Implementation of Federated Learning for classifying the sentiments of Twitter data

Bhavesh Kilaru

Spring, 2023

DSA 5900

4 credit hours

April 28, 2023

Under the supervision of Dr.Trafalis, Theodore B.

Contents

1 Introduction

This project intends to implement the concept of Federated Learning(FL) for sentiment classification of tweets. Since, my professor focuses his research on Federated Learning, he wanted to analyse the results of FL on text classification.

2 Objectives

As mentioned above, the main motive behind the project to check viability of FL in predicting the sentiment of tweets.

2.1 Technical Objectives

The technical objectives behind the project are:-

1. Applying the concept of FL to classify the sentiment of tweet.
2. To understand the affect each hyper parameter on FL(similar to hyper parameter tuning).
3. To check how adding or removing or stacking different layers to the architecture affects the accuracy of the model (centralized learning).

2.2 Individual objectives

The individual motives behind the project are:-

1. To familiarize with the concept of FL.
2. Getting exposed to the TensorFlow and TensforFlow Federated frameworks.
3. To understand Natural Language processing techniques.

3 Literature

Federated Learning(FL) is a machine learning setting where numerous entities (clients) work together to solve a machine learning problem under the direction of a central server or service provider. The data of each client is neither exchanged or transferred to the central server, instead the data for each is stored locally and the model for each client is trained locally and the model updates are transferred to the central server for immediate aggregation. The central server is responsible for the orchestration. It combines the concept of focused collection and data minimization which can reduce many systemic privacy issues and also the conventional costs caused by traditional, centralized machine learning.

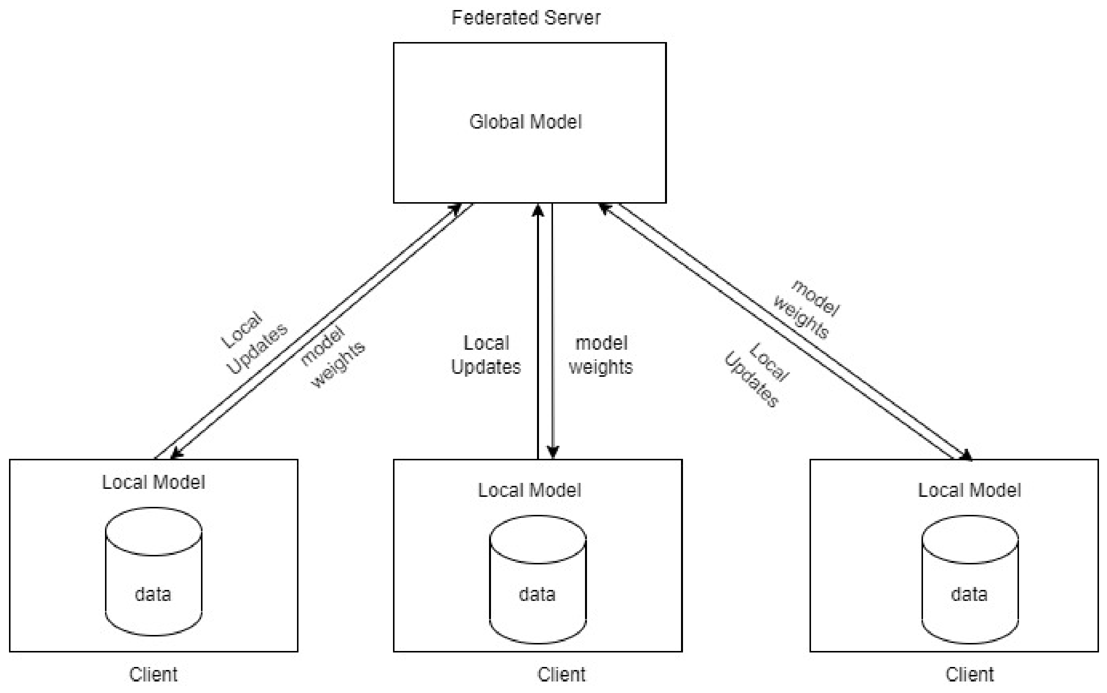


Figure 1: Federated Learning architecture.

There are three types of federated learning techniques.

1. **Data Center Distributed learning**: - It refers to a distributed machine learning approach that involves training machine learning models on large amounts of data stored across multiple data centers. The Clients are compute nodes in a single cluster or data center.

2. **Cross-silo federated learning**: - It refers to the FL technique that trains the model on data that is located across different organisations(silo) e.g., medical or financial. Data from each silo is stored locally and is not shared with other organisation. Typically involves 2-100 clients.

3. **Cross-device federated learning**:- In this type of FL, the data is stored across large number of devices like IOT devices or mobile phones. Massive number of clients upto 10^{10} may be involved in training but in a round of communication, all the clients may not participate in training. Here, clients cannot be addressed directly. This has high chances of client failure (may be due to power or communication issues).

3.1 Life cycle of model in Federated Learning

The typical flow in developing a federated learning involves the following stages.

1. **Problem Identification**: - The model engineer identifies a challenge that FL can address.

2. **Client Instrumentation**: - This is optional phase. Clients should be be provided the necessary resources to store the data and metadata locally as well as to compute the model.

3. **Simulation prototyping**: - This is also an optional phase.The model engineer has to come up with the right architecture and set of hyper parameters to be used in FL using a simulation dataset.

4. **Federated model training** :- A number of training activities have to be initialized to train different variations of the model or to find the correct set of hyper parameters for optimization.

5. **model evaluation**: - Once the model has been for a certain period, the model has to be analysed for appropriate candidates. This can be done in two ways. One way is to evaluate the model in the centralized server with standard data sets. While the other is to do the federated evaluation ie., pushing the centralized model weights to certain clients in order to get them evaluated on the local data.

