IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 23 March 2023, accepted 9 April 2023, date of publication 12 April 2023, date of current version 18 April 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3266640

RESEARCH ARTICLE

An Experimental Analysis of Deep Neural Network Based Classifiers for Sentiment Analysis Task

MRIGANK SHUKLA AND AKHIL KUMAR
School of Computer Science and Engineering, Vellore Institute of Technology, Chennai 600127, India
Corresponding author: Akhil Kumar (akhil.kumar@vit.ac.in)

ABSTRACT The application of natural language processing (NLP) in sentiment analysis task by using textual data has wide scale application across various domains in plethora of industries. We have methodically studied pre-existing models and proposed new models for examining sentiment analysis task. The models proposed were analysed with three widely popular word embeddings separately and in combined approach using all embeddings as unique channels. We combined deep neural network models such as Bidirectional Long Short-Term Memory (BiLSTM) and Convolutional Neural Network (CNN) so that integrated models complement each other with their unique architectures. The word embeddings used had profound impact in accuracy of models owing to performative changes. The best word embedding was Word2Vec giving highest accuracy in almost all implemented models, followed by GloVe. FastText embedding performed consistently worse, giving much lower accuracy than other embeddings. We also observed that adding transformer encoder layers with CNN improves accuracy by 2% when compared to CNN without any transformer layers. An accuracy improvement of 2-3% over CNN-BiLSTM model was also observed by utilizing transformer encoder layer in conjunction with both BiLSTM and CNN. The proposed model achieved an accuracy of 89.04% on SST-2 dataset. We also compared larger pretrained language model used in sentiment analysis task with our proposed approach. The accuracy values obtained through combination of embeddings and models can be useful for other researchers when selecting word embeddings for their models.

INDEX TERMS Sentiment analysis, deep learning, natural language processing, word embeddings, text classification.

I. INTRODUCTION

The massive rise in popularity of internet and its ease of access to general population has created an exponential rise in user generated content on online platforms [1]. People express their feelings and opinions on social media platforms through text, images and videos. Users are also able to present their views about specific products and services through interactive comments on commercial business websites and fun social interaction platforms as well. The textual content generated on these platforms has valuable insights into the commentor's opinion and feelings [2] on the topic being discussed upon. These opinions can be better understood and utilized effectively through a systematic application of Natural Language Processing (NLP) techniques on texts collected. This process of identifying, extracting and categorizing information that is subjective in nature from unstructured textual content is called as Sentiment Analysis, historically also referred to as opinion mining [3]. A system of classification of sentiment analysis can be the size of text content that we need to analyse, based on this Document, Sentence and Aspect Based Sentiment Analysis [4] are performed. This research work focuses on sentence level sentiment analysis, which will classify a given sentence to a specific class from possible classes.

Traditional approach to sentiment analysis involves the identification and creation of features from text manually. These traditional models used Term Frequency-Inverse

The associate editor coordinating the review of this manuscript and approving it for publication was Tao Huang.

Document Frequency to help represent the text numerically, these models worked well with simple sentence structures, but failed when used with sentences with more diverse and complex language structure [5]. Deep learning has emerged as a strong tool for solving a wide range of issues in several disciplines such as computer vision and speech recognition. These deep-learning based models have proven to be useful in natural language processing use cases as well for their ability to extract complex features from data and usefulness in large scale text processing [6]. Convolutional neural networks (CNNs) [7] have been applied for sentiment analysis, these models use word embeddings to represent text in form of numerical vectors that can map similar meaning words to similar vectors, and enable performing mathematical operations on these words. The CNN layers use fixed length representation from textual embeddings with the use of filters and pooling layers. These models are able to capture local features from text, but completely ignore the impact of context of words due to their positions in text. This does not allow for these models to capture long term feature information [8], but they are able to compensate by using numerous filters which are able to learn more. Recurrent neural networks (RNNs) [9] have also been used for sentiment analysis as they are able to retain information in sequence, using the information learned from previous sequence to be used in the next sequence of text. The RNN models suffer from the problem of vanishing and exploding gradient and hence are not able to retain large amount of information extracted from textual embeddings. Bidirectional long short-term memory (BiLSTM) model which is a type of RNN have also shown remarkable performance [10], as they are able to focus on the textual sequence from both direction by using two LSTM cells. The introduction of attention mechanism has helped in making great strides in the natural language processing. Applying attention mechanism [11] to textual sentiment analysis helps model focus on relevant information from text allowing it to remember long term dependency information [12]. New embeddings have been developed since the introduction of Word2Vec [14] and GloVe such as FastText which when combined together to give a multi-channel representation have yet to be tested with these models. There has also been a sharp rise in development of large-scale language models such as BERT [36] that have been pre-trained on massive datasets. Transfer learning from these large models have consistently shown great results in many NLP tasks. These models have been pre-trained using large datasets such as Wikipedia (2,500M words) in case of BERT. The pre-training requires heavy use of tensor processing units (TPUs) like, for $\text{BERT}_{\text{base}}$ 16 TPU chips were required and the training lasted a total of 4 days. These pre-trained models need to be further fine-tuned on downstream tasks which further requires training time and graphical processing units (GPU) with large memory capacity to fit these models. These resources are not easily available to smaller companies or individuals. In this work, we have focussed on comparatively smaller models that use pre-trained embeddings and are intended to be trained for use on task specific datasets only. The architecture of the model proposed in this work use lesser number of trainable parameters (6M) as compared to large pre-trained language models such as $\text{BERT}_{\text{base}}$ (110M). This difference in number of trainable parameters can be leveraged by smaller companies and individuals to fit their computational budget. This work is aimed to test various small-scale deep learning models, with different permutations of word embeddings, to test their performance in sentiment analysis task. This work proposes a novel Transformer-BiLSTM-CNN model that gives great result on sentiment analysis task, other models such as CNN (Multichannel), CNN-BiLSTM, BiLSTM-CNN-BiLSTM, Transformer-CNN have also been tested in this work with combinations of different word embeddings. The application of sentiment analysis in web based commercial business is tremendous. Analysing customer's sentiment from opinionated texts allows organizations to make important business level decisions of recommending products, creating new products and assessing overall reaction of consumers towards a product [13]. Hence different light-weight models utilizing lesser computation resources can be highly useful for these businesses. This work analyses combination of latest models in natural language processing field for sentiment analysis, that will also serve as a comparable baseline for other researchers to build upon their improved work. The major contributions of this work are:

- Exhaustive study of different models with comparison of pre-trained word embeddings and their combinational impact. Furthermore, we have trained and tested these models with random (*Rand*) embeddings that are fine-tuned during training to provide baseline for comparison.
- Proposal of a novel Transformer-BiLSTM-CNN model for sentiment analysis task.
- A standard dataset has been used for training and testing different models and the model performing best with highest accuracy is Transformer-BiLSTM-CNN with Word2Vec embeddings achieving a value of 89.04%.
- The embedding which performs best with almost all the models is Word2Vec.

II. RELATED WORK

Motivated by the deeply valued application of NLP in the field of sentiment analysis by using opinions expressed by general public several works are published in recent years. He et al. [15] used deep learning-based LSTM-CFR model with a convolution neural network to perform analysis of student's opinions that they express in online social media platforms. They performed the task using Chinese and English text used frequently by college going students to express themselves in online forums. Instead of relying on sentiment lexicon and machine learning models that are inherently dependent on tedious handcrafted features to make the models perform better, they used word embeddings to perform task of word vectorization. Using LSTM network and

CFR jointly for the purpose of adaptive word segmentation focused on Chinese text these segmented words were then vectorized. Word vectorization through embedding and use of LSTM-CFR helps map dependencies between valuable features and sentimental opinions. The results were then passed to a deep learning CNN model which was able to classify the text. Model developed was tested on Chinese datasets as well as widely used English sentiment analysis dataset. Khasanah [16] tested FastText [17] embedding with simple single layer CNN and BiGRU models. The researchers used FastText embedding to construct a matrix of words and relevant embedding. In the matrix, each row was represented as a 300-dimensional feature vector that was calculated using FastText embedding. In CNN model, a convolution layer was used with size of kernel fixed as 2, to convolve over the 300-dimensional vectors at once, it generated a reduced dimensional representation of words. Similarly, the word embeddings were passed through BiGRU layer which used combination of a forward GRU and subsequent reverse GRU to create a bidirectional representation. A pooling layer was used with max pooling and average pooling for both the models, results from both were concatenated to be used in class prediction. CNN model proved to be a better choice giving better results than BiGRU, with best accuracy of 83.9%. Kasri et al. [18] proposed a new sentiment aware embedding which they called Word2Sent, this embedding was aimed to capture implicitly the sentiments for words leveraging the surrounding context. This gave the embedding semantic, syntactic and sentiment aware embedding which had reduced dimension, when compared with tradition embeddings. These were developed using Continuous Bag of Words (CBoW) [14] and SentiWordNet [19] which is a famous lexical resource for sentiment analysis. The embeddings were then used as textual representation input for a CNN model, and gave slightly improved performance than traditional embeddings. Khan et al. [20] proposed a method to extract sentiment aware features from noisy texts, with dimensionality reduction statistical algorithms such as mRMR [21] and PCA [22] for condensed feature selection. Wide coverage sentiment lexicon (WCSL) [23] was integrated with linguistic rules to identify features from sentence enriched with semantic and sentiment knowledge. Different embeddings were then applied on identified sentiment features to convert them into word vectors. The curated features were then processed in a CNN model, which achieved an accuracy of 85.10%. When working with sentences in text processing, models usually accept input of fixed length only. To address this issue, researchers have used padding for this purpose, in which sentences shorter than fixed length are padded with vectors filled with zeroes and larger sentences are trimmed to adhere to the decided length. Giménez et al. [24] used semantic based padding to reduce noise in training sentences caused by zero padding. The authors used the text present in the sentence to pad to maximum length, this helped in learning relationship within words in long-distance context. As the ending of sentence was composed of text from beginning of sentence instead of meaningless padding. The applied semantic padding performed better when compared with sentences padded with zero vector, in all the word embeddings used.

