May 7, 2024

Human vs. AI-Generated Text Detection

https://github.com/SherryKu/Huamn_VS_AI_Text_Detection_Project/tree/main

Sherry Liu,
Veronica Zhao,
Yanchen Zhou

Table of Contents

M I N I
M A L

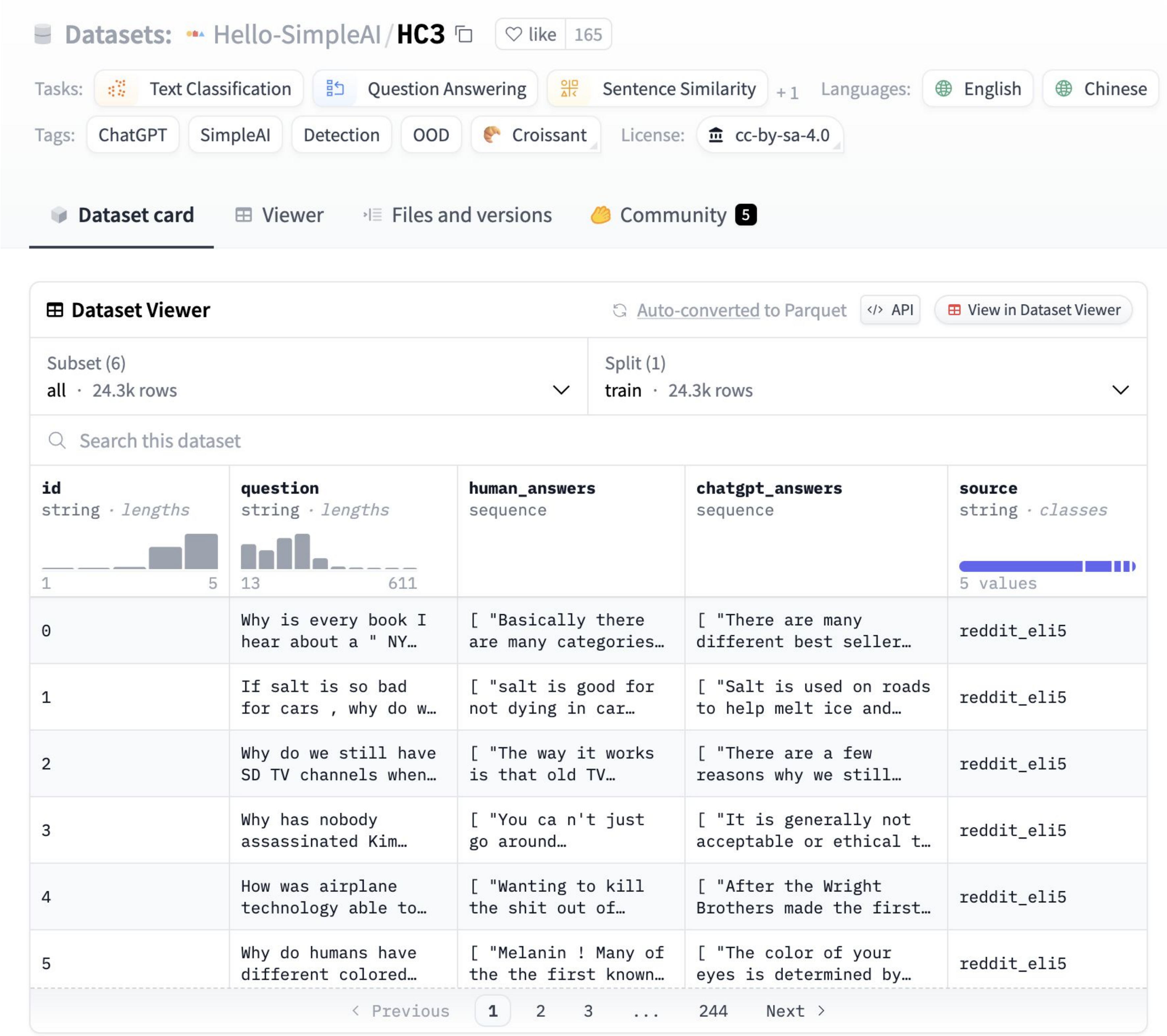
B U S I
N E S
S

1

Executive summary

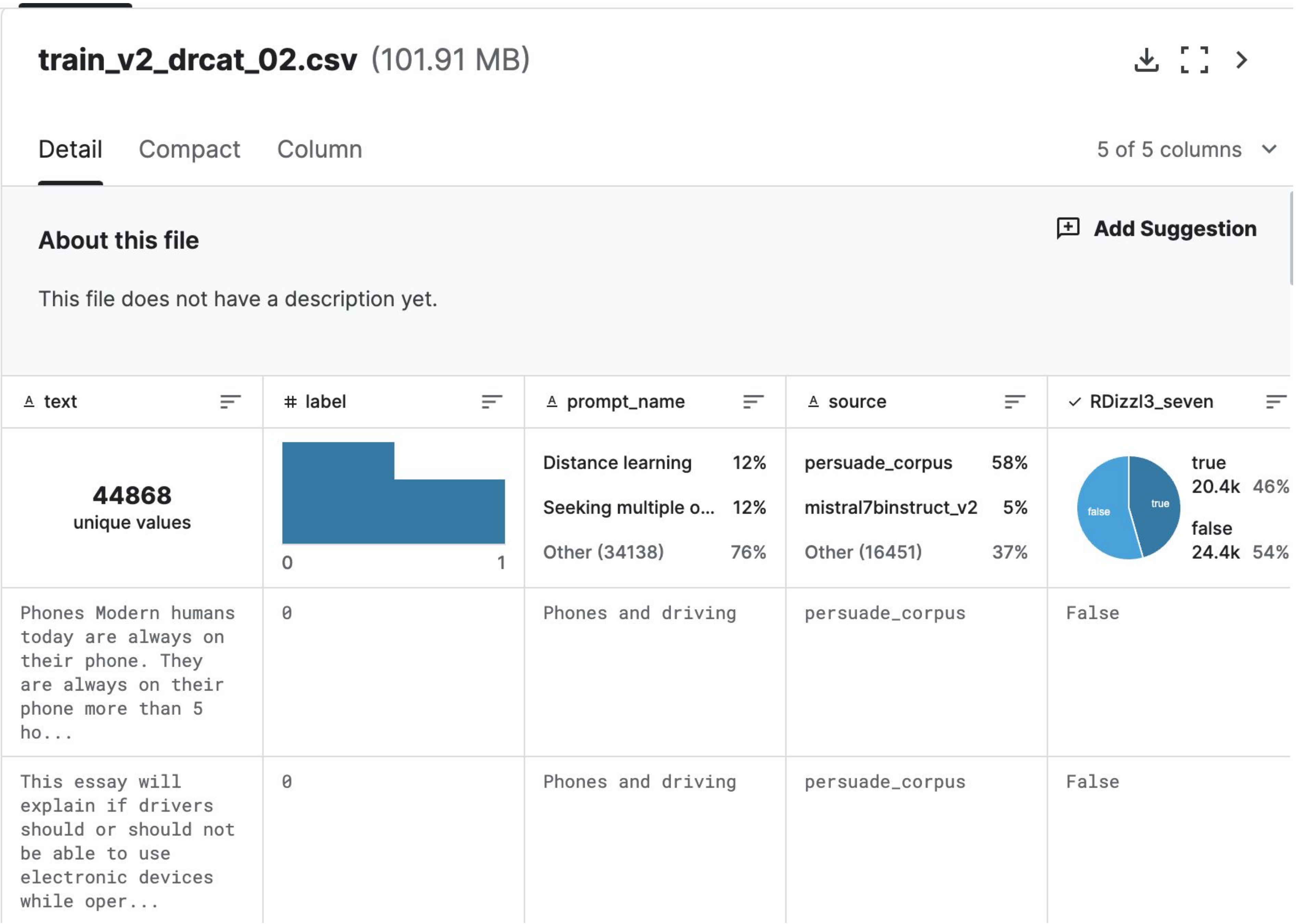
- Problem statement
- Goal
- Technical challenges solution
- Approach
- Value of our solution

Train, Validation, Test Dataset



English Split	Source	Source License	Note
reddit_eli5	ELI5	BSD License	
open_qa	WikiQA	PWC Custom	
wiki_csai	Wikipedia	CC-BY-SA	
medicine	Medical Dialog	Unknown	Asking
finance	FiQA	Unknown	Asking by

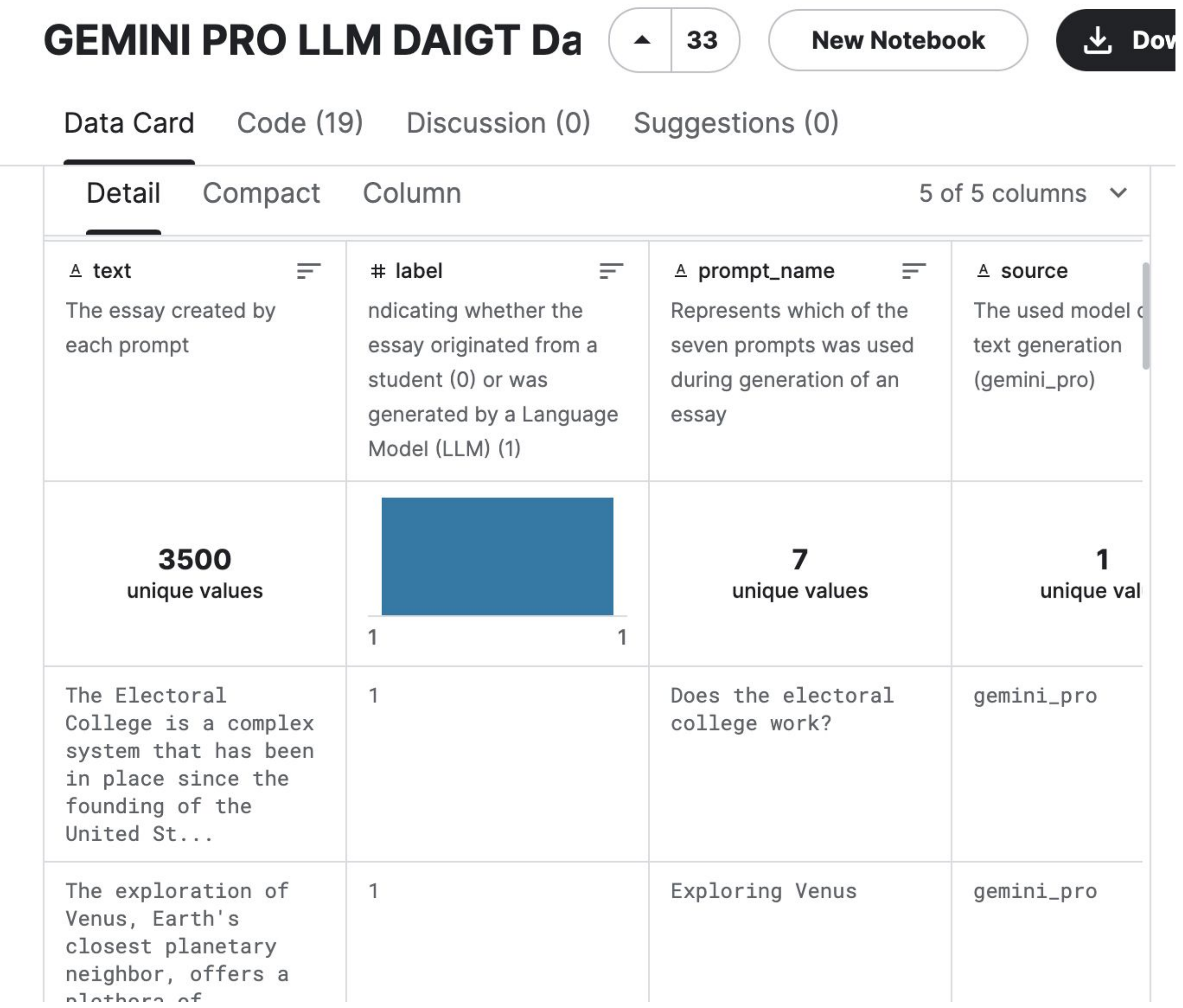
Train, Validation, Test Dataset



- persuade_corpus
- mistral7binstruct_v1
- mistral7binstruct_v2
- chat_gpt_moth
- llama2_chat

