Deep Learning Tutorial: Final Project

NLP: Sentiment and Emotion Analysis

Minerva University

Introduction

In this paper, I built upon my previous assignment on sentiment analysis of IMDB movie review data using BERT. In the last assignment, I tried several types of small BERT models and a few different values of hyperparameter and the best result was a 86% binary accuracy. In that paper, I hypothesized that by using larger BERT models and more optimized hyperparameters, we could cross the 90% threshold. In this paper, I first tackled the same problem (IMDB sentiment analysis) but with larger BERT models and using a more systematic way of hyperparameter tuning. Then using this experience, I applied the same hyperparameter tuning strategy to a more complex problem of emotion analysis, where we need to classify the text into six different emotions; sadness, joy, love, anger, fear, and surprise.

Part 1: Sentiment Analysis

Previous Assignment Summary

Data Collection and Preparation

The IMDB dataset was collected from the Large Movie Review Dataset (Andrew et al., 2011). As the data have only the train and test directory, I made a validation set with 20% of the training data. The number of text in train, validation and test set are respectively 20,000, 5000, and 5000.

For preprocessing, I used the preprocessing unit provided by the tensorflow corresponding to each of the BERT models. In this step, first, we standardized the data by removing punctuations and HTML tags, etc. Then tokenized them by splitting the sentences into

words and adding special tokens. Finally, they are vectorized by embedding the tokens into same-sized vectors.

Base Model

As the base model, I used small BERT models (Turc et al., 2019), which are a smaller version of BERT. They have less number of transformer blocks (L), smaller hidden embedding size (H), and less attention head (A). I tried several different small BERT architectures. In the model, first the input texts are going through a preprocessing layer, then it goes through the main BERT model. The *pooled_output* of the BERT model corresponds to a single movie review. Thus we add a dense layer at the end which takes the pooled output as input and outputs a single number. But before the dense layer, a dropout layer have been added to reduce overfitting. Lastly, the activation function of the last layer is the sigmoid to have a result between 0 to 1, where 1 represents positive and 0 represents negative review.

As suggested by the original BERT paper (Devlin et al, 2018), I used the AdamW optimizer, Adam with weight decay instead of momentum). A linear schedule is used to reduce the learning rate over time to avoid being stuck in a plateau. As we have a single output (from 0 to 1) and the test labels are binary (0: negative, 1: positive), I used binary cross-entropy and binary accuracy as the loss and accuracy metrics.

Model Comparison Summary

The following table shows the validation accuracy with different models:

Model No.	Batch Size	Model	Initial Learning Rate	Scheduler	Dropout Rate	Epoch	Validation Accuracy	Runtime (min)
1	32	L=6, H=512, A=8	3e-5	yes	0.2	3	0.8532	21.43
2	32	L=6, H=512, A=8	3e-4	yes	0.2	3	0.8284	21.15
3	32	L=6, H=512, A=8	5e-5	yes	0.2	3	0.8552	20.8
4	32	L=6, H=512, A=8	5e-5	No	0.2	3	0.8516	20.9
5	32	L=8, H=512, A=8	5e-5	yes	0.2	3	0.8626	26.6
6	32	L=8, H=512, A=8	7e-5	yes	0.3	3	0.8554	26.51
7	32	L=8, H=512, A=8	3e-5	yes	0.3	3	0.8510	26.51
8	32	L=12, H=512, A=8	5e-5	yes	0.2	3	**0.8632**	37.51

Almost all the initial learning rates were of the same magnitude and best model is the model with the most transformer layer. The one model with a smaller magnitude learning rate and the model without a scheduler showed less accuracy in general.

Results

The following table shows the train, validation, and test accuracy with the best model.

Dataset	Loss	Accuracy
Train	0.1421	0.9507
Validation	0.4901	0.8632
Test	0.4659	0.8673

We only achieved 86% accuracy in the test dataset. Also, the difference in accuracy in train and test datasets suggests a possibility of overfitting. In order to improve our model, we will use a more complex BERT and a more systematic approach to hyperparameter tuning.

Extension: Larger Models

First I used the best hyperparameter from the last model and apply two more complex BERTs: the original BERT model (Devlin et al, 2018) L = 12, H = 768, and A = 12, and a modified lite

version: A Lite BERT (Lan et al (2019)). The following table shows that the original BERT works better and even runs faster. So we will use the original BERT for our further analysis.

