# Capstone Project

**Rohan Asnani**

# Machine Learning Engineer Nanodegree

**July 15, 2020**

## Efficient detection of deceptive content using Deep Learning

## 1. Definition

### Project Overview

Fake News or deception is an emerging topic that has received a lot of attention since the 2016 US presidential election, where many reckon that the spread of false information on social networks had a significant influence on its outcome. Most of the current work focuses either on the actual content of the news articles or the user that shares the news article on social media. However, social media platforms where fake news spread can be easily modelled as graphs and the goal of our project is to leverage techniques from deep Learning on Graphs for design better models for fake news detection.

### Problem Statement

The proliferation of digital fake news is one of the most pressing needs of modern society, with several academics as well as tech companies investing heavy resources for this task.

We categorize this as an Artificial Intelligence problem, and suggest Deep Learning methods to tackle it. The problem is viewed as the classification of a single news story based on the features of the online posts users make about the news, using Twitter15 and Twitter16 datasets to show the same.

## 2. Analysis

### Data Exploration

Several datasets and models of various neural nets have been implemented by the deceptive content detection community. A first step toward contributing to this research is to mix and match various datasets with models and observe the performance. This report summarizes such datasets and models used.

### Datasets

1. Fake News [1]
2. Fake or Real News [2]

3. Election Day Tweets [3]
4. Fake News Data [4]
5. LIAR
6. TWITTER15&16

| Fake News Dataset | |
|---|---|
| Columns/Attributes | 1. ID<br>2. Title<br>3. Author<br>4. Text<br>5. Label (0, 1) |
| Rows | 20,800 |
| Type | Content-based |
| Attributes considered | Title, Text, Label |
| Split ratio (train/val/test) | 16640 10400 10400 |

| | |
|---|---|
| Colu | |
| Rows | |
| Type | |
| Attri | |
| Split | |

| Election Day Tweets | |
|---|---|
| Columns/Attributes | 1. ID<br>2. Text<br>3. Retweet count<br>4. Username<br>5. (12 more) |
| Rows | 1327 |
| Type | Content-based |
| Attributes considered | text, label, retweet |
| Split ratio (train/val/test) | 1061 664 663 |

| | |
|---|---|
| Colu | |
| Rows | |
| Type | |
| Attri | |
| Split | |

## LIAR

| SPLIT | TRAIN | VALIDATION | TEST |
|---|---|---|---|
| SIZE | 9727 | 6144 | 4095 |
| FALSE | 1892 | 1218 | 776 |
| TRUE | 1582 | 1008 | 668 |
| MOSTLY-TRUE | 1870 | 1142 | 820 |
| HALF-TRUE | 2000 | 1261 | 853 |
| PANTS ON FIRE | 798 | 503 | 336 |
| BARELY TRUE | 1585 | 1012 | 642 |

| Column1 | barely_true_c | false_c | half_true_c | mostly_true_c | pants_on_fire_c |
|---|---|---|---|---|---|
| count | 10237 | 10237 | 10237 | 10237 | 10237 |
| mean | 11.53433623 | 13.28768194 | 17.13539123 | 16.43586988 | 6.202012308 |
| std | 18.97434865 | 24.11380828 | 35.84786188 | 36.15308885 | 16.12959871 |
| min | 0 | 0 | 0 | 0 | 0 |
| 25% | 0 | 0 | 0 | 0 | 0 |
| 50% | 2 | 2 | 3 | 3 | 1 |
| 75% | 12 | 12 | 13 | 11 | 5 |
| max | 70 | 114 | 160 | 163 | 105 |

DESCRIPTION OF THE LIARDATASET

TWITTER 15&16

For experimental evaluation, we use two publicly available Twitter datasets released by Ma et al. (2017), namely Twitter15 and Twitter164 , which respectively contains 1,381 and 1,181 propagation trees (see (Ma et al., 2017) for detailed statistics). In each dataset, a group of wide spread source tweets along with their propagation threads, i.e., replies and retweets, are provided in the form of tree structure. Each tree is annotated with one of the four class labels, i.e., non-rumor, false rumor, true rumor and unverified rumor. We remove the retweets from the trees since they do not provide any extra information or evidence contentwise. We build two versions for each tree, one for the bottom-up tree and the other for the top-down tree, by flipping the edges' direction.

