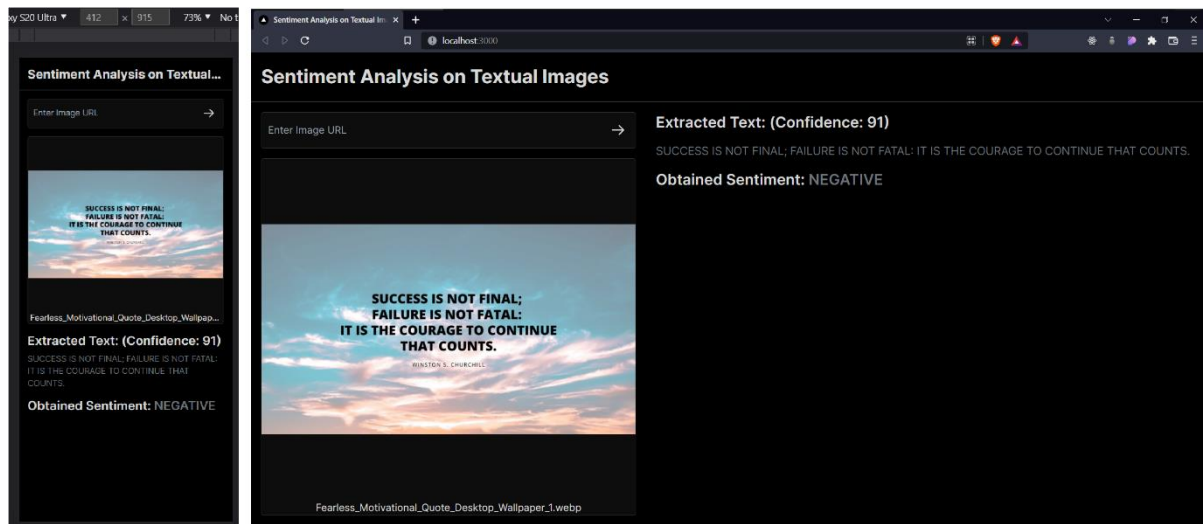


figure. 5.1.3.1 Responsive UI design with Tailwind CSS

Our entire application uses Tailwind CSS for styling. As shown in figure 5.1.3.1 our application changes its layout according to the user device size. This is made possible through Tailwind CSS' screen variants, which conditionally applies the classes that are attached after the screen size' shortcut prefix. These classes use CSS media queries to conditionally apply the styles.

5.1.4 Other Developer Utilities

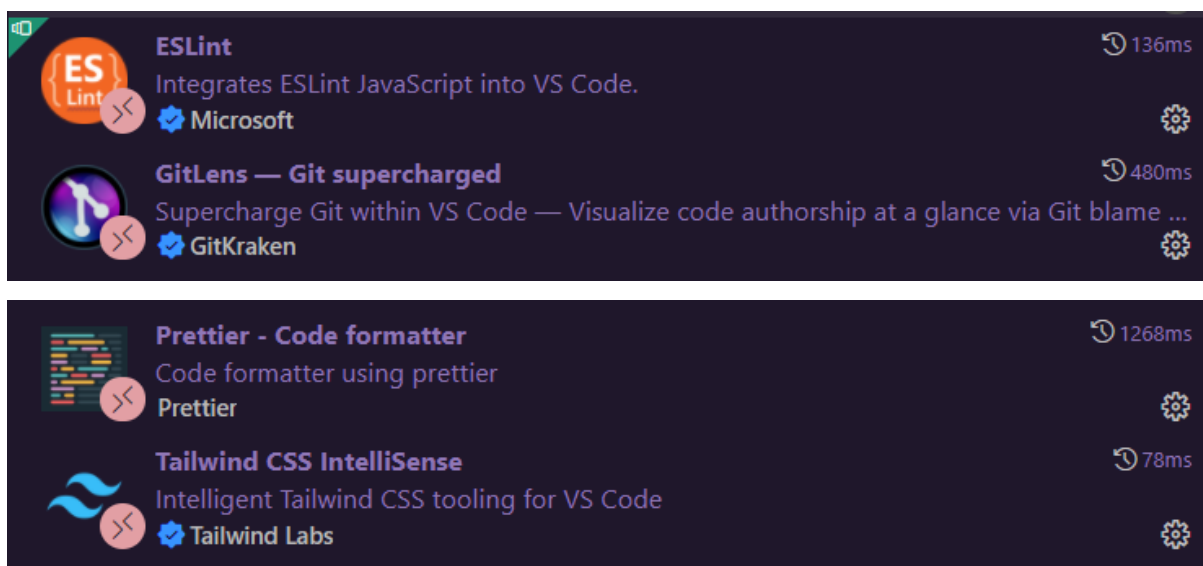


figure 5.1.4.1 VSCode extensions

The use VSCode as our primary code editor and its integrated terminal for development.
The use the extensions shown above.

```
{
  "semi": true,
  "tabWidth": 2,
  "printWidth": 80,
  "singleQuote": true,
  "trailingComma": "all",
  "arrowParens": "avoid",
  "bracketSpacing": true
}
```

figure 5.1.4.2 Prettier config

The use ESLint and Prettier for linting and formatting our code, to ensure that code styles and conventions are consistent throughout our application. The above picture shows our prettier configuration.

The use NPM for managing the application dependencies and to run development, build and other utility scripts.