The application of RNN based models which could overcome the exploding and vanishing gradient problems has shown promising results in the field of text processing. RNN based models such as LSTM and gated recurrent neural network, commonly referred to as GRU [25] have been used alone and in combination with convolutional neural networks. Wang et al. [26] used a combined approach of CNN and RNN based models by implementing a unique pooling strategy in their CNN model. Instead of using max pooling over the feature map obtained as a result of convolution operations, they used pairwise max pooling to produce a reduced dimension feature map. This has helped to maintain the sequential information of sentence being passed through convolution layer. The result obtained after CNN layer were later passed to LSTM in one variation of model and GRU in the second variation of model for their unique ability to capture long-term dependencies. Application of BiGRU model have been explored in sentiment analysis as mentioned above. To further focus on relevant words of text instead of giving equal importance to all sequences, attention [11] layers prove out to be very useful. Liu and Gao [27] combined BiLSTM and CNN with an attention layer. The model utilizes an CNN layer to convolve over word embeddings using different filters which were later passed into a BiLSTM layer. The forward context was passed through an attention layer and backward context obtained from BiLSTM was passed through another attention layer and the resulting vectors were concatenated and passed into SoftMax layer for classification.The type of attention and the way it is applied to the features obtained from convolutional filters of CNN layer was studied by Usama et al. [28] in their sentiment analysis model consisting of CNN and RNN. They used attention mechanism [29] to calculate score for features obtained from convolution operation with two variations. In the first variant contextual features generated from convolution operating were averaged to calculate the attention score while in the second variant attention scores were obtained by performing calculations on averaged features context being generated from max pooling layer of CNN. The context features and attention score were used together as input to RNN and processed sequentially to pass the output to a fully connected layer with a softmax layer at the end. They experimented with both randomized and pre-trained word vectors for generating word vectors in embedding layer. Zhu et al. [30] used a similar architecture based on attention mechanism word embeddings which were passed through a BiGRU layer for capturing long-distance contextual semantics. Self-attention is then applied on BiGRU layer's output to compute word similarity across sentence subsequently emphasizing word with strong emotions. Resulting output was passed through a convolution layer with depth separable convolution and

dilated convolution was used with global average pooling that aided in reducing the number of parameters passed to fully connected layer. Focal loss was used to train the model as it accounts for sample size difference between categories in dataset. Using connected models can sometimes lead to overfitting and increase the training complexity as hyperparameters increases due to increase in possible combinations. Xu et al. [31] proposed a hybrid attention based robust neural network. The network used pre-trained embeddings and passed them separately to a BiLSTM network and a CNN network. The output of every hidden state of BiLSTM and pre pooled output of CNN were fused dynamically using an attention layer. The output of attention layer, the last hidden state of BiLSTM and max pooled features of CNN were later concatenated to be passed to a fully connected layer and classified using a softmax layer. Zhang et al. [32] used fused parts of speech tagging with embedding, which further help increase syntactic and semantic knowledge capture when passed through attention layers. Huang et al. [33] used transformer block instead of using only an attention layer. They used multiple attention heads in the transformer block that helped in contextual knowledge enrichment and long-term dependency memorization. The transformer block was jointly trained on two tasks with polarity prediction for individual words which was further combined with classification of sentence as whole. They achieved fascinating results with the use of a single transformer block.

Recent developments in deep learning for natural language has also seen the introduction of massive pretrained models that can be used for variety of tasks. Model such as ELMo [34] is able to give a contextualized vector representation for a word by representing it as a function of the complete input sentence and is trained on a massive text corpus. ELMo is used for generating pretrained representations that are used as input to other deep learning layers such as BiLSTM. This approach is referred as feature based aggregation and significant performance increase in the task of sentiment analysis have been observed by using this strategy as demonstrated by Wang et al. [35]. Another model that effectively utilizes pretraining to build language model is Bidirectional Encoder Representations from Transformers commonly abbreviated as BERT [36]. The model can be used in other downstream tasks such as sentiment analysis through classification, question answering etc. by fine-tuning the already trained parameters by using task specific dataset. This strategy is commonly referred to as ''fine tuning'' in context of such giant pretrained language representation models. Unlike other large models developed for pre-trained representations, BERT is able to use to use context from both the left and right side by using a masking technique allowing for development of richer representations. Munikar et al. [37] used BERT for sentiment classification task by adding a regularization layer and softmax on top of it and fine-tuned it on their chosen dataset for the task giving much exceptionally good results. Efforts have been made to use these models with additional neural network architecture to give improved results on language tasks. Zhang et al. [38] developed a new model which they named as Broad MultiTask Transformer Network that utilizes both the feature and fine-tuning based approach. They used pretrained BERT model and fine-tuned using a multitask approach using different datasets that are linked with specific tasks. The shared parameters in contextual representation layers were able to learn better representations from the training data of different tasks which is due to implicit data augmentation and regularization that are benefits from a multitask learning approach. They used the broad learning system approach suggested by Chen and Liu [39] on top of the multitask deep learning network for the task of classification in sentiment analysis. The powerful representation of deeply contextual nature obtained from multitask network were passed to a set of mapping feature nodes and enhancement nodes. In the proposed model the usage of incremental learning algorithms helped in fast remodeling that is advantageous than the approach of training the network weights from scratch. The model obtained performed slightly better than the original BERT model when compared on accuracy of results. In this work, we have used the results of these approaches that have pre-trained language models as major components to compare the performance of our much smaller models.

III. MATERIALS AND METHODS

To carry out a systematic study of performance comparison of deep neural network models with different word embedding combinations, we have used SST-2 [40] dataset for training and testing of models. We have used CNN with Word2Vec, FastText and GloVe [41] as three input channels. Later the impact of BiLSTM layer for learning contextual information through sequence was studied by using two different models that is, BiLSTM-CNN where the BiLSTM layer was used only in front and BiLSTM-CNN-BiLSTM in which it was used in both the sides of CNN layer. Transformer blocks with multi-head attention layer have been extensively used in natural language processing filed due to their ability to capture information in a weighted manner from different positions of text. We have used transformer block with CNN and BiLSTM layers to study combination of models which are Transformer-CNN, CNN-Transformer, Transformer-BiLSTM-CNN and Transformer-Transformer-CNN. The performance of all these models were compared with the usage of the above specified word embeddings.

A. DATASET

To carry out this work, the dataset used for training and testing all the models is Stanford sentiment treebank [40] binary labeled dataset commonly referred as SST-2 dataset. It consists of movie reviews initially collected by Pang et al. [42] for purpose of sentiment analysis, this was parsed using Stanford parser [43] to parse the sentences which resulted in spitting of single sentences to multiple phrases. Further, these were labeled by human judges in three categories. For SST-2 dataset, task was set as binary classification hence

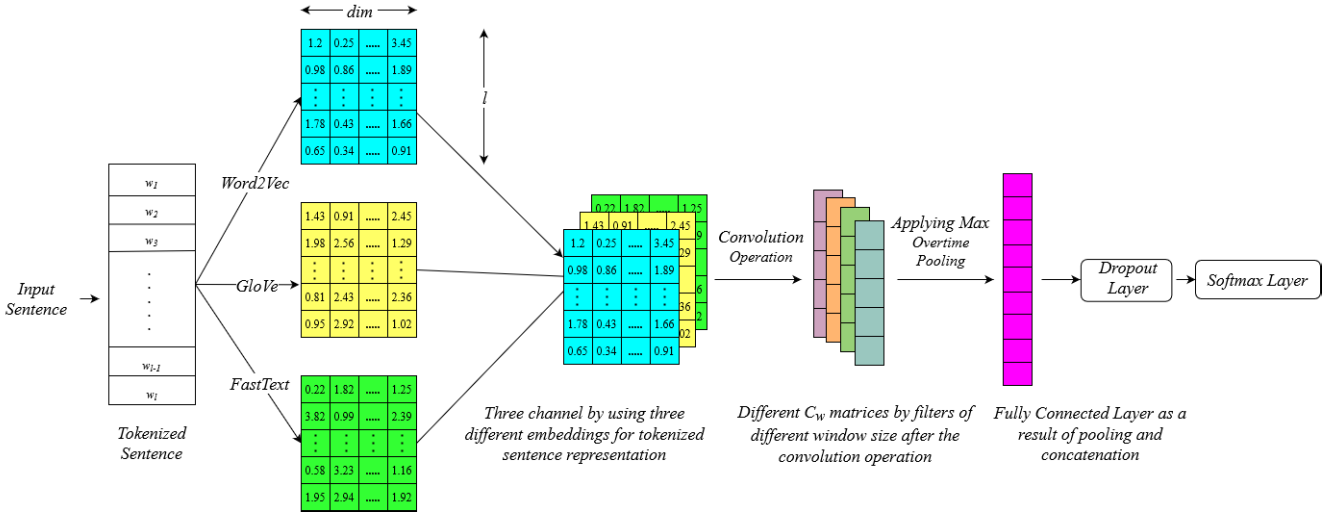

FIGURE 1. The multi-channel implementation of CNN model.

sentences with neutral labels were removed from the dataset. The resulting dataset was split into three sets with 6920 sentences in training, 872 sentences in validation and 1821 sentences in test set. To get more data for training many researchers have used the phrase level training dataset which consists of 67,349 rows with binary labels. Following the strategy of other researchers, we have also followed this methodology. The sentences for validation and testing set were used as whole sentences and were not split.