6. **deployment**: - Once a model is selected, it will be deployed in the production after testing it.

3.2 Federated Learning training process

A typical federated learning process is shown in Fig 2. Here, the service provider (centralized server) is responsible for orchestration ie., managing communication with the clients both to and forth, aggregating the weights and so on.

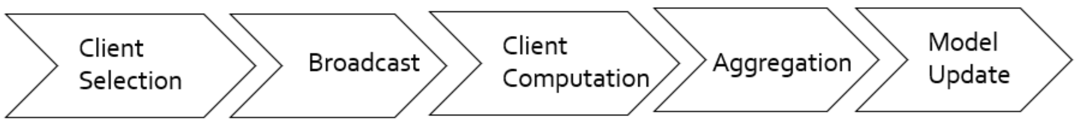


Figure 2: Federated Learning training process.

1. **Client Selection**: - The server selects the clients based on particular criteria. For example, in cross-device FL, the server may choose only the devices that have a stable connection or that are connected to power source.

2. **Broadcast**:- Once the server chooses the clients and a communication is established between the clients and the server, the server will send the model weights to the selected clients.

3. **Client computation** :- Each of the client will train the model with its data and updates the model locally.

4. **Aggregation** :- The server collects the updated weights from the devices (clients) and aggregates the results. It may drop some clients if it got the sufficent number of results.

5. **Model update**: - The server will update its weights with the aggregated ones.[1]

4 Data

Choosing correct set of data is a crucial task in for modelling. As a part of this project, ”Twitter Sentiment Analysis” is considered for modelling. This dataset is taken from Kaggle. Two datasets namely `"twitter_training"` and `"twitter_validation"` are provided as a part of the dataset.

4.1 Exploration

Each of the above two datasets have four attributes namely ”id”, ”entity”, ”sentiment” and ”text”. The twitter_training dataset has 74,682 examples whereas twitter_validation has 1000 examples. The latter is considered as the test dataset due to low data size. The ”sentiment” attribute is the class label i.e., the output to be predicted. It has four different labels namely Positive, Negative, Neutral and Irrelevant. The percentage of each label in train set is shown in figure 3.

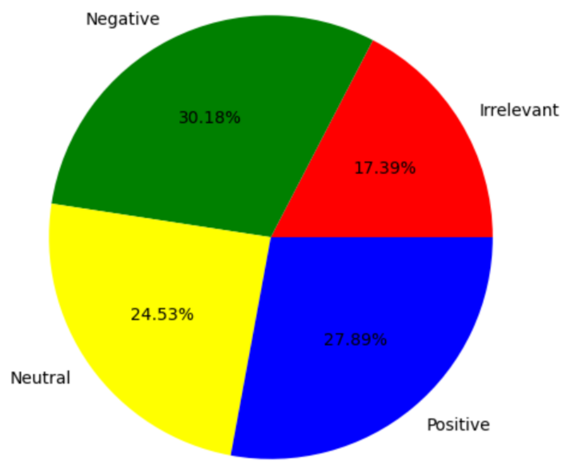


Figure 3: percentage decomposition of tweets

Only records with positive and negative sentiment are considered as a part of the project.The world clouds of tweets corresponding to positive and negative tweets are shown in fig 4 and 5. These word clouds are generated after the initial data processing. The larger the word in word cloud, the more the word occurs in the document or in tweet(s).