```
vishn@vishnu:~/codes/sentiment-analysis-textual-images$ npm i
> my-app@0.1.0 prepare
> husky install

husky - Git hooks installed

up to date, audited 346 packages in 1s

81 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

figure 5.1.4.3 NPM and husky

The configured Git hooks via Husky to run TypeScript checker, ESLint and Prettier before every commit.

5.2 BACKEND

A server, an application, and a database make up a website's back end. A back-end developer creates and maintains the technology that enables the components that allow the website's user-facing side to exist in the first place.

5.2.1 Python

Python is a high-level general-purpose programming language that is interpreted. Its design concept emphasises code readability by employing heavy indentation. Its language components and object-oriented approach are intended to assist programmers in writing concise, logical code for small and large-scale projects alike.

Python is garbage-collected and dynamically typed. It is compatible with a wide range of programming paradigms, including structured (particularly procedural), object-oriented, and functional programming. Because of its extensive standard library, it is frequently referred to as a "batteries included" language.

Python's huge standard library, which is frequently recognised as one of its greatest features, provides tools suitable for a wide range of applications.

Many common formats and protocols, such as MIME and HTTP, are supported for Internet-

facing applications. It comes with modules for constructing graphical user interfaces, connecting to relational databases, producing pseudorandom numbers, arithmetic with arbitrary-precision decimals, manipulating regular expressions, and unit testing.

Some elements of the standard library are specified (for example, the Web Server Gateway Interface (WSGI) implementation `wsgiref` adheres to PEP 333), however the majority of modules are not.

5.2.1.1 spaCy

spaCy is an open-source natural language processing software package written in Python and Cython. The library is released under the MIT licence, and its primary creators are Matthew Honnibal and Ines Montani, the founders of the software firm Explosion.

Some of the features of spaCy are Non-destructive tokenization, Built-in support for trainable pipeline components such as Named entity recognition, Part-of-speech tagging, dependency parsing, Text classification, Entity Linking etc, Production-ready training system, Built-in syntax and named entity visualizers, simple model packaging, deployment, and workflow management.

5.2.1.2 PyTorch

PyTorch is a Python-based open-source machine learning library based on the Torch library, which was mostly developed by Facebook's AI Research division for applications such as computer vision and natural language processing. Its free software published under the Modified BSD licence.

The torch package was utilised because it includes data structures for multidimensional tensors and defines mathematical operations on these tensors.

To import the general data loaders, abstractions, and iterators for text (including vocabulary and word vectors) torchtext, The utilised torchtext.legacy.

5.2.1.3 Pandas

pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license.

5.2.1.4 PIL (Python Imaging Library)

Python Imaging Library is a free and open-source additional library for the Python programming language that adds support for opening, manipulating, and saving many different image file formats. It is available for Windows, Mac OS X and Linux.

5.2.1.4.1 Image:

The most important class in the Python Imaging Library is the Image class, defined in the module with the same name. The can create instances of this class in several ways; either by loading images from files, processing other images, or creating images from scratch..

5.2.1.5 NumPy:

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, Fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.

5.2.1.6 Math:

Python provides the math module to deal with such calculations. Math module provides functions to deal with both basic operations such as addition (+), subtraction (-), multiplication (*), division (/) and advance operations like trigonometric, logarithmic, exponential functions.

5.2.1.7 Scikit-learn:

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

5.2.1.7.1 sklearn.metrics:

The sklearn.metrics module implements functions assessing prediction error for specific purposes. These metrics are detailed in sections on Classification metrics, Multilabel ranking metrics, Regression metrics and Clustering metrics.

5.2.1.7.2 sklearn.metrics.precision_score:

To Compute the precision score The use precision_score function from sklearn.metrics module. The precision is the ratio $tp / (tp + fp)$ where tp is the number of true positives and fp the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative. The best value is 1 and the worst value is 0.

5.2.1.7.3 sklearn.metrics.recall_score

To Compute the recall score The use recall_score from sklearn.metrics module. The recall is the ratio $tp / (tp + fn)$ where tp is the number of true positives and fn the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.

5.2.1.7.4 sklearn.metrics.f1_score :

To Compute the F1 score, also known as balanced F-score or F-measure. The F1 score can be interpreted as a harmonic mean of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal.

The formula for the F1 score is:

$$F1 = 2 * (precision * recall) / (precision + recall)$$

5.2.1.8 shutil:

Python shutil module provides the facility to perform the high-level file operation. It can operate with the file object and offers us the ability of copy and remove the files. It handles the low-level semantic such creating and closing file objects after performing all operations.

5.2.1.8.1 shutil.copyfile

`shutil.copy()` method in Python is used to copy the content of source file to destination file or directory. It also preserves the file's permission mode but other metadata of the file like the file's creation and modification times is not preserved.

Syntax:

```
shutil.copyfile(source, destination, *, follow_symlinks = True)
```

5.2.1.9 keras :

Keras is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear & actionable error messages. It also has extensive documentation and developer guides.

5.2.1.9.1 Sequential:

The core idea of Sequential API is simply arranging the Keras layers in a sequential order and so, it is called Sequential API. Most of the ANN also has layers in sequential order and the data flows from one layer to another layer in the given order until the data finally reaches the output layer.

5.2.1.9.2 Keras layer:

Layers are the basic building blocks of neural networks in Keras. A layer consists of a tensor-in tensor-out computation function (the layer's call method) and some state, held in TensorFlow variables (the layer's weights).