# Current results

[1]

| | |
|---|---|
| Author | Yang Yang Beihang University Beijing, China and all |
| Model | TI-CNN (Text and Image information based Convolutional Neural Network |
| Detection | NEWS CONTENT AND CONTEXT BASED |
| Dataset | The dataset in this paper contains 20,015 news, i.e., 11,941 fake news and 8,074 real news. It is available online1. For fake news, it contains text and metadata scraped from more than 240 websites by the Megan Risdal on Kaggle2. The real news is crawled from the well known authoritative news websites, i.e., the New York Times, Washington Post, etc. The dataset contains multiple information, such as the title, text, image, author and website. To reveal the intrinsic differences between real and fake news, we solely use the title, text and image information. |
| Method | TI-CNN-1000 |
| Precision | 0.9220 |
| Recall | 0.9277 |
| F1-measure | 0.9210 |

[2]

| | |
|---|---|
| Author | ArnaudAutef Department of Management Science and Engineering |

| | |
|---|---|
| | arnaud15@stanford.edu and all |
| Model | GraphicalModels |
| Detection | NEWS CONTEXT BASED |
| Dataset | Twitter15 and Twitter16 |
| Method | -(SEIZ)ContagionModelforFakeNewsDetection<br>-GraphNeuralNetworks GNNs |
| Accuracy | TRAIN VAL TEST TEST BINARY CLASSIFICATION<br>Twitter 15 1.00 0.719 0.690 0.88<br>Twitter 16 0.859 0.841 0.750 0.88 |

[3]

| | |
|---|---|
| Author | Shivangi Singhal IIIT-Delhi Delhi, India shivangis@iiitd.ac.in and all |
| Model | SpotFake |
| Dataset | The training is performed on two publicly available datasets i.e., Twitter and Weibo |
| Detection | NEWS CONTENT AND CONTEXT BASED |
| Method | Textual Feature Extractor:- |

| | |
|---|---|
| | Bidirectional Encoder Representations from Transformers<br>(BERT) which is then passed through a fully-connected layer<br>Visual Feature Extractor:-<br>The pre-trained VGG-19. The second last layer of VGG-19 **convolutional network pre-trained on ImageNet dataset** (denoted by Vg) and pass it through a fully connected layer to reduce down to final dimension of length 32. |
| Dataset | Twitter Dataset |
| Accuracy | 0.7777 |
| | REAL FAKE |
| Precision | 0.751 0.832 |
| Recall | 0.900 0.606 |
| F1-measure | 0.82 0.701 |
| Dataset | Weibo Dataset |
| Accuracy | 0.8923 |
| | REAL FAKE |
| Precision | 0.902 0.847 |
| Recall | 0.964 0.656 |
| F1-measure | 0.932 0.739 |

[4]

| | |
|---|---|
| Author | Yaqing Wang, Department of Computer Science, the State University of New York at Buffalo, Buffalo, New York and all |
| Model | EventAdversarialNeuralNetwork (EANN) |
| Dataset | Twitter and Weibo |
| Detection | NEWS CONTEXT BASED |
| Method | Convolutional neural networks(CNN)as the core module for textual feature extractor and Event Discriminator |
| Dataset | Twitter Dataset |
| Accuracy | 0.715 |
| Precision | 0.822 |
| Recall | 0.638 |
| F1-measure | 0.719 |
| Dataset | Weibo Dataset |
| Accuracy | 0.827 |
| Precision | 0.847 |
| Recall | 0.812 |
| F1-measure | 0.829 |

[5]

| | |
|---|---|
| Author | Natali RuchanskyUniversityofSouthernCalifornia Los Angeles, California natalir@bu.edu and all |

| | |
|---|---|
| Model | CSI which is composed of three modules: Capture, Score, and Integrate |
| Dataset | Twitter and Weibo |
| Detection | NEWS CONTEXT BASED |
| Method | RecurrentNeuralNetwork(RNN) |
| Dataset | Twitter Dataset |
| Accuracy | 0.892 |
| F1-measure | 0.894 |
| Dataset | Weibo |
| Accuracy | 0.953 |
| F1-measure | 0.954 |