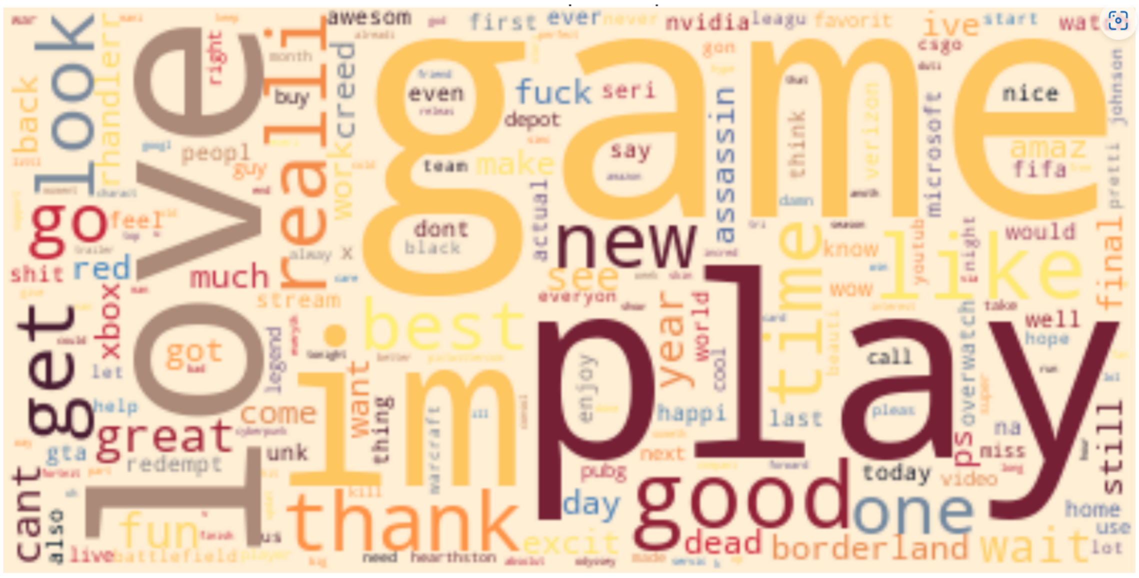


Figure 4: Word Cloud of Positive Tweets

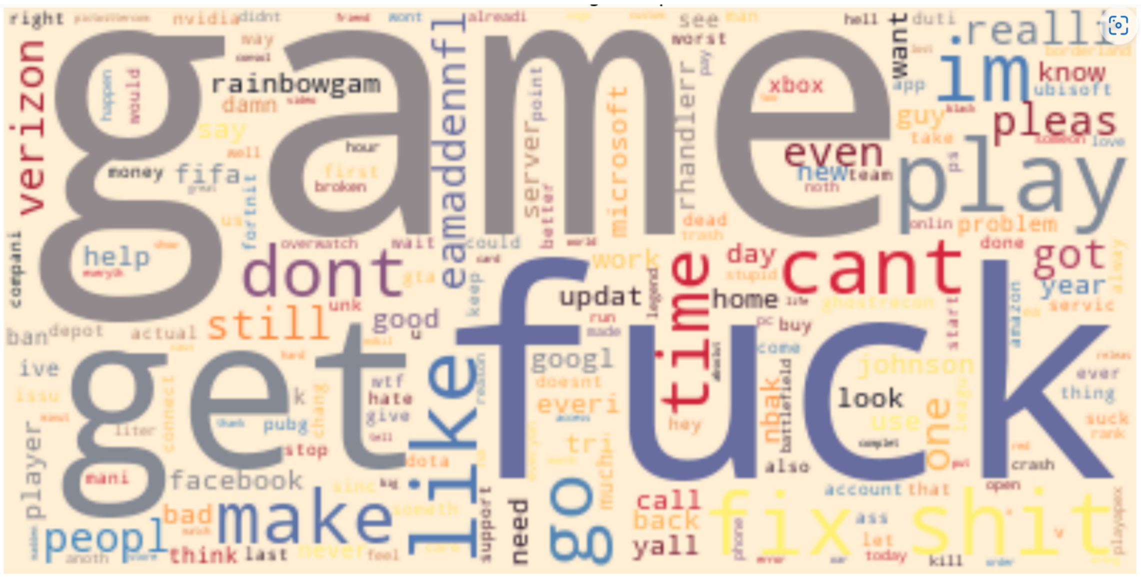


Figure 5: Word Cloud of Negative Tweets

4.2 Preparation

Preprocessing is an integral step and plays a critical role in any Machine learning project. The nature of tweets as short, informal, and often containing noisy language such as hash tags, URL's and emotions. It unique challenges that can affect the accuracy of sentiment classification models.[2] Since neural networks cannot understand any character or strings, the sentiment of the tweet are converted into numbers. All the tweets with positive polarity are labelled as 1 whereas the tweets with negative polarity are labelled as zero. Each tweet's text is processed as shown in figures 6 and 7 using packages like Spacy, Regex, NLTK.

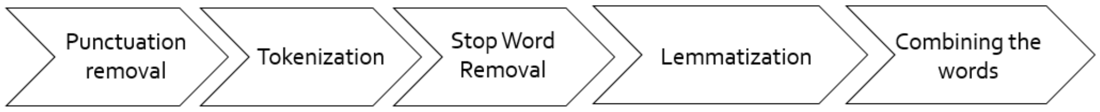


Figure 6: Initial Data processing

Initially, all the punctuation marks are removed from the text and then it is divided into words using word tokenizer. Stop words are removed and next, the words are lemmatized which basically converts the word into its root word. Finally, the cleaned words are combined to form the sentence.

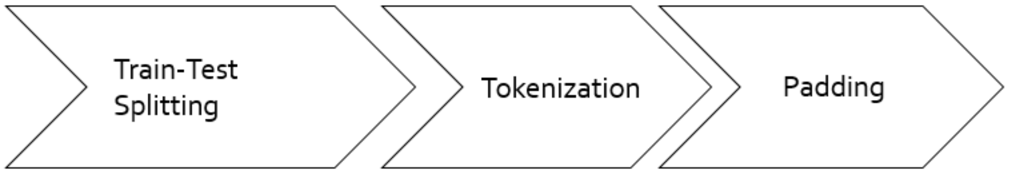
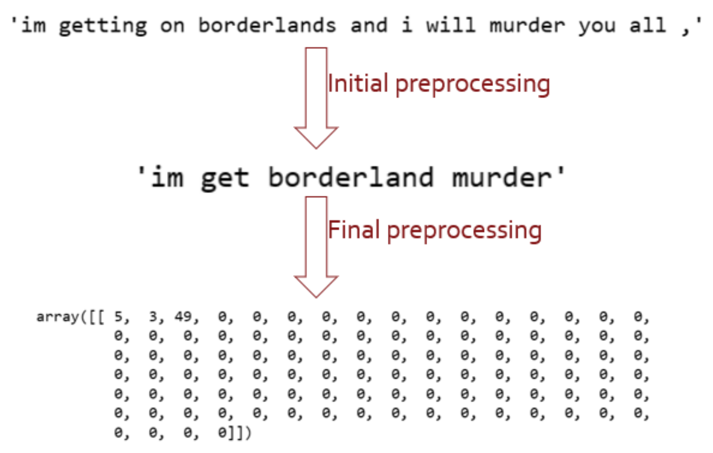


Figure 7: Final Data processing

After cleaning the text, the train data is split into train and validation set in 80:20 ratio. A tokenizer is used to convert the text into numbers (tokens). This tokenizer is fitted on processed text from the train data and then the train set, validation set and test set are converted into tokens using this tokenizer. Since LSTM's can take sequences of equal length [3], all the three datasets are padded to ensure all examples are of similar length.



```
'im getting on borderlands and i will murder you all ,'
```

```
'im get borderland murder'
```

```
array([[ 5,  3, 49,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
         0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
         0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
         0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
         0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
         0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
         0,  0,  0,  0]])
```