Model No.	Batch Size	Model	Initial Learning Rate	Scheduler	Dropout Rate	Epoch	Validation Accuracy	Runtime (min)
1	32	L=6, H=512, A=8	3e-5	yes	0.2	3	0.8532	21.43
2	32	L=6, H=512, A=8	3e-4	yes	0.2	3	0.8284	21.15
3	32	L=6, H=512, A=8	5e-5	yes	0.2	3	0.8552	20.8
4	32	L=6, H=512, A=8	5e-5	No	0.2	3	0.8516	20.9
5	32	L=8, H=512, A=8	5e-5	yes	0.2	3	0.8626	26.6
6	32	L=8, H=512, A=8	7e-5	yes	0.3	3	0.8554	26.51
7	32	L=8, H=512, A=8	3e-5	yes	0.3	3	0.8510	26.51
8	32	L=12, H=512, A=8	5e-5	yes	0.2	3	0.8632	37.51
9	32	L=12, H=768, A=12 (Original BERT)	3e-5	yes	0.2	3	**0.8876**	32.02
10	32	L=12, H=768, A=12 (A Lite Bert, ALBERT)	3e-5	yes	0.2	3	0.8800	36.70

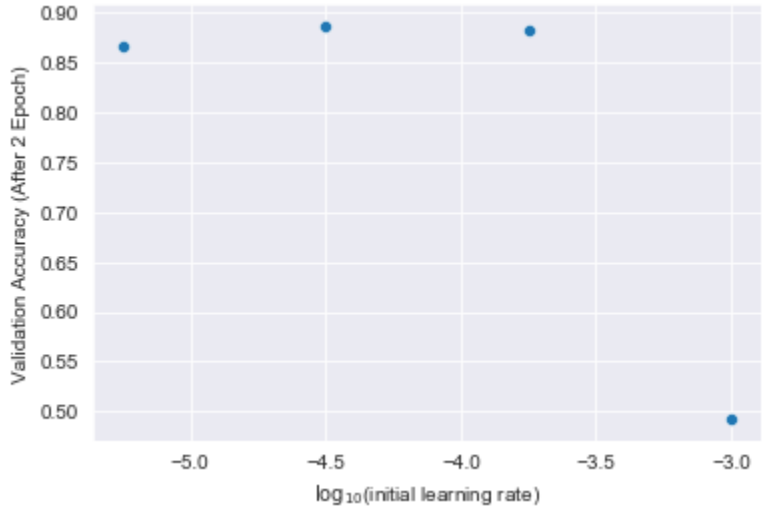
Extension: Hyperparameter Optimization

Primary: Gridsearch in Broader Range

The important hyperparameters in our problems are batch size, initial learning rate, dropout rate, and the use or not use of the scheduler. From the previous analysis, we found that the model without any learning rate scheduler often gets stuck in a plateau, thus it is recommended to use the scheduler. Similarly, as we increase the batch size, the model gets better in performance but the computation time increases a lot. I tried with a batch size of 64 in this problem, but I could not train the model using google colab GPU without crossing the time and memory limit. Thus I will use a batch size of 32.

The most important hyperparameter is the learning rate. We would first start with a broader search in the learning rate. For now, we want to find the optimal region for further investigation. A small additive change in learning rate does not change the model dynamics much. Thus it is important to search for the different magnitude of values. In this phase, we will

use grid-search for two reasons. One, we are only interested to know the optimal range instead of the precise optimal value. Second, a random search in a broader range will need more iterations to cover the entire range. As the BERT paper suggested a value around 3e-5, the range I considered is $[\,10^{-6},\ 10^{-3}]$. Considering the resource constraint, I only search for 5 different values in this range chosen by a NumPy *linspace* function for only 2 epochs. The model with the lowers learning rate results in an OOM (out of memory). The following figure shows the validation accuracy for different learning rates.