Generalization Test Dataset

- Tweepfake: tweets generated by RNN, LSTM, GPT-2, and human tweets
- Gemini



- Chinese

Chinese Split	Source	Source License	Note
open_qa	WebTextQA & BaikeQA	MIT license	
baike	Baidu Baike	None	
nlpcc_dbqa	NLPCC-DBQA	Unknown	Asking
medicine	Chinese Medical Dialogue	CC-BY-NC 4.0	
finance	FinanceZhidao	CC-BY 4.0	
psychology	On Baidu AI Studio	CC0	
law	LegalQA	Unknown	Asking

M I N I
M A L

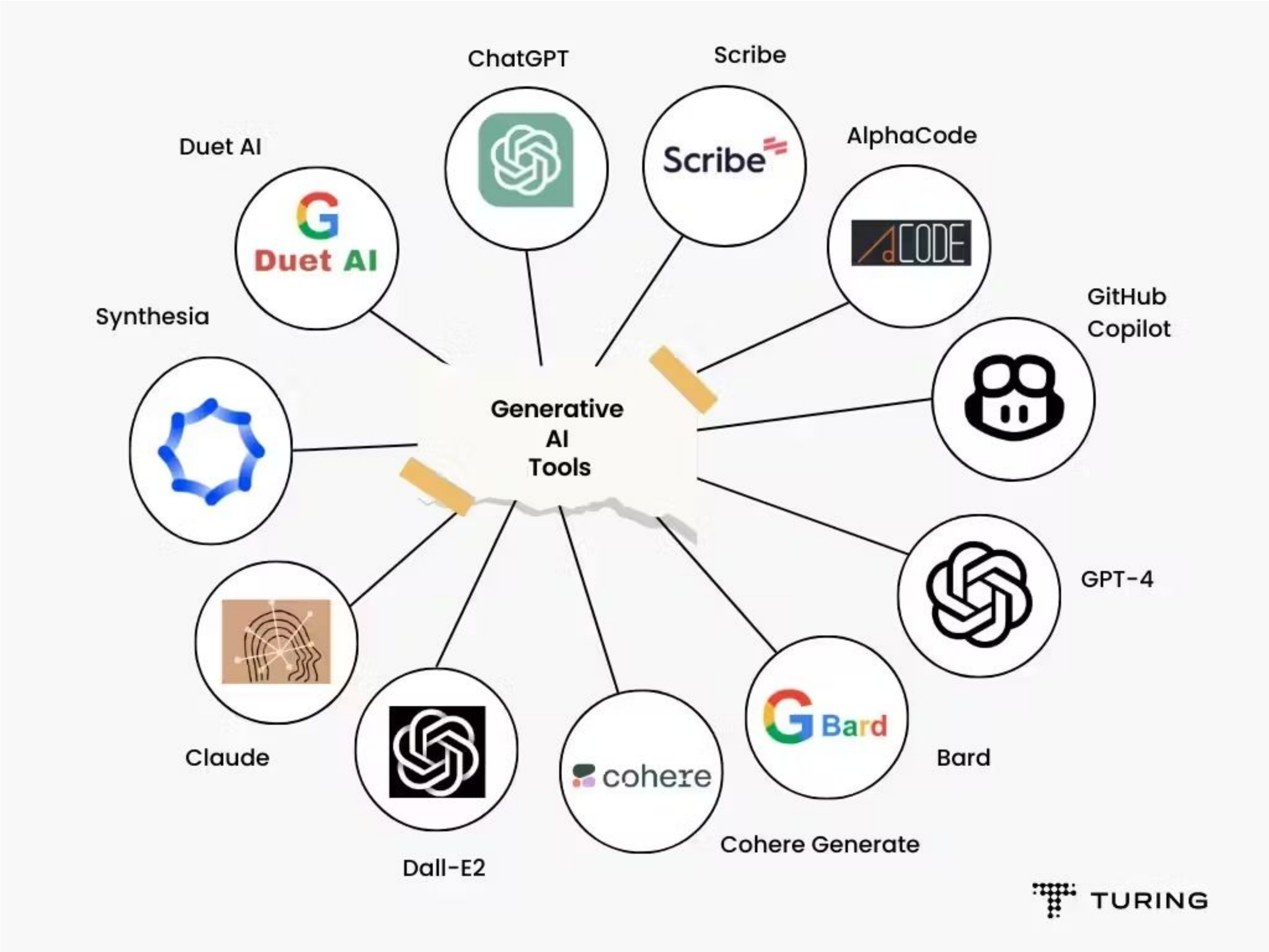
B U S I
N E S
S

2

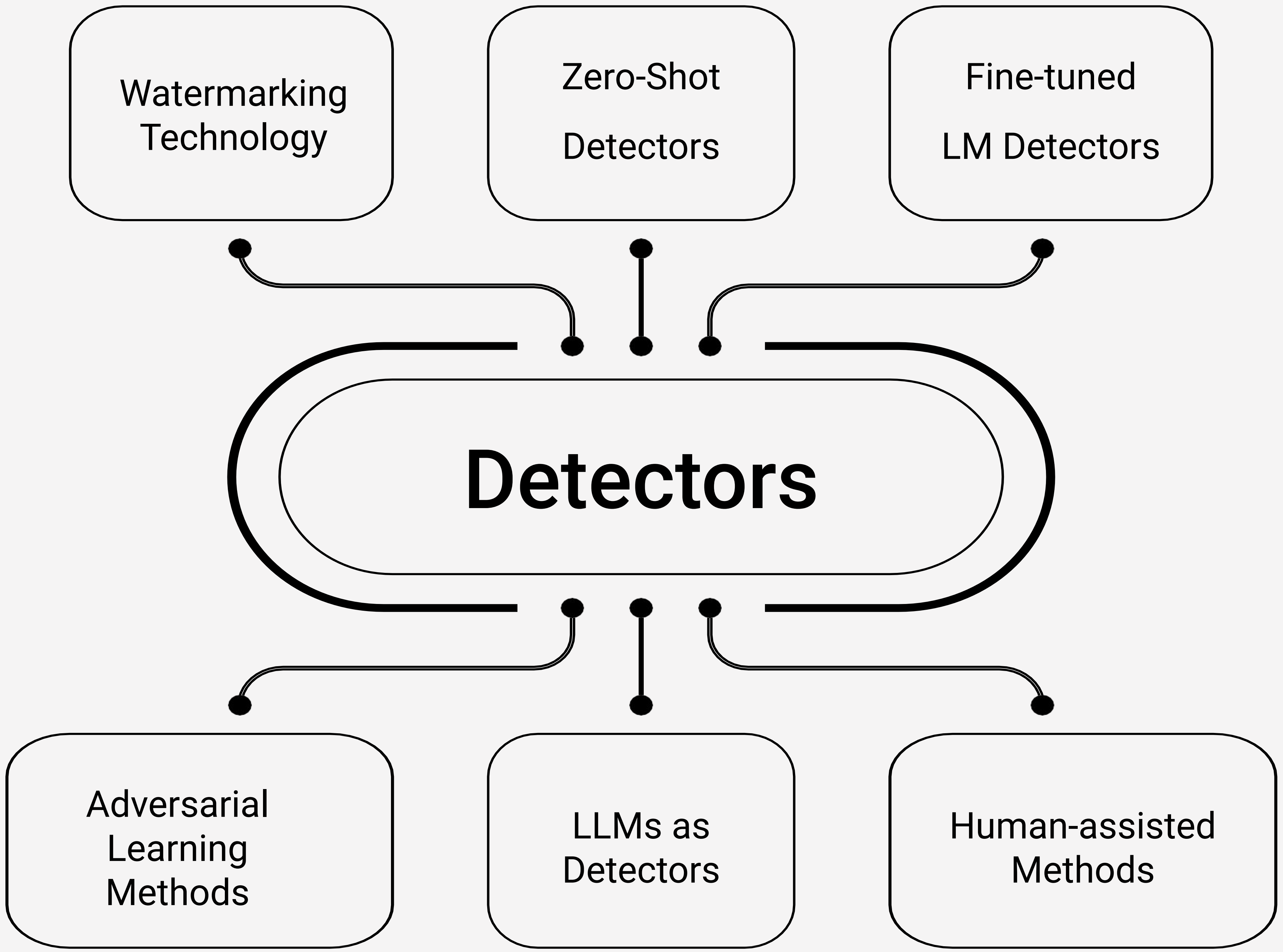
Motivation
& Related Works

Motivations

Booming of AI Generation Tools



Concerns？



Related Works in AI-text Detection

Challenges and Limitations:

- Detection methods struggle with high similarities between AI and human texts.
- Limited test on non-English language detections with current large models mostly trained on English texts