Figure 8: Example of preprocessing

5 Methodology

5.1 Techniques

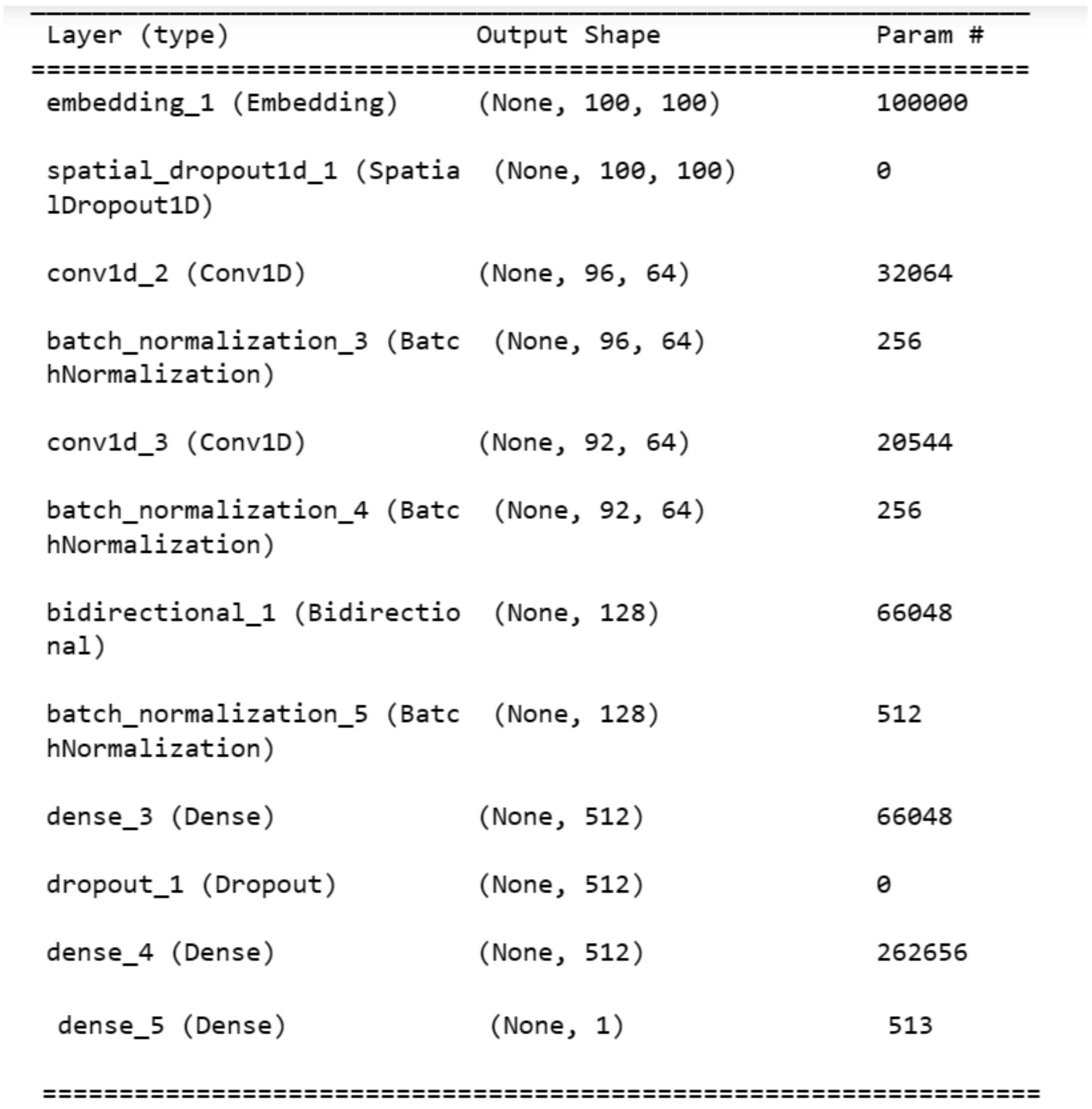
RNNs are a subset of neural networks in which the current inputs of hidden layers are influenced by their past outputs. As a result, they can handle time sequences with temporal relations, such speech recognition. RNNs are discovered to be more effective in sentiment analysis than CNNs in a previous comparison of RNN and CNN in natural language processing. In recurrent neural networks, as the sequence length increases, the network weights may become too large or too small to handle. To address this issue of vanishing gradients during the training of traditional RNNs, Long Short-Term Memory (LSTM) was introduced. LSTM allows the network to learn long-term dependencies over extended time periods by incorporating forget gates along with input and output gates[4]. So, LSTM's are used in all the newtorks.

GloVe which stands for Global Vectors, is a type of unsupervised learning algorithm which is used to that to generate word embeddings, which are dense vector representations of words in a vocabulary. These word embeddings capture the semantic relationships between words. The 'glove.6B.100d.txt' file specifically which contains 100-dimensional embeddings for a vocabulary of 400,000 words trained on a corpus of 6 billion tokens is used for this project. This can save significant time and resources compared to training a new embedding model from scratch. An embedding layer is used an initial layer to all our models to map discrete values, i.e., Bag of Words, to dense vectors of real numbers. This dense vector representation helps the model to capture the context and meaning of the words in the input text, and it can also help to reduce the dimensionality of the input data, making it easier for the model to learn from.

Finally, the last layer of all our models is a layer having single neuron with Sigmoid activation. This is due to the fact that the problem to be solved is a binary classification. Apart from the above several layers were tried as hidden layers to get the best performance.