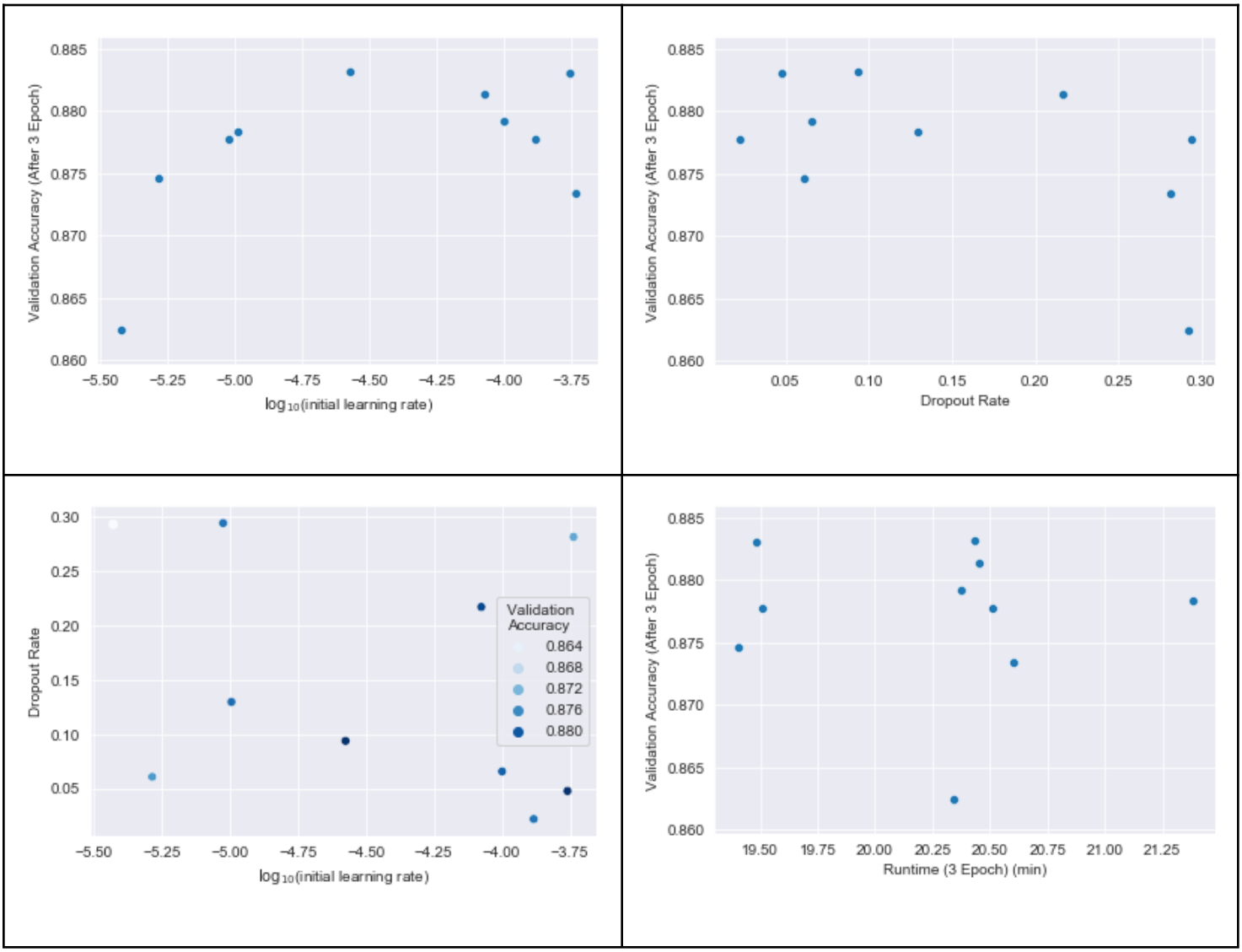
The figure suggests that the optimal value should lie in the $[\,10^{-5.5},\ 10^{-3.5}]$. range. For smaller values, we are getting OOM and for larger values, the accuracy decreases.

Secondary: More Granular Random Search

Once we get the smaller optimal region for learning rate, now we can do a more granular search for both learning rate and dropout rate. We will check the dropout rate in the range of (0, 0.4). We used random search instead of grid-search because of its two advantages (Bergstra &

Bengio, 2012). One, we can set to do a finite number of iterations in a random search instead of completing all the grid points, which is computationally less costly. Two, if we sample randomly, there is a higher chance that we could explore the important part of the parameters more than a grid search, especially when we have a large number of parameters.

For each iteration, I randomly chose a learning rate (*10**np.random.uniform (-5.5,-3.5)*) and the dropout rate (*np.random.uniform(0,0.4)*). The more iterations I will do, I can do a more extensive parameter search but that will take more computation power. I was able to run only for 10 different iterations. The comparison summary and the visualizations are presented below.