Methodology on Models

- LSTM
- VNN, CNN, FNN
- DistilBert, RoBerta, AlBert

3

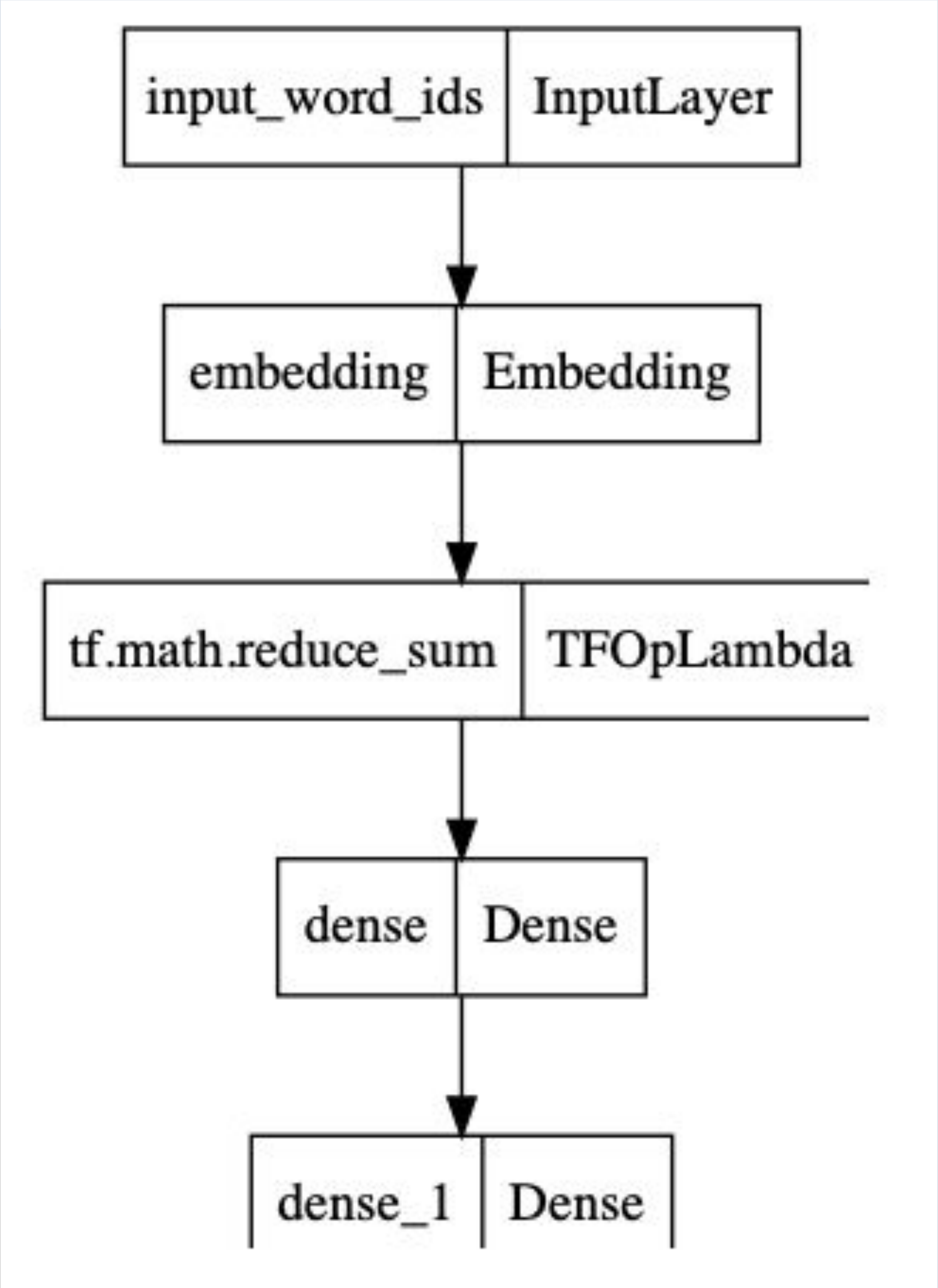
LSTM

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_word_ids (InputLayer  [(None, 512)]             0
 )

 embedding (Embedding)       (None, 512, 768)          91812096

 spatial_dropout1d (Spatial  (None, 512, 768)          0
 Dropout1D)

 lstm (LSTM)                 (None, 512, 128)          459264

 lstm_1 (LSTM)               (None, 512, 128)          131584

 attention_weighted_average  (None, 128)               128
  (AttentionWeightedAverage
 )

 dense (Dense)               (None, 64)                8256

 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 92411393 (352.52 MB)
Trainable params: 599297 (2.29 MB)
Non-trainable params: 91812096 (350.24 MB)
_________________________________________________________________
```

VNN with DistilBERT

Simplistic design, high parameter count due to embedding layer.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_word_ids (InputLayer  [(None, 512)]             0
 )

 embedding (Embedding)       (None, 512, 768)          91812096

 tf.math.reduce_sum (TFOpLa  (None, 512)               0
 mbda)

 dense (Dense)               (None, 128)               65664

 dense_1 (Dense)             (None, 1)                 129

=================================================================
Total params: 91877889 (350.49 MB)
Trainable params: 65793 (257.00 KB)
Non-trainable params: 91812096 (350.24 MB)
_________________________________________________________________
```

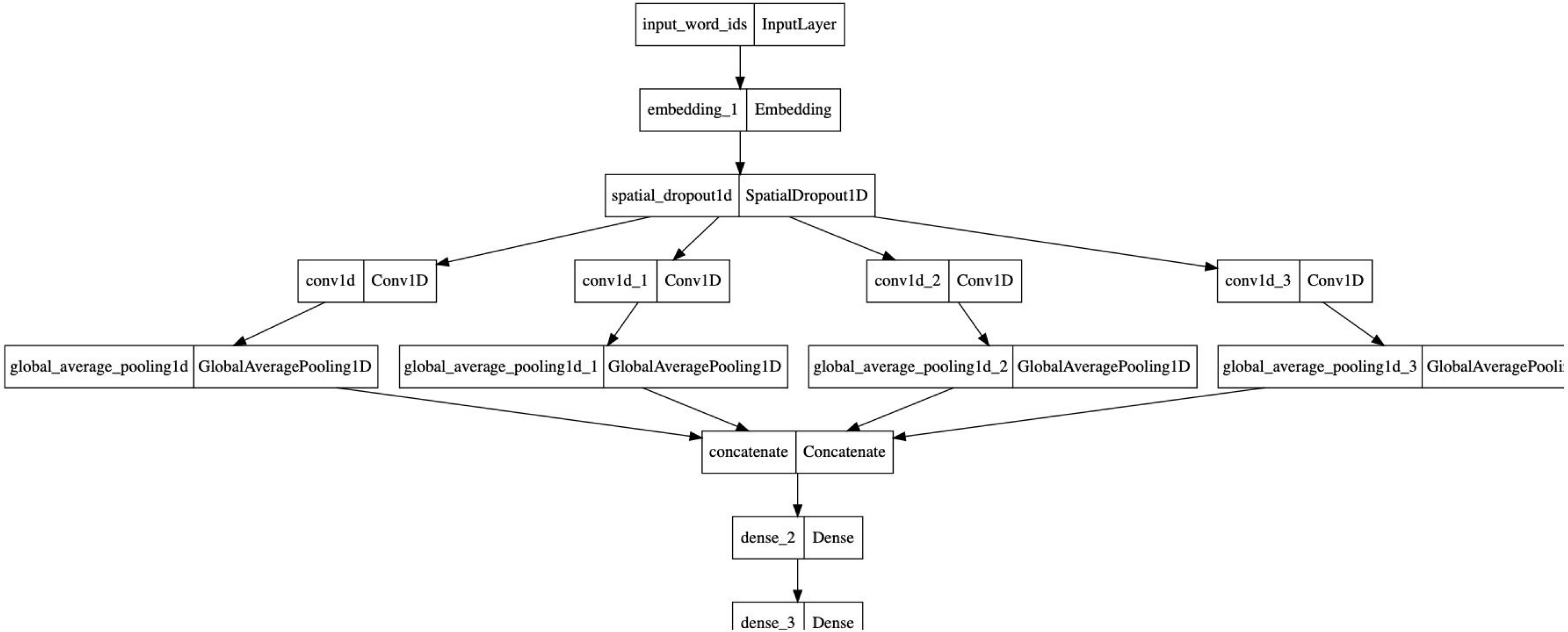


CNN with DistilBERT

```
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_word_ids (InputLayer    [(None, 512)]        0           []
 )

 embedding_1 (Embedding)        (None, 512, 768)     9181209     ['input_word_ids[0][0]']
                                                     6

 spatial_dropout1d (Spatial    (None, 512, 768)     0           ['embedding_1[0][0]']
 Dropout1D)

 conv1d (Conv1D)                (None, 511, 64)      98368       ['spatial_dropout1d[0][0]']

 conv1d_1 (Conv1D)              (None, 510, 64)      147520      ['spatial_dropout1d[0][0]']

 conv1d_2 (Conv1D)              (None, 509, 64)      196672      ['spatial_dropout1d[0][0]']

 conv1d_3 (Conv1D)              (None, 508, 64)      245824      ['spatial_dropout1d[0][0]']

 global_average_pooling1d (    (None, 64)           0           ['conv1d[0][0]']
 GlobalAveragePooling1D)

 global_average_pooling1d_1    (None, 64)           0           ['conv1d_1[0][0]']
  (GlobalAveragePooling1D)

 global_average_pooling1d_2    (None, 64)           0           ['conv1d_2[0][0]']
  (GlobalAveragePooling1D)

 global_average_pooling1d_3    (None, 64)           0           ['conv1d_3[0][0]']
  (GlobalAveragePooling1D)

 concatenate (Concatenate)      (None, 256)          0           ['global_average_pooling1d[0][
                                                                 0]',
                                                                  'global_average_pooling1d_1[0
                                                                 ][0]',
                                                                  'global_average_pooling1d_2[0
                                                                 ][0]',
                                                                  'global_average_pooling1d_3[0
                                                                 ][0]']

 dense_2 (Dense)                (None, 64)           16448       ['concatenate[0][0]']

 dense_3 (Dense)                (None, 1)            65          ['dense_2[0][0]']
```

Advanced text processing with multiple convolutional layers and global pooling.

CNN with DistilBERT



Transition to GloVe Tokenizer for Subsequent Models

(Global Vectors for Word Representation)
https://nlp.stanford.edu/projects/glove/

- **Objective:** Evaluate tokenizer impact on performance.
- **Comparison:**
  - *DistilBERT*: Contextual, computationally intensive.
  - *GloVe*: Fixed embeddings, efficient and scalable.
- **Reasons for Transition:**
  - Efficiency and scalability.
  - Performance evaluation across scenarios.

GloVe: Global Vectors for Word Representation

Jeffrey Pennington, Richard Socher, Christopher D. Manning

Introduction

GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.

