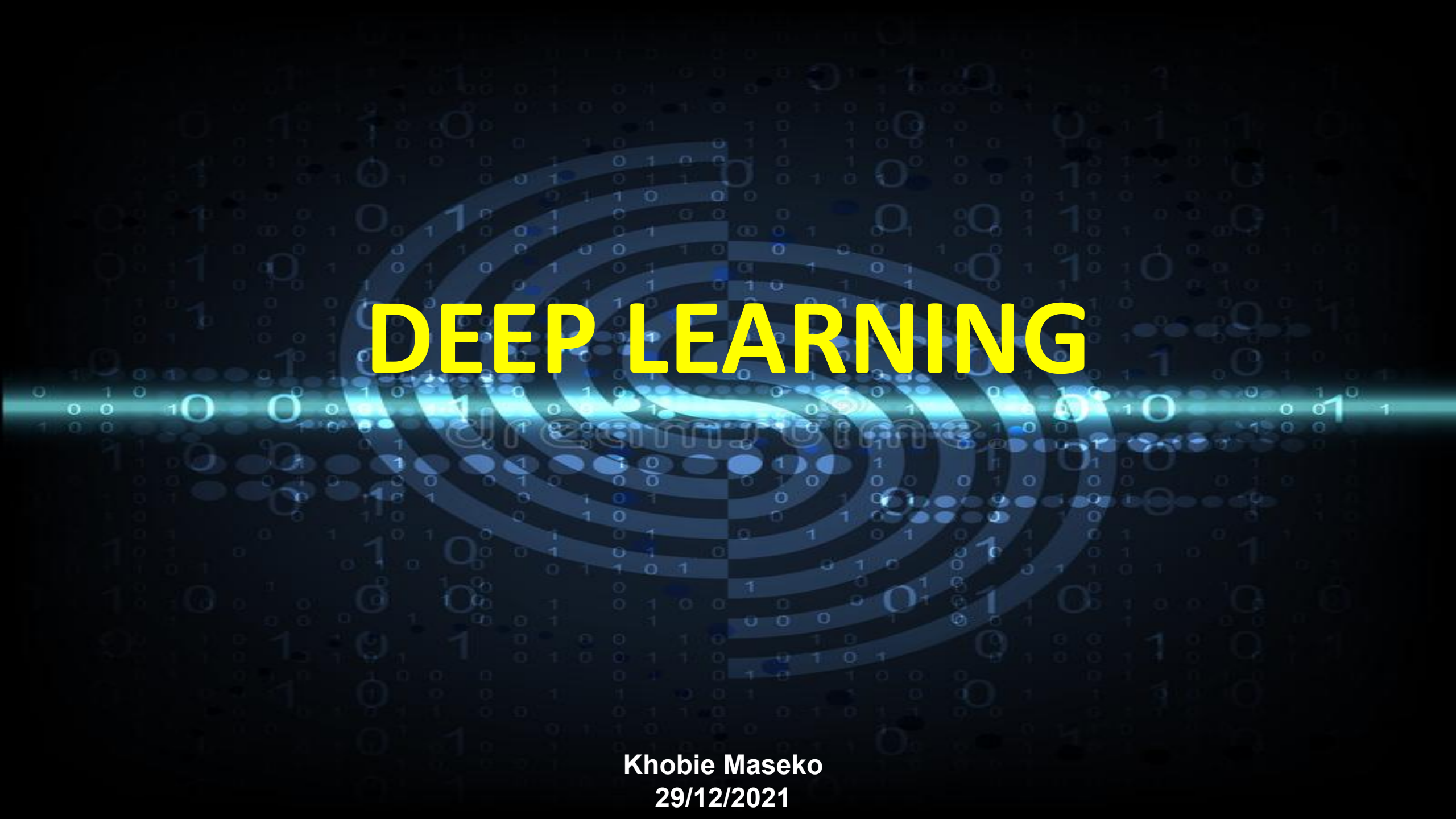


Khobie Maseko
29/12/2021

Project Background and Objective

- This data was obtained from a dataset by Tensorflow titled “reuters”.
- This dataset consists of 11,228 newswires from the Reuters news agency.
- The **objective** of this project is to train a Recurrent Neural Networks (RNN) to classify news items into different topics. We will also design and fit a Random Trees Classifier for the same purpose.
- Dateset source: https://www.tensorflow.org/api_docs/python/tf/keras/datasets/reuters.
- We have to note however, that whilst RNN is generally good for sequential data, its major weakness is that it is currently constructed, which makes it hard to leverage information from the distant parts, or in other words, early on in our sequence. Our conclusion will address how to negate this weakness.