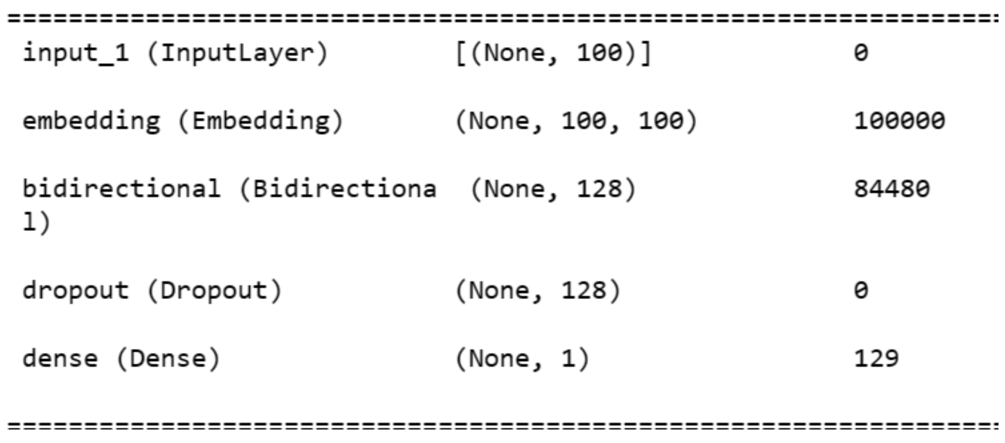
5.2 Procedure

Different types of network architectures are tried to get the best architecture that performs well on the data. Many factors like the complexity of the network, test accuracy, F1 score and ROC_AUC scores were considered in choosing the network. All these networks are build using TensorFlow package of python. The selected network will then be used in training the FL model. Figure 9, 10 and 11 shows the different architectures that were tried to pick the best network based on its performance on the test data.



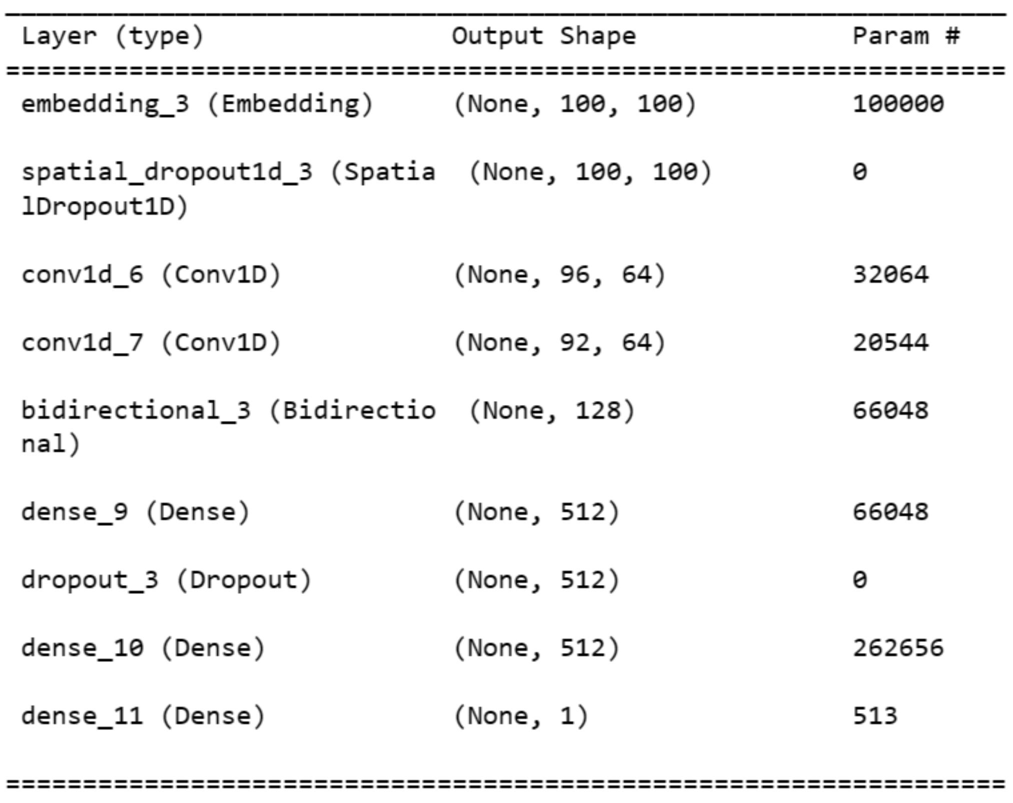
```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, 100, 100)          100000

 spatial_dropout1d_1 (Spatia  (None, 100, 100)         0
 lDropout1D)

 conv1d_2 (Conv1D)           (None, 96, 64)            32064

 batch_normalization_3 (Batc  (None, 96, 64)           256
 hNormalization)

 conv1d_3 (Conv1D)           (None, 92, 64)            20544

 batch_normalization_4 (Batc  (None, 92, 64)           256
 hNormalization)

 bidirectional_1 (Bidirectio  (None, 128)              66048
 nal)

 batch_normalization_5 (Batc  (None, 128)              512
 hNormalization)

 dense_3 (Dense)             (None, 512)               66048

 dropout_1 (Dropout)         (None, 512)               0

 dense_4 (Dense)             (None, 512)               262656

  dense_5 (Dense)             (None, 1)                 513

=================================================================
```

Figure 9: Model 2



```
================================================================
 input_1 (InputLayer)        [(None, 100)]             0

 embedding (Embedding)       (None, 100, 100)          100000

 bidirectional (Bidirectiona  (None, 128)              84480
 l)

 dropout (Dropout)           (None, 128)               0

 dense (Dense)               (None, 1)                 129

================================================================
```