Getting started (Code download)

- Download the latest latest code (licensed under the Apache License, Version 2.0). Look for "Clone or download"
- Unpack the files: unzip master.zip
- Compile the source: cd GloVe-master && make
- Run the demo script: ./demo.sh
- Consult the included README for further usage details, or ask a question

Download pre-trained word vectors

- Pre-trained word vectors. This data is made available under the Public Domain Dedication and License v1.0 whose full text can be found at: http://www.opendatacommons.org/licenses/pddl/1.0/.
  - Wikipedia 2014 + Gigaword 5 (6B tokens, 400K vocab, uncased, 50d, 100d, 200d, & 300d vectors, 822 MB download): glove.6B.zip
  - Common Crawl (42B tokens, 1.9M vocab, uncased, 300d vectors, 1.75 GB download): glove.42B.300d.zip
  - Common Crawl (840B tokens, 2.2M vocab, cased, 300d vectors, 2.03 GB download): glove.840B.300d.zip
  - Twitter (2B tweets, 27B tokens, 1.2M vocab, uncased, 25d, 50d, 100d, & 200d vectors, 1.42 GB download): glove.twitter.27B.zip
- Ruby script for preprocessing Twitter data

Citing GloVe

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. [pdf] [bib]

Highlights

1. **Nearest neighbors**

The Euclidean distance (or cosine similarity) between two word vectors provides an effective method for measuring the linguistic or semantic similarity of the corresponding words. Sometimes, the nearest neighbors according to this metric reveal rare but relevant words that lie outside an average human's vocabulary. For example, here are the closest words to the target word *frog*:

0. *frog*
1. frogs
2. toad
3. litoria
4. leptodactylidae
5. rana
6. lizard
7. eleutherodactylus

Optimizing Neural Networks with Hyperparameter Tuning

https://www.tensorflow.org/tutorials/keras/keras_tuner

- **Model Framework:** TensorFlow and Keras Tuner.
- **Model Structure:** Input (GloVe vectors), Flatten, Dynamic Dense Layer, Sigmoid Output.
- **Hyperparameters Tuned:** Neuron count (64-256), Learning rate, Batch size.
- **Optimization Method:** Hyperband, focusing on validation accuracy.
- **Results:** Enhanced model performance: Accuracy, Precision, Recall.

FNN with GloVe

Basic feedforward structure, optimized with learning rate reduction.

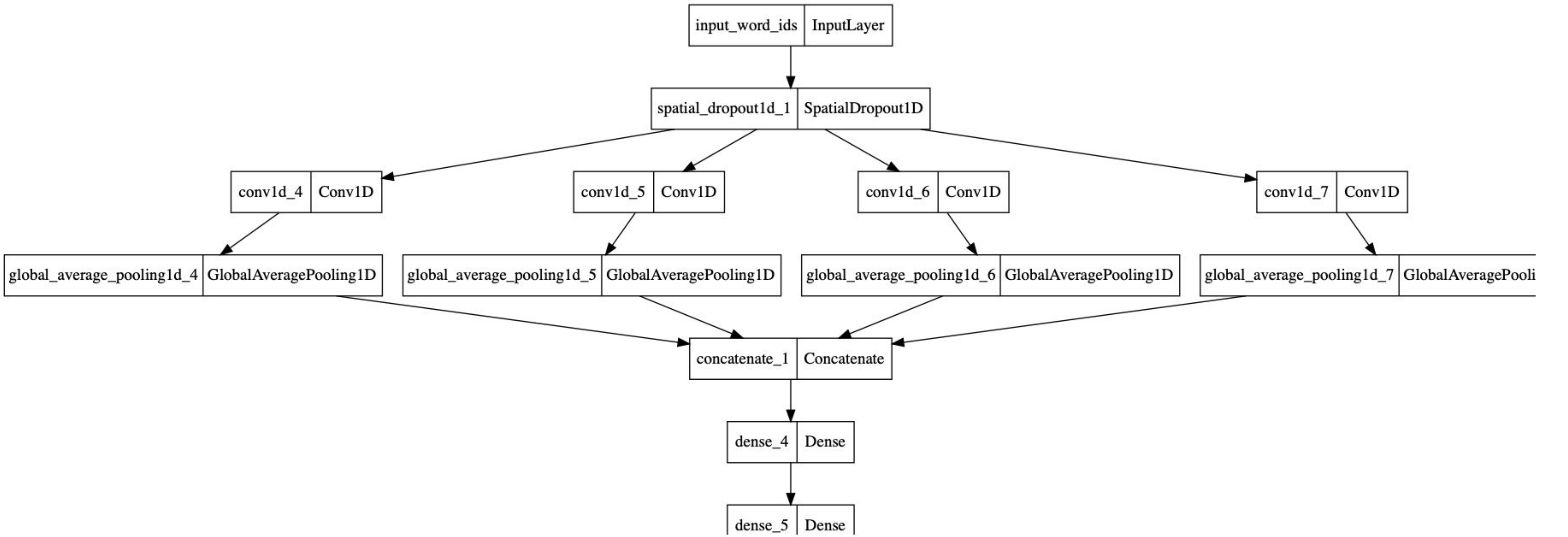
- **Model Type:** Feedforward Neural Network (FNN).
- **Input:** GloVe tokenized text data, reshaped to 100x50.
- **Architecture:** Flatten → Dense (128, ReLU) → Dense (1, Sigmoid).
- **Training:** Adam optimizer, binary cross-entropy, accuracy, precision, and recall metrics.
- **Adjustments:** Learning rate reduction on plateau.

CNN #1 with GloVe

```
_____________________________________________________________________________________________
 Layer (type)                Output Shape                 Param #   Connected to
=============================================================================================
 input_word_ids (InputLayer  [(None, 100, 50)]            0         []
 )

 spatial_dropout1d_1 (Spati  (None, 100, 50)              0         ['input_word_ids[0][0]']
 alDropout1D)

 conv1d_4 (Conv1D)           (None, 99, 64)               6464      ['spatial_dropout1d_1[0][0]']

 conv1d_5 (Conv1D)           (None, 98, 64)               9664      ['spatial_dropout1d_1[0][0]']

 conv1d_6 (Conv1D)           (None, 97, 64)               12864     ['spatial_dropout1d_1[0][0]']

 conv1d_7 (Conv1D)           (None, 96, 64)               16064     ['spatial_dropout1d_1[0][0]']

 global_average_pooling1d_4  (None, 64)                   0         ['conv1d_4[0][0]']
  (GlobalAveragePooling1D)

 global_average_pooling1d_5  (None, 64)                   0         ['conv1d_5[0][0]']
  (GlobalAveragePooling1D)

 global_average_pooling1d_6  (None, 64)                   0         ['conv1d_6[0][0]']
  (GlobalAveragePooling1D)

 global_average_pooling1d_7  (None, 64)                   0         ['conv1d_7[0][0]']
  (GlobalAveragePooling1D)

 concatenate_1 (Concatenate  (None, 256)                  0         ['global_average_pooling1d_4[0
 )                                                                  ][0]',
                                                                     'global_average_pooling1d_5[0
                                                                    ][0]',
                                                                     'global_average_pooling1d_6[0
                                                                    ][0]',
                                                                     'global_average_pooling1d_7[0
                                                                    ][0]']

 dense_4 (Dense)             (None, 64)                   16448     ['concatenate_1[0][0]']

 dense_5 (Dense)             (None, 1)                    65        ['dense_4[0][0]']

=============================================================================================
Total params: 61569 (240.50 KB)
Trainable params: 61569 (240.50 KB)
Non-trainable params: 0 (0.00 Byte)
```

Multiple convolutions and concatenation for feature integration.

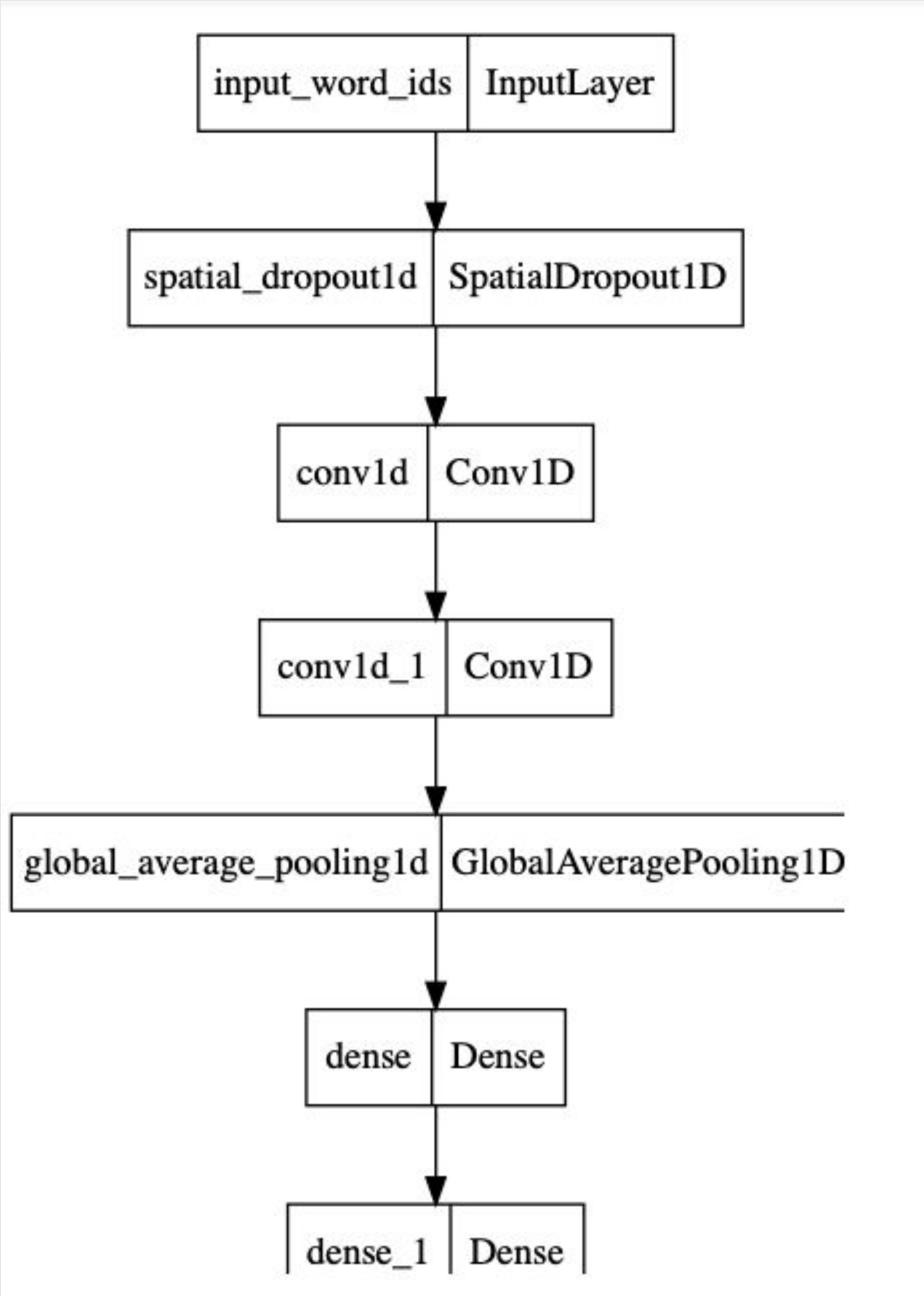
CNN #1 with GloVe



CNN #2 with GloVe

Fewer convolutional layers,
streamlined design for efficiency.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_word_ids (InputLayer  [(None, 100, 50)]         0
 )

 spatial_dropout1d (Spatial  (None, 100, 50)           0
 Dropout1D)

 conv1d (Conv1D)             (None, 98, 96)            14496

 conv1d_1 (Conv1D)           (None, 95, 64)            24640

 global_average_pooling1d (  (None, 64)                0
 GlobalAveragePooling1D)

 dense (Dense)               (None, 128)               8320

 dense_1 (Dense)             (None, 1)                 129

=================================================================
Total params: 47585 (185.88 KB)
Trainable params: 47585 (185.88 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```



Comparison of VNN, FNN, CNN

- **VNN with DistilBERT:** Simplistic design, high parameter count due to embedding layer.
- **CNN with DistilBERT:** Advanced text processing with multiple convolutional layers and global pooling.
- **FNN with GloVe:** Basic feedforward structure, optimized with learning rate reduction.
- **CNN #1 with GloVe:** Multiple convolutions and concatenation for feature integration.
- **CNN #2 with GloVe:** Fewer convolutional layers, streamlined design for efficiency.