[6]

| | |
|---|---|
| Author | Feng Qian, Peking University and all |
| Model | Two-Level Convolutional Neural Network with User Response Generator (TCNN-URG) |
| Dataset | Weibo Dataset and self-collected |
| Method | TCNN-URG |
| Dataset | Weibo Dataset |
| % of all data used as training data | 10% 20% 30% 40% 50% 60% 70% 80% 90% |

| | |
|---|---|
| Accuracy | 79.00 84.52 85.51 86.26 88.05 88.41 88.43 88.56 89.84 |
| Dataset | Self-collected Dataset |
| % of all data used as training data | 10% 20% 30% 40% 50% 60% 70% 80% 90% |
| Accuracy | 77.47 77.71 79.38 81.92 83.98 86.13 86.68 88.28 88 |

# Model Architectures

## Introduction

Neural Network model architectures have recurring structures based on the type of neural network: such as convolutional, recurrent, etc. In this section the types of common layers, their purposes and functions are briefly discussed.

## CNN[5]

A Convolutional Neural Network is a deep learning algorithm that can recognize and classify features in data such as images or text. It is a multi-layer (deep when layers are dense) neural network designed to analyze visual inputs and perform tasks such as image classification, segmentation, object detection, or text classification, which are useful in various countless scenarios.[6]

A CNN is composed of several kinds of layers:

- **Convolutional layer:** creates a feature map to predict the class probabilities for each feature by applying a filter that scans the whole image, few pixels at a time.
- **Pooling layer (downsampling):** scales down the amount of information the convolutional layer generated for each feature and maintains the most essential information (the process of the convolutional and pooling layers usually repeats several times).
- **Fully connected input layer:** “flattens” the outputs generated by previous layers to turn them into a single vector that can be used as an input for the next layer.
- **Fully connected layer:** applies weights over the input generated by the feature analysis to predict an accurate label.

- **Fully connected output layer:** generates the final probabilities to determine a class for the image.

## LSTM[7]

The German researchers, Hochreiter and Schmidhuber, introduced the idea of long short-term memory networks in a paper published in 1997. LSTM is a unique type of Recurrent Neural Network (RNN) capable of learning long-term dependencies, which is useful for certain types of prediction that require the network to retain information over longer time periods, a task that traditional RNNs struggle with.[8]

The chain-like architecture of LSTM allows it to contain information for longer time periods, solving challenging tasks that traditional RNNs struggle to or simply cannot solve. The three major parts of the LSTM include:

- **Forget gate**—removes information that is no longer necessary for the completion of the task. This step is essential to optimizing the performance of the network.

- **Input gate**—responsible for adding information to the cells

- **Output gate**—selects and outputs necessary information

**Other Key Terms**

**Batch Normalization**: method to standardize inputs going into a network, in order to proceed the activations of a Preceding layer and its inputs directly.

**Regularization**: This is a form of regression, that constrains/ regularizes or shrinks the coefficient estimates towards zero. In other words, this technique discourages learning a more complex or flexible model, so as to avoid the risk of overfitting.[9]