Figure 10: Model 1



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_3 (Embedding)     (None, 100, 100)          100000

 spatial_dropout1d_3 (Spatia  (None, 100, 100)         0
 lDropout1D)

 conv1d_6 (Conv1D)           (None, 96, 64)            32064

 conv1d_7 (Conv1D)           (None, 92, 64)            20544

 bidirectional_3 (Bidirectio  (None, 128)              66048
 nal)

 dense_9 (Dense)             (None, 512)               66048

 dropout_3 (Dropout)         (None, 512)               0

 dense_10 (Dense)            (None, 512)               262656

 dense_11 (Dense)            (None, 1)                 513

=================================================================
```

Figure 11: Model 3

Once the network is selected, the train data and validation data are divided among the clients in order to perform FL. Figure 12 represents the pie chart showing the distribution of data for each Client. Then these client data will be converted into tensors and the divided into batches. This is preprocessing data for FL.

A federated learning algorithm is build to average the weights from the clients and then assign them to the server, with model architecture, learning rates each for a client and the server. This project utilizes the unweighted averaging algorithm to aggregate the weights. Then the model is initialized and trained over certain round of communications. In each round, as described in life cycle of FL, clients are selected randomly to participate in the learning.

Some of the hyperparameters that should be tuned for better learning include the number of clients between whom the data should be divided, number of clients to be participated in learning, learning rates for both client and the server and so on.

TensorFlow_Federated framework will be used for preprocessing the data, building the FL averaging algorithm and for learning.

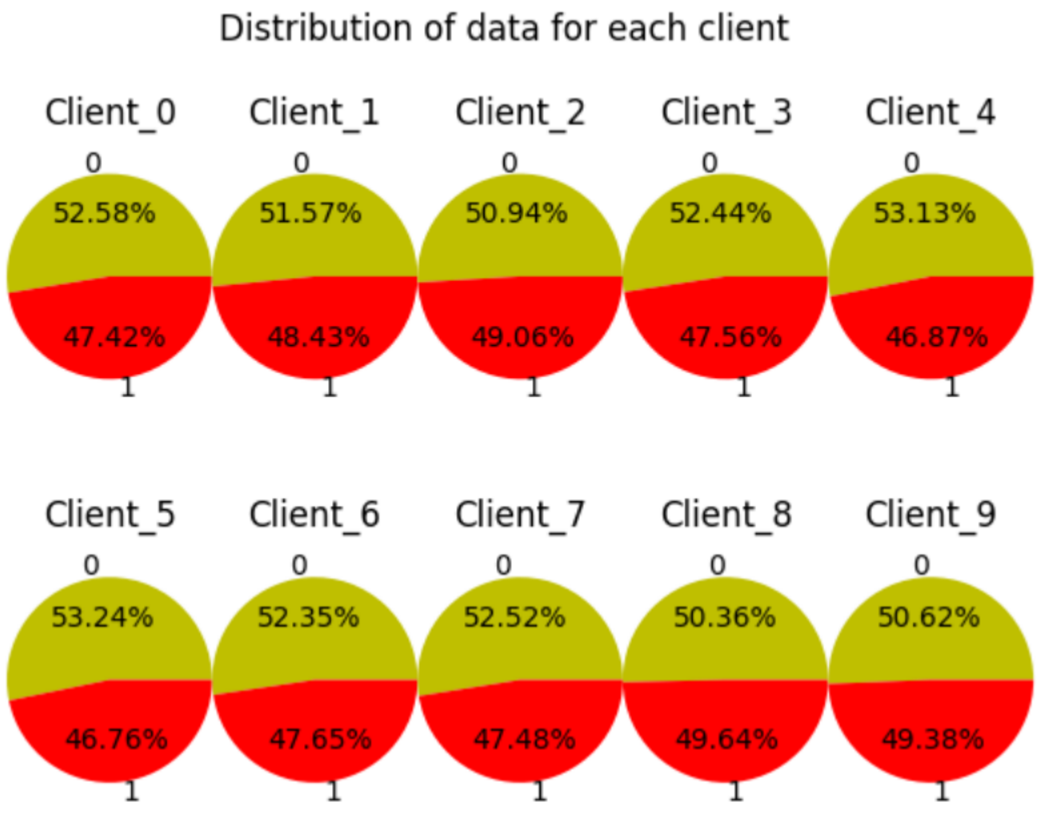


Figure 12: Distribution of data for each Client

6 Results and Analysis

The results for the centralized learning and Federated learning were documented separately.

6.1 Centralized Learning results

Table 1: Metrics for different models using Centralized Learning

	Accuracy	F1 Score	ROC_AUC Score
Model 1	95.76	95.69	0.9909
Model 2	94.10	94.24	0.9835
Model 3	94.29	94.41	0.9805

Table 1 shows the performance of different metrics on test data. It can be clearly noticed that model 1 performs better when compared with other architectures. Additionally, model 1 has a simple architecture than the other two. This implies lower training times and less resource consumption. So, Model 1 was used in Federated Learning. Figure 13 compares the train loss with the validation loss and train accuracy with validation accuracy for three networks.

