MINION vs BERT: Efficient Fine-Tuning

Gero Embser, Carlos Vonessen, Leo Widmer
Banana-Fueled NN
Department of Computer Science, ETH Zurich, Switzerland

Abstract—**The past decades have seen an explosion in the use of microblogging and text messaging as popular communication channels. This rapid growth has resulted in an unprecedented volume of data, creating a pressing need for automated methods to analyze and understand the sentiments expressed in text. However, state-of-the-art methods are often expensive to train and exhibit overconfidence when faced with unseen dataset distributions. Parameter-efficient transfer learning (PETL) attempts to reduce the training cost associated with training large models, but often still requires significant computational resources to train models. In this paper, building on previous work, we implement a novel MINIature Orthogonal Network (MINION), that trains a small sequence-to-sequence recurrent neural network (RNN) alongside a large frozen transformer model, thus avoiding backpropagation through the larger transformer. We show that MINION requires lower training costs compared to previous implementations and full fine-tuning while reducing model overconfidence and maintaining high accuracy for text sentiment classification.**

I. Introduction

Text sentiment analysis is a problem that has been widely approached with neural networks (NN) and in recent years specifically large language models (LLM) [1], [2], [3], [4]. LLMs such as RoBERTa [1] create high-accuracy predictions for a wide range of NLP tasks and have become the standard in recent years. However, LLMs require very large amounts of data, high computational resources, and are very sensitive to choices in hyperparameters. Lacking time and resources, these difficulties make training an LLM or only fine-tuning it untenable in most cases. Research into parameter-efficient transfer learning (PETL) aims to address these issues. Instead of fine-tuning a full model, PETL methods introduce a few additional parameters to a pre-trained model or train a small subset of model weights, to adapt a pre-trained model to a specific task. Recently, the implementation of a REcurrent ADaptation (READ) model, that uses an RNN as a side model to a frozen LLM, has shown a significant reduction in training cost while exceeding the accuracy of fully finetuned models on several NLP tasks [5]. A further complication of modern neural networks is that they often exhibit overconfidence in their predictions [6], i.e. generate high probabilities for predictions with lower accuracy rate. Models with bad calibration reduce the interpretability of their outputs and increase risk in safety-critical applications [7], [6].

Faced with the task of training a well-calibrated, high-accuracy model with limited time and resources that is able to compete with an industry-standard like RoBERTa, we build on the side-tuning approach first implemented in READ [5].

1) We expand on READ by implementing a new side model configuration, that freezes the transformer and trains only a sequence-to-sequence RNN,
2) We show higher accuracy using the same RNN hidden layer size and fewer parameters in our side model compared to READ,
3) We show that our approach yields a significant reduction in training costs compared to full fine-tuning while maintaining high accuracy and reducing calibration error.

II. Models and Methods

An overview of the implemented models can be found in Table I.

A. Baseline Models

To evaluate the performance of our approach, we implement two conventional baseline models and one baseline model more specific to our implementation.

1) FastText: In the context of natural language processing (NLP) tasks, like sentiment analysis, FastText [8] is a common and straightforward baseline. It is a shallow neural network as it only has two layers: An embedding layer and a classification layer. In contrast to most state-of-the-art models, it is not based on a transformer architecture but relies on the ”bag-of-words” model. The text embeddings are averaged together to create a fixed-length representation of the input. It then uses a shallow NN for classification. Compared to LLMs, it has extremely low training costs and very high inference speeds. This makes it a good baseline, as using it is often an attractive alternative in NLP tasks when one has limited data and computational resources.

2) BERTweet: The ubiquitous use of LLMs in NLP tasks has made such models meaningful baselines. In this paper, we fully fine-tune the BERTweet model [2] on our dataset. BERTweet is already fine-tuned on Twitter data and is based on the BERT architecture [4]. In using BERTweet as a baseline, we explore what model performance and computational costs are created by using an out-of-the-box LLM and fine-tuning it for our specific task. BERTweet is much more computationally expensive than FastText but

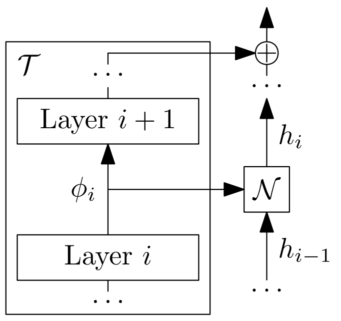

Figure 1. The READ architecture that we use for our experiments.

also achieves significantly higher accuracies on most NLP tasks [2]. Furthermore, fully fine-tuned BERTweet is an attractive baseline to us, as we use BERTweet as a part of our own implementations which gives a direct comparison for accuracy, computational cost, and calibration.