# Model Architectures Used for Given Datasets

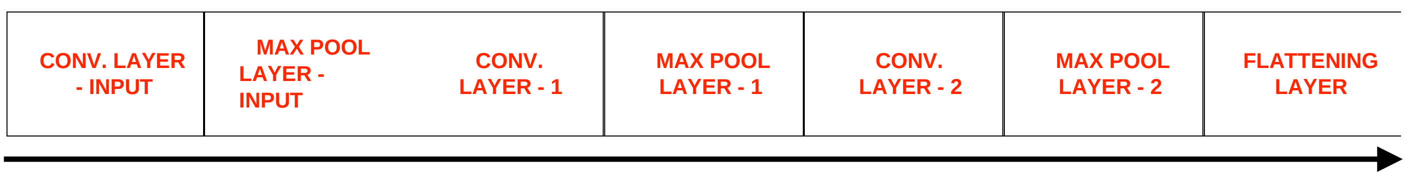
CNN Model

```
for fsz in filter_sizes:
l_conv =
Conv1D(nb_filter=128,filter_length=fsz,activation='relu')(embed-
ded_sequences)
    l_pool = MaxPooling1D(5)(l_conv)
    convs.append(l_pool)

l_merge = concatenate(convs, axis=1)
l_cov1= Conv1D(filters=128, kernel_size=5, activation='relu')
(l_merge)
l_pool1 = MaxPooling1D(5)(l_cov1)
l_cov2 = Conv1D(filters=128, kernel_size=5, activation='relu')
(l_pool1)
l_pool2 = MaxPooling1D(30)(l_cov2) l_flat
= Flatten()(l_pool2)
l_dense = Dense(128, activation='relu')(l_flat) preds
= Dense(2, activation='softmax')(l_dense)

model2 = Model(sequence_input, preds)
model2.compile(loss='categorical_crossentropy',
              optimizer='adadelta',
```

As we can see in this code snippet where the CNN model is being designed, we have the following layers in the CNN architecture for Fake News Dataset:

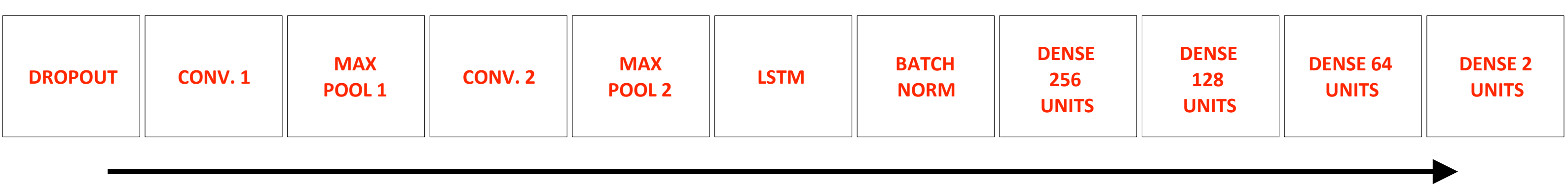



**Activation Function:** Input => ReLU

Output => softmax

**Loss Function:** Categorical Cross Entropy

**Optimizer:** Adadelta




LSTM Model

```
embedding_vecor_length = 32 modell
= Sequential()
modell.add(embedding_layer)
modell.add(Dropout(0.2))
modell.add(Conv1D(filters=32, kernel_size=5, padding='same', activa-
tion='relu'))
modell.add(MaxPooling1D(pool_size=2))
modell.add(Conv1D(filters=64, kernel_size=3, padding='same', activa-
tion='relu'))
modell.add(MaxPooling1D(pool_size=2))
modell.add(LSTM(100, dropout=0.2, recurrent_dropout=0.2))
modell.add(BatchNormalization())
modell.add(Dense(256, activation='relu'))
modell.add(Dense(128, activation='relu'))
modell.add(Dense(64, activation='relu'))
modell.add(Dense(2, activation='softmax'))
modell.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
```

As we can see in this code snippet where the LSTM model is being designed, we have the following layers in the LSTM architecture for Fake News Dataset: AFTER AN INITIAL *EMBEDDING LAYER*

**Activation Function:** Input => ReLU

Output => softmax

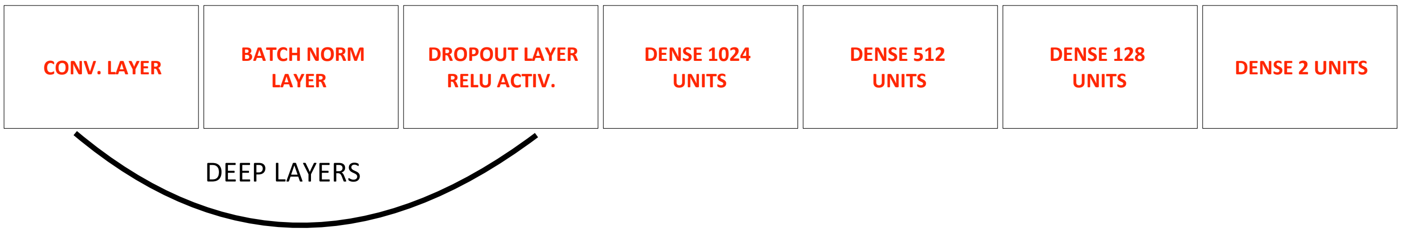
**Loss Function:** Categorical Cross Entropy **Optimizer:** Adam