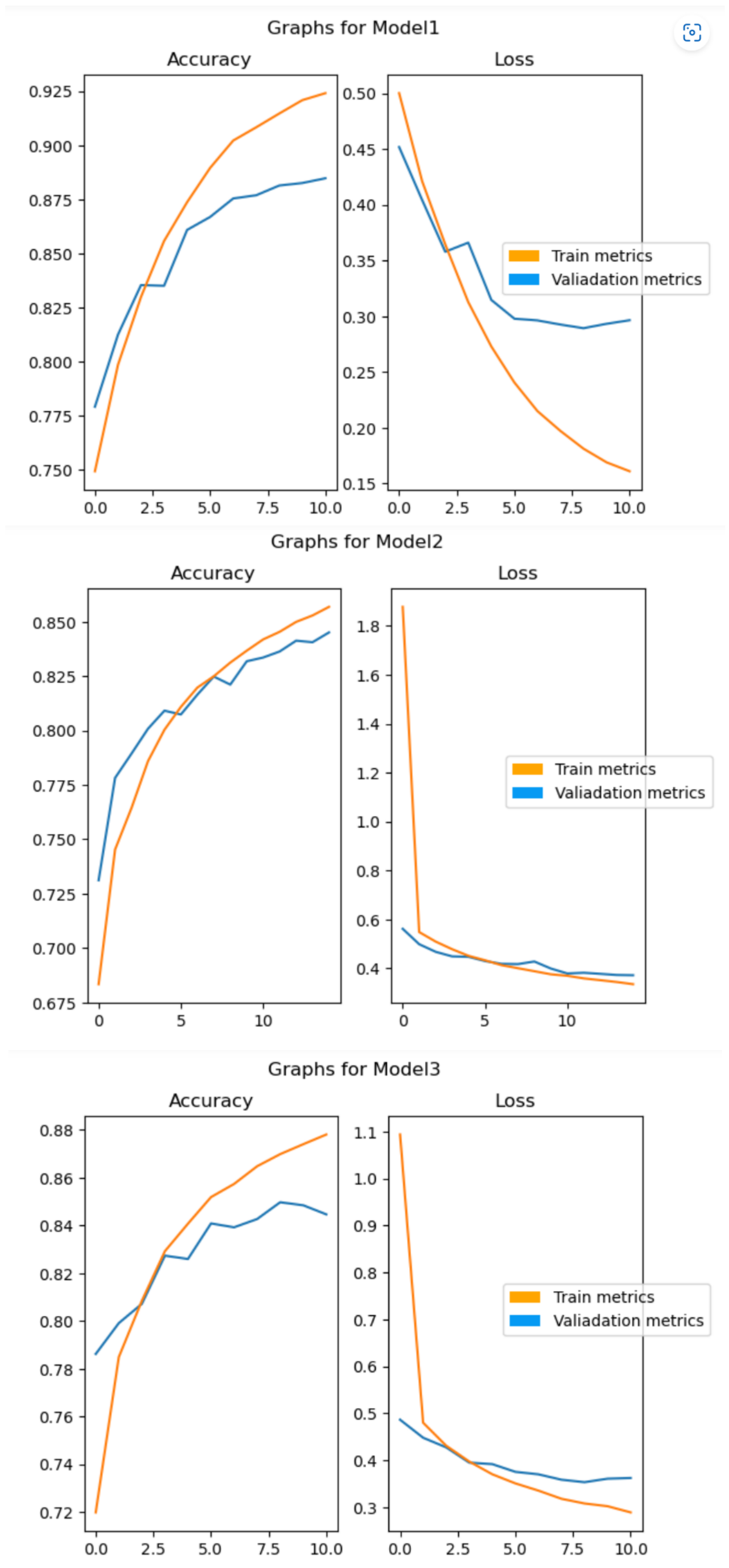


Figure 13: Plots of metrics for different network

6.2 Federated Learning results

A number of experiments were performed on FL model to find the best set of hyper-parameters. The best result was found with Adams optimizer for a batch size of 16, with a client learning rate 0.05 and a server learning rate 0.2. Table 2 shows the metrics for Federated Learning model whereas figure 14 shows the plot for accuracy and loss on validation and train data for the FL model.

Table 2: Metrics for Federated Learning

Accuracy	89.31
F1 Score	89.29
ROC_AUC Score	0.9549

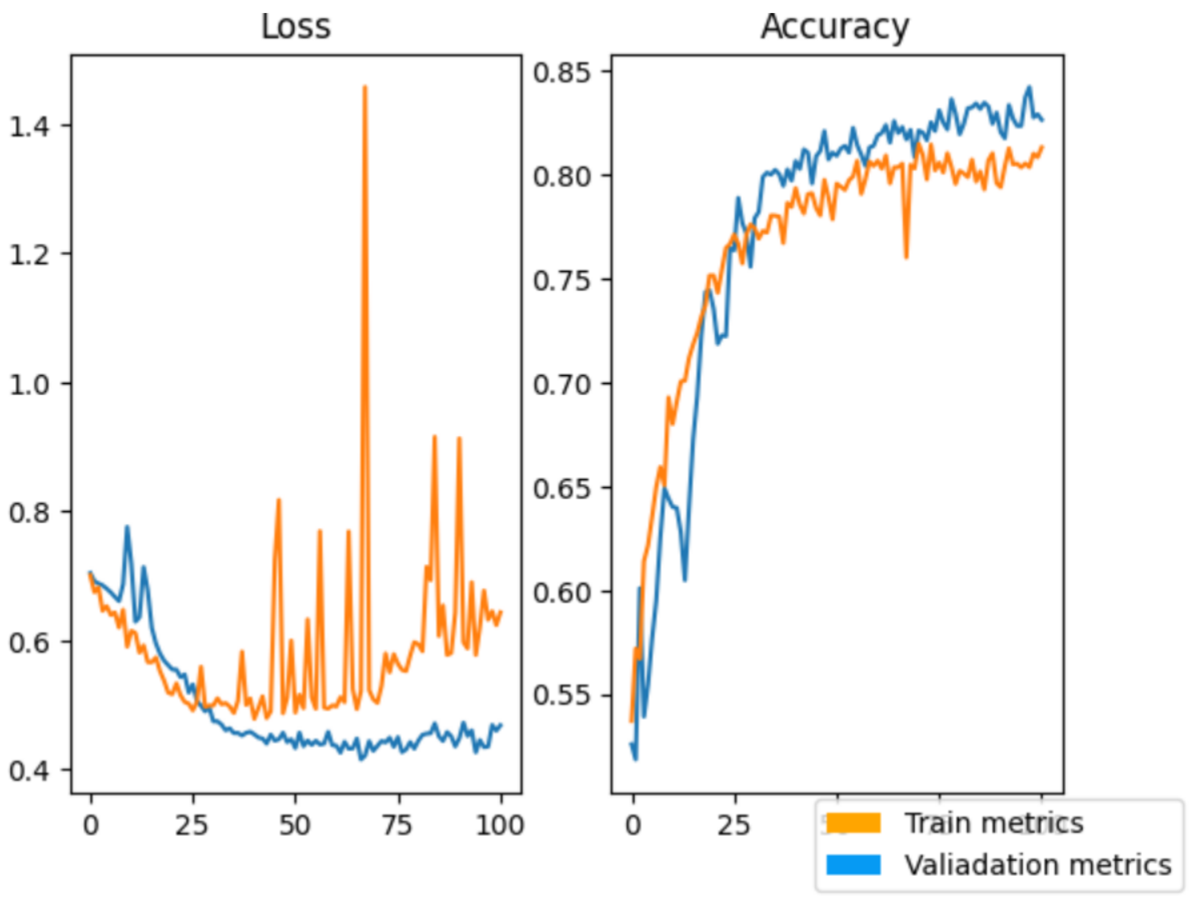


Figure 14: Plots of metrics for Federated Learning

6.3 Analysis

From table 1, it can be observed that the model perfomance degrades with increase in the complexity of the model. Complex networks also results in higher training times.

The series of experiments that were ran for finding the best set of hyper-parameters to be used in FL revealed that the hyper-parameter if not tuned properly had resulted the model to never converge in some instances. Also in some instances, it leads to convergence till some point and then started diverging instead of oscillating around the global maxima.

Overall, The trained Federated Learning model achieved an accuracy of 89.31% on the test set, with F1 score and ROC_AUC of 0.8929 and 0.9529 respectively. From the figure 14, it can be noticed that eventhough the train loss increases drastically ocassionally, it does not effect the train accuracy.

7 Deliverables

This project demonstrated the feasibility and effectiveness of Federated Learning for Twitter Sentiment Classification. The Federated Learning approach used in this project can be extended to other natural language processing tasks and can be used in a variety of applications where privacy is a concern. Overall, this project ensured that the concept of Federated Learning can be successfully applied to classify the sentiment of the tweet.

As a part of this project, the effect of different hyper parameters on learning task of FL is understood. It was revealed hyperparameters such as batch size, learning rate for client, learning rate for server and the choice of optimizer is crucial for the convergence of FL model. Hyperparameter when not selected correctly may even lead to divergence of the algorithm.

Also, the affects of adding or removing or stacking different layers to the model on the performance was analyzed. It was found adding more layers to the network lead to overfitting and reduce the generalization performance of the model.