We can see that around a value of $10^{-4.6}$ for the learning rate, the model works better. But we can be more confident and precise in our results if we could do more iterations. For the dropout rate, until a certain dropout rate (around 30%), we can get similar accuracy for small variations of the dropout rate. The 3rd picture shows that better accuracy is usually found with a smaller dropout rate and larger than $10^{-4.6}$ initial learning rate. The runtime vs accuracy plot shows that with a good hyperparameter value, it is possible to get good accuracy without running the model for more time.

Model Comparison Summary

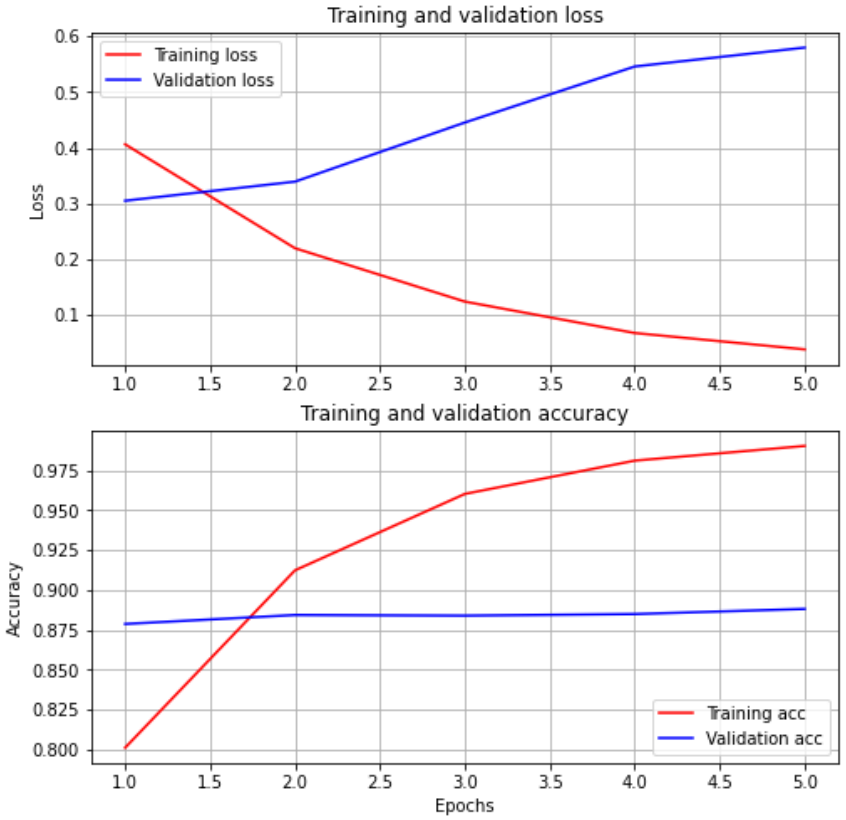
	Dropout Rate	Initial Learning Rate	Runtime (min)	Validation Accuracy
0	0.065645	0.000100	20.377045	0.8792
1	0.292564	0.000004	20.344030	0.8624
2	0.060683	0.000005	19.402505	0.8746
3	0.280994	0.000184	20.604098	0.8734
4	0.021925	0.000131	19.511795	0.8778
5	0.093675	0.000027	20.435145	0.8832
6	0.129455	0.000010	21.380731	0.8784
7	0.216669	0.000084	20.455474	0.8814
8	0.047688	0.000174	19.484247	0.8830
9	0.293704	0.000009	20.510026	0.8778

From all our iterations, we chose the value that gave the best validation accuracy. Then we run the model for 5 epochs before finally evaluating it with the test data.

Results

The accuracy for three different datasets are provided below. We reached an 89% accuracy with the test data. The difference in accuracy between train and test data in the table and the difference between test and validation in the plot shows that it still there are overfitting happening, though the validation accuracy continues to increase slightly over the epochs.