DeepCNN

```
modell.add(embedding_layer) for i in range(0,2):     modell.add(Conv1D(filters=1024,
kernel_size=1, padding='same', activation='relu'))
modell.add(BatchNormalization())

for i in range(0,5):     modell.add(Conv1D(filters=32, kernel_size=5,
padding='same', activation='relu'))
    modell.add(BatchNormalization())
modell.add(Activation('relu'))
     for i in range(0,5):     modell.add(Conv1D(filters=64, kernel_size=3,
padding='same', activation='relu'))
    modell.add(BatchNormalization())
modell.add(Activation('relu'))
     for i in range(0,3):     modell.add(Conv1D(filters=64, kernel_size=5,
padding='same', activation='relu'))
    modell.add(BatchNormalization())
modell.add(Activation('relu'))
    modell.add(Conv1D(filters=128, kernel_size=3, padding='same', activation='relu'))
    modell.add(BatchNormalization())
modell.add(MaxPooling1D(pool_size=2))
modell.add(Activation('relu'))
     for i in range (0,7):     modell.add(Conv1D(filters=128, kernel_size=5,
padding='same', activation='relu'))
    modell.add(BatchNormalization())     modell.add(Activation('relu')) for i in
range (0,5):     modell.add(Conv1D(filters=256, kernel_size=3, padding='same',
activation='relu'))
    modell.add(BatchNormalization())
modell.add(Activation('relu'))
     for i in range (0,3):     modell.add(Conv1D(filters=256, kernel_size=5,
padding='same', activation='relu'))
    modell.add(BatchNormalization())
     for i in range (0,5):     modell.add(Conv1D(filters=512, kernel_size=3,
padding='same', activation='relu'))
    modell.add(BatchNormalization())
    modell.add(Dropout(0.1))
     for i in range(0,2):     modell.add(Conv1D(filters=768, kernel_size=5,
padding='same', activation='relu'))
    modell.add(BatchNormalization())     modell.add(MaxPooling1D(pool_size=2))
modell.add(Activation('relu')) for i in range(0,2):
modell.add(Conv1D(filters=1024, kernel_size=3, padding='same', activation='relu'))