B. TEXT PROCESSEING

The text present in the dataset is already clean to most extent. We performed basic text processing steps such as removal of special symbols that may pose hindrance while calculating word embeddings. Then tokenization was performed to split the text into singular words which serve as tokens. These tokens serve as basic units on which computation will be performed further, the number of tokens present in a sentence is considered as the sentence length. To make computation easier through batches, sentences were padded to make sentence length uniform throughout training, validation and testing dataset. The length of longest sentence was used as maximum length for padding. These tokenized sentences with padding were used later as input for word vectorization using different embeddings.

C. METHODS

We have used different deep learning models and tested unified combinations of the models. The combined usage of models help in compensating for each other drawbacks giving a better result in most cases. Along with this, effectiveness of GloVe, Word2Vec and FastText embeddings individually and used in combination to different models have been analyzed which is used to determine the best embedding for word vectorization to be used with different deep learning models.

1) CNN

Convolution neural networks (CNNs) architecture proposed by Kim [7] which itself adopted many insights for various layers from Collobert et al. [44] has shown great performance in the field of classification tasks in natural language processing. Our proposed architecture is similar to the original but we have used multi channels consisting of different trainable embeddings instead of a single embedding. We have used Word2Vec, FastText and GloVe embedding as three input channels to the CNN models. The architecture of CNN models learns effectively using convolution operation by grouping words similar to n-grams through its multiple filters. The model also reduces the dimension of word vector input which makes it easy for the fully connected layer to learn effectively. The model architecture explained below is also used in combination with other deep learning models, to further increase the accuracy of classification task.

a: WORD EMBEDDINGS

Natural language words cannot be directly used with deep learning models as these models require numerical vectors as input. Word embeddings are used to represent words in a machine understandable numerical format. It is aimed to capture relationship between words in vector format which closely represent their usage in texts. Word embeddings are able to replicate and store contextual information in vectors with compact dimensions, so that word similar to each other have the same relationship represented in these vectors. Word2Vec is a word embedding that has been trained to capture relationship between words in specified window size. However, this embedding is not able to use the whole document while calculating these vectors due to its short window size and subword information is also not captured by it. Furthermore, it is unable to handle out of vocabulary words. This embedding is unable to decipher word disambiguation,

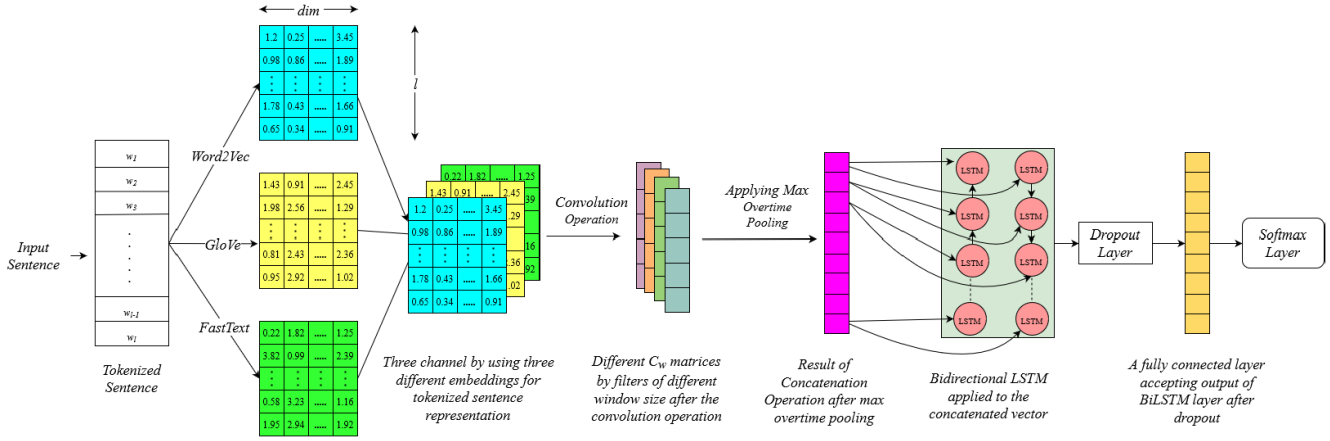

FIGURE 2. The CNN-BiLSTM model.

making it ineffective to give different embedding for same word based on its context of usage. FastText embedding help improve shortcomings of Word2Vec. It uses skipgram model similar to Word2Vec but uses n-gram at character level enriching it with subword information and also enabling it to handle out of vocabulary words, which means words that are not seen during training are recognized by it. GloVe uses the statistical information gathered over the whole corpus and utilizes co-occurrence probabilities to calculate the word embeddings. A tokenized sentence is passed as input to this embedding and corresponding embedding is extracted for each token. Tokens used for padding are initialized with zeroes and out of vocabulary words are randomly initialized wherever applicable. Using GloVe, if we choose l as the length for all the sentences and pad as the shorter sentences then a sentence is represented as $S = \{w_1, w_2, w_3, \ldots, w_l\}$ where w_x represent the tokens extracted from sentence which are words and, in some cases, padding tokens and when passed through word embeddings resultant matrix of embeddings can be represented as in (1).

$$Em = \{wvec_1, wvec_2, wvec_3, \ldots.., wvec_l\} \in \mathbb{R}^{l \times dim} \quad (1)$$

In (1), $wvec_i$ is the word embedding for the i^{th} token in the sentence and *dim* is the dimension of word embedding used. These word embeddings convert each word into corresponding vector and are set as trainable parameters so that they can improve the training of model used with. We have used Word2Vec, GloVe and FastText to obtain three different embeddings for the same sentence that will be passed as three input channels to convolution layers as presented in Fig. 1.

b: CONVOLUTION FILTERS

The result of word embedding operation is a matrix of numbers in which each row represents embedding vector of a token in input sentence and column represent the embedding dimension. The length of the matrix is l for each sentence and width is *dim* which represents the dimension of word embedding used. Here, *dim* is kept same for all three employed embeddings. We have used filters with width equal to embedding dimension to effectively treat a row as meaningful computational unit. Further, we have varied the *length* of the filter to focus on specific number of consecutive adjacent rows, referred to as window size, which effectively treats tokens as n-gram features. If the sentence is represented as embedding vector matrix $Em \in \mathbb{R}^{l \times dim}$, where consecutive rows from i^{th} position to j^{th} position, both inclusive, are represented by $Em[i^{th}{:}j^{th}]$. The filter is represented by F_w^n where w represents the window size and n represents the nth filter for window size w, giving filter dimension of $w \times dim$. The filter is applied by convolution operation ($*$) over consecutive w rows in a sliding manner to calculate element wise dot product and further summed to give a singular value.

This operation can be represented as in (2).

$$C(i)_w^n = F(w^n * Em[i : i + w - 1] + b^n) \quad from\ i = 1\ to\ i = l - w + 1 \quad (2)$$

In (2), b^n is the bias term added for nth filter and F is a non-linear activation function which is rectified linear unit (ReLU) in this work. The convolution operation performed by a single filer gives an output in form of a single dimension matrix $C_w^n = [c(1)_w^n, c(2)_w^n, c(3)_w^n, \ldots, c(1 - h + 1)_w^n]$, where $C \in \mathbb{R}^{l-h+1}$. We have used multiple filters n for same window size w along with different window sizes. After the convolution operation, we have applied max overtime pooling operation on each C_w^n extracting maximum value from each window and the resulting values are concatenated. The concatenated vector can be represented mathematically by using (3).

$$V = [max\left(C_{w1}^{n1}\right), max\left(C_{w1}^{n2}\right), max\left(C_{w2}^{n1}\right) \ldots, max\left(C_{wk}^{nj}\right)] \quad (3)$$

In (3), V represents the concatenated vector, $\max(C_{w1}^{n1})$ is the result of max overtime pooling operation on vector resultant of convolution performed by $n1$ filter of window size $w1$. The total number of window sizes decided are k and number of