5.2.1.9.2.1 Dense layer:

The dense layer is a neural network layer that is connected deeply, which means each neuron in the dense layer receives input from all neurons of its previous layer. The dense layer is found to be the most commonly used layer in the models. In the background, the dense layer performs a matrix-vector multiplication. The values used in the matrix are actually parameters that can be trained and updated with the help of backpropagation. The output generated by the dense layer is an 'm' dimensional vector.

Thus, dense layer is basically used for changing the dimensions of the vector. Dense layers also applies operations like rotation, scaling, translation on the vector.

5.2.1.9.2.2 Flatten:

Flatten is used to flatten the input. For example, if flatten is applied to layer having input shape as (batch size, 2,2), then the output shape of the layer will be (batch size, 4).

5.2.1.10 seaborn:

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

5.2.1.11 sys module:

The sys module in Python provides various functions and variables that are used to manipulate different parts of the Python runtime environment. It allows operating on the interpreter as it provides access to the variables and functions that interact strongly with the interpreter.

5.2.1.12 OS Module:

The OS module in Python provides functions for interacting with the operating system. OS comes under Python's standard utility modules. This module provides a portable way of using operating system-dependent functionality.

5.2.2 Amazon Web Services

AWS (Amazon Web Services) is a comprehensive, evolving cloud computing platform provided by Amazon that includes a mixture of infrastructure as a service (IaaS), platform as a service (PaaS) and packaged software as a service (SaaS) offerings. AWS services can offer an organization tool such as compute power, database storage and content delivery services.

5.2.2.1 Amazon SageMaker

Amazon SageMaker, a cloud machine-learning platform, was introduced in November 2017. SageMaker lets developers create, train, and deploy machine learning models on

the cloud. SageMaker may also be used by developers to deploy ML models on embedded systems and edge devices.

One of SageMaker's finest characteristics is its modular architecture can train elsewhere and only use SageMaker for deployment if you choose to train your model and leverage its hyper-parameter tweaking capability if you like. This is one of the best features of SageMaker.

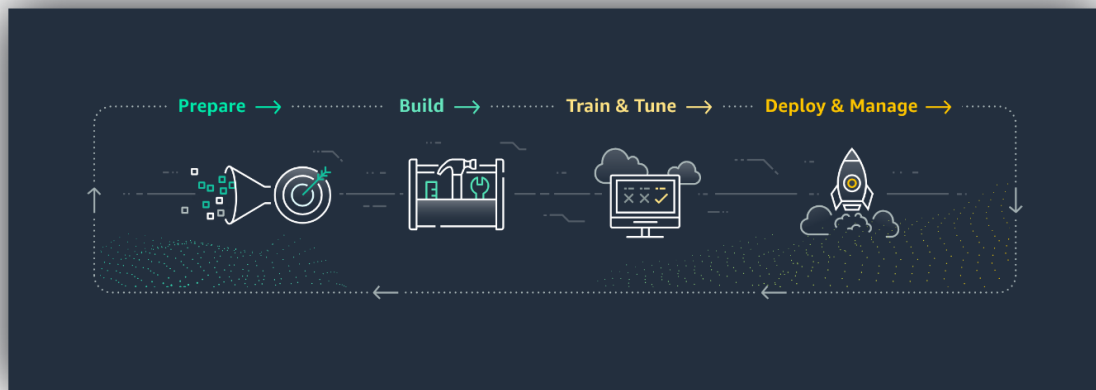


figure 5.2.2.1 Working process of Amazon SageMaker.

5.2.2.2 Working of Amazon SageMaker

As The can see from the figure 5.2.2.1, the working goes like this. First the step is preparing the model, the model is sent to Ground Truth i.e., Label training data for machine learning, then, Data Wrangler i.e., Aggregate and prepare data for machine learning, Processing Built-in Python, BYOR/Spark. Then the data is set to Feature Store, update, retrieve, and share features.

Then the model is Clarify, i.e., detect bias and understand model predictions. These are the working in process going on first step i.e., preparing the model.

On the second step, the built model in the algorithm is optimized, Test and prototype on our local machine, automatically created machine learning models with full visibility and JumpStarted i.e., Pre-built solutions for common use cases.

Next step is train and tune the model, here the distribute the model, capture and compare every step, automatically model is tuned and debugged.

This model then goes to the last step that is deployment, here the maintained and checked the accuracy of the model and deployed.

5.2.2.3 API gateway

API is an abbreviation for Application Programming Interface. This programme acts as a "middleman" between two apps that desire to connect with one another. Any requests you make are sent to the server first, and any responses you get follow the same path. This bridges the gap between the two apps (the one on your phone and the company's software) while also adding an extra layer of protection. When you use a social networking app, for example, you are interacting with an API. Without immediate touch, all potential hazards offered by one to the other are mitigated.

The API provides a list of methods the two applications can use to communicate. They include:

- GET for retrieving data.
- POST for creating data.
- PUT for updating data.
- DELETE for removing data.

5.2.2.4 REST API

A REST API (sometimes spelled RESTful API) is a form of application programming interface (API or web API) that corresponds to the limits of the REST architectural style and allows interaction with RESTful web services. The acronym REST stands for "representative state transfer."

A REST API functions in much the same manner as a webpage. The HTTP protocol is used by a client to submit a request to a server, and the server responds with data. Using the REST API, a transaction is divided down into discrete components. Every component focuses on a different part of a transaction. It is a versatile development

technique due to its flexibility. When a developer asks the Twitter API to obtain a user's object (a resource), the API will provide the user's current state, their name, followers, and shared tweets. Because of API integration, this is achievable. A REST API in between the model and the web application to request and get the data or in other words to make a way to communicate between the web application and the model.