Takeaway:

- VNN and CNN with DistilBERT are parameter-heavy, whereas GloVe-based models are more parameter-efficient.
- CNN models potentially offer better localized feature extraction compared to VNN and FNN.

DistilBERT

Pros:

- Efficiency
- Resource-friendly

Cons:

- Reduced Model Complexity
- Trade-off Between Performance and Size

RoBERTa

Pros:

- Improved Fine-Tuning
- Higher Performance

Cons:

- Resource Intensive
- Slower Inference

ALBERT

Pros:

- Parameter Reduction Techniques
- Scalability

Cons:

- Complex Implementation
- Potential Underfitting

M I N I
M A L

B U S I
N E S
S

4

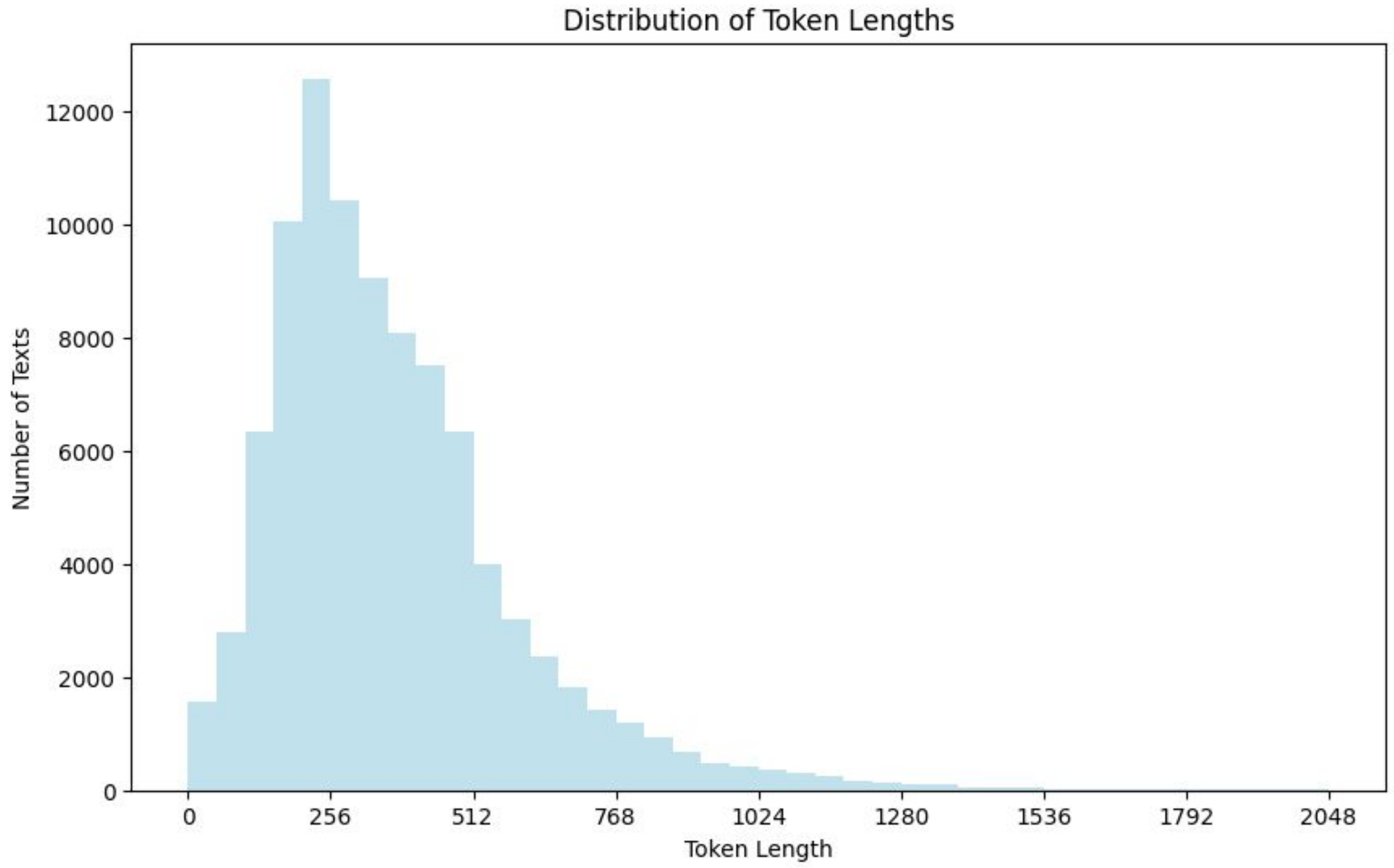
Experimental Details:

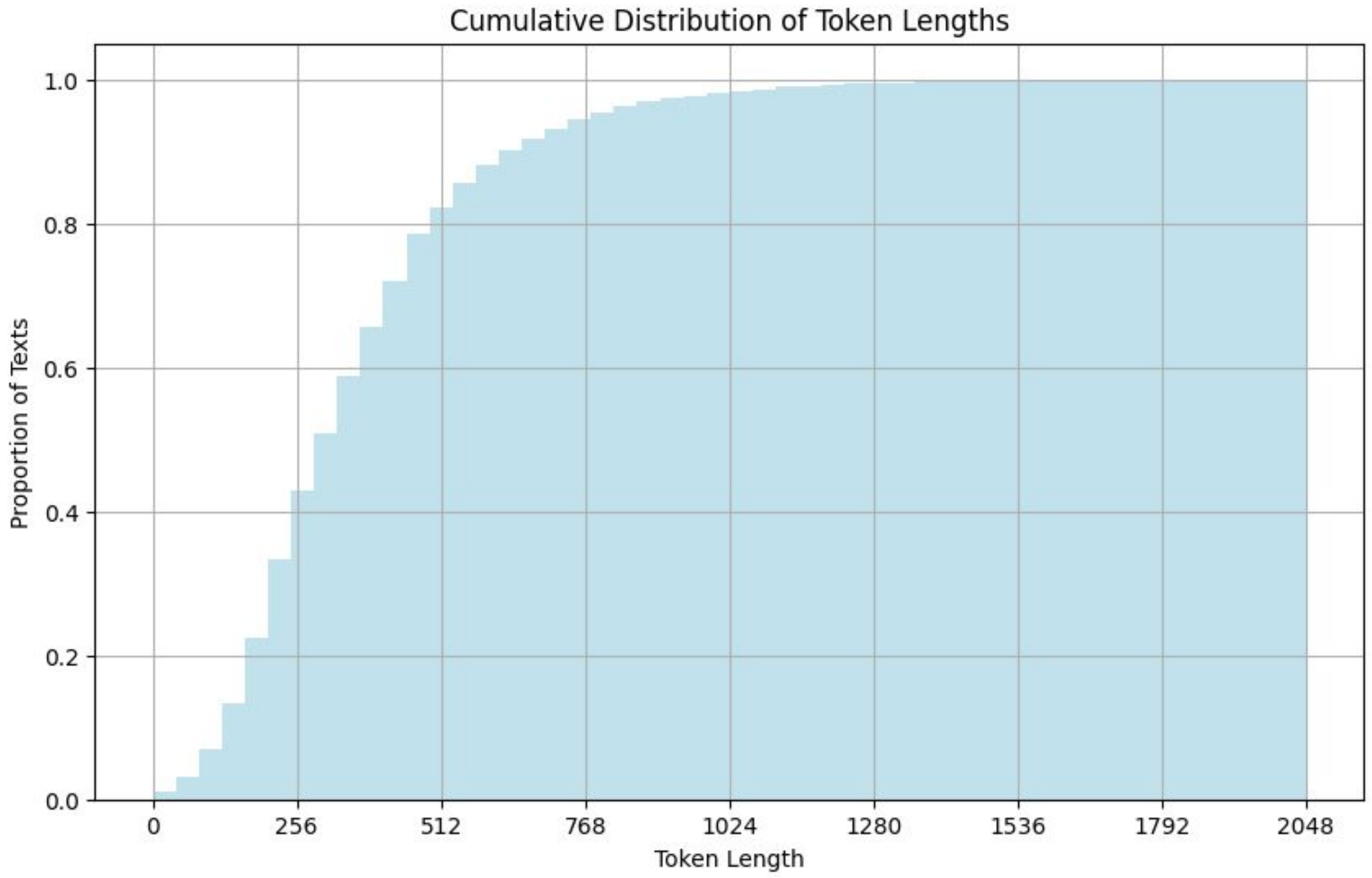
Ratur mos eseque estiis dolutati optassi molori to maio odi veliquae pel et, nihit re iur, ommolup tatende moluptae por sapit ulparu jmaio idis es di doluptaero odite esendunt.

M I N I M A L

B U S I N E S S

Empirical Test to Optimize Token's max_legnths





Challenges on Resources!

DistilBERT Hyper Parameters Configurations

- Configuration 1: `max_len = 1024`, `batch_size = 64` DEAD
- Configuration 2: `max_len = 512`, `batch_size = 64` DEAD
- Configuration 3: `max_len = 512`, `batch_size = 32` DEAD
- Configuration 4: `max_len = 512`, `batch_size = 16` `Work!`
- Configuration 5: `max_len = 256`, `batch_size = 64` `Work!`

......