Dataset	Loss	Accuracy
Train	0.0382	0.9904
Validation	0.5796	0.8882
Test	0.5670	0.8872



Part 2: Emotion Analysis

Data Collection and Preparation

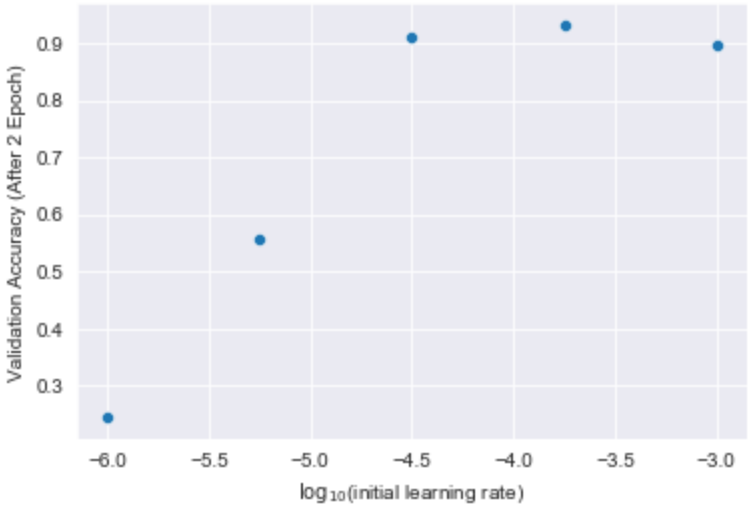
Now for this part we used the emotion analysis dataset (Saravia et al 2019), where we have texts and their classification label, with possible values including sadness (0), joy (1), love (2), anger (3), fear (4), and surprise (5). The datasets package provides the test, train and validation datasets of 16000, 2000, and 2000 entries correspondingly.

We follow the same structure as the previous model. But as we have six different output, the final output layer is a shape of 6. Similarly, instead of sigmoid, we used softmax as the final activation function to find the probability distribution of being into each of the six categories. The loss and accuracy loss metrics are the multiclass categorical cross-entropy and categorical accuracy. The original BERT model is used as the main neural network model.

Hyperparameter Optimization

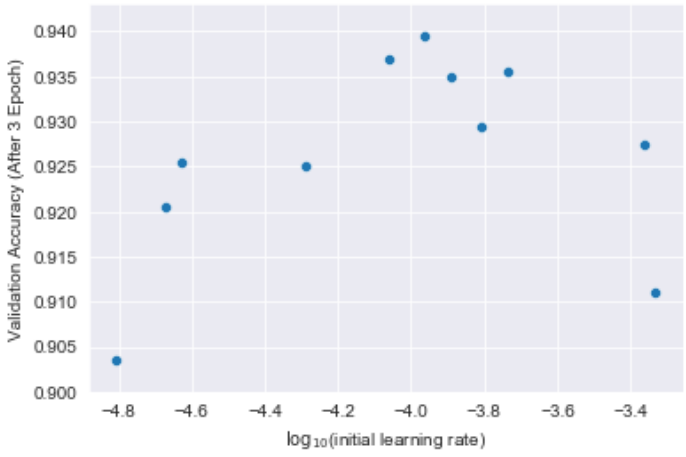
Primary: Grid Search

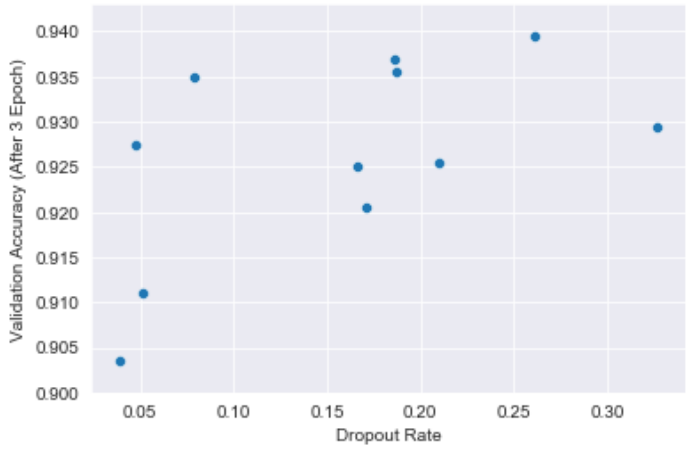
First we searched the initial learning rate for a broader range using grid search. The following plot shows that the optimal value likely lies between $[10^{-5}, 10^{-3}]$.

