TI-CNN: Convolutional Neural Networks for Fake News Detection

Team-7
Gaurav Kumar(214101018)
Jawade Anand Shyam(214101022)
Megha Dhruvil kumar(214101028)
Neha Afreen(214101034)

Introduction-

The problem of fake news has existed since the appearance of the printing press, but only gained a lot of momentum and visibility during the age of social media. This is due to the large audience, easy access and fast dissemination mechanism of social media, where more and more users are consuming news in a daily basis Traditional methods for verifying the veracity of news that rely on human experts, despite being reliable, do not scale well to the massive volume of news nowadays. This renders the automatic detection of fake news on social media an important problem, drawing a lot of attention from both the academic and industrial communities.

Deep learning (DL) has recently become an emerging technology among the research community and has proven to be more effective in recognizing fake news than traditional ML methods. DL has some particular advantages over ML, such as

a) automated feature extraction
b) lightly dependent on data pre-processing
c) ability to extract high-dimensional feature

From the survey we know that for detecting fake news along with text information, images play important role in news. Images in fake news are also different from that in real news. Cartoons, irrelevant images (mismatch of text and image, no face in political news) and altered low-resolution images are frequently observed in fake news. We have proposed model to consider both the image and text information in fake news detection. We capture the explicit feature of images and text and model is used for combine this feature and it used for identify the fake news.

Problem Statement-

Given a set of m news articles containing the text and image information, we can represent the data as a set of text-image tuples A = {(AT_i , AI_i)} . In the fake news detection problem, we want to predict whether the news articles in A are fake news or not. We can represent the label set as Y = {0,1}, where 1 denotes real news while 0 represents the fake news.

Related Work –

Deception detection is a hot topic in the past few years. Deception information includes scientific fraud, fake news, false tweets etc. Fake news detection is a subtopic in this area. A TICNN based model has been proposed. In this model besides the explicit features, it innovatively utilize two parallel CNNs to extract latent features from both textual and visual information. And then explicit and latent features are projected into the same feature space to form new representations of texts and images. At last, it proposes to fuse textual and visual representations together for fake news detection

TI-CNN model consider both text and image information in fake news detection. Beyond the explicit features extracted from the data, as the development of the representative learning, convolutional neural networks are employed to learn the latent features which cannot be captured by the explicit features. Finally, it utilizes TI-CNN to combine the explicit and latent features of text and image information into a unified feature space, and then use the learned features to identify the fake news.

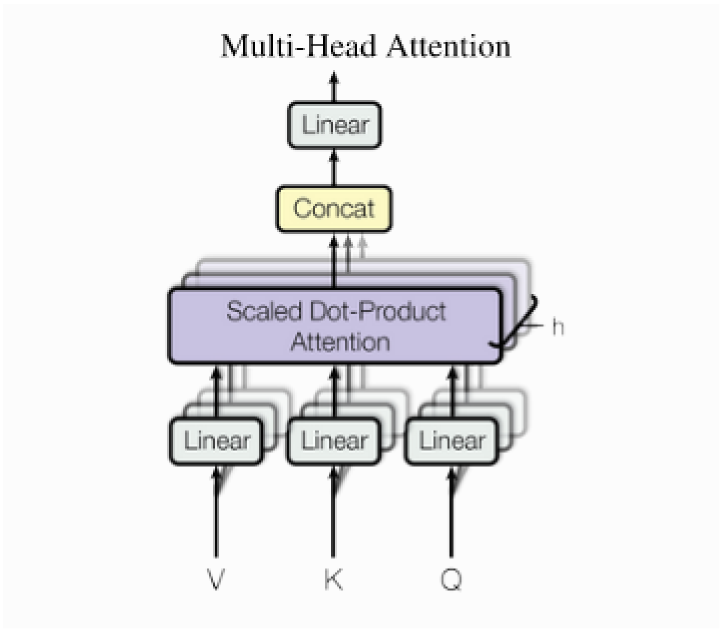
Overview of Implementation

1. Yolo:- An image is given input in yolo pre-trained model, which can identify the objects in the image and can give labels as output.
2. Bert:- Labels of the images is converted into a string and passed as input to Bert. It will convert the string into (number of words)*768 size of matrix. The text part of the news then passed as input to Bert and we will get same output as previous one.
3. Multi-head Attention:- Outputs from the Bert is then passed as input to multi-head attention.
4. Dense layer:- The output of the Multi-head Attention is then passed to the Dense layer.