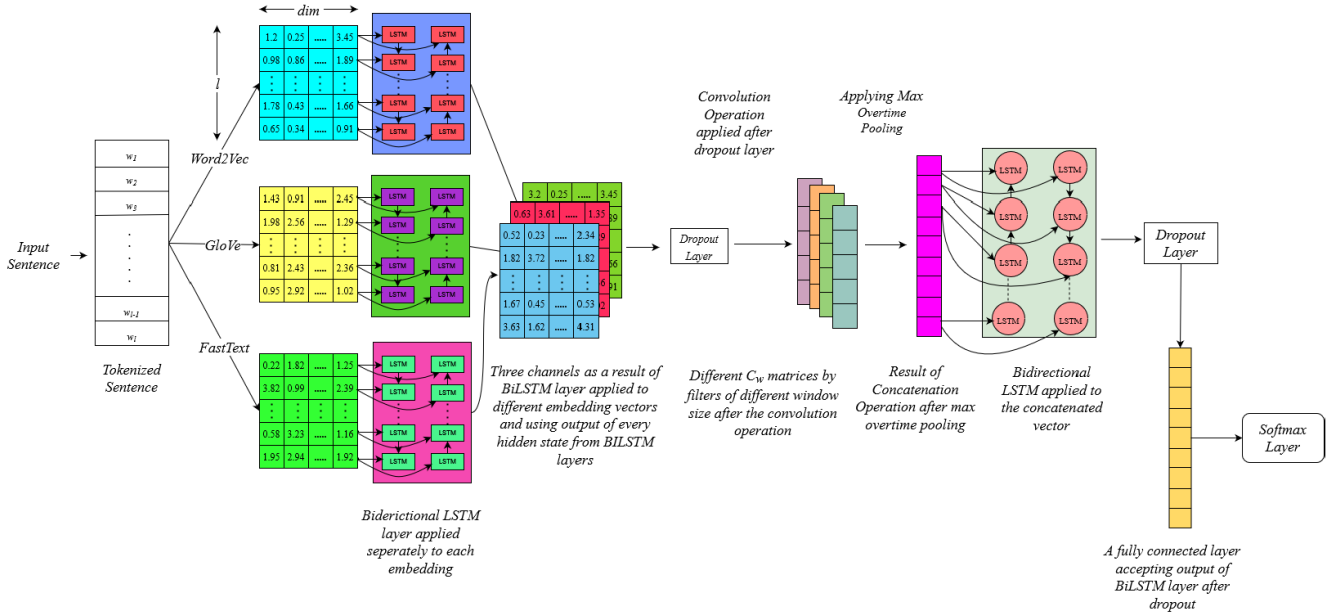

FIGURE 3. BiLSTM-CNN-BiLSTM model.

filters in each are j, which can vary for each window size as well. To avoid possible overfitting due to accumulated features from various filters, we have added a dropout [45] layer to the model which assists in regularization. The resultant output after dropout is passed through a fully connected layer and then a softmax layer, after which cross-entropy loss is used as training loss for minimization objective. Then backpropagation is used to update parameters for each layer including the embedding layer.

2) CNN-BILSTM

LSTM [46] model successfully overcomes exploding and vanishing gradient problem that is observed during backpropagation of RNN models. This model is widely used in time series data analysis where input sequences to model have some dependency based on sequence and can benefit from information learned from previous input. BiLSTM further improves the sequence-based learning by using information passed from both the direction. This is accomplished by using two different LSTM networks, one network allows information to flow from present sequence to future sequence, the other network learns by information flowing in opposite direction that is, from future to past sequence. This allows the model to learn better by combining contextual information learned from both directions. The output is a concatenation of output from last states of both the forward and backward LSTM network. We have combined CNN model with BiLSTM layer as explored in [47], but instead of using a single channel in CNN layer, we have incorporated three different channels from three different embeddings. We converted the tokenized sentences into word embeddings and Word2Vec, GloVe and FastText has been used to obtain three different embedding representation for the same sentence, using these embeddings as three input channels. We have then passed the embeddings to CNN layer as explained in equations (1), (2), and (3) to obtain a pooled representation consisting of combined knowledge from all the filters of different window size. The pooled input is then passed to the BiLSTM layer as illustrated in Fig. 2. After this, a dropout layer is added for regularization. A fully connected layer is applied after the dropout layer to pass the resulting output to a softmax layer. We have used minimization of cross-entropy loss as training objective to train all the layers of this model.

3) BILSTM-CNN-BILSTM

The previously discussed CNN-BiLSTM model uses the BiLSTM layer after the pooling operation of convolution network allowing the model to obtain a reduced dimension for BiLSTM to work on and to get a global view of the data. However, it leads to loss of sequential information due to pooling operation. To overcome this, we have proposed an additional BiLSTM layer before Convolutional layer allowing it to use sequential information to learn relevant part of sequential tokens from input embedded sentences before being passed to CNN layer. The LSTM layer uses forget gate that enables the network to forget the irrelevant information from current input state and previous hidden state. Input gate is applied for updating cell state of network along with output gate for calculating the next hidden state of network. These gates are composed of non-linear activation functions such as tanh and sigmoid, which enable them to achieve the objective. We use two LSTM networks one learning from processing sequences in forward direction while the other learns from backward processing of same sequence. The resultant outputs of both

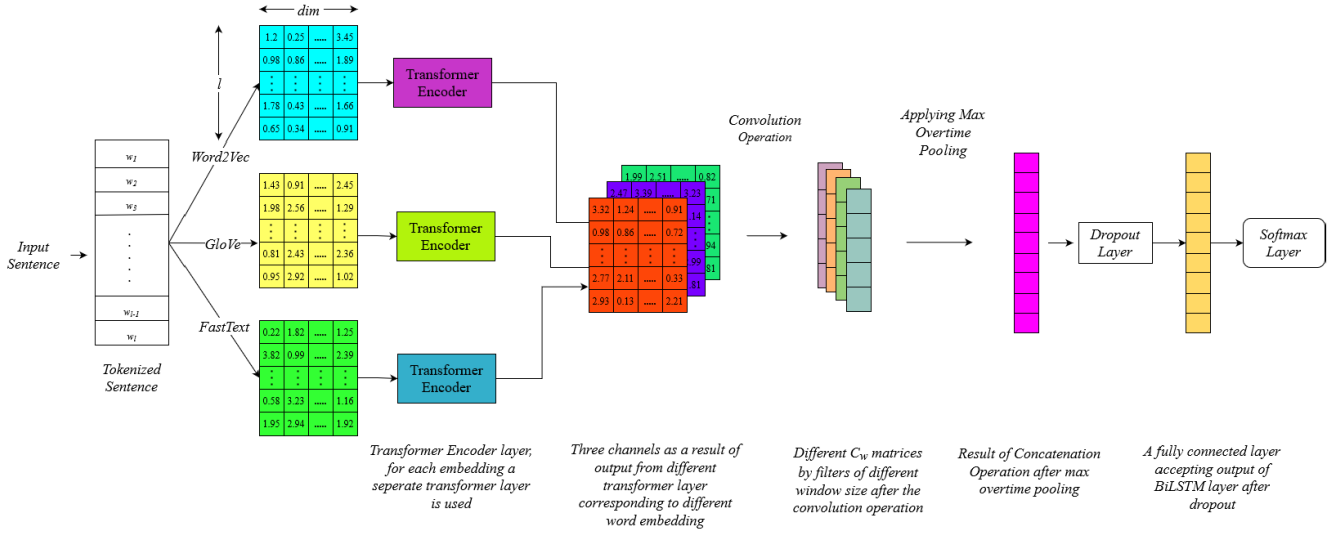

FIGURE 4. The transformer-CNN model.

are concatenated giving us bidirectional LSTM or BiLSTM. We have kept the dimension of hidden state to half of the input sequence so that the concatenation operation generates the same dimension as input embedding sequence. As we are using three embeddings for word vector representation, each word embedding vector is passed through a separate BiLSTM layer. So Word2Vec has its own BiLSTM layer, GloVe is processed through a separate BiLSTM layer and a third BiLSTM layer is used for FastText embeddings. The LSTM layer is implemented using the mathematical operations proposed in [48] and [49] and can be represented using (4), (5), (6), (7), (8), and (9).

$$input_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \tag{4}$$
$$forget_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \tag{5}$$
$$g_t = tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \tag{6}$$
$$output_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \tag{7}$$
$$c_t = forget_t \odot c_{t-1} + input_t \odot g_t \tag{8}$$
$$h_t = output_t \odot \tanh(c_t) \tag{9}$$

In the above equations, h_t is used to represent state of hidden layer at time t, the cell state at time t by c_t, input state by x_t at time t and h_{t-1} is used for hidden layer state at time *t-1*. The input gate, forget gate, cell gate and output gates are represented by the symbols $input_t, forget_t, g_t, output_t$ respectively. The non-linear activation function sigmoid is represented by σ, whereas $\odot$ is used for Hadamard product. As we use two LSTM layer to form a single BiLSTM layer for forward and backward layers therefore, the final output is stacked together. Each embedding is passed through its own BiLSTM layer and then stacked together as three channels matrix input to CNN layer. To achieve this, we have used the output of all hidden states of BiLSTM unlike implementations in the related work, where the output of only last hidden state is used. This is done so as to give an extended word embedding with long term sequence information making this BiLSTM layer's output the same dimension as our embedding layer. The resultant output from BiLSTM layer can be represented as in (10)

$$B_{res} = \{h_1, h_2, h_3 \ldots .., h_l\} \in \mathbb{R}^{l \times 2 * hidden\,dim} \tag{10}$$

In (10), B_{res} is the resultant output of a BiLSTM layer which is a embedding passed from an embedding layer, the h_i represent the hidden state at time step i of the last layer which is in this case is token at position i. As we are using bidirectional LSTM, the size of resultant output is twice the size of dimension of hidden units result of concatenation of forward and backward layer. Dropout layer has been proven to increase the performance of RNN based networks [50], therefore, we have applied a dropout layer to stacked matrix and further passed the output to the CNN layer. The implementation design for this model is represented in Fig. 3. The result of CNN layer after max overtime pooling operation is further passed to another BiLSTM layer for global level features extraction, which is further passed to another dropout layer before being passed to a fully connected layer. Softmax layer is then applied to final output with cross-entropy loss minimization as training objective for the network as whole.