5.2.2.5 AWS API gateway

Amazon API Gateway is a fully managed service that allows developers to effortlessly create, publish, maintain, monitor, and secure APIs of any size. APIs are the "front door" via which apps gain access to data, business logic, or functionality from your backend services. API Gateway enables the creation of RESTful APIs and WebSocket APIs for real-time two-way communication applications. API Gateway supports containerized and serverless workloads, as well as web applications.

Amazon API Gateway handles all aspects of accepting and processing hundreds of thousands of concurrent API calls, such as traffic management, CORS compliance, authorization and access control, throttling, monitoring, and API version management. API Gateway offers no minimum fees or initial charges. A must pay for the API calls get as well as the amount of data transferred out, and with the API Gateway tiered pricing model, The may be able to reduce our expenses as our API usage rises.

5.2.2.6 Senti API

In this proposed model a REST API and AWS API gateway was used to integrate our API with the machine learning models which is deployed in the sagemaker connects the web application.

CHAPTER 6

RESULT AND DISCUSSION

6.1 USER INTERFACE

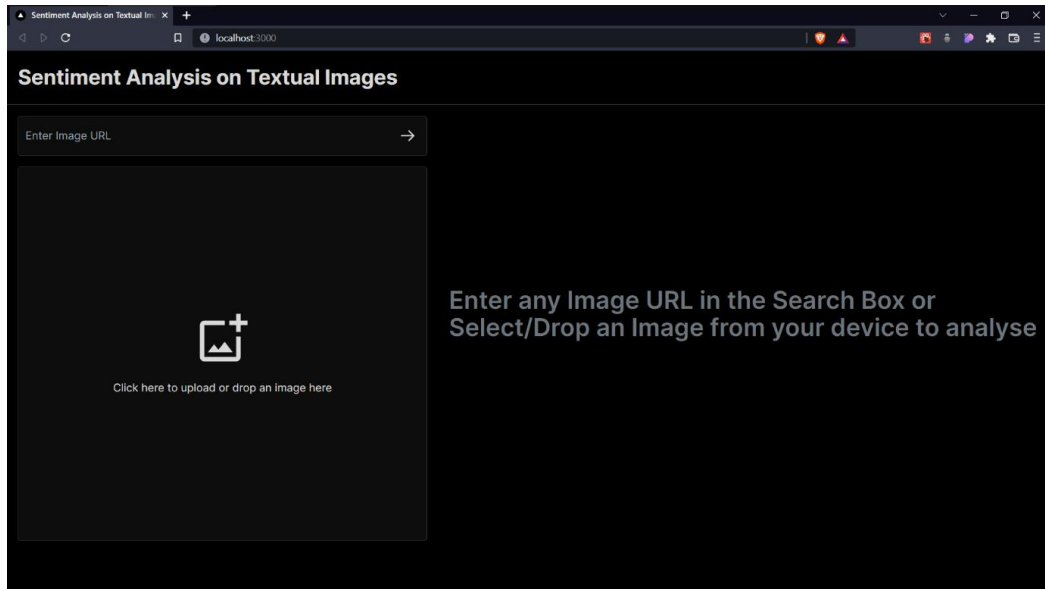


Figure:6.1 UI of web application

This (figure:6.1) is the web app's user interface, here the user can upload the textual image as file and can also provide the URL of the image.

6.2 UPLOADING A TEXTUAL IMAGE

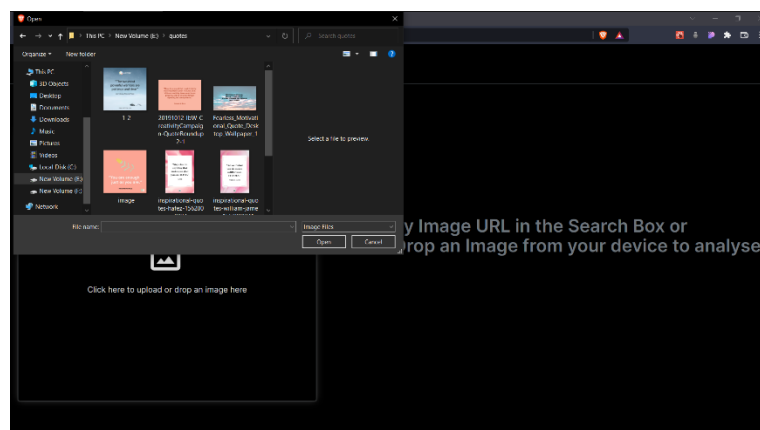


Figure 6.2.1 User uploading a textual image

Figure 6.2.1 shows that the user is uploading a textual image into the web application, figure 6.2.2 shows the uploaded picture and figure 6.2.3 is the output of the textual image which is having the extracted text from the image and sentiment of it.