- Configuration 9: `max_len = 128`, `batch_size = 128` `Work!`
- Configuration 10: `max_len = 128`, `batch_size = 64` `Work!` →

System RAM: 51GB

GPU RAM: 16GB

RoBERTa
ALBERT

Optimization used Adam optimizer with a learning rate of 3e-5

Loss evaluation used sparse categorical cross-entropy loss to measure performance

VNN with DistilBERT Result

```
 ROC-AUC - epoch: 2 - score: 0.710473
1744/1744 [==============================] - 31s 18ms/step - loss: 0.6195 - accuracy: 0.6423 - lr: 0.0010
Epoch 3/10
1741/1744 [============================>.] - ETA: 0s - loss: 0.6150 - accuracy: 0.6461WARNING:tensorflow:Lear

 ROC-AUC - epoch: 3 - score: 0.709967
1744/1744 [==============================] - 30s 17ms/step - loss: 0.6149 - accuracy: 0.6461 - lr: 0.0010
Epoch 4/10
1743/1744 [============================>.] - ETA: 0s - loss: 0.6127 - accuracy: 0.6475WARNING:tensorflow:Lear

 ROC-AUC - epoch: 4 - score: 0.703699
1744/1744 [==============================] - 31s 18ms/step - loss: 0.6127 - accuracy: 0.6475 - lr: 0.0010
Epoch 5/10
1743/1744 [============================>.] - ETA: 0s - loss: 0.6111 - accuracy: 0.6513WARNING:tensorflow:Lear

 ROC-AUC - epoch: 5 - score: 0.709908
1744/1744 [==============================] - 35s 20ms/step - loss: 0.6112 - accuracy: 0.6513 - lr: 0.0010
Epoch 6/10
1741/1744 [============================>.] - ETA: 0s - loss: 0.6087 - accuracy: 0.6539WARNING:tensorflow:Lear

 ROC-AUC - epoch: 6 - score: 0.710768
1744/1744 [==============================] - 35s 20ms/step - loss: 0.6088 - accuracy: 0.6540 - lr: 0.0010
Epoch 7/10
1742/1744 [============================>.] - ETA: 0s - loss: 0.6065 - accuracy: 0.6569WARNING:tensorflow:Lear

 ROC-AUC - epoch: 7 - score: 0.712407
1744/1744 [==============================] - 33s 19ms/step - loss: 0.6066 - accuracy: 0.6568 - lr: 0.0010
Epoch 8/10
1743/1744 [============================>.] - ETA: 0s - loss: 0.6049 - accuracy: 0.6579WARNING:tensorflow:Lear

 ROC-AUC - epoch: 8 - score: 0.711606
1744/1744 [==============================] - 31s 18ms/step - loss: 0.6049 - accuracy: 0.6579 - lr: 0.0010
Epoch 9/10
1743/1744 [============================>.] - ETA: 0s - loss: 0.6035 - accuracy: 0.6605WARNING:tensorflow:Lear

 ROC-AUC - epoch: 9 - score: 0.712692
1744/1744 [==============================] - 33s 19ms/step - loss: 0.6035 - accuracy: 0.6605 - lr: 0.0010
Epoch 10/10
1743/1744 [============================>.] - ETA: 0s - loss: 0.6027 - accuracy: 0.6613WARNING:tensorflow:Lear

 ROC-AUC - epoch: 10 - score: 0.712310
1744/1744 [==============================] - 30s 17ms/step - loss: 0.6027 - accuracy: 0.6613 - lr: 0.0010
```

CNN with DistilBERT Result

```
Epoch 1/2
1744/1744 [==============================] - ETA: 0s - loss: 0.4719 - accuracy: 0.7651
 ROC-AUC - epoch: 1 - score: 0.934770
1744/1744 [==============================] - 591s 338ms/step - loss: 0.4719 - accuracy: 0.7651 - val_loss: 0.3731 - val_accuracy: 0.8194 - lr: 0.0010
Epoch 2/2
1744/1744 [==============================] - ETA: 0s - loss: 0.4085 - accuracy: 0.8146
 ROC-AUC - epoch: 2 - score: 0.945020
1744/1744 [==============================] - 617s 354ms/step - loss: 0.4085 - accuracy: 0.8146 - val_loss: 0.3207 - val_accuracy: 0.8780 - lr: 0.0010
```

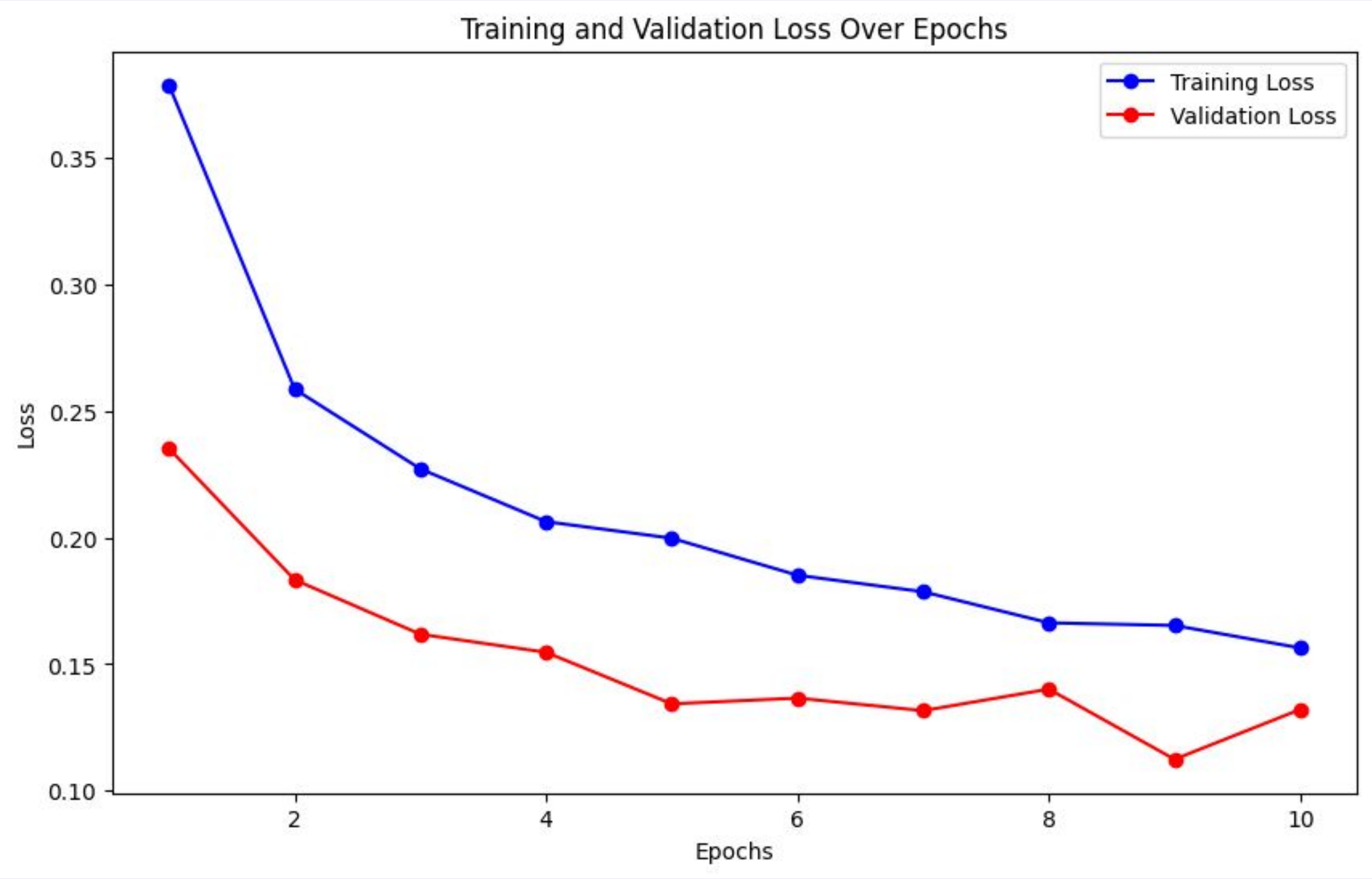
FNN with GloVe Result

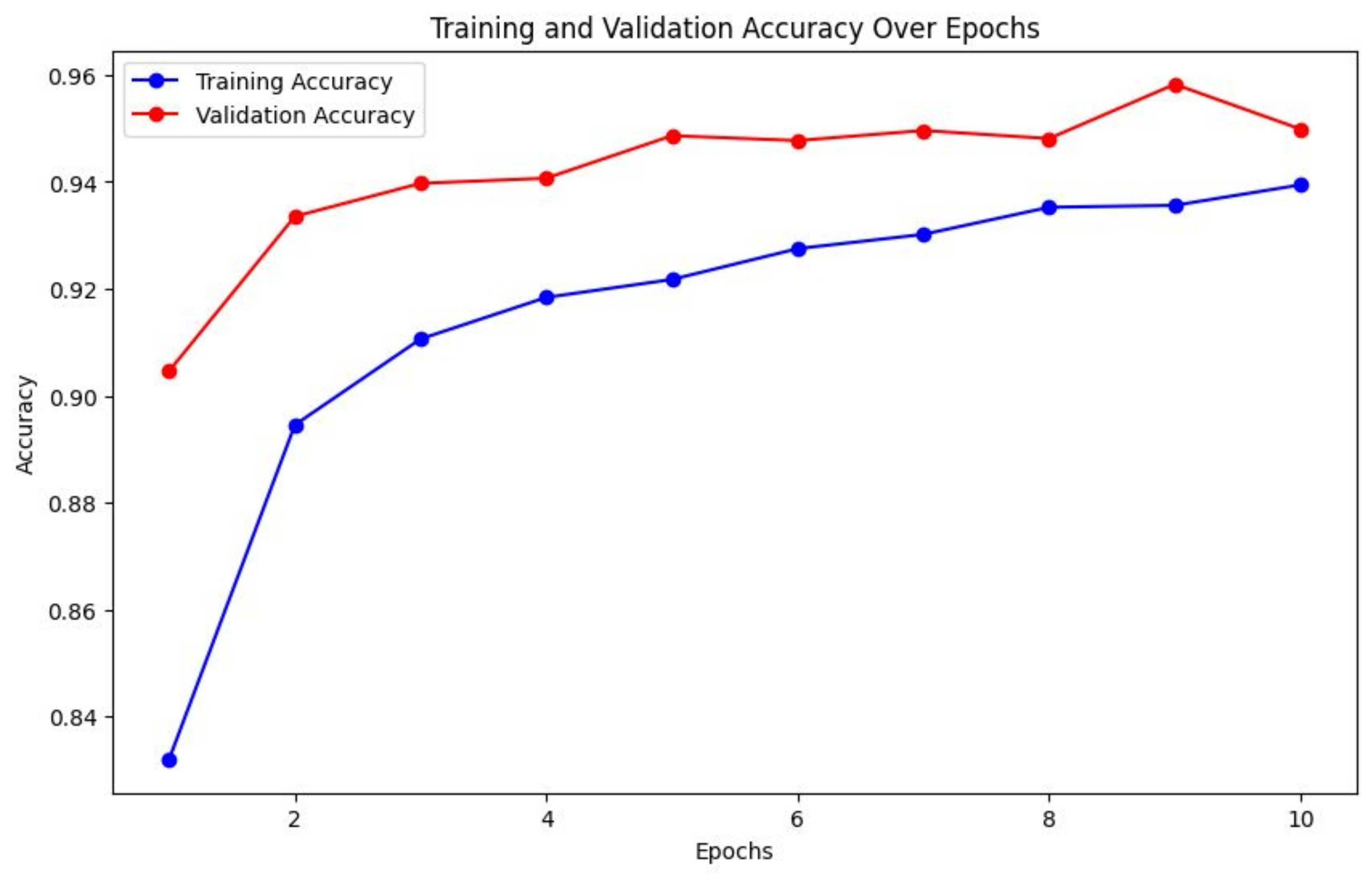
Metric	Details
Trial Complete	Trial 167 Complete [00h 03m 27s]
val_accuracy	0.8827574849128723
Best val_accuracy	So Far: 0.8827574849128723
Total elapsed time	01h 53m 21s
Search	Running Trial #168