3) READ: As our final base model we use the REcurrent ADaptation (READ) model presented in [5], on whose principles we build our own implementation. It is useful as a final base model, as it is closest to our own implementation and thus clearly shows the difference our additions make. The goal of READ is to fine-tune a large pre-trained transformer $\mathcal{T}$ efficiently in terms of memory and energy consumption. Here, we explain how READ works if $\mathcal{T}$ has only an encoder and no decoder. $\mathcal{T}$ is frozen all throughout training. First, a forward pass is executed on $\mathcal{T}$ where one caches the results of all the layers, i.e. states $\phi_1, \ldots, \phi_N$. If we had fine-tuned the transformer directly we could obtain the desired states $\phi'_1, \ldots, \phi'_N$. Instead of doing this, READ tries to approximate so-called corrections $\delta\phi_i := \phi'_i - \phi_i$, which we then can add to ϕ_i to obtain the fine-tuned ϕ'_i. These approximated corrections, denoted as h_i, are computed using an RNN, which is unrolled alongside $\mathcal{T}$. Starting with a sequence of length m, we iterate through the states for i from 1 to N and compute

$$h_i^\alpha = \mathcal{N}\left(\phi_i^\alpha, h_{i-1}^\alpha\right) \ \forall \alpha \in \{1, ..., m\}, \tag{1}$$

where $\mathcal{N}$ is one step in the time of the RNN. We assume the embedding state needs no correction, so $h_0 = 0$. In the end, we map h_N to the dimension of the output state ϕ_N and return the approximation $\phi_N + h_N \approx \phi'_N$. The output sequence is then averaged and fed into a linear classifier. The basic structure is again illustrated in Figure 1. We implement the RNN as a bidirectional RNN using Gated Recurrent Units (GRUs). As a pre-trained network $\mathcal{T}$ we use BERTweet [2].

B. MINION

Our model works similarly to READ but uses a different function $\mathcal{N}$ in the computation of the corrections h_i. Here, $\mathcal{N}$ first computes $x_i = \mathcal{F}\phi_i + h_{i-1}$, where $\mathcal{F}$ is a linear layer, and then feeds the sequence $x_i = x_i^1, ..., x_i^m$ into a sequence-to-sequence RNN which returns $h_i^1, ..., h_i^m = h_i$. In the current description of READ, h_i^α and $h_i^{\alpha'}$ for $\alpha \neq \alpha'$ are independent of each other. In contrast, by using a sequence-to-sequence RNN, we expect that MINION efficiently captures the dependency structure arising from natural text, thus requiring fewer parameters for similar performance. See Figure 2 for the difference between READ and MINION in terms of information flow. Fewer parameters will also prevent the very large transformer model from overfitting on the relatively small data set. We call our RNN configuration an orthogonal RNN, as it also transmits information orthogonally to the transformer layers, in contrast, we call the original READ network a parallel RNN.
The sequence-to-sequence RNN is implemented with bidirectional Gated Recurrent Units (GRUs). We experiment with hidden sizes much smaller than in the original READ paper [5], beginning at 8 and increasing through successive powers of two until 256, which was the hidden size selected in the original paper. As the pre-trained network we use BERTweet [2].

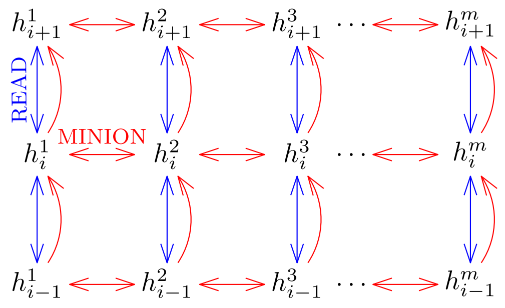

Figure 2. Information flow of READ vs. that of MINION.

