Using Language Models on Low-end Hardware

Anonymous ACL submission

Abstract

This paper evaluates the viability of using fixed language models for training text classification networks on low-end hardware. We combine language models with a CNN architecture and put together a comprehensive benchmark with 8 datasets covering single-label and multi-label classification of topic, sentiment, and genre. Our observations are distilled into a list of trade-offs, concluding that there are scenarios, where not fine-tuning a language model yields competitive effectiveness at faster training, requiring only a quarter of the memory compared to fine-tuning.

1 Introduction

The transition to neural networks as primary machine learning paradigm in natural language processing (NLP), and especially pre-training language models, has led to dramatic effectiveness improvements in most any NLP task. Current state-of-the-art approaches utilize pre-trained language models, which are fine-tuned to a given set of target variables (i.e., by training all parameters of the language model). Training neural networks requires calculating a gradient for every layer and batch element, thus easily tripling the required memory. Meanwhile, this practice often exceeds the capabilities of end-user graphics cards. Nevertheless, such graphics cards can still get by without fine-tuning, through fitting a fixed parameter language model. This leads us to the following questions: (1) Can older or cheaper graphics cards be used to train neural language models for text classification without fine-tuning? (2) How does their effectiveness compare to the state-of-the-art fine-tuning methods? (3) What are the trade-offs between effectiveness and efficiency? A less resource-intensive means to using language models is especially beneficial in cases where older graphics cards are available (e.g., due to the presently prohibitive upgrade costs), and where outsourcing to cloud services is not an option (e.g., for privacy, security, or budget reasons). Among others, our experiments show that the drop of performance for single-label topic classification is marginal at around 1%; other tasks suffer more severe drops at around 10%. At the lower end, training a fixed parameter language model is always superior to word embeddings.

2 Related Work

Currently, text classification is mostly tackled using neural networks. Virtually every state-of-the-art result for text classification has been achieved using neural language models (Yang et al., 2019; Aly et al., 2019; Pal et al., 2020). Typically, a pre-trained language model is used, attaching an extra layer for projection to task-specific target variables, and fine-tuning both (Devlin et al., 2019). This either requires graphics cards with large memory, or low batch sizes. A substantial amount of research focuses on methods to downsize these pre-trained language models (Sanh et al., 2019; Mikolov et al., 2013; Clark et al., 2020). But, while these approaches reduce the number of parameters drastically, the memory during fine-tuning can still exceed 10 GiB due to gradient calculations.

Prior to language models, the most popular neural approach was to leverage pre-trained word embeddings, like GloVe (Pennington et al., 2014) and Word2Vec (Mikolov et al., 2013), feed them to task-specific neural architectures (Xiao et al., 2019; Kim, 2014), and then train on the task data at hand. Although pre-trained word embeddings also range at around 100M parameters, they are computationally efficient, since no gradient calculations are required. However, these approaches suffer from problems, such as out-of-vocabulary words and context insensitivity. We evaluate the performance of neural networks using a fixed-parameter language model as a middle ground between fine-tuning and word embeddings, investigating the trade-off between efficiency and effectiveness.

3 Experimental Setup

We employ the Huggingface library (Wolf et al., 2020) as a reference implementation for various state-of-the-art language models, and a PyTorch (Paszke et al., 2019) implementation of a CNN feature extraction module, which has proven useful for word-embedding-based models in different combinations (Kim, 2014; Rios and Kavuluru, 2018). The following language models are compared to word embeddings: The base versions of BERT (*bert-base-cased*) and RoBERTa (*roberta-base*) with standard hyperparameters. Moreover, a heavily downsized BERT-Tiny, a 2-layer version of BERT-base (*bert_uncased_L-2_H-768_A-12*), and a version with reduced hidden size and attention heads (*bert_uncased_L-12_H-128_A-2*).

The CNN consists of c convolutional layers with k_i kernel sizes, $1 \leq i \leq c$, and f filters per layer. Its resulting feature vector is projected to the number of target classes of the corresponding dataset. Combinations of the CNN layer with either a language model or word embeddings are trained with and without fine-tuning. We use $c = 4$ convolutional layers with kernel sizes of 3,4,5, and 6, and filter size $f = 100$. The CNN model is trained using Adam (Kingma and Ba, 2015) with a learning rate of $5e-5$. The number of input tokens is set to 200. For multi-label tasks, a sigmoid activation of outputs and binary cross-entropy loss is used, and for single-label tasks, a softmax activation with categorical cross-entropy. For feature extraction, a batch size of 50 is used across all datasets while it has to be adjusted to 40 for fine-tuning to circumvent memory errors. We run each setting 3 times and report mean and standard deviation. Training epochs for each dataset are listed in Appendix A1.

The aforementioned models are evaluated on 8 datasets for a broad view of their effectiveness and efficiency compared to word embeddings. In both test cases, we feed the output of the language model into the CNN module and train both in combination. All datasets used are English. Each multi-label dataset has an unbalanced label distribution. The following datasets are included:

AG News: News articles from the 4 largest topics of the corpus for a total of 30,000 training and 1,900 test samples per topic (Zhang et al., 2015).