Value	Best Value So Far	Hyperparameter
96	224	units
tanh	tanh	activation
rmsprop	rmsprop	optimizer
0.0057044	0.000663	initial_lr
0.00070378	0.0015202	lr
0.00017971	0.00027451	learning_rate
64	32	batch_size
40	40	tuner/epochs
14	14	tuner/initial_epoch
1	1	tuner/bracket
1	1	tuner/round
0159	0161	tuner/trial_id

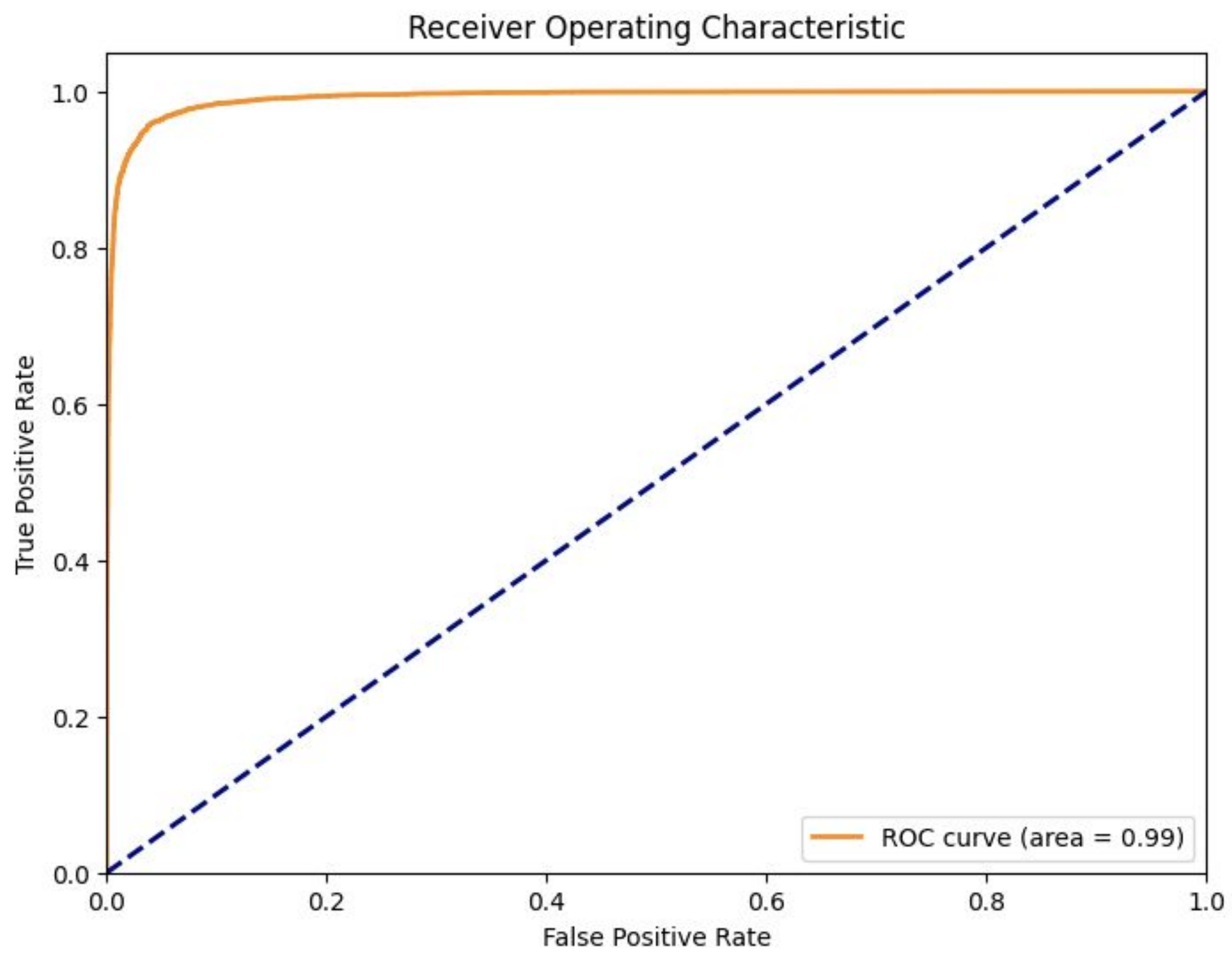
Epoch	Steps	Loss	Accuracy	Precision	Recall	Val Loss	Val Accuracy	Val Precision	Val Recall
15/40	873	0.0059	0.9999	0.9999	0.9999	0.4428	0.8742	0.8709	0.8458
16/40	873	0.0042	0.9998	0.9999	0.9998	0.4723	0.8749	0.8890	0.8249
17/40	873	0.0086	0.9993	0.9993	0.9992	0.5130	0.8719	0.8755	0.8339
...	...	...	...	...	...	...	...	...	...
36/40	873	9.0399e-04	0.9999	1.0000	0.9999	0.6927	0.8727	0.8613	0.8545
37/40	873	0.0042	0.9988	0.9987	0.9986	0.6706	0.8709	0.8794	0.8265

```
Test Accuracy: 87.99%
Test Loss: 0.39
Test Precision: 0.86
Test Recall: 0.87
582/582 [==============================] - 1s 2ms/step
Confusion Matrix:
[[9360 1156]
 [1080 7015]]
Classification Report:
              precision    recall  f1-score   support

     Class 0       0.90      0.89      0.89     10516
     Class 1       0.86      0.87      0.86      8095

    accuracy                           0.88     18611
   macro avg       0.88      0.88      0.88     18611
weighted avg       0.88      0.88      0.88     18611
```

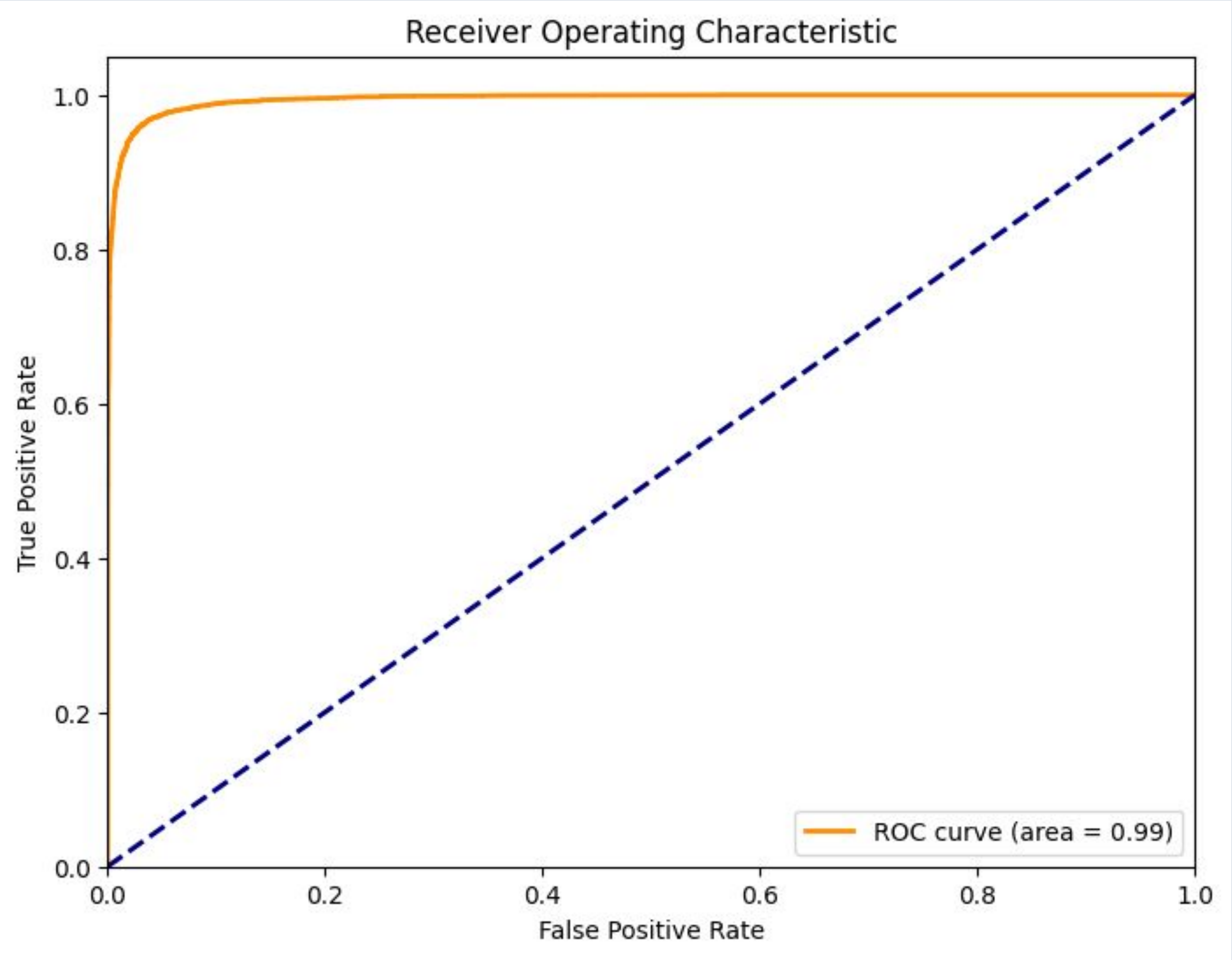




CNN #1 with GloVe Result




```
Confusion Matrix:
 [[9709  807]
 [ 180 7915]]
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.92      0.95     10516
           1       0.91      0.98      0.94      8095

    accuracy                           0.95     18611
   macro avg       0.94      0.95      0.95     18611
weighted avg       0.95      0.95      0.95     18611
```



CNN #2 with GloVe Result

```
582/582 [==============================] - 1s 2ms/step
Confusion Matrix:
[[10211   305]
 [  341  7754]]

Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.97      0.97     10516
           1       0.96      0.96      0.96      8095

    accuracy                           0.97     18611
   macro avg       0.96      0.96      0.96     18611
weighted avg       0.97      0.97      0.97     18611


582/582 [==============================] - 2s 3ms/step - loss: 0.0942 - accuracy: 0.9653
Test Accuracy: 0.9652893543243408
```

Comparative Results of VNN, FNN, CNN

- **VNN with DistilBERT:** Moderate accuracy and gradual increase in ROC-AUC.
- **CNN with DistilBERT:** High accuracy and ROC-AUC, quick convergence.
- **FNN with GloVe:** High accuracy, good precision and recall, efficient hyperparameter tuning.
- **CNN #1 with GloVe:** Exceptional accuracy and ROC curve, excellent validation performance.
- **CNN #2 with GloVe:** Highest accuracy and precision among models, excellent ROC curve.