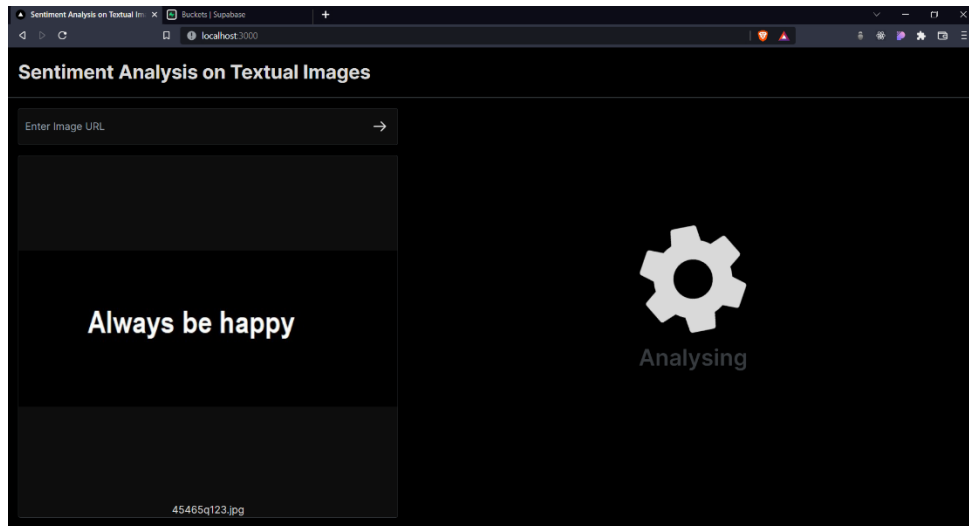


Figure 6.2.2 Uploaded picture

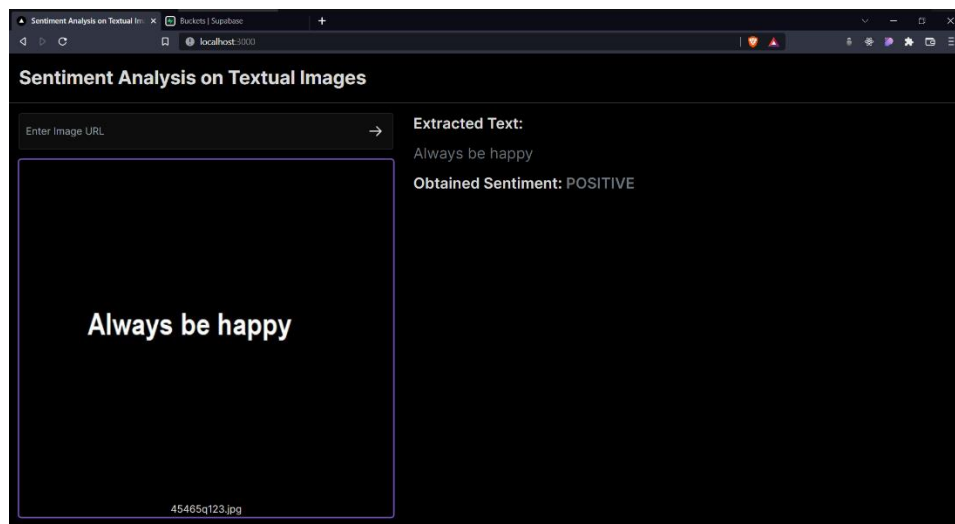


Figure 6.2.3 Sentiment of extracted Text

6.3 UPLOADING THE LINK OF A TEXTUAL IMAGE

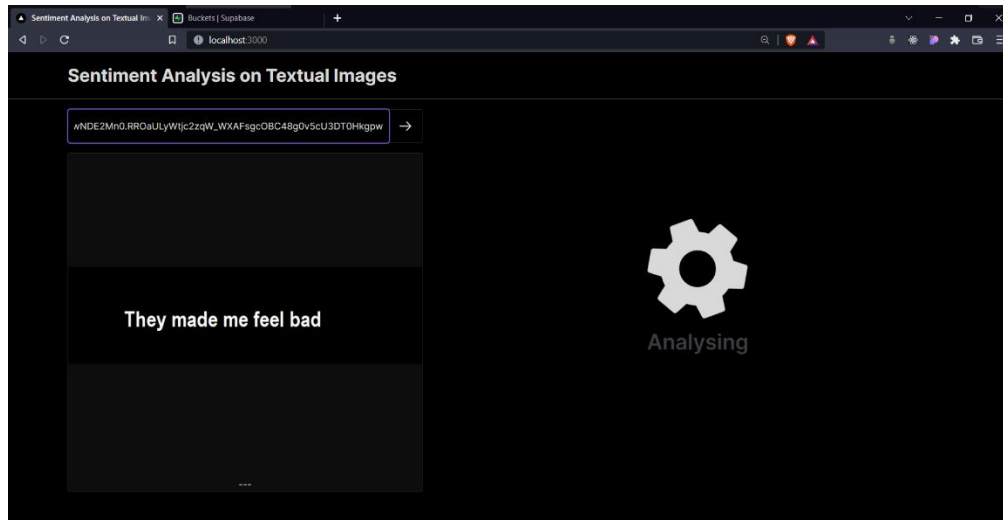


Figure 6.3.1 Entering the URL of the image

Figure 6.3.1 shows that the user is entering the URL of a textual image into the web application and figure 6.3.2 is the output of the textual image which is having the extracted text from the image and sentiment of it.