    modell.add(BatchNormalization())
modell.add(Activation('relu'))


modell.add(Dense(1024, activation='relu'))
modell.add(Dense(512, activation='relu'))
modell.add(Dense(128, activation='relu'))
modell.add(Dense(2, activation='softmax'))
modell.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```




**Activation Function:** Input => ReLU

Output => softmax

**Loss Function:** Binary Cross Entropy

**Optimizer:** adam

**Challenge: - Develop an efficient deep neural network for detection of deceptive content**

**Dataset: -** Twitter15 and Twitter16 dataset

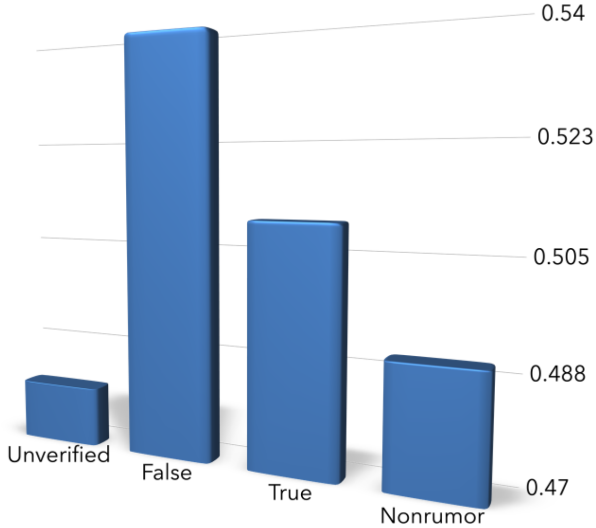
The datasets we work with are Twitter15 and Twitter16. These two datasets share the same exact structure. Both of them contain the tweets and re-tweets from a thousand of news articles published in 2015 and 2016. For each news article, the data contains the first tweet that shared it on Twitter, and a sequence of re-tweets following this initial post. We show one such data point (initial tweet and first two re-tweets). Each event is labeled according to the initial news article, the label is taken out of four possible classes: "true", "false", "unverified", "non-rumor". Labels are evenly distributed in both datasets**.**

## Previous benchmark Results: - Previous benchmark results (Till Now) are shown below and accuracy is not so high (approx. 69%) using twitter15 dataset.

| | Twitter15 | | | Twitter16 | | |
|---|---|---|---|---|---|---|
| **Split** | **Train** | **Val** | **Test** | **Train** | **Val** | **Test** |
| Recursive Tree[8] | NA | NA | 0.723 | NA | NA | 0.737 |
| RNN+CNN[3]* | NA | NA | 0.842 | NA | NA | 0.863 |
| GBDT_user | 0.962 | 0.629 | 0.628 | 1.00 | 0.671 | 0.647 |
| GBDT_seiz | 0.672 | 0.412 | 0.360 | 0.741 | 0.506 | 0.377 |
| Ens_GBDT | 0.959 | 0.635 | 0.577 | 0.995 | 0.617 | 0.618 |