Project Workflow

The following are the steps that we will take to determine which model is the one best suited to meeting our objective for this dataset:

1. Data Overview
2. Data Preparation
3. RNN Model 1
4. RNN Model 2
5. RNN Model 3
6. Random Forest Model
7. Key Findings and Insights
8. Recommendations
9. Conclusion

Data Overview - Data Attributes

To the right are the attributes of the datase as per the Keras interface that we have used to import the dataset.

We imported this information using: keras.datasets.reuters.get_word_index()


```
Output exceeds the size limit. Open the full output data in a text editor
{'mdbl': 10996,
 'fawc': 16260,
 'degussa': 12089,
 'woods': 8803,
 'hanging': 13796,
 'localized': 20672,
 'sation': 20673,
 'chanthaburi': 20675,
 'refunding': 10997,
 'hermann': 8804,
 'passsengers': 20676,
 'stipulate': 20677,
 'heublein': 8352,
 'screaming': 20713,
 'tcby': 16261,
 'four': 185,
 'grains': 1642,
 'broiler': 20680,
 'wooden': 12090,
 'wednesday': 1220,
 'highveld': 13797,
 'duffour': 7593,
 '0053': 20681,
 'elections': 3914,
 '270': 2563,
...
 'clock': 16407,
 'teape': 20985,
 'teapa': 20986,
 'prevailed': 10093,
 'hebei': 9407,
 ...}
```

Data Preparation

To prepare the data for our model designs, we did the following:

- Set the maximum number of words in a news item to 1000.
- Split data into Train/Test sets and used Tokenizer to transform both features.
- Transformed both y_train and y_test into one-hot encoded vectors.

```
array([[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., 1., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 1., 0., 0.]], dtype=float32)
```

RNN Model 1

We:

- initialised a Keras Sequential model and added three layers to it.
- set the model to pick 1000 of the most common words.
- compiled our Keras model with a training configuration.
- defined the batch_size for training as 32 and trained the model for 5 epochs on train sets.
- calculated the score for our trained model by running evaluation on test sets.

This model achieved a **accuracy** of **94.44%**.

```
Epoch 1/5
246/246 [==============================] - 2s 6ms/step - loss: 1.3018 - accuracy: 0.7089
Epoch 2/5
246/246 [==============================] - 1s 6ms/step - loss: 0.6913 - accuracy: 0.8387
Epoch 3/5
246/246 [==============================] - 1s 6ms/step - loss: 0.5131 - accuracy: 0.8764
Epoch 4/5
246/246 [==============================] - 1s 6ms/step - loss: 0.4221 - accuracy: 0.8946
Epoch 5/5
246/246 [==============================] - 1s 6ms/step - loss: 0.3426 - accuracy: 0.9133
246/246 [==============================] - 1s 3ms/step - loss: 0.2257 - accuracy: 0.9444
```

RNN Model 2

We:

- set the model to pick 5000 of the most common words.
- compiled our Keras model with a training configuration.
- defined the batch_size for training as 80 and trained the model for 10 epochs on train sets.
- calculated the score for our trained model by running evaluation on test sets.