Attention

Attention is mapping of set of key-values pair to an output, Where the query, keys, values and output all are vectors. The output is computed as a weighted sum of values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

Multi-Head Attention-

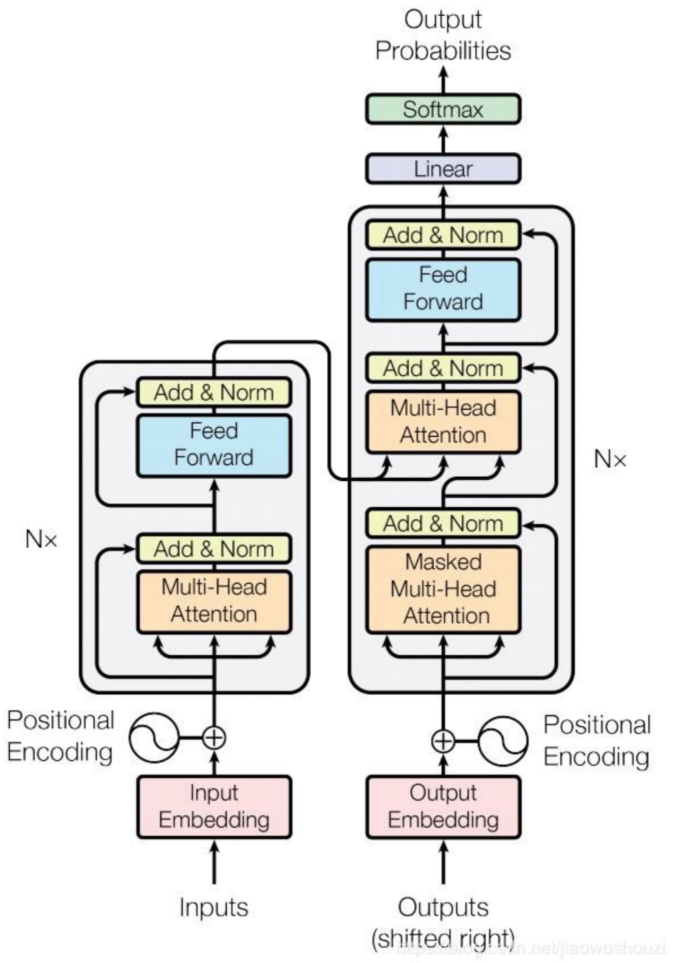


The scaled dot product attention allows a network to attend over a sequence. However, often there are multiple different aspects a sequence element wants to attend to, and a single weighted average is not a good option for it. This is why we extend the attention mechanisms to multiple heads, i.e. multiple different query-key-value triplets on the same features. Specifically, given a query, key, and value matrix, we transform those into sub-queries, sub-keys, and sub-values, which we pass through the scaled dot product attention independently. Afterward, we concatenate the heads and combine them with a final weight matrix. Mathematically, we can express this operation as:

$$\begin{aligned}\text{Multihead}(Q,K,V) &= \text{Concat}(\text{head}_1,\ldots,\text{head}_h)W^O\\ \text{where head}_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)\end{aligned}$$

We refer to this as Multi-Head Attention layer with the learnable parameters $W_{1\ldots h}^{Q} \in \mathbb{R}^{D\times d_k}$, $W_{1\ldots h}^{K} \in \mathbb{R}^{D\times d_k}$, $W_{1\ldots h}^{V} \in \mathbb{R}^{D\times d_v}$, and $W^{O} \in \mathbb{R}^{h\cdot d_k\times d_{out}}$ $(D$