20NEWS: Messages from Usenet newsgroups with 20 topic classes. for a total of 11,314 training and 7,532 test samples (Lang, 1995).

DBpedia: An ontology dataset with 14 classes for a total 40,000 training and 5,000 test samples per class, randomly chosen from DBpedia 2014 (Zhang et al., 2015).

TREC: Question classification dataset consisting of open-domain, fact-based questions. We use the versions with 6 and 50 classes, each containing 5,452 training and 500 test samples.

Yelp: A sentiment classification dataset containing 650,000 Yelp reviews used as training samples, as well as 50,000 test samples. Each review may have a rating between 1 and 5 stars for classes.

RCV1-v2: Topic classification dataset created by categorization of newswire stories. This version consists of 103 classes for a total of 23,149 training and 781,265 test samples with a label density of 3.12 (Lewis et al., 2004).

BlurbGenreCollection_EN: Genre classification dataset using book blurbs made up of a short abstracts describing a given book with 152 classes for a total of 58,715 training and 18,394 test samples with a label density of 3.01 (Aly et al., 2019).

Ohsumed: Medical dataset split into 23 different cardiovascular diseases for classes for a total of 7,643 training and 6,286 test samples with a label density of 1.64 (Hersh et al., 1994).

4 Results

Effectiveness: We report accuracy on the single-label datasets in Table 1 and micro Precision/Recall/F1 on multi-label datasets in Table 2. The results of feature extraction and fine-tuning are also compared to the current state-of-the-art results of the chosen datasets. On both single-label and multi-label datasets, fine-tuning on average performs better than feature extraction. In most cases, the smallest language model, BERT-Tiny, trained with feature extraction is on par with the word embeddings. However, when increasing model size or using fine-tuning, the word embeddings fall behind regarding recall. DBpedia is the only dataset on which BERT achieves better results with fine-tuning and feature extraction. We argue that this can be attributed to the much higher similarity between pre-training and downstream task compared to RoBERTa which was also observed by (Peters et al., 2019). While feature extraction can achieve good results on single-label data compared to fine-tuning it unfortunately falls short when training on multi-label data.

Memory: We train on an NVIDIA 1080Ti with

11GiB of VRAM. We compare the differences in memory usage by model between single-label and multi-label datasets using the feature extraction and fine-tuning approaches. As shown in App. A2, when training with feature extraction the memory usage of the larger models is generally the same, hovering between 1.6-1.7 GiB. The same can be said about fine-tuning where the usage stays between 10.3-10.6 GiB when applying BERT and RoBERTa. One must consider that even with a batch size reduction of 20% these models take up almost all our available VRAM. Therefore, the actual memory savings with equalized batch sizes are larger. While the memory usage of GloVe is the same as BERT-base concerning feature extraction, a large reduction is possible with the smallest BERT models for both feature extraction and fine-tuning. In the case of BERT-Tiny 1GiB of VRAM or less is needed in both cases.

Time: There are two aspects to training time: Time per epoch and overall training time. To evaluate epoch time, we calculate the mean of each dataset's runs and divide by BERT-FE time to get relative values. The results are presented in App. A3. Generally speaking fine-tuning takes around **twice to thrice** as long per epoch than feature extraction. On single-label datasets RoBERTa-FE overall takes longer to train than BERT, with the increase in training time conforming to the increase in model size. GloVe and the small BERT models only require a fraction of time compared to their larger counterparts. Time savings of around 95% using feature extraction and 90% using fine-tuning are possible with BERT-Tiny. While per epoch time advantage is substantial for feature extraction, to be fair, we have to look at the net training time in App. A4. To achieve the best results with feature extraction, it takes about 2-3 times more epochs to compete. This sometimes leads to feature extraction taking more overall time than fine-tuning, thus nullifying the gain per epoch.

Trade-offs: There are a number of observations to be made from these experiments, which are:

Feature extraction can be a viable option for single-label classification, losing only 1.06% of relative performance on average compared to its fine-tuning counterpart for topic classification, while using only 1.7 GiB VRAM. Sentiment and question classification seem to draw more benefit from fine-tuning with a relative performance gain of 7.08% and 13%, respectively.

Feature extraction loses significantly on multi-label classification, 8% decrease in performance on average, while also requiring more overall training time. The only argument for using feature extraction in multi-label tasks would be strong memory constrictions.

Feature extraction is a viable option if you have memory restrictions (even 2GiB or lower), since the larger feature extraction models hover around 1.65 GiB of VRAM with the smallest at 700MiB. For example, when having a restriction of 2GiB it is still better to use feature extraction on a large model than to fine-tune a small model.

Per epoch time efficiency: In all feature extraction cases you get a better per epoch time efficiency. As discussed above, this amortizes when comparing the overall run times or hyperparameters. If time per epoch is a relevant metric (e.g., in split training sessions) this can be useful.

Very small language models are competitive, when compared to word embeddings. The small memory footprint enables training on older hardware while maintaining a comparable performance combined with faster individual epochs.