This model achieved a **accuracy** of **4.67%**.

```
Epoch 1/10
281/281 [==============================] - 5s 15ms/step - loss: -28.4963 - accuracy: 0.0478 - val_loss: -99.7126 - val_accuracy: 0.0467
Epoch 2/10
281/281 [==============================] - 4s 15ms/step - loss: -547.9193 - accuracy: 0.0481 - val_loss: -1689.9034 - val_accuracy: 0.0467
Epoch 3/10
281/281 [==============================] - 4s 15ms/step - loss: -17136.0195 - accuracy: 0.0481 - val_loss: -69151.0781 - val_accuracy: 0.0467
Epoch 4/10
281/281 [==============================] - 4s 15ms/step - loss: -1008389.7500 - accuracy: 0.0481 - val_loss: -4292768.5000 - val_accuracy: 0.0467
Epoch 5/10
281/281 [==============================] - 4s 15ms/step - loss: -61021936.0000 - accuracy: 0.0481 - val_loss: -258857040.0000 - val_accuracy: 0.0467
Epoch 6/10
281/281 [==============================] - 4s 15ms/step - loss: -3329280000.0000 - accuracy: 0.0481 - val_loss: -13403151360.0000 - val_accuracy: 0.0467
Epoch 7/10
281/281 [==============================] - 4s 15ms/step - loss: -152297521152.0000 - accuracy: 0.0481 - val_loss: -596596359168.0000 - val_accuracy:
0.0467
Epoch 8/10
281/281 [==============================] - 4s 15ms/step - loss: -6004218527744.0000 - accuracy: 0.0481 - val_loss: -22344357642240.0000 - val_accuracy:
0.0467
Epoch 9/10
281/281 [==============================] - 4s 15ms/step - loss: -202655632719872.0000 - accuracy: 0.0481 - val_loss: -711048210415616.0000 -
val_accuracy: 0.0467
Epoch 10/10
281/281 [==============================] - 4s 15ms/step - loss: -5659339893243904.0000 - accuracy: 0.0481 - val_loss: -19329897600122880.0000 -
val_accuracy: 0.0467
```

RNN Model 3

We:

- set the model to pick 2000 of the most common words.
- compiled our Keras model with a training configuration.
- defined the batch_size for training as 32 and trained the model for 5 epochs on train sets.
- calculated the score for our trained model by running evaluation on test sets.

This model achieved a **accuracy** of **95.18%**.

```
Epoch 1/5
281/281 [==============================] - 3s 9ms/step - loss: 1.1812 - accuracy: 0.7417
Epoch 2/5
281/281 [==============================] - 3s 9ms/step - loss: 0.5537 - accuracy: 0.8743
Epoch 3/5
281/281 [==============================] - 3s 9ms/step - loss: 0.3986 - accuracy: 0.9084
Epoch 4/5
281/281 [==============================] - 3s 9ms/step - loss: 0.3436 - accuracy: 0.9233: 0s - loss: 0.3
Epoch 5/5
281/281 [==============================] - 3s 9ms/step - loss: 0.3085 - accuracy: 0.9281
281/281 [==============================] - 1s 3ms/step - loss: 0.1971 - accuracy: 0.9518
```

Random Forest Model

We:

- trained the RF Model for a 100 trees in the forest.
- calculated the score for our trained model by running evaluation on test sets.

This model achieved a **accuracy** of **61%**.

```
Random Forest Classifier accuracy is 0.610
```

We note that the accuracy score for the RF Classifier is 61%.

Key Findings and Insights

The different models have the following accuracy scores:

- **RNN Model 1: 94.44%**
- **RNN Model 2: 4.67%**
- **RNN Model 3: 95.18%**
- **Random Forest Model: 61.00%**

We noted that Model 2 uses the RELU activation in the first dense layer as opposed to the other RNN models which used the ‘sigmoid’ and ‘softmax’ respectively. That may have greatly contributed to its underperformance due to RELU’s propensity to not let nodes that are being zeroed out learn. A sequential model requires the exact opposite regarding its nodes.