C. Data

The data used for training the models is provided to us by the Data Analytics Lab of ETH Zurich as a part of the Text Sentiment Classification Competition in the Computational Intelligence Lab 2023. The unprocessed data consists of 2.5 million tweets, where exactly half are labeled to have positive sentiment, and the other half are labeled with negative sentiment. The data is anonymized, meaning that references to specific users, for example, "`@realDonaldTrump`" have been replaced with the tag "`<user>`". Any URLs in a tweet have also been replaced with the tag "`<url>`".
We processed the data to remove duplicate tweets, which leaves 1,127,628 positive sentiment tweets and 1,142,785 negative sentiment tweets. Furthermore, since hashtags often contain a lot of the sentiment expressed in the text, we used an automated word separation library[1] based on [9] to convert hashtags into normal text, i.e. "`<user> omg #ilovethislab`" becomes "`<user>`

[1] https://github.com/grantjenks/python-wordsegment

`omg i love this lab`". This preprocessing step was done to increase the model's semantic understanding of the hashtags. FastText especially profits from this, as it uses a bag-of-words implementation and thus does not understand hashtags. The other models are predominantly trained on normal text and thus also stand to benefit from converting the hashtags to simple words.

D. Energy Consumption

To compare the cost of training each model we evaluate the approximate energy consumption and memory utilisation of each model during training.

We calculate estimated energy consumption based on the implementation presented in [5]. We track GPU utilization u_i (in percent) and memory usage m_i (in MiB) every ten seconds during training. We further assume a linear relationship between power consumption and GPU utilization and use $p_0 = 0.4\text{kW}$ [2] as the maximum power utilization constant. Using H as the number of hours an experiment runs, we approximate the energy consumption E (kWh) with the average u_i value

$$E = \int_0^H p_0 \frac{u(t)}{100} dt \approx \frac{H}{360H} \sum_i \frac{p_0 \cdot u_i}{100} = \frac{p_0}{36000} \bar{U} \quad (2)$$

Here $\bar{U} = \sum_i u_i$ and one gets the 360, from the $6 \cdot 60 = 360$ datapoints per hour. The power is then calculated using the standard formula $P = E/H$.

E. Calibration Error

We call a model well-calibrated when its outputs, interpreted as probability distributions, are consistent with the empirically observed frequencies. In this binary classification task, the predictions of models regarding positive and negative sentiment can be interpreted as probabilities of class assignment or as the degree to which a tweet has positive or negative sentiment. This interpretation is only possible reliably if the model is well-calibrated.

Calibration errors are computed by dividing the range $[0, 1]$ into K intervals. For each interval, one computes the average predicted probability of a positive label ($conf$) and the fraction of true positive labels (acc) in that bucket B_k. The closer they match, the better the model calibration is. We quantitatively assess the total calibration error using the Adaptive Calibration Error (ACE) [10] and the Maximum Calibration Error (MCE) [6]. ACE spaces the intervals such that each interval contains an equal number of samples, while MCE spaces the intervals equally wide. We define

$$\text{ACE} := \frac{1}{K} \sum_k |acc(B_k) - conf(B_k)|, \quad (3)$$

$$\text{MCE} := \max_k \{|acc(B_k) - conf(B_k)|\}. \quad (4)$$

[2]See this NVIDIA datasheet

F. Implementation

All our models and their evaluations are implemented in Python. To create and expand the models we used TensorFlow [11]. We used the transformers API provided by HuggingFace [12] to add the pre-trained models into our workflow.

G. Experiments

In addition to BERTweet and FastText, we train READ and our newly developed MINION model. For READ and MINION, we perform a grid search to find the optimal configurations of the RNN hidden size and the learning rate. For the RNN hidden size, we explored the values $\{8, 16, 32, 64, 128, 256\}$, while for the learning rate, we considered $3 \cdot 10^{-4}$ and $1 \cdot 10^{-3}$. This results in a total of 12 different configurations for each of the two RNN-based models. We evaluate all metrics on a selection of these models. The final choices can be found in Table I.