Takeaway:

- VNN and CNN with DistilBERT are parameter-heavy, whereas GloVe-based models are more parameter-efficient.
- CNN models potentially offer better localized feature extraction compared to VNN and FNN.

MINI
MAL

BUSINESS

Experimental Evaluation:

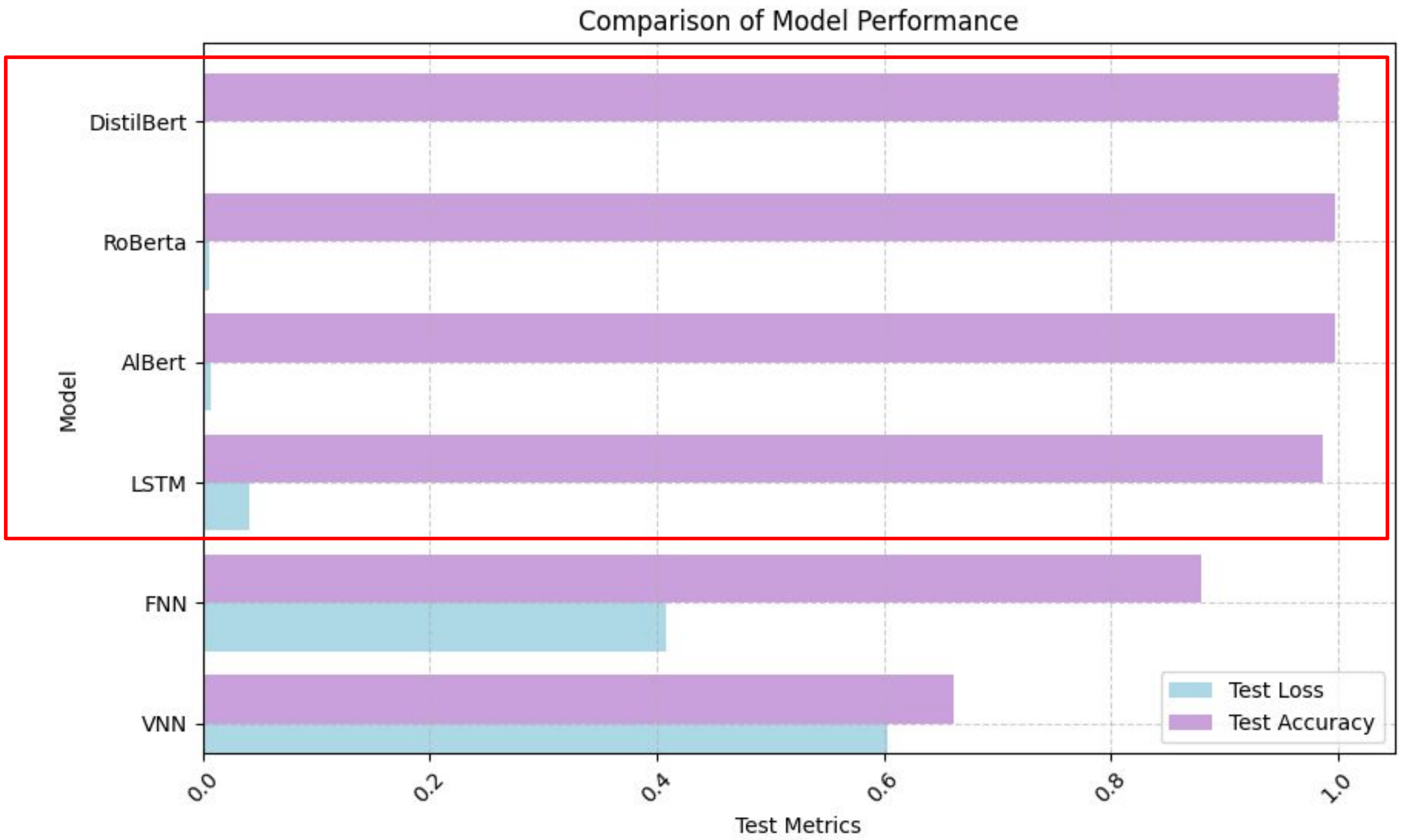
5

2023 PRESENTATION

M I N I
M A L

Models Competition on HC3 Dataset

B U S I
N E S S

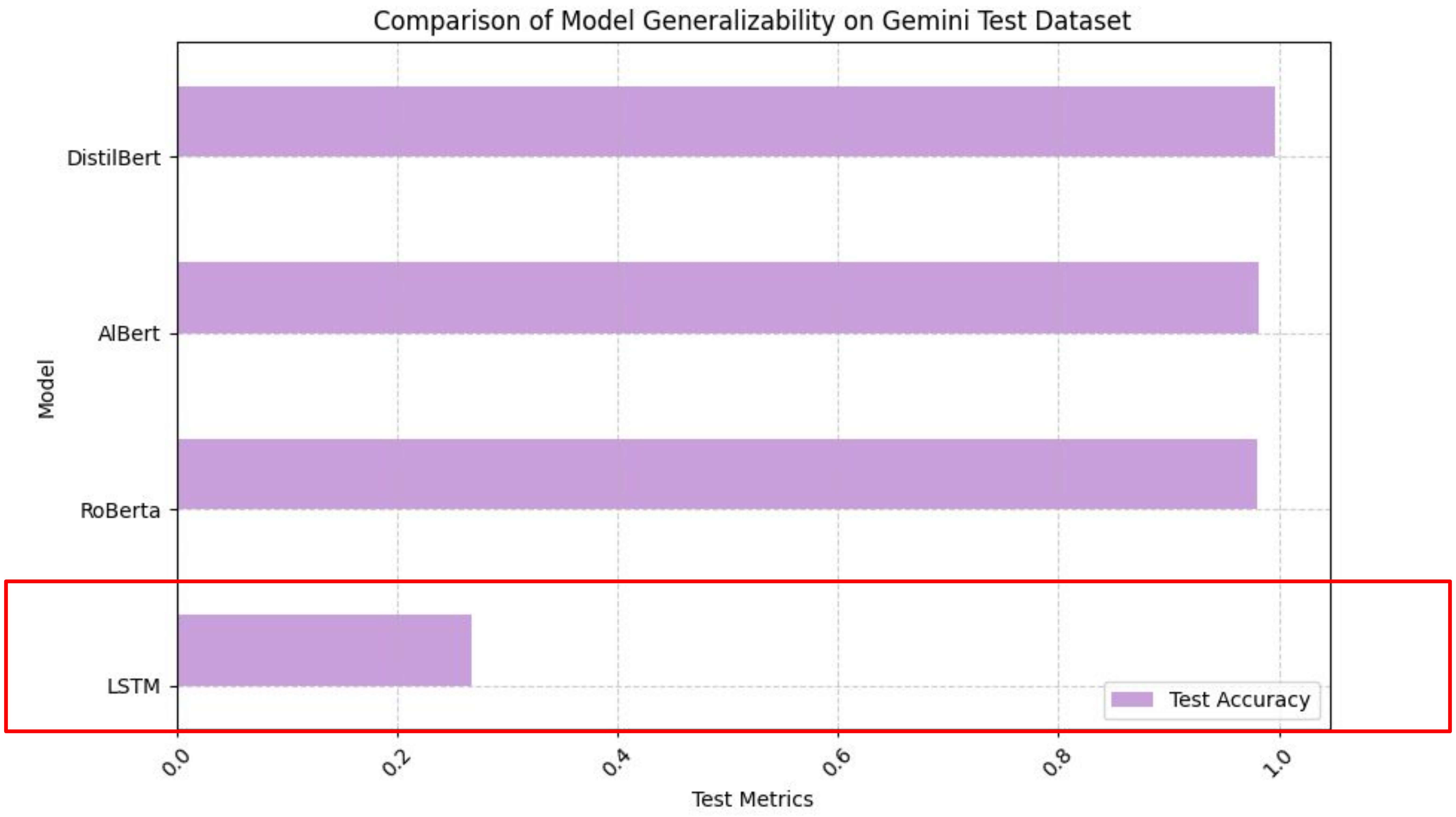


Continue to compete their generalizability

M I N I
M A L

Models Competition on Gemini Dataset

B U S I N E S S



Very poor generalization, out of the game.

M I N I
M A L

Models Competition on Overall Performance

B U S I N E S S

Models	Average Training Times per Epoch (s)	Test Accuracy on HC3 Dataset	Generalizability on Gemini Test Dataset	Generalizability on HC3-Chinese Test Dataset
DistilBert	298.67	0.9957	0.9734	0.5247
RoBerta	641.5	0.9982	0.9797	0.5000
AlBert	569	0.9977	0.9820	0.5293

Takeaway:

- Model Efficiency vs. Performance
- Generalizability Across Datasets

M I N I
M A L

Conclusions:

B U S I
N E S
S

6

Future works:

1. Explore broader model architectures, generate new datasets, and increase computational efficiency.
2. Enhancing the models' capabilities to handle diverse and multilingual datasets to ensure broader applicability and robustness.
3. Emphasis on developing detectors that can evolve with advancing AI capabilities.

M I N I
M A L

Background Works

B U S I
N E S
S

- **Kaggle notebooks:**

  Jigsaw Multilingual Toxicity : EDA + Models

- **GitHub References:**

  Awesome LLM-generated Text Detection

- **Papers:**

  A Survey on LLM-generated Text Detection: Necessity, Methods, and Future Directions.

  BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

  DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter

  ALBERT: A Lite BERT for Self-supervised Learning of Language Representations

  RoBERTa: A Robustly Optimized BERT Pretraining Approach

  GloVe: Global Vectors for Word Representation

M I N I
M A L

BUSINESS

Our
Professional Team..

Am no an listening depending up believing. Enough around it remove to burton's agreed regret in or it. there Advantage to burton's be comndedw

Thanks for Listening
Q&A