The Random Forest model also performed relatively bad on this dataset even though we tried it with a few different hyperparameters. This accuracy score was the highest out of all of them

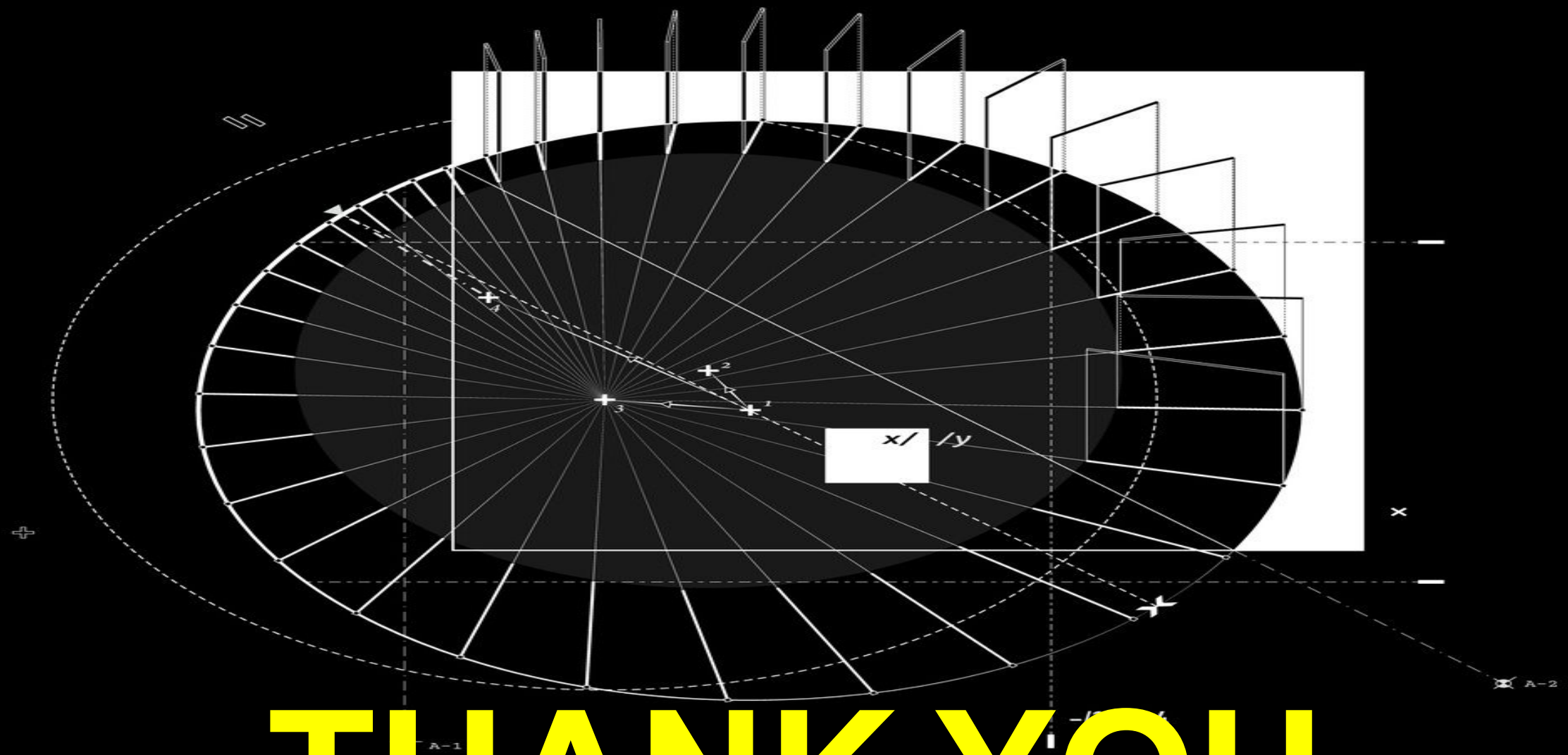
Recommendations

I would recommend us using the Leaky RELU activation to improve RNN Mode 2 as it is keeps the advantage of a steady learning rate between the nodes. It solves the Dying RELU problem.

We can also use LSTMs as our RNN models because they attack the main weakness of ordinary RNNs, their transitional nature. They have poor memory because the transition Matrix unnecessarily weakens signals.

Conclusion

We may conclude that RNN Method 3 is the best model for the purposes of this project although it does not exceed RNN Method 1 by much.



THANK YOU