4) TRANSFORMER-CNN

Sequence to sequence models such as BiLSTM applied in the above models only focus on the past sequence at time *t-1* while computing over time sequence *t*. This does not allow parallelization for fast processing and also poses a bottleneck for longer sequence length, where sequence at a time step may relate differently with each sequence from past and not necessarily be related strongly to the previous few sequences alone. Attention mechanism [51], [52] works to solve this problem by allowing these models to focus varyingly on all

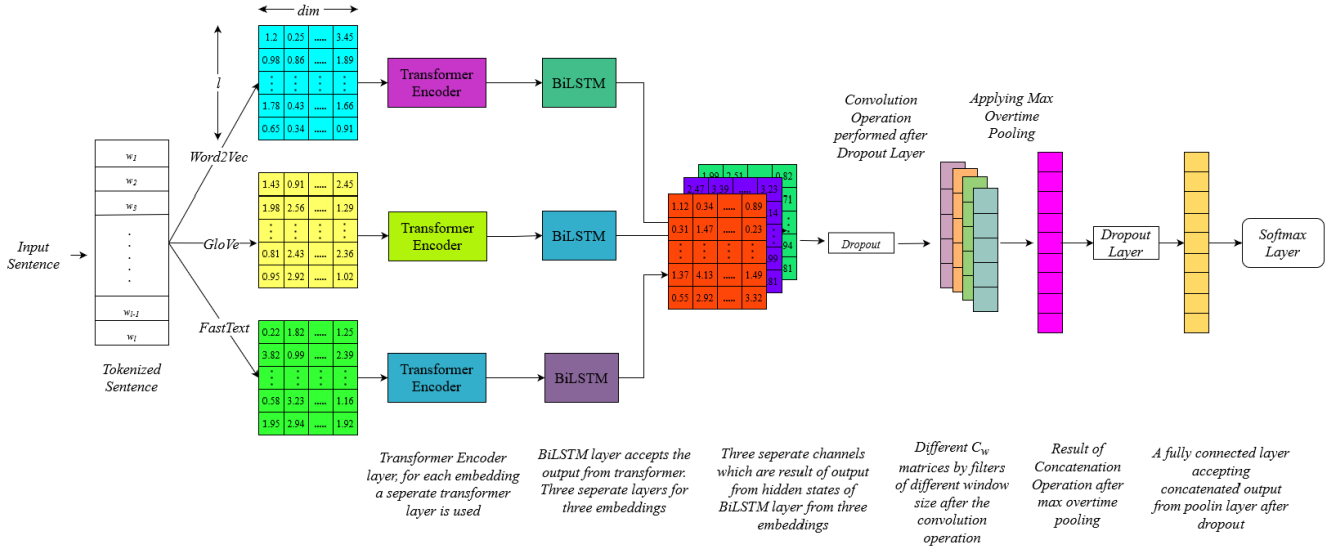

FIGURE 5. Architecture of transformer-BiLSTM-CNN model.

input hidden state sequences passed and further to consider relevant parts with higher weights and lower weights for irrelevant parts of input sequence. Transformer model [11] used the concept of attention without any sequence-to-sequence layers that allowed the model to perform higher parallelization and achieve better results. In this work, we have used only the encoder layer of transformer model to enrich our word embeddings with positional context by considering the whole sentence at once using self-attention mechanism. The transformer model uses positional embedding combined with word embedding as input to the first encoder layer where, positional embeddings allow the model to learn specific patterns by using position of words in the sentence [53]. The positional encoding combined by addition with word embeddings are passed to encoder layer whose main components are self-attention layer and a feed-forward network. The word embeddings of sequence are passed all at once to the encoder layer. The self-attention layer allows each sequence to have a look at all other sequences for calculating relations that effectively allows it to focus more on relevant sequences. This is achieved by using query, key and value vectors, which are computed to a lower dimension by using three different weight matrices. As we have used self-attention, the input sequence was used to calculate these vectors. The query vector of a single word in sequence whose attention score has to be computed undergoes a dot product operation with keys of all the words in the sequence and scaled to a score with square root of dimension of key vector. All these scores undergo through a softmax operation which allows the model to make learned decision by giving more importance to certain words by assigning high softmax score and suppressing irrelevant words with low softmax score. The value vectors are multiplied by these softmax scores and summed up to be passed to feed-forward network for a given sequence. The self-attention mechanism [11] mentioned above is mathematically expressed in (11) where "Query" is abbreviated as Qu, "Key" as Ke, "Value" as Va and d_k represents the dimension of "Key".

$$Attention\,(Qu, Ke, Va) = softmax\left(\frac{Qu \cdot Ke^T}{\sqrt{d_k}}\right) Va \quad (11)$$

The transformer encoder layer as discussed above uses a multi-headed attention which allows parallelization and helps the model to learn information using different representation subspaces. In this model, the self-attention layer is followed by a normalization layer with residual connection from input embeddings which is passed to a feed-forward network having one more normalization layer with a residual connection. This operation give same dimensional features for input sequences as the input embeddings which are passed to a CNN layer with max overtime pooling, fully connected layer, dropout and a softmax layer as explained in previously presented models. We have tested different embeddings' performance as input to these models and also a combined approach in which each word embedding is passed through its own transformer encoder layer and the results are used as different channels to CNN model as presented in Fig. 4. Further, we have used two variations of this model configuration, one with a single transformer encoder layer and the other with two transformer encoder layers. In the second variation the output of first encoder is passed to second layer followed by CNN model. Cross entropy loss minimization task is chosen as training objective for the whole model and every layer is trained including the word embeddings.

5) TRANSFORMER-BILSTM-CNN

Bidirectional LSTM has been applied as a feature extractor in different models for sequential data [53]. The BiLSTM layer

is effectively able to capture information from sequences even in texts with longer lengths by using a combined forward and backward combination of cells that allows analyses of text from both the directions. Self-attention mechanism helps in finding crucial dependency between individual tokens or words in a sequence at once by using all the tokens from the input sentence. Transformer uses self-attention with other different layers in its encoder module to effectively leverage self-attention while computing features for word embeddings. We have used transformer-encoder layer to use its powerful self-attention mechanism to further improve the word embedding vectors of input sentences with contextual information. As self-attention is able to use its query, key and value to compute dependence between different words in a single sentence as a whole. This allows words in sentence to calculate effectively the impact of other words at small or large distances on them. Hence, when they update their embeddings, they can use other words in sentences with weights dependent on their relevance to effectively capture their context in sentence with reference to other words. We have used different word embeddings to represent the tokens in sentences to corresponding embedding vectors. The word embedding vectors are combined with positional embeddings and passed to muti-headed attention layer in transformer encoder. We have used [54] as reference to better explain the operations being performed in the transformer-encoder as expressed in (12), (13), (14), (15), (16), and (17).

$$Qe^{(h)}(x_i) = W_{h,q}^T x_i,\ Ke^{(h)}(x_i) = W_{h,k}^T x_i,\ Va^{(h)}(x_i) = W_{h,v}^T x_i,\ where\ W_{h,q}, W_{h,k}, W_{h,v} \in \mathbb{R}^{dim \times k} \tag{12}$$

$$\alpha_{i,j}^{(h)} = Softmax_j\left(\langle Qe^{(h)}(x_i)\rangle / \sqrt{k}\right) \tag{13}$$

$$u_i' = \sum_{h=1}^{H} W_{c,h}^T \sum_{j=1}^{n} \alpha_{i,j}^{(h)} Va^{(h)}\left(x_j\right),\ where\ W_{c,h} \in \mathbb{R}^{k \times dim} \tag{14}$$

$$u_i = LayerNorm\left(x_i + u_i'; \gamma_1, \beta_1\right),\ where\ \gamma_1, \beta_1 \in \mathbb{R}^{dim} \tag{15}$$

$$z_i' = W_2^T ReLU\left(W_1^T u_i\right),\ where\ W_1 \in \mathbb{R}^{dim \times m},\ W_2 \in \mathbb{R}^{m \times dim} \tag{16}$$

$$z_i = LayerNorm\left(u_i + z_i'; \gamma_2, \beta_2\right),\ where\ \gamma_2, \beta_2 \in \mathbb{R}^{dim} \tag{17}$$