For all experiments, we shuffled the preprocessed data and split it into an 80/10/10 training, validation, and test set split. All transformer-based models (BERTweet, READ, and MINION) were trained for 5 epochs, with a batch size of 256 and the Adam optimizer [13] with a first momentum value of 0.9 and a second momentum value of 0.999. FastText is optimized with a batch size of one using stochastic gradient descent (SGD).

During the experiments of our transformer-based models (BERTweet, READ, and MINION) we measure GPU utilization and memory using NVIDIA's System Management Interface (SMI)[3] to assess the computational effort required to train each model. To ensure comparability, the models were all trained on an NVIDIA A100 SXM GPU with 40GB RAM. We further evaluate the accuracy and the calibration error of each model, where we compare the ACE and MCE of all models.

III. Results

Table II shows the key metrics of selected models after training. We present the READ and MINION models with hidden unit sizes $\{8, 64, 256\}$, as they show model performances for the smallest and largest number of hidden units. Furthermore, we present MINION-64 as it achieves the best model calibration. To evaluate the calibration of the models we use the ACE and MCE with 20 buckets. We plot a selection of the calibration errors in Figure 3 to further illustrate our findings.

For the Kaggle competition [14] we submitted an evaluation of the MINION-64 trained on the entire dataset. It achieved a score of 0.9056 on the public leaderboard.

[3]https://developer.nvidia.com/nvidia-system-management-interface

Model Name	Model Type	Training Method	Trainable Params. (%)	Learning Rate
FastText	Shallow NN	Full Training	31.1	10^{-1}
BERTweet	Transformer	Full Fine-Tuning	100	$2 \cdot 10^{-5}$
READ-8 / 64 / 256	T + parallel RNN	RNN Side-Tuning	0.03 / 0.27 / 1.3	10^{-3}
MINION-8 / 64 / 256	T + orthogonal RNN	RNN Side-Tuning	0.02 / 0.2 / 1.44	10^{-3} / $3 \cdot 10^{-4}$ / $3 \cdot 10^{-4}$

Table I

MODEL ARCHITECTURES, THEIR TRAINING METHOD, THE NUMBER OF PARAMETERS, AND THE FINAL CHOSEN LEARNING RATE. WE USE THE FIRST THREE MODELS AND THEIR CONFIGURATIONS AS A BASELINE. WE USE T+RNN TO DENOTE A TRANSFORMER WITH AN RNN SIDE MODEL. THE 100% IN BERTWEET CORRESPONDS TO ITS $1.35 \cdot 10^8$ PARAMETERS, WHICH IS ALSO THE NUMBER OF FROZEN PARAMETERS IN THE T+RNN MODELS.

Model Name	Accuracy (%)	ACE (%)	MCE (%)	Energy (kWh)	Power (kW)	Memory (GiB)
FastText	82.83	0.79	1.75	-	-	-
BERTweet	90.42	4.74	13.11	1.39	0.39	33.11
READ-8/64/256	89.38 / 89.71 / 89.79	2.50 / 3.12 / 2.73	7.33 / 9.47 / 9.52	0.80 / 0.85 / 0.99	0.33 / 0.34 / 0.36	5.26 / 9.26 / 17.26
MINION-8/64/256	89.69 / 90.15 / 90.33	0.36 / 0.48 / 0.94	3.91 / 2.35 / 5.00	0.72 / 0.79 / 0.97	0.37 / 0.37 / 0.37	5.26 / 5.26 / 9.26

Table II

COMPARISON OF KEY METRICS ACROSS TRAINED MODELS. THE ACCURACY, ACE, AND MCE ARE EVALUATED ON OUR TEST DATA SPLIT. THE ENERGY AND MEMORY METRICS ARE OMITTED FOR FASTTEXT, AS THE COMPUTATIONAL EFFORT FOR TRAINING IT IS NEGLIGIBLE AND WAS NOT DONE ON A GPU. MEMORY ALWAYS DENOTES THE GPU PEAK TRAINING MEMORY.