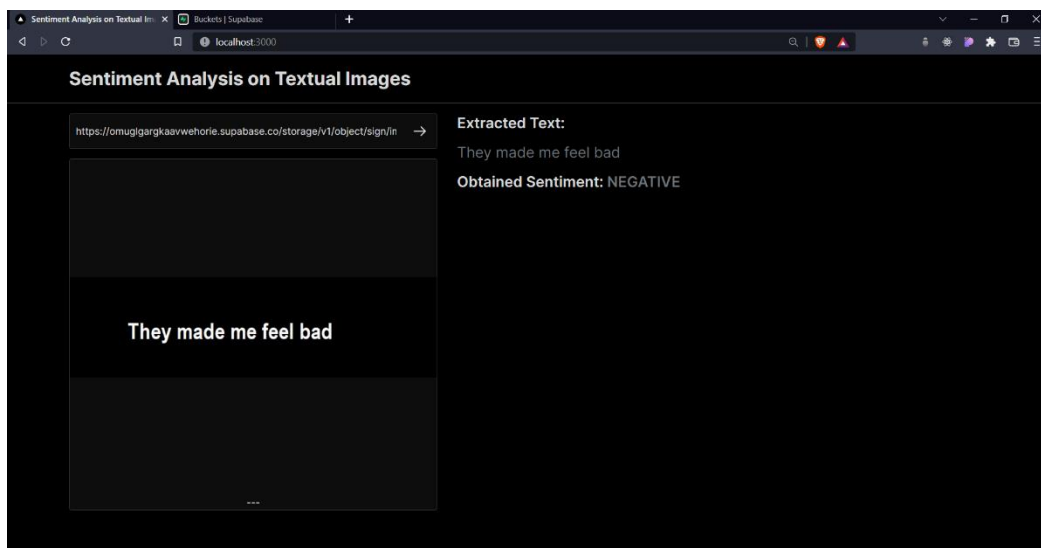


Figure 6.3.2 Output of the image from the URL

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

Sentiment analysis, often known as opinion mining, is a field of study that investigates people's feelings, attitudes, and emotions about certain things. In this project the sentiment polarity classification from textual images is accomplished. Sentence level sentiment categorization was the strategy followed. This classifies sentiments on the basis of different categories i.e., positive and mixed sentiment. Optical character recognition or optical character reader is the electronic or mechanical conversion of images of typed, handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a scene-photo or from subtitle text superimposed on an image. On average, the accuracy of our proposed model was 92 percent. The analysed results were depicted on a webpage after classification.

7.2 FUTURE SCOPE

Sentiment analysis is incredibly strong and has a wide variety of applications. The capacity to derive insights from social data is becoming increasingly popular among businesses throughout the world.

7.2.1 Social media monitoring

The most common use of a sentiment analysis system is social media monitoring. Companies may monitor brand perception by monitoring reviews and mentions on social media sites like posts, news, and comments. Social media monitoring enables us to discover areas that require our attention as well as immediately reply to specific remarks.

7.2.2 Brand Monitoring

Brand monitoring is one of the most prominent applications of sentiment analysis in business. Bad reviews may spread quickly on the internet, and the longer us to leave

them, the worse the problem will get. Sentiment analysis tools will alert you to negative brand references as soon as they occur.

7.2.3 Product Analysis

Using sentiment analysis, a user can examine product statistics to keep track of what works and what doesn't. The most enticing benefit of tracking product analytics is that the input provided by customers is genuine. They could offer adjustments to a product based on the customer reviews and make changes that would be beneficial for both the parties and utilise their feedback to modify a feature that is profitable.

CHAPTER 8

REFERENCES

- [1] Moraes R, Valiati J.F, Neto W.P. Document-level sentiment classification: an empirical comparison between SVM and ANN. *Expert Systems with Applications*. 2013.
- [2] Le Q, Mikolov T. Distributed representations of sentences and documents. In *Proceedings of the International Conference on Machine Learning (ICML 2014)*, 2014.
- [3] Glorot X, Bordes A, Bengio Y. Domain adaption for large-scale sentiment classification: a deep learning approach. In *Proceedings of the International Conference on Machine Learning (ICML 2011)*, 2011.
- [4] Zhai S, Zhongfei (Mark) Zhang. Semisupervised autoencoder for sentiment analysis. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI 2016)*, 2016.
- [5] Johnson R, Zhang T. Effective use of word order for text categorization with convolutional neural networks. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2015)*, 2015.
- [6] Tang D, Qin B, Liu T. Document modelling with gated recurrent neural network for sentiment classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, 2015.
- [7] Tang D, Qin B, Liu T. Learning semantic representations of users and products for document level sentiment classification. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL 2015)*, 2015.
- [8] Chen H, Sun M, Tu C, Lin Y, and Liu Z. Neural sentiment classification with user and product attention. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, 2016.

- [9] Dou ZY. Capturing user and product Information for document level sentiment analysis with deep memory network. In Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP 2017), 2017.
- [10] Xu J, Chen D, Qiu X, and Huang X. Cached long short-term memory neural networks for document-level sentiment classification. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2016), 2016.
- [11] Yang Z, Yang D, Dyer C, He X, Smola AJ, and Hovy EH. For document classification, hierarchical attention networks are used. The Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2016) is published in the Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2016).
- [12] Zhou X, Wan X, Xiao J. Attention-based LSTM network for cross-lingual sentiment classification. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2016), 2016.
- [13] Li Z, Zhang Y, Wei Y, Wu Y, and Yang Q. End-to-end adversarial memory network for cross-domain sentiment classification. In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2017), 2017.
- [14] Socher R, Pennington J, Huang E.H, Ng A.Y, and Manning C.D. Semi-supervised recursive autoencoders for predicting sentiment distributions. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2011), 2011.
- [15] Socher R, Huval B, Manning C.D, and Ng A.Y. Semantic compositionality through recursive matrix-vector spaces. In Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP 2012), 2012.
- [16] Socher R, Perelygin A, Wu J. Y, Chuang J, Manning C.D, Ng A. Y, and Potts C. Recursive deep models for semantic compositionality over a sentiment treebank. In

Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP 2013), 2013.

[17] Kalchbrenner N, Grefenstette E, Blunsom P. A convolutional neural network for modelling sentences. In Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL 2014), 2014.