In (12-17), x_i is used to represent the i^{th} token from a sequence of length n where each token from the sequence is represented using the vector embedding of dimension dim specific to the word embeddings used with added position encoding. The symbols γ, β are used from the layer normalization function as mentioned in [55]. Equations (12) and (13) are essentially representing the multi headed self-attention mechanism where h is one of the heads from a total of H heads. The symbol $\alpha_{i,j}^{(h)}$ is used to represent the attention weights, which is used to calculate the value of attention that x_i has with x_j with respect to head h. The operations being performed in (14) are a more detailed view of the operations from (11). Equations (15) and (17) represent the layer normalization mechanism with residual connections [56], with (16) representing the fully connected layer. The resultant output z_i is of the same dimension as the input embedding used, which here is represented by dim. The symbol m is used to represent the dimension of fully connected layer or the feed forward network. Each word embedding representation is passed to its own transformer encoder layer. Multi-headed attention and feed forward network layer with residual connections and normalization enrich the word embeddings with additional information. These word embedding vectors from transformer encoder layer are then passed to their own BiLSTM layer as shown in Fig. 5, where each hidden state in sequence is used as output. If we are using combination of different word embedding vectors, the resultant vector matrices are stacked and then passed through a dropout layer. In case of single word embedding, no stacking is used before passing to a dropout layer. The output from dropout layer is then used as an input to CNN layer, which includes convolution operation using different window length in various filters, max overtime pooling operation. The concatenated output from CNN layer is passed through a dropout layer for purpose of regularization. Then a fully connected layer along with a softmax layer is used. With this model, we have used cross-entropy loss minimization as training objective.

IV. EXPERIMENTS AND EVALUATIONS

We have carried out all the experiments using the Google Colaboratory platform. All the experiments were carried out using the GPUs provided by the Google Colaboratory. We have used Word2Vec, GloVe and FastText as the three pre-trained word embeddings for all the models. To provide a baseline for comparison we have used randomized embeddings (*Rand)* with all the exploited models to get classification results where the word embeddings are arbitrarily initialized. The Rand embedding returns a tensor filled with random integers generated uniformly between inclusive and exclusive values for a given sentence. The dimension of embeddings used were same for all the three variants being 300. When using more than one embedding for word vectorization, we have treated each as separate entity while being passed through initial layers of BiLSTM or transformer encoders. Embeddings before being applied to the CNN layer, we have stacked them as three channels input to CNN layer. The hyperparameters which obtained best result for CNN model used as standalone and kept same for subsequent models where, CNN was used as local feature extractor in later part. We used precision, recall, F1 Score, and accuracy as metrics for evaluating the performance of all the models. The Tables 1, 2, 3, 4 and 5 show the hyperparameter used for the implemented models.

TABLE 1. The hyperparameter setting for CNN model.

Parameter	Value	Parameter	Value
Embedding Dimension	300	Filter Window Size	[3,4,5]
Batch Size	128	Number of Filters	[100,100,100]
Epochs	15	Optimizer	Adam
Dropout	0.3	Activation Function	ReLU
Loss	Cross-entropy	L2 Regularization	0.001

TABLE 2. The hyperparameter setting for CNN-BiLSTM model.

Parameter	Value	Parameter	Value
Embedding Dimension	300	Epochs	15
Batch Size	128	Optimizer	Adam
BiLSTM Hidden Size	150	Dropout	0.3
L2 Regularization	0.001	CNN Window Size	[3,4,5]
CNN Filters	[100,100,100]	Loss	Cross-entropy

TABLE 3. The hyperparameter setting for transformer-CNN model.

Parameter	Value	Parameter	Value
Embedding Dimension	300	Epochs	15
Batch Size	128	Optimizer	Adam
Transformer Encoder Batch First	True	Transformer Encoder Input Dimension	300
Transformer Encoder Attention Heads	15	Transformer Encoder Feedforward Dimension	1200
CNN Filters	[100,100,100]	CNN Window Size	[3,4,5]
Dropout	0.3	L2 Regularization	0.001

A. EVALUATION AND RESULTS

The employed models with different input embeddings are evaluated using accuracy, precision, recall and F1 Score. The models are trained for binary classification task, hence these metrics help evaluate the performance when assigning predicted emotional label to the input sentences. Precision is used to represent number of correctly predicted positive label sentences out of all the sentences predicted to have positive label. Recall is the number of truly positive labeled sentences out of all the positive sentences in the dataset. The harmonic mean of precision and recall is used to calculate F1 Score. Equations (18), (19), (20) and (21) are mathematical representations of the above-mentioned performance metrics.

$$Precision = \frac{TP}{TP + FP} \tag{18}$$

$$Recall = \frac{TP}{TP + FN} \tag{19}$$

$$F1Score = 2 * \frac{Precision \times Recall}{Precision + Recall} \tag{20}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{21}$$

True Positive (TP) signifies positive labeled sentences predicted as positive, True Negative (TN) signifies negative labelled sentence predicted as negative, False Negative (FN)

TABLE 4. The hyperparameter settings for transformer-BiLSTM-CNN model.

Parameter	Value	Parameter	Value
Embedding Dimension	300	Epochs	15
Batch Size	128	Optimizer	Adam
Transformer Encoder Batch First	False	Transformer Encoder Input Dimension	300
Transformer Encoder Attention Heads	15	Transformer Encoder Feedforward Dimension	1200
BiLSTM Hidden Size	150	BiLSTM Input Size	300
CNN Filters	[100,100,100]	CNN Window Size	[3,4,5]
1st Dropout	0.3	2nd Dropout	0.3
L2 Regularization	0.001	Loss	Cross-entropy

TABLE 5. The hyperparameter setting for BiLSTM-CNN-BiLSTM model.

Parameter	Value	Parameter	Value
Embedding Dimension	300	Epochs	15
Batch Size	128	Optimizer	Adam
1st BiLSTM Hidden Size	150	1st BiLSTM Input Size	300
1st BiLSTM Batch First	True	1st Dropout	0.3
CNN Filters	[100,100,100]	CNN Window Size	[3,4,5]
2nd BiLSTM Input Size	300	2nd BiLSTM Hidden Size	150
2nd Dropout	0.3	L2 Regularization	0.001

signifies sentences predicted negative but are positive, False Positive (FP) represents negative sentences misclassified as positive. The results of different models with the embeddings used are presented in Table 6.

B. FINDINGS

We have exploited different models for sentiment analysis and used different word embeddings to give a complete view of impact of different layers and embeddings. The pairing of different deep learning layers leveraging specific information from text using their own unique architectural design and its impact to the process of sentiment analysis is fascinating. The detailed findings with employed models are as follow:

1) The best performing model among the ones we proposed in terms of precision, recall, F1 Score and accuracy in non-pretrained category is Transformer-BiLSTM-CNN model. Using the combination of transformer layer providing self-attention, BiLSTM layer helping leverage sequential information and CNN layer capturing additional features using its many filters provide better accuracy than other exploited models.

2) The choice of word embedding has a profound impact on accuracy of models where different word embeddings can impact the final performance of the same model. Though we have used the word embeddings of same dimension, the difference in architectural design used in calculating these word embeddings during pre-training phases can cause accuracy to vary by 3%.

3) The word embedding giving the best accuracy in almost all the models is Word2Vec, with GloVe coming second with almost the same performance. FastText embedding though providing word embedding even for out of vocabulary words by using its character-based n-gram model performed the worst with almost all the models.

4) Also using the combination of different word embeddings by using them as channels in CNN model had varied impact. With a few of the models, the use of combination of word embedding gave better result than the best performing single word embedding and with a few it performed slightly

TABLE 6. Comprehensive results of different models with different word embeddings compared on Precision, Recall, F1 Score and Accuracy.

Model	Embedding	Precision	Recall	F1 Score	Accuracy
CNN-BiLSTM	Word2Vec	85.24	85.17	85.19	85.27
	GloVe	83.51	83.46	83.47	83.52
	FastText	82.69	82.66	82.67	82.71
	Word2Vec+GloVe+FastText	85.31	85.26	85.20	85.32
	Embedding (Rand)	81.65	81.62	81.63	81.64
BiLSTM-CNN-BiSLTM	Word2Vec	83.84	83.81	83.82	83.93
	GloVe	84.19	84.20	84.17	84.21
	Word2Vec+GloVe+FastText	84.74	84.74	84.74	84.85
	Embeddings (Rand)	80.40	80.24	80.14	80.16
CNN	Word2Vec	84.29	84.28	84.28	84.43
	GloVe	84.15	84.01	84.03	84.15
	FastText	82.24	81.91	81.93	81.99
	Word2Vec+GloVe+FastText	85.44	85.45	85.43	85.54
	Embeddings (Rand)	79.40	79.17	79.18	79.24
Transformer-CNN	Word2Vec	85.92	85.92	85.89	86.09
	GloVe	84.33	84.11	84.13	84.29
	FastText	83.85	83.77	83.71	83.84
	Word2Vec+GloVe+FastText	84.84	84.70	84.72	84.85
	Embeddings (Rand)	83.09	82.98	83.00	83.03
Transformer-Transformer-CNN	Word2Vec	86.24	86.25	86.24	86.43
	GloVe	84.40	84.39	84.40	84.44
	FastText	84.77	84.57	84.60	84.79
	Word2Vec+GloVe+FastText	85.62	85.11	85.13	85.24
	Embeddings (Rand)	83.53	83.45	83.47	83.49
Transformer-BiLSTM-CNN	**Word2Vec**	**88.99**	**88.91**	**89.01**	**89.04**
	GloVe	85.27	85.18	85.22	85.32
	FastText	84.95	84.96	84.95	84.97
	Word2Vec+GloVe+FastText	85.17	85.02	85.09	85.11
	Embeddings (Rand)	84.62	84.28	84.15	84.19

worse than the best performing word embedding. Further, utilization of Rand embeddings achieved lesser accuracy as compared to other exploited embeddings. This is due to arbitrary initialization of word embeddings by Rand embeddings.