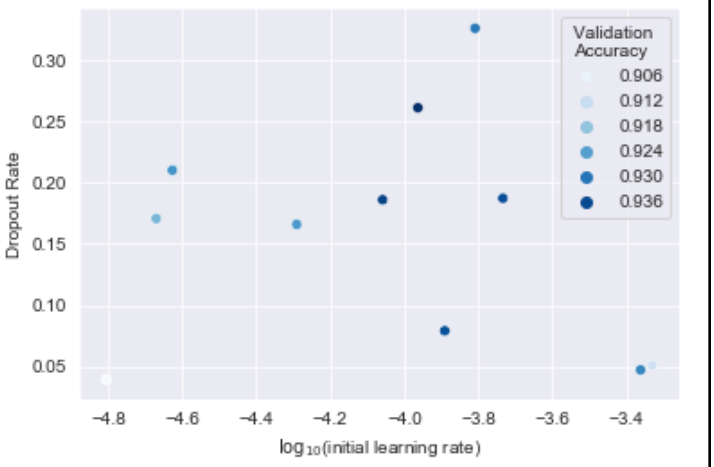


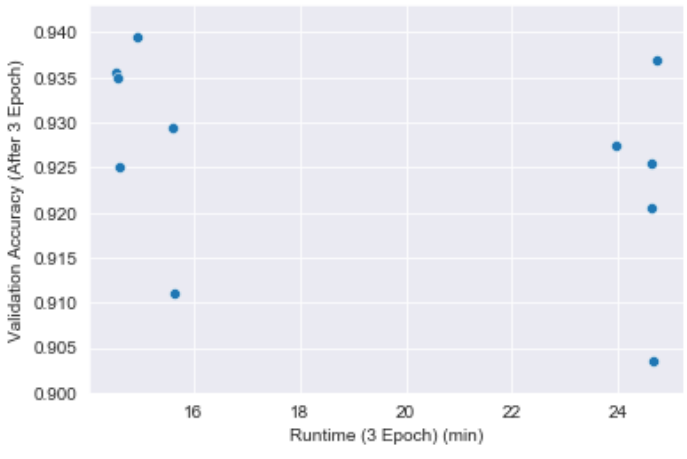
Secondary: Granular Random Search

Now again I did 11 different iterations of random search, where we chose a random learning rate from the optimal range and from the dropout range of (0,0.4).









The learning rate around 10^{-4} seems like the best learning rate for this problem, while there is no strong association with the dropout rate. We found a good model, for almost all ranges of dropout rates. The 3rd figure shows it more clearly that as long as the initial learning rate is around 10^{-4}, the validation accuracy is better. Similar to the last problem, the runtime does not influence the validation accuracy too much as long as we have good hyperparameters.