7.1 Problems faced during the practicum

Since the concept of Federated Learning is still a research field, it is hard to find any resources. Knowing how to tackle an issue is really the hardest part of this practicum because of the lack of resources. Additionally, finding a dataset that actually fits in FL concept is also a biggest task. Running a Federated Learning code is both time consuming and resource consuming. It is also difficult to find the version on TensorFlow Federated that is compatible with TensorFlow.

8 Future Work

Federated Learning algorithm can be implemented using a different averaging process and it effects can be analyzed. A method to implement Learning rate decay can be found for FL. Cross device FL environment can be simulated to study how FL works in real time.

9 References

[1] Brendan McMahan et al. "Advances and Open Problems in Federated Learning". In: *Foundations and Trends® in Machine Learning* 14.1-2 (2019), pp. 1–210. DOI: `https://oadoi.org/10.1561/2200000083`.

[2] Ying Bao et al. "The Role of Pre-processing in Twitter Sentiment Analysis". In: *Lecture Notes in Computer Science* (2014), pp. 615–624. DOI: `10.1007/978-3-319-09339-0_62`.

[3] Mahidhar Dwarampudi and N V Subba Reddy. *Effects of padding on LSTMs and CNNs*. 2019. arXiv: `1903.07288 [cs.LG]`.

[4] Jenq Haur Wang et al. "An LSTM approach to short text sentiment classification with word embeddings". In: *Proceedings of the 30th Conference on Computational Linguistics and Speech Processing, ROCLING 2018*. Ed. by Chi-Chun Lee et al. Proceedings of the 30th Conference on Computational Linguistics and Speech Processing, ROCLING 2018. The Association for Computational Linguistics and Chinese Language Processing (ACLCLP), Oct. 2018, pp. 214–223.

[5] *Google Colab notebook*. `https://colab.research.google.com/drive/14L6TS1wMJ_jS3kC_sRO6mdn1zLxgpz45?usp=sharing`.

10 Self-Assessment

As a part of this project, I got to know and familiarize the concept of Federated Learning and its applications. I also got familiarized with different techniques of natural language processing like Stop word removal, lemmatization, filtering text using various packages like NLTK, Spacy, Regex. Finally, through this project, I got an opportunity to work on deep learning Frameworks TensorFlow, TensorFlow Federated. I also got to know the applications of various layers, different optimizers and different TensorFlow preprocessing techniques like padding, tokenizer so on. Additionally, I got an exposure to Latex since the documentation was made using latex,