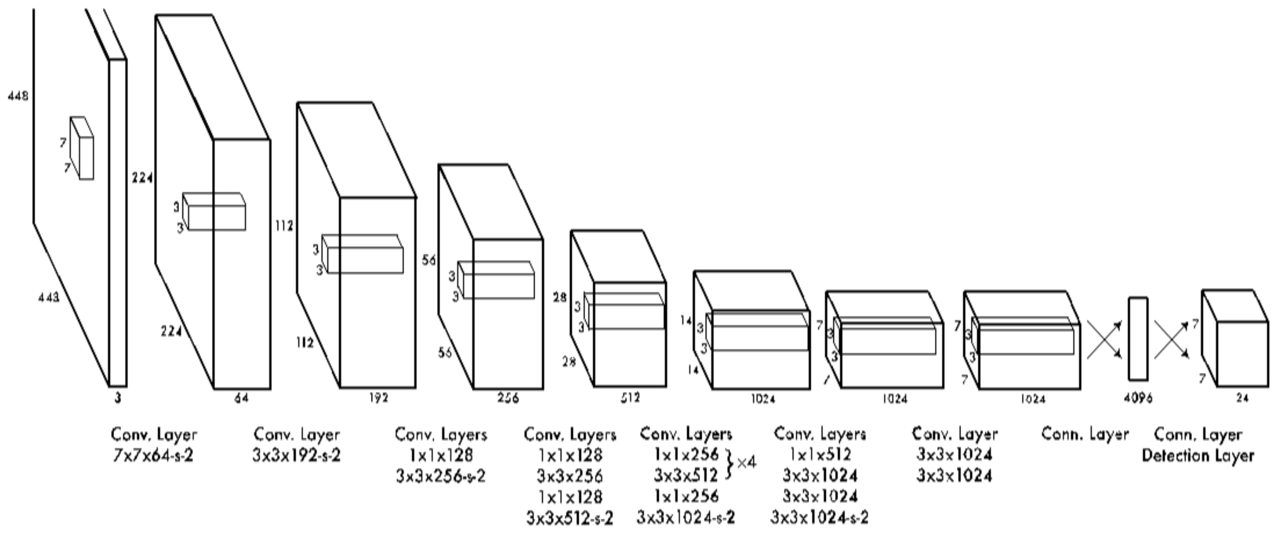
BERT



Bidirectional Encoder Representations from Transformers (BERT) is a transformer-based machine learning technique for natural language processing (NLP) pre-training developed by Google. BERT is at its core a transformer language model with a variable number of encoder layers and self-attention heads. BERT was pretrained on two tasks: language modelling (15% of tokens were masked and BERT was trained to predict them from context) and next sentence prediction (BERT was trained to predict if a chosen next sentence was probable or not given the first sentence). As a result of the training process, BERT learns contextual embeddings for words. After pretraining, which is computationally expensive, BERT can be finetuned with less resources on smaller datasets to optimize its performance on specific tasks.

YOLO Algorithm :

- YOLO is a state-of-the-art object detection algorithm that is incredibly fast and accurate
- The YOLO algorithm works by dividing the image into *N* grids, each having an equal dimensional region of S x S.
- Each of these *N* grids is responsible for the detection and localization of the object it contains. Correspondingly, these grids predict B bounding box coordinates relative to their cell coordinates, along with the object label and probability of the object being present in the cell.
- This process greatly lowers the computation as both detection and recognition are handled by cells from the image, but—
- It brings forth a lot of duplicate predictions due to multiple cells predicting the same object with different bounding box predictions. YOLO makes use of Non Maximal Suppression to deal with this issue.


448
448
3
224
224
64
112
112
192
56
56
256
28
28
512
14
14
1024
7
7
1024
7
7
1024
4096
7
7
24
Conv. Layer 7x7x64-s-2
Conv. Layer 3x3x192-s-2
Conv. Layers 1x1x128 3x3x256-s-2
Conv. Layers 1x1x128 3x3x256 1x1x128 3x3x512-s-2
Conv. Layers 1x1x256 3x3x512 }×4 1x1x256 3x3x1024-s-2
Conv. Layers 1x1x512 3x3x1024 3x3x1024 3x3x1024-s-2
Conv. Layer 3x3x1024 3x3x1024
Conn. Layer
Conn. Layer Detection Layer