[18] Kim Y. Convolutional neural networks for sentence classification. In Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL 2014), 2014.

[19] dos Santos, C. N., Gatti M. Deep convolutional neural networks for sentiment analysis for short texts. In Proceedings of the International Conference on Computational Linguistics (COLING 2014), 2014.

[20] Wang X, Liu Y, Sun C, Wang B, and Wang X. Predicting polarities of tweets by composing word embeddings with long short-term memory. In Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL 2015), 2015.

[21] Graves A, Schmidhuber J. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Networks, 2005.

[22] Wang J, Yu L-C, Lai R.K., and Zhang X. Dimensional sentiment analysis using a regional CNN-LSTM model. In Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL 2016), 2016.

[23] Wang X, Jiang W, Luo Z. Combination of convolutional and recurrent neural network for sentiment analysis of short texts. In Proceedings of the International Conference on Computational Linguistics (COLING 2016), 2016.

[24] Guggilla C, Miller T, Gurevych I. CNN-and LSTM-based claim classification in online user comments. In Proceedings of the International Conference on Computational Linguistics (COLING 2016), 2016.

[25] Huang M, Qian Q, Zhu X. Encoding syntactic knowledge in neural networks for sentiment classification. ACM Transactions on Information Systems, 2017.

- [26] Akhtar MS, Kumar A, Ghosal D, Ekbal A, and Bhattacharyya P. A multilayer perceptron based ensemble technique for fine-grained financial sentiment analysis. In Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP 2017), 2017.
- [27] Guan Z, Chen L, Zhao W, Zheng Y, Tan S, and Cai D. Weakly-supervised deep learning for customer review sentiment classification. In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2016), 2016.
- [28] Teng Z, Vo D-T, and Zhang Y. Context-sensitive lexicon features for neural sentiment analysis. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2016), 2016.
- [29] Yu J, Jiang J. Learning sentence embeddings with auxiliary tasks for cross-domain sentiment classification. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2016), 2016.
- [30] Zhao Z, Lu H, Cai D, He X, Zhuang Y. Microblog sentiment classification via recurrent random walk network learning. In Proceedings of the Internal Joint Conference on Artificial Intelligence (IJCAI 2017), 2017.
- [31] Mishra A, Dey K, Bhattacharyya P. Learning cognitive features from gaze data for sentiment and sarcasm classification using convolutional neural network. In Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL 2017), 2017.
- [32] Qian Q, Huang M, Lei J, and Zhu X. Linguistically regularized LSTM for sentiment classification. In Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL 2017), 2017.
- [33] Dong L, Wei F, Tan C, Tang D, Zhou M, and Xu K. Adaptive recursive neural network for target-dependent Twitter sentiment classification. In Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL 2014), 2014.

- [34] Vo D-T, Zhang Y. Target-dependent twitter sentiment classification with rich automatic features. In Proceedings of the Internal Joint Conference on Artificial Intelligence (IJCAI 2015), 2015.
- [35] Tang D, Qin B, Feng X, and Liu T. Effective LSTMs for target-dependent sentiment classification. In Proceedings of the International Conference on Computational Linguistics (COLING 2016), 2016.
- [36] Ruder S, Ghaffari P, Breslin J.G. A hierarchical model of reviews for aspect-based sentiment analysis. In Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP 2016), 2016.
- [37] Zhang M, Zhang Y, Vo D-T. Gated neural networks for targeted sentiment analysis. In Proceedings of AAAI Conference on Artificial Intelligence (AAAI 2016), 2016.
- [38] Wang Y, Huang M, Zhu X, and Zhao L. Attention-based LSTM for aspect-level sentiment classification. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2016), 2016.
- [39] Yang M, Tu W, Wang J, Xu F, and Chen X. Attention-based LSTM for target-dependent sentiment classification. In Proceedings of AAAI Conference on Artificial Intelligence (AAAI 2017), 2017.
- [40] Liu J, Zhang Y. Attention modeling for targeted sentiment. In Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017), 2017.
- [41] Tang D, Qin B, and Liu T. Aspect-level sentiment classification with deep memory network. arXiv preprint arXiv:1605.08900, 2016.
- [42] Lei T, Barzilay R, Jaakkola T. Rationalizing neural predictions. In Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP 2016), 2016.

- [43] Li C, Guo X, Mei Q. Deep memory networks for attitude Identification. In Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM 2017), 2017.
- [44] Ma D, Li S, Zhang X, Wang H. Interactive attention networks for aspect-Level sentiment classification. In Proceedings of the Internal Joint Conference on Artificial Intelligence (IJCAI 2017), 2017.
- [45] Chen P, Sun Z, Bing L, and Yang W. Recurrent attention network on memory for aspect sentiment analysis. In Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP 2017), 2017.
- [46] Tay Y, Tuan LA, Hui SC. Dyadic memory networks for aspect-based sentiment analysis. In Proceedings of the International Conference on Information and Knowledge Management (CIKM 2017), 2017.