| MLP text | 0.931 | 0.568 | 0.536 | 0.882 | 0.634 | 0.549 |
| LSTM text | 0.899 | 0.584 | 0.622 | 0.922 | 0.622 | 0.587 |
| GraphSage text | 0.954 | 0.624 | 0.622 | 0.866 | 0.756 | 0.712 |
| GCN all (Our best) | 1.00 | 0.719 | 0.690 | 0.859 | 0.841 | 0.750 |

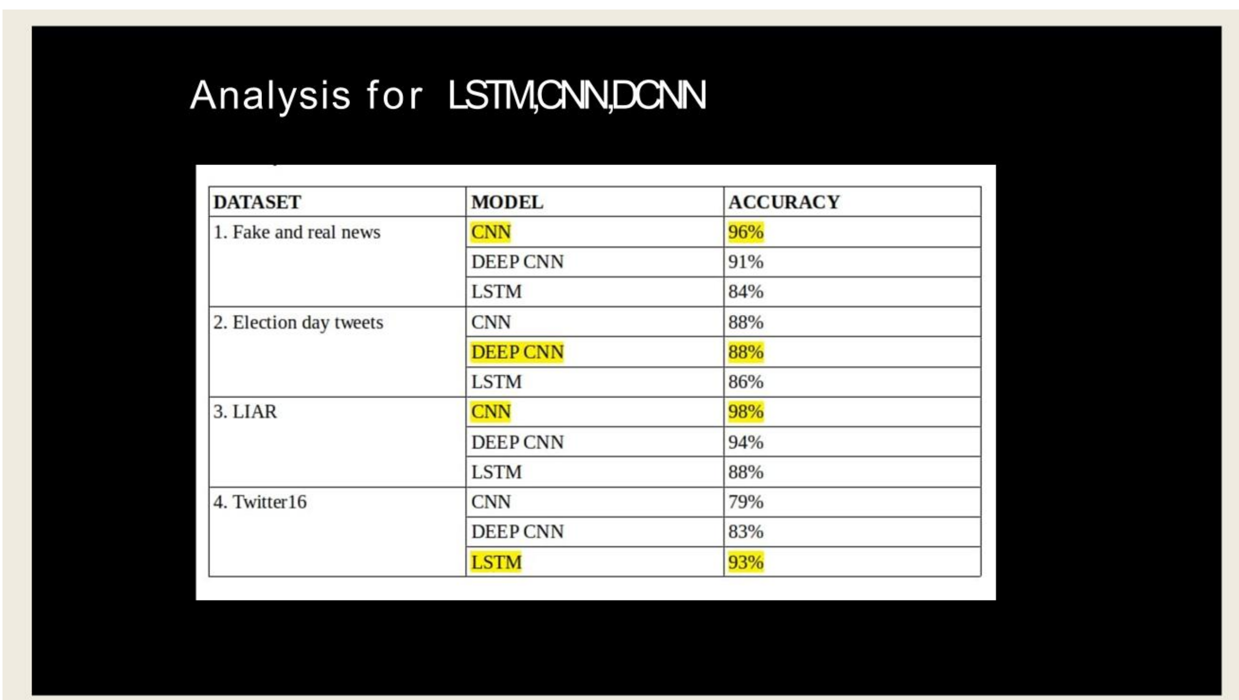
**my Goal: -**To apply deep learning models using ALL THE datasets for fake news Detection that are mentioned.

# Our results



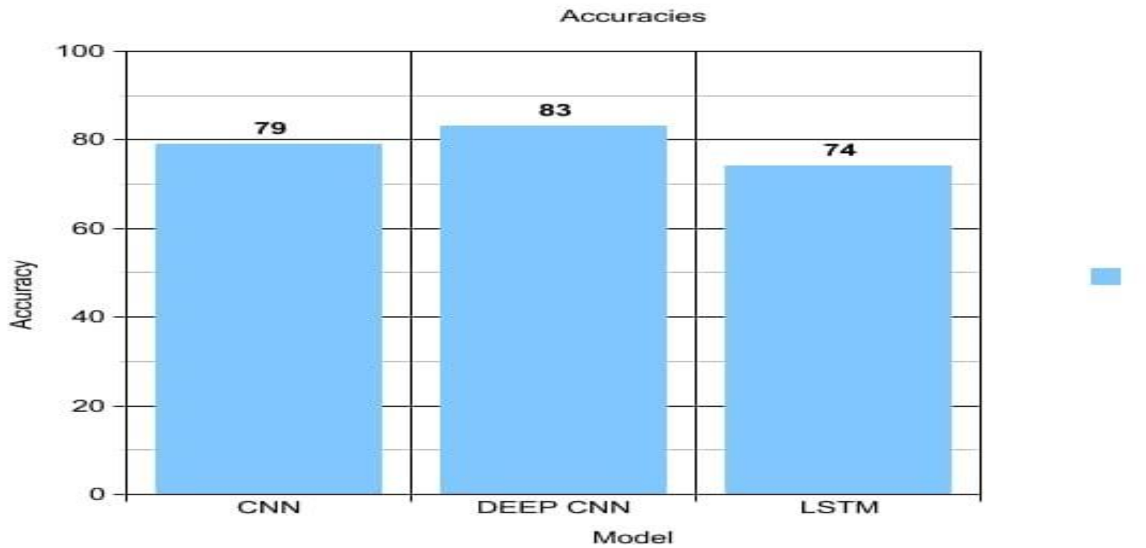

## Results

Observed results of the models we experimented along with proposed model.



Analysis for LSTM,CNN,DCNN

| DATASET | MODEL | ACCURACY |
|---|---|---|
| 1. Fake and real news | CNN | 96% |
| | DEEP CNN | 91% |
| | LSTM | 84% |
| 2. Election day tweets | CNN | 88% |
| | DEEP CNN | 88% |
| | LSTM | 86% |
| 3. LIAR | CNN | 98% |
| | DEEP CNN | 94% |
| | LSTM | 88% |
| 4. Twitter16 | CNN | 79% |
| | DEEP CNN | 83% |
| | LSTM | 93% |




# Future Research and Conclusion

Several sub-problems and pre-problems exist in the area of fake news detection, one such problem being to detect whether a post is important or dangerous enough to employ fake news detection. It is also a challenge to detect fake news prior to its propagation, when serious effects of it have little chance of having already taken place.

Deep Learning is greatly advantageous in improving over present performance levels. Application of improved and customized CNN, Deep CNN, LSTM, and RNN models to the different detection methods can yield constantly improving results as datasets continue to improve.

Today, technology and social media companies are realizing the importance of the quality of information their users are posting and receiving. And they are slowly beginning to prioritize this over maximizing attention of users in their platforms. Seeing as attention has been the highest currency on the internet until now, this is a welcome change.