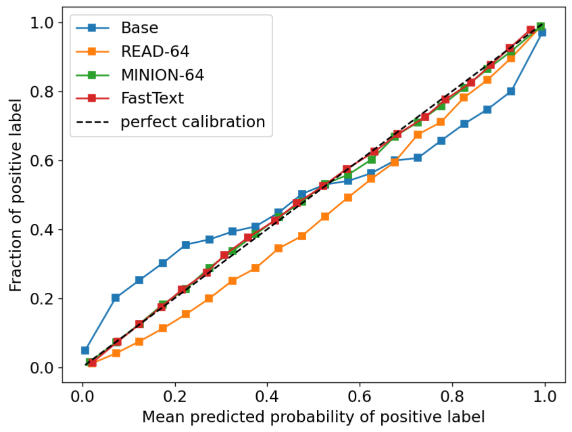

Figure 3. Calibration curves of the trained models for predicting positive sentiment. The dashed line represents the ideal model calibration. Regions, where the calibration curve of a model lies under the dashed line, represent overconfidence of the model in its predictions, and regions where it lies above represent underconfidence.

IV. DISCUSSION

BERTweet achieves the highest accuracy on the test set but is ahead of the other transformer-based models by less than one percent. MINION-8, 64, and 256 respectively only require about 52%, 57% and 71% of the energy required for fully fine-tuning BERTweet. The low-cost alternative FastText achieves much lower accuracy but only requires negligible resources.

The MINION models also show slightly higher accuracy compared to READ for each hidden unit size respectively, while requiring less energy and memory utilisation during training. One should note that our implementations of READ and MINION are not optimized, and thus this marginal improvement of energy consumption and memory utilization is tentative and may also relate to the internal implementation of each model.

The calibration errors in Table II clearly show an improvement in the model calibration of all models compared to BERTweet on both ACE and MCE. This suggests that model calibration is significantly improved when using recurrent side models compared to full fine-tuning. Comparing the calibration of READ to MINION, we can see a marked reduction in ACE and MCE. This could either be explained by the influence of the bidirectional GRU in each hidden layer of the transformer, or with the slight reduction in the number of parameters in MINION compared to READ. We can also observe a reduction in the ACE for MINION-8 compared to MINION-64. In line with previous findings [6] this suggests, that larger models exhibit worse calibration. However, considering that here only two models are compared and only one dataset is evaluated, more research has to be done to find more definitive answers.

V. SUMMARY

Overall, our implementation appears viable as a PETL method for sentiment classification, as it significantly reduces the computational cost required for training compared to full fine-tuning. MINION, by using a sequence-to-sequence RNN as a side model, further improves on READ [5] achieving slightly higher accuracy with lower computational cost with the same number of hidden units. Finally, on this classification task, our approach shows much better calibration compared to full fine-tuning and slightly better calibration compared to READ. However, more research on different NLP tasks will be necessary, to confirm whether it performs well in general.

References

[1] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," 2019. [Online]. Available: https://arxiv.org/abs/1907.11692

[2] D. Q. Nguyen, T. Vu, and A. T. Nguyen, "Bertweet: A pre-trained language model for english tweets," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2020, pp. 9–14.

[3] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2018.

[5] S. Wang, J. Nguyen, K. Li, and C.-J. Wu, "Read: Recurrent adaptation of large transformers," 2023. [Online]. Available: https://arxiv.org/abs/2305.15348

[6] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," 2017.

[7] J. Vaicenavicius, D. Widmann, C. Andersson, F. Lindsten, J. Roll, and T. Schön, "Evaluating model calibration in classification," in *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 2019, pp. 3459–3467.

[8] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching Word Vectors with Subword Information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 06 2017. [Online]. Available: https://doi.org/10.1162/tacl_a_00051

[9] T. Segaran and J. Hammerbacher, *Beautiful data: the stories behind elegant data solutions*. " O'Reilly Media, Inc.", 2009.

[10] J. Nixon, M. Dusenberry, G. Jerfel, T. Nguyen, J. Liu, L. Zhang, and D. Tran, "Measuring calibration in deep learning," 2020.

[11] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," 2016.

[12] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Huggingface's transformers: State-of-the-art natural language processing," 2020.

[13] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[14] C. Lecture, "Ethz cil text classification 2023," 2023. [Online]. Available: https://kaggle.com/competitions/ethz-cil-text-classification-2023

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

MINION vs BERT: Efficient Fine-Tuning

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
Embser	Gero
Vonessen	Carlos
Widmer	Leo

With my signature I confirm that
- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
Zurich, 31.07.2023	Gero Embser
	[illegible]
	L. Widmer

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.