- [47] Tirath Prasad Sahu, Sanjeev Ahuja, “Sentiment Analysis of Movie Reviews: A study on Feature Selection & Classification Algorithms”, IEEE, (2016)
- [48] Rachana Bandana, “Sentiment Analysis of Movie Reviews Using Heterogeneous Features”, IEEE, (2018) Page 9/11
- [49] Purtata Bhoir, Shilpa Kolte, “Sentiment Analysis of Movie Reviews Using Lexicon Approach”, IEEE, (2015)
- [50] Rosy Indah Permatasari, M. Ali Fauzi, Putra Pandu Adikara, “Twitter Sentiment Analysis of Movie Reviews using Ensemble Features Based Naïve Bayes”, IEEE, (2018)
- [51] K. Singh, R. Piryani, A. Uddin, “Sentiment Analysis of Movie Reviews and Blog Posts- Evaluating SentiWordNet with different Linguistic Features and Scoring Schemes”, IEEE, (2012)
- [52] Vasilija Uzunova and Andrea Kulakov, “Sentiment Analysis of Movie Reviews Written in Macedonian Language”, Advances in Intelligent Systems and Computing, ICT Innovations, pp. 311, (2015)

- [53] Benaddy M, El Meslouhi O, Es-saady Y, Kardouchi M. Handwritten Tifinagh Characters Recognition Using Deep Convolutional Neural Networks. *Sensing and Imaging*. 2019;(20).
- [54] Roy S, Das N, Kundu M, Nasipuri M. Handwritten isolated Bangla compound character recognition: A new benchmark using a novel deep learning approach. *Pattern Recognition Letters*. 2017 April; 90.
- [55] Jangid M SS. Handwritten Devanagari Character Recognition Using Layer-Wise Training of Deep Convolutional Neural Networks and Adaptive Gradient Methods. *Journal of Imaging*. 2018; 4(41). 41
- [56] Yin Y, Zhang W, Hong S, Yang J, Xiong J, Gui G. Deep Learning-Aided OCR Techniques for Chinese Uppercase Characters in the Application of Internet of Things. *IEEE Access*. 2019; 7: p. 47043-47049.
- [57] Valuevaa MV, Nagornovb NN, Lyakhova PA, Valueva MV, Chervyakova NI. Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. *Mathematics and Computers in Simulation*. 2020 November; 177: p. 232–243.
- [58] Parhami B. *Computer Arithmetic: Algorithms and Hardware Design*. 2nd ed. New York: Oxford University Press; 2010.
- [59] Vincent L. Google code. [Online].; 2006 [cited 2020 May 15]. Available from: <https://googlecode.blogspot.com/2006/08/announcing-tesseract-ocr.html>.
- [60] Willis N. Linux.com. [Online].; 2006 [cited 2020 May 15]. Available from: <https://www.linux.com/news/googles-tesseract-ocr-engine-quantum-leapforward/>.

PROJECT OUTCOME

Project EXPO at ECE Association (**ECHO'2022**)


DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

(NBA Accredited)

ECHO

CERTIFICATE

This is to certify that JOEMON JOHNSON of IV YEAR has participated in the PROJECT EXPO event held on 29th March 2022 organized by ECE Association - ECHO '22.


Dr. A. Amir Anton Jone
Association Coordinator



Dr. D. Nirmal
HOD - ECE

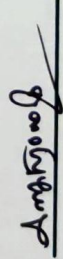
DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING


(NBA Accredited)

ECHO

CERTIFICATE

This is to certify that PRAVIN BOSCO J of IV YEAR has participated in the PROJECT EXPO event held on 29th March 2022 organized by ECE Association - ECHO '22.


Dr. A. Amir Anton Jone
Association Coordinator


Dr. D. Nirmal
HOD - ECE



Karunya

DEEMED UNIVERSITY
SOLVING HUMAN PROBLEMS



ECE
ASSOCIATION

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

(NBA Accredited)

ECHO

CERTIFICATE

This is to certify that ALLWYN K of IV YEAR has participated in
the PROJECT EXPO event held on 29th March 2022 organized
by ECE Association - ECHO '22.

Amir Anton Jone

Dr. A.Amir Anton Jone
Association Coordinator

Dr.D.Nirmal

Dr.D.Nirmal
HOD - ECE

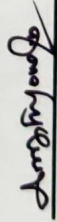
DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

(NBA Accredited)

ECHO

CERTIFICATE

This is to certify that R K VISHNUMOHAN of IV YEAR has participated in the PROJECT EXPO event held on 29th March 2022 organized by ECE Association - ECHO '22.



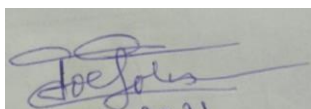
Dr. A. Amir Anton Jone
Association Coordinator



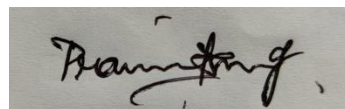
Dr. D. Nirmal
HOD - ECE

DECLARATION

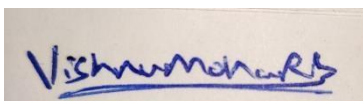
- I understand that Karunya Institute of Technology and Sciences shall hold the **copyrights of all thesis/dissertations** submitted to the University.
- I will republish the entire thesis /extracts of the thesis only with the permission of Karunya University and I am liable to pay 40% of royalty to Karunya Institute of Technology and Sciences.
- If I engage in documenting any research findings with an intention of publishing it for commercial purpose, **I shall obtain a NOC from the Office of the Registrar** prior to engaging in such activities.



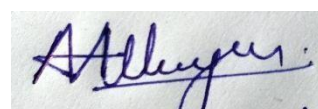
JOEMON JOHNSON (URK18EC010)



PRAVIN BOSCO J (URK18EC060)



VISHNUMOHAN R K (URK18EC072)



ALLWYN K GEEVARUGHESE (URK18EC070)

Signature of the candidates with Reg.No