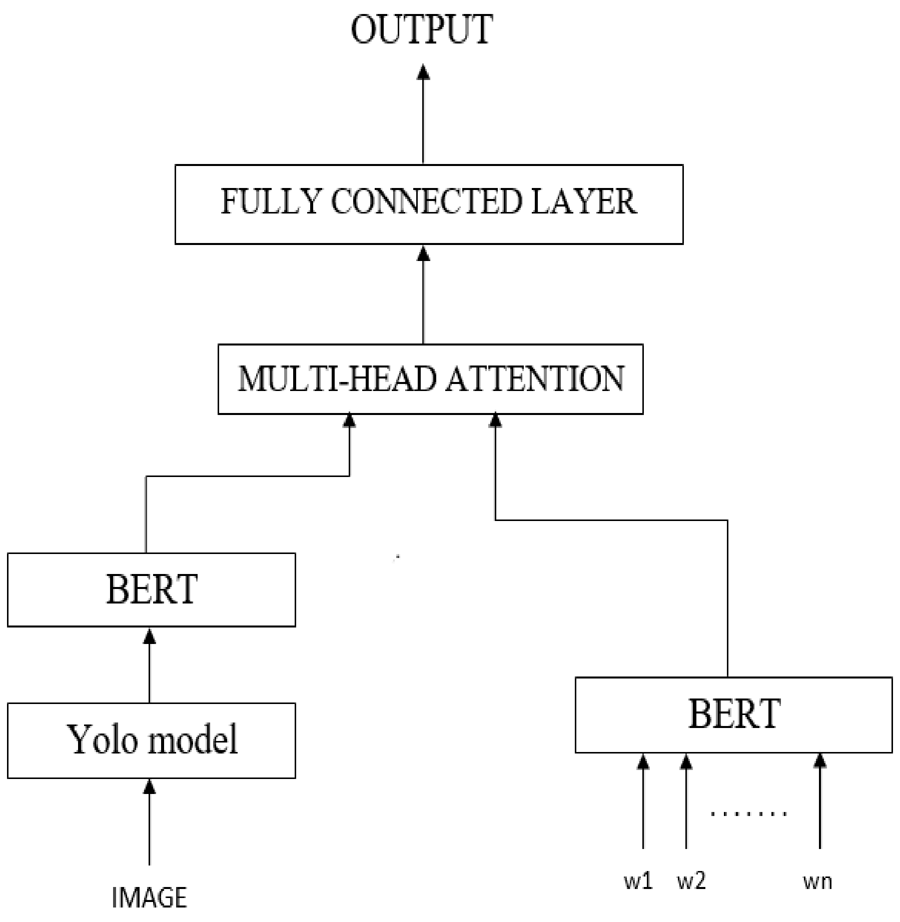
Proposed Model

In this model we have divided the input layer into two branches viz text branch and image branch .

Text branch: We have used Bert to convert text into vector form. After conversion every word will be a vector of size 768. If a sentence contains n words then the size of input vector formed will be nx768. As the number of words in different sentences may vary, we have applied padding at the end of every vector representation of sentences.

Image branch: We have used pre4trained yolo model which takes image as input and outputs list of labels related to that image. We then have converted those list into vector by passing those lists into bert. Again we have applied padding here to make them equal sized vectors.

Combining text and image branch: We have used multi-headed attention layer to combine both image and text part. We have taken vector from text branch as key and value. We have taken image as query for multiheaded attention layer.
Output from the attention layer is padded, flattened and then passed to the fully connected layer with input layer with (9984)13x768 neurons and output layer with 2 neurons. We have used softmax function in the output layer of fully connected layer.



BLOCK DIAGRAM

Experimental Result

Metrics	***Values***
Accuracy	80.96
Precision	72.38
Recall	70.53
F1 Score	71.44

References-

1. Short Text Sentiment Analysis Based on Multi-Channel CNN With Multi-Head Attention Mechanism
2. https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270
3. https://www.google.com/search?q=bert+diagram&tbm=isch&chips=q:bert+diagram,online_chips:transformer+architecture:EQGKk5KszJg%3D&hl=en&sa=X&ved=2ahUKEwj-37CE5c33AhWexnMBHRkTCa0Q4lYoAHoECAEQHQ&biw=1498&bih=730#imgrc=E5QY6Ky78lpUnM
4. https://towardsdatascience.com/transformers-explained-visually-part-3-multi-head-attention-deep-dive-1c1ff1024853