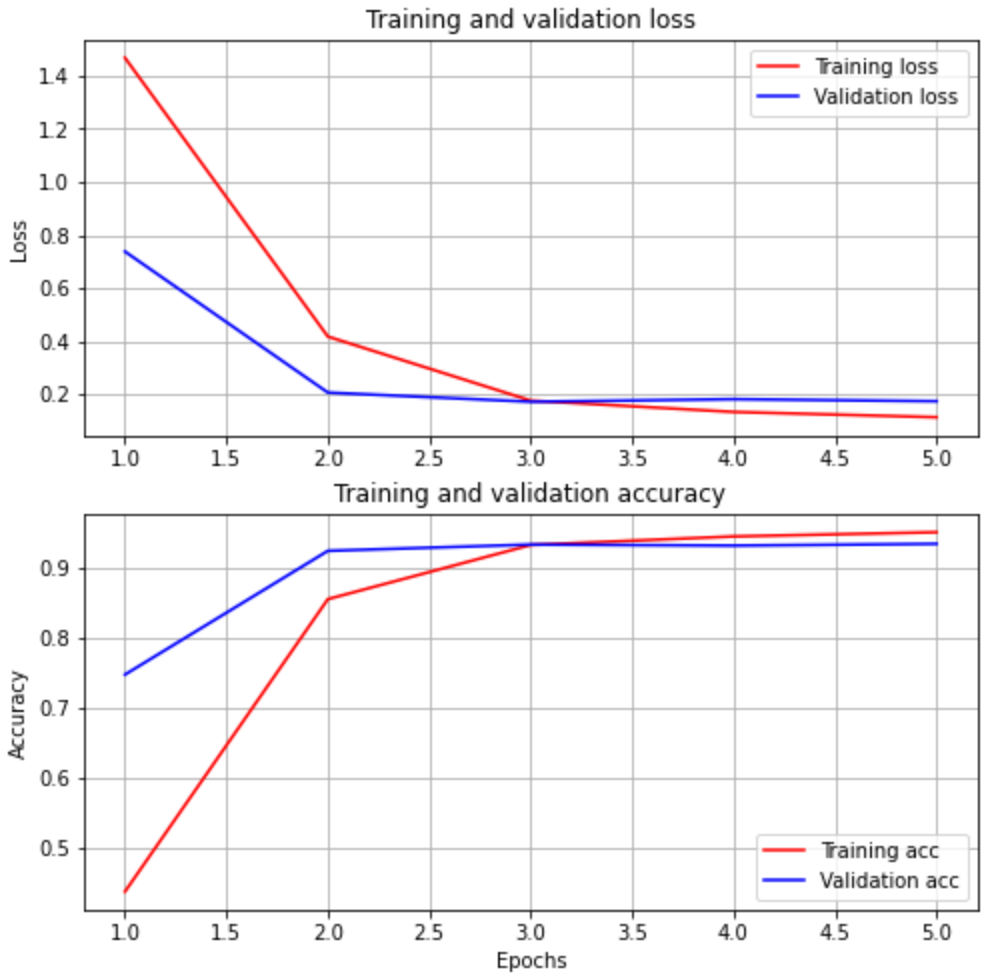
Model Comparison Summary

	Dropout Rate	Initial Learning Rate	Runtime (min)	Validation Accuracy
0	0.050994	0.000467	15.625041	0.9110
1	0.166048	0.000051	14.610250	0.9250
2	0.187358	0.000185	14.541972	0.9355
3	0.079060	0.000129	14.578095	0.9350
4	0.326302	0.000155	15.600972	0.9295
5	0.261312	0.000109	14.927945	0.9395
6	0.186133	0.000087	24.715508	0.9370
7	0.170812	0.000021	24.636170	0.9205
8	0.039383	0.000016	24.656373	0.9035
9	0.047166	0.000434	23.964654	0.9275
10	0.210300	0.000024	24.623683	0.9255

Results

We choose the model with the best validation accuracy and run it for 5 epochs. This results in a 93% test set accuracy. The training and validation both increase only slightly over the epochs.

Dataset	Loss	Accuracy
Train	0.1138	0.9511
Validation	0.1741	0.9345
Test	0.2248	0.9300



Discussion and Future Work

We should try with a more number of iterations to get explore extensively the parameter space. Though the original paper suggests using AdamW, we can try to use Adam, RMSProp and other optimizers. The associated hyperparameters like weight decay, momentum beta parameter, and regularizer alpha, could also be added in the get_random_param() function to find the optimal combinations. Here we only added a last layer to the BERT model. We could try with different architectures of BERT and also maybe add two or more layers with different sizes of hidden layers.

References

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). Learning Word Vectors for Sentiment Analysis. _The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011)_ http://www.aclweb.org/anthology/P11-1015

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. Journal of machine learning research, 13(2). http://jmlr.org/papers/v13/bergstra12a.html

Devlin, J., Chang, M., Lee, K., Toutanova, K., (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", 2018. https://arxiv.org/abs/1810.04805

Lan Z., Mingda C., Sebastian G, Kevin G, Piyush S., Radu S. (2019) ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. arXiv preprint https://arxiv.org/abs/1909.11942 , 2019

Saravia, E., Liu, H., Huang, Y., Wu, J., Chen, Y. (2018) Contextualized Affect Representations for Emotion Recognition, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, https://doi.org/10.18653/v1/D18-1404

TensorFlow Tutorial (2022), Classify Text with BERT. Retrieved from: https://www.tensorflow.org/text/tutorials/classify_text_with_bert

Turc, I., Chang, M., Lee, K., Toutanova, K. (2019). "Well-Read Students Learn Better: On the Importance of Pre-training Compact Models". https://arxiv.org/abs/1908.08962

Code Appendix

Sentiment Analysis Notebook:

https://colab.research.google.com/drive/140rkzOL414Hn3ZXEIuKJQyG-4pcqUNdh?usp=sharing

Emotion Analysis Notebook:

https://colab.research.google.com/drive/1n5OvPsQMxaXMXYvGoZbwl9PcUtg0sgRq?usp=sharing