5) The addition of BiLSTM layer when used in back of CNN layer or in combination as both front and back layer of CNN model had no additional impact on performance of the core CNN layer, which on its own has a good accuracy. In both the models the accuracy decreased in comparison to the CNN layer in our implementations.

6) The use of transformer layer positively impacted the sentiment classification task. The use of transformer layer as an input layer to CNN increased its performance by leaps and bounces. We also used two transformer layers, which further increased the accuracy during our experimentation. Adding additional transformer layer after the first two layers had no impact in improvement of the accuracy. This can be attributed to the need for more training data with increase in transformer layers.

7) The model architectures such as, BERT and BMT-NET, etc. that have pre-trained language models as major component achieves better results as compared to the best performing model presented in this work. This is due to their size that is, having a large number of encoder transformer layers as compared to the models exploited in this work and the scale of pre-training on large corpus they have received.

C. COMPARISON WITH RELATED WORK

We have compared the performance of our proposed models with other state-of-the-art models proposed in recent years

TABLE 7. Comparison of accuracy of our models compared to others on SST-2 dataset.

Work	Model	Accuracy
Khan et al. [20]	Using DNN models with GloVe and FastText with mRMR and PCA for feature selection	85.10
He et al. [15]	Using LSTM-CRF network for word segmentation combined with CNN model	87.60
Zhang et al. [32]	Parts of speech embedding with self-attention mechanism	87.99
Khasanah [16]	BiLSTM and CNN models with FastText embedding	83.90
Huang et al. [33]	Jointly trained sentiment aware transformer	84.34
Giménez et al. [24]	Convolution network with semantically padded texts	82.45
Usama et al. [28]	Attention with Convolution and RNN	89.62
Wang et al. [26]	Combination of Convolution and Recurrent neural network	89.95
Wang et al. [35]	Using BiLSTM+ELMo+Attention	90.20
Devlin et al. [36]	BERT	93.50
Zhang et al. [38]	BMT-NET	94.00
Ours	BiLSTM-CNN-BiLSTM	84.85
Ours	Transformer-CNN	86.09
Ours	Transformer-Transformer-CNN	86.43
Ours	Transformer-BiLSTM-CNN	**89.04**

with best performing architectural implementation of their models. We have used the results from our model with the best performing word embedding combination. The results of compared accuracy are presented in Table 7. The reason of the higher performance with the state-of-the-art models is due to their large architectures and pre-training on large corpus consisting of text from varying domains and languages.

V. CONCLUSION

We have performed comparison between different deep learning models by using combination of varied neural network layers. Implementing different architectures to focus on specific aspect of natural language processing problem and the subsequent application in textual sentiment analysis. The impact of most common word embeddings used widely in research papers on the models have been studied and found Word2Vec embedding performs best in all the cases. The combination of word embedding as channels to CNN model had slight impact on the accuracy of model but the result deviated for better in some model and caused accuracy to decrease in other. Our proposed model, Transformer-BiLSTM-CNN gives comparable accuracy to other model of its size by achieving a value of 89.04%. BERT, ELMo, and other models built on their architecture have been pre-trained on massive corpus and are bigger than our model when measured against trainable parameters thereby, resulting in accuracy differences. The Transformer-BilSTM-CNN performing best in this work has only 6M parameters when compared to $\text{BERT}_{\text{base}}$ having 110M parameters and ELMo's (original) having 93.6M parameters. The proposed Transformer-BiLSTM-CNN model with a larger number of encoder transformer layers and pre-training on a large corpus can produce better accuracy results.

Furthermore, in context to our experimentations, the BiLSTM layer when used solely with CNN model did not perform as per our expectations, both CNN-BiLSTM and BiLSTM-CNN-BiLSTM had no visible impact on accuracy as it increased for some embedding while decreasing in other word embedding, in both cases with no high margin. The comparative analysis by using different embeddings can be beneficial to other researchers when choosing embeddings for their models. The proposed models can have an impact in booming field of natural language processing in general and textual sentiment analysis in particular. The accuracy achieved by proposed model can help industries relying specifically on sentiment analysis models of smaller size than other large pre-trained language models for tasks such as filtering content on platform can provide better recommendations to users suiting their business needs with minimal computational resources utilization.

ACKNOWLEDGMENT

The authors are thankful to the Vellore Institute of Technology, Chennai, India for providing necessary support to carry out this research work.

REFERENCES

[1] L. Li, D. Novillo-Ortiz, N. Azzopardi-Muscat, and P. Kostkova, "Digital data sources and their impact on people's health: A systematic review of systematic reviews," *Frontiers Public Health*, vol. 9, May 2021, Art. no. 645260. Accessed: Sep. 1, 2022. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fpubh.2021.645260/full

[2] Y. Gorodnichenko, T. Pham, and O. Talavera, "Social media, sentiment and public opinions: Evidence from #Brexit and #USElection," *Eur. Econ. Rev.*, vol. 136, Jul. 2021, Art. no. 103772, doi: 10.1016/j.euroecorev.2021.103772.

[3] B. Liu, "The problem of sentiment analysis," in *Sentiment Analysis and Opinion Mining* (Synthesis Lectures on Human Language Technologies), 1st ed. Cham, Switzerland: Springer, 2012, ch. 2, pp. 9–22. [Online]. Available: https://link.springer.com/book/10.1007/978-3-031-02145-9

[4] G. Brauwers and F. Frasincar, "A survey on aspect-based sentiment classification," *ACM Comput. Surv.*, vol. 55, no. 4, pp. 1–37, Apr. 2023. Accessed: Oct. 22, 2022, doi: 10.1145/3503044.

[5] E. Cambria, B. Schuller, Y. Xia, and C. Havasi, "New avenues in opinion mining and sentiment analysis," *IEEE Intell. Syst.*, vol. 28, no. 2, pp. 15–21, Mar./Apr. 2013, doi: 10.1109/MIS.2013.30.

[6] A. R. Pathak, B. Agarwal, M. Pandey, and S. Rautaray, "Application of deep learning approaches for sentiment analysis," in *Deep Learning-Based Approaches for Sentiment Analysis*, B. Agarwal, R. Nayak, N. Mittal, and S. Patnaik, Eds. Singapore: Springer, 2020, pp. 1–31.

[7] Y. Kim, "Convolutional neural networks for sentence classification," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, Doha, Qatar, 2014, pp. 1746–1751. [Online]. Available: https://aclanthology.org/D14-1181

[8] M. J. Er, Y. Zhang, N. Wang, and M. Pratama, "Attention pooling-based convolutional neural network for sentence modelling," *Inf. Sci.*, vol. 373, pp. 388–403, Dec. 2016, doi: 10.1016/j.ins.2016.08.084.

[9] P. Liu, X. Qiu, and X. Huang, "Recurrent neural network for text classification with multi-task learning," 2016, *arXiv:1605.05101*.

[10] T. Chen, R. Xu, Y. He, and X. Wang, "Improving sentiment analysis via sentence type classification using BiLSTM-CRF and CNN," *Expert Syst. Appl.*, vol. 72, pp. 221–230, Apr. 2017, doi: 10.1016/j.eswa.2016.10.065.

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017, *arXiv:1706.03762*.

[12] A. Kumar and R. Rastogi, "Attentional recurrent neural networks for sentence classification," in *Innovations in Infrastructure* (Advances in Intelligent Systems and Computing), vol. 757, D. Deb, V. Balas, and R. Dey, Eds. Singapore: Springer, Sep. 2019, pp. 549–559. [Online]. Available: https://link.springer.com/chapter/10.1007/978-981-13-1966-2_49

[13] Z. Drus and H. Khalid, "Sentiment analysis in social media and its application: Systematic literature review," *Proc. Comput. Sci.*, vol. 161, pp. 707–714, Jan. 2019, doi: 10.1016/j.procs.2019.11.174.

[14] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, *arXiv:1301.3781*.

[15] M. He, C. Ma, and R. Wang, "A data-driven approach for university public opinion analysis and its applications," *Appl. Sci.*, vol. 12, no. 18, p. 9136, Sep. 2022, doi: 10.3390/app12189136.

[16] I. N. Khasanah, "Sentiment classification using fastText embedding and deep learning model," *Proc. Comput. Sci.*, vol. 189, pp. 343–350, Jan. 2021, doi: 10.1016/j.procs.2021.05.103.

[17] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Trans. Assoc. Comput. Linguistics*, vol. 5, pp. 135–146, Dec. 2017. Accessed: Sep. 10, 2022, doi: 10.1162/tacl_a_00051.

[18] M. Kasri, M. Birjali, and A. Beni-Hssane, "Word2Sent: A new learning sentiment-embedding model with low dimension for sentence level sentiment classification," *Concurrency Comput., Pract. Exp.*, vol. 33, no. 9, May 2021, Art. no. e6149, doi: 10.1002/cpe.6149.

[19] S. Baccianella, A. Esuli, and F. Sebastiani, "SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining," in *Proc. LREC*, Valletta, Malta, 2010, pp. 2200–2204.

[20] J. Khan, N. Ahmad, A. Alam, and Y. Lee, "Leveraging semantic and sentiment knowledge for user-generated text sentiment classification," in *Proc. 8th Workshop Noisy User-Generated Text (W-NUT)*, Gyeongju, South Korea, 2022, pp. 101–105. [Online]. Available: https://aclanthology.org/2022.wnut-1.11

[21] C. Ding and H. Peng, "Minimum redundancy feature selection from microarray gene expression data," *J. Bioinf. Comput. Biol.*, vol. 3, no. 2, pp. 185–205, Apr. 2005, doi: 10.1142/s0219720005001004.

[22] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics Intell. Lab. Syst.*, vol. 2, nos. 1–3, pp. 37–52, 1987.

[23] J. Khan and Y.-K. Lee, "LeSSA: A unified framework based on lexicons and semi-supervised learning approaches for textual sentiment classification," *Appl. Sci.*, vol. 9, no. 24, Dec. 2019, Art. no. 5562, doi: 10.3390/app9245562.

[24] M. Gimnez, J. Palanca, and V. Botti, "Semantic-based padding in convolutional neural networks for improving the performance in natural language processing. A case of study in sentiment analysis," *Neurocomputing*, vol. 378, pp. 315–323, Feb. 2020, doi: 10.1016/j.neucom.2019.08.096.

[25] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder–decoder approaches," 2014, *arXiv:1409.1259*.

[26] X. Wang, W. Jiang, and Z. Luo, "Combination of convolutional and recurrent neural network for sentiment analysis of short texts," in *Proc. COLING*, Osaka, Japan, 2016, pp. 2428–2437. [Online]. Available: https://aclanthology.org/C16-1229

[27] G. Liu and J. Guo, "Bidirectional LSTM with attention mechanism and convolutional layer for text classification," *Neurocomputing*, vol. 337, pp. 325–338, Apr. 2019, doi: 10.1016/j.neucom.2019.01.078.

[28] M. Usama, B. Ahmad, E. Song, M. S. Hossain, M. Alrashoud, and G. Muhammad, "Attention-based sentiment analysis using convolutional and recurrent neural network," *Future Gener. Comput. Syst.*, vol. 113, pp. 571–578, Dec. 2020, doi: 10.1016/j.future.2020.07.022.

[29] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014, *arXiv:1409.0473*.

[30] Q. Zhu, X. Jiang, and R. Ye, "Sentiment analysis of review text based on BiGRU-attention and hybrid CNN," *IEEE Access*, vol. 9, pp. 149077–149088, 2021, doi: 10.1109/ACCESS.2021.3118537.

[31] K. Xu, B. Liu, J. Tao, Z. Lv, C. Fan, and L. Song, "AHRNN: Attention-based hybrid robust neural network for emotion recognition," *Cogn. Comput. Syst.*, vol. 4, no. 1, pp. 85–95, Mar. 2022, doi: 10.1049/ccs2.12038.

[32] R. Zhang, B. Chen, and H. Tang, "A novel sentiment classification architecture based on self-attention mechanism," in *Proc. Int. Conf. Data Process. Techn. Appl. Cyber-Phys. Syst.*, in Advances in Intelligent Systems and Computing, vol. 1379, C. Huang, Y. W. Chan, and N. Yen, Eds. Singapore: Springer, 2021, pp. 685–692.

[33] H. Huang, Y. Jin, and R. Rao, "Sentiment-aware transformer using joint training," in *Proc. IEEE 32nd Int. Conf. Tools Artif. Intell. (ICTAI)*, Nov. 2020, pp. 1154–1160, doi: 10.1109/ICTAI50040.2020.00175.

[34] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, vol. 1, New Orleans, LA, USA, 2018, pp. 2227–2237. [Online]. Available: https://aclanthology.org/N18-1202.pdf

[35] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman, "GLUE: A multi-task benchmark and analysis platform for natural language understanding," in *Proc. EMNLP Workshop BlackboxNLP, Analyzing Interpreting Neural Netw. NLP*, 2019, pp. 353–355, doi: 10.18653/v1/W18-5446.

[36] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, vol. 1, Minneapolis, MN, USA, 2019, pp. 4171–4186. [Online]. Available: https://aclanthology.org/N19-1423

[37] M. Munikar, S. Shakya, and A. Shrestha, "Fine-grained sentiment classification using BERT," in *Proc. Artif. Intell. Transforming Bus. Soc. (AITB)*, Nov. 2019, pp. 1–5, doi: 10.1109/AITB48515.2019.8947435.

[38] T. Zhang, X. Gong, and C. L. P. Chen, "BMT-Net: Broad multitask transformer network for sentiment analysis," *IEEE Trans. Cybern.*, vol. 52, no. 7, pp. 6232–6243, Jul. 2022, doi: 10.1109/TCYB.2021.3050508.

[39] C. L. P. Chen and Z. L. Liu, "Broad learning system: An effective and efficient incremental learning system without the need for deep architecture," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 1, pp. 10–24, Jan. 2018, doi: 10.1109/TNNLS.2017.2716952.

[40] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, Seattle, WA, USA, 2013, pp. 1631–1642. [Online]. Available: https://aclanthology.org/D13-1170

[41] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, Doha, Qatar, 2014, pp. 1532–1543. [Online]. Available: https://aclanthology.org/D14-1162

[42] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up? Sentiment classification using machine learning techniques," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, Philadelphia, PA, USA, 2002, pp. 79–86. [Online]. Available: https://aclanthology.org/W02-1011

[43] D. Klein and C. D. Manning, "Accurate unlexicalized parsing," in *Proc. 41st Annu. Meeting Assoc. Comput. Linguistics*, Sapporo, Japan, 2003, pp. 423–430. [Online]. Available: https://aclanthology.org/P03-1054

[44] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *J. Mach. Learn. Res.*, vol. 12, no. 76, pp. 2493–2537, 2011. Accessed: Sep. 29, 2022. [Online]. Available: https://www.jmlr.org/papers/volume12/collobert11a/collobert11a.pdf

[45] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 56, pp. 1929–1958, 2014. Accessed: Oct. 5, 2022. [Online]. Available: https://dl.acm.org/doi/10.5555/2627435.2670313

[46] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[47] L. Xiaoyan, R. C. Raga, and S. Xuemei, "GloVe-CNN-BiLSTM model for sentiment analysis on text reviews," *J. Sensors*, vol. 2022, Oct. 2022, Art. no. 7212366, doi: 10.1155/2022/7212366.

[48] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., Dec. 2019, pp. 8024–8035. [Online]. Available: https://papers.nips.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf

[49] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition," 2014, *arXiv:1402.1128*.

[50] V. Pham, T. Bluche, C. Kermorvant, and J. Louradour, "Dropout improves recurrent neural networks for handwriting recognition," in *Proc. 14th Int. Conf. Frontiers Handwriting Recognit.*, Hersonissos, Greece, Sep. 2014, pp. 285–290, doi: 10.1109/ICFHR.2014.55.

[51] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Lisbon, Portugal, 2015, pp. 1412–1421. [Online]. Available: https://aclanthology.org/D15-1166/

[52] J. Alammar. *The Illustrated Transformer*. Accessed: Oct. 16, 2022. [Online]. Available: https://jalammar.github.io/illustrated-transformer/

[53] J.-L. Wu, Y. He, L.-C. Yu, and K. R. Lai, "Identifying emotion labels from psychiatric social texts using a bi-directional LSTM-CNN model," *IEEE Access*, vol. 8, pp. 66638–66646, 2020, doi: 10.1109/ACCESS.2020.2985228.

[54] J. Thickstun. *The Transformer Model in Equations*. Accessed: Feb. 13, 2023. [Online]. Available: https://johnthickstun.com/docs/transformers.pdf

[55] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016, *arXiv:1607.06450*.

[56] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.

MRIGANK SHUKLA was born in Raebareli, Uttar Pradesh, India. He is currently pursuing the B.Tech. degree in CSE with the Vellore Institute of Technology, Chennai, India.

He is an Amid Scholar with several laureates to his name in academics at the undergraduate education level. He is passionate about programming. His current research interests include deep learning and machine learning for natural language processing applications.

AKHIL KUMAR was born in New Delhi, India. He received the dual B.Tech. and M.Tech. degree in information technology from the University School of Information Technology, Guru Gobind Singh Indraprastha University, New Delhi, in 2012, and the Ph.D. degree in computer science from Himachal Pradesh University, Shimla, in 2021.

He is currently an Assistant Professor (Senior Grade) with the Vellore Institute of Technology, Chennai, India. His research interests include computer vision, natural language processing, machine learning, and deep learning.