No matter the hardware restrictions there is always a language model better and more efficient to use. If the language model crosses a minimum size it consistently outperforms word embeddings while maintaining a smaller memory footprint.

5 Summary

In this paper, we evaluated using language models without fine-tuning for text classification. We surveyed a set of publicly available datasets to distill a list of trade-offs regarding performance, time, and memory, to decide whether keeping a fixed language model is a viable option in scenarios with limited hardware resources. We found that there are use cases in which using a larger model and not fine-tuning proves to be a good way to go, the main reason being memory restrictions. The largest model in our experiments (RoBERTa) without fine-tuning and a batch size of 50 still has a memory footprint comparable to fixed word embeddings, while improving on every dataset by 6.61% on average. We hope this analysis may help with regards to choosing the appropriate hardware and model combination.

Method	AG News	20NEWS	DBpedia	TREC-6	TREC-50	YELP
GloVe-FE	91.84 ±0.18	79.85 ±0.04	98.71 ±0.01	92.33 ±0.77	84.13 ±0.57	58.75 ±0.06
GloVe-FiT	92.14 ±0.19	80.06 ±0.15	98.79 ±0.02	92.33 ±0.09	77.80 ±0.71	59.63 ±0.15
BERT-Tiny-FE	90.74 ±0.04	78.29 ±0.28	98.74 ±0.02	88.47 ±0.19	71.13 ±0.41	60.74 ±0.09
BERT-Tiny-FiT	92.92 ±0.13	80.52 ±0.19	99.01 ±0.02	91.33 ±0.25	81.33 ±1.20	63.57 ±0.09
BERT-L-2-FE	93.03 ±0.11	84.12 ±0.26	99.18 ±0.02	94.33 ±0.38	78.40 ±0.16	63.57 ±0.06
BERT-L-2-FiT	93.76 ±0.15	84.90 ±0.08	99.23 ±0.02	95.00 ±0.33	89.73 ±0.25	65.13 ±0.05
BERT-L-12-FE	91.37 ±0.09	78.88 ±0.39	98.95 ±0.01	90.27 ±0.57	73.53 ±0.47	60.77 ±0.01
BERT-L-12-FiT	93.47 ±0.25	82.50 ±0.47	99.08 ±0.02	94.60 ±0.43	87.93 ±0.94	65.24 ±0.18
BERT-FE	92.97 ±0.06	84.25 ±0.43	99.19 ±0.02	94.87 ±0.34	78.87 ±0.62	63.25 ±0.03
BERT-FiT	94.01 ±0.06	84.69 ±0.29	99.26 ±0.03	97.13 ±0.25	92.00 ±0.65	66.44 ±0.17
RoBERTa-FE	92.49 ±0.06	85.81 ±0.16	99.04 ±0.01	78.53 ±0.50	55.13 ±0.62	64.16 ±0.09
RoBERTa-FiT	94.84 ±0.25	85.59 ±0.3	99.21 ±0.01	96.67 ±0.09	90.67 ±1.32	68.70 ±0.05
SOTA	**95.55** Yang et al.	**88.5** Wu et al.	**99.4** Yang et al.	**98.07** Cer et al.	**97.2** Tayyar Madabushi and Lee	**73.28** Abreu et al.

Table 1: Test Accuracy (%) averaged over 3 runs on the single-label datasets. We compare feature extraction (FE) with our baselines and state-of-the-art (SOTA) models.

Dataset	Method	Precision	Recall	F1
RCV1	GloVe-FE	90.33 ±0.09	67.69 ±0.13	77.38 ±0.06
	GloVe-FiT	90.70 ±0.13	67.89 ±0.10	77.65 ±0.02
	BERT-Tiny-FE	89.20 ±0.12	70.21 ±0.12	78.57 ±0.05
	BERT-Tiny-FiT	81.99 ±0.45	78.22 ±0.13	80.06 ±0.16
	BERT-L-2-FE	92.41 ±0.02	73.37 ±0.12	81.79 ±0.07
	BERT-L-2-FiT	84.99 ±0.34	83.31 ±0.31	84.14 ±0.07
	BERT-L-12-FE	91.58 ±0.09	68.93 ±0.14	78.66 ±0.08
	BERT-L-12-FiT	82.77 ±0.17	82.79 ±0.24	82.78 ±0.06
	BERT-FE	87.88 ±0.18	77.97 ±0.43	82.63 ±0.16
	BERT-FiT	86.12 ±0.20	86.39 ±0.19	86.26 ±0.08
	RoBERTa-FE	87.62 ±0.11	81.34 ±0.08	84.36 ±0.04
	RoBERTa-FiT	86.93 ±0.52	87.30 ±0.62	87.11 ±0.07
	MAGNET (Pal et al., 2020)	-	-	**88.5**
Ohsumed	GloVe-FE	69.25 ±0.43	56.99 ±0.13	62.53 ±0.19
	GloVe-FiT	68.71 ±0.22	59.14 ±0.09	63.57 ±0.14
	BERT-Tiny-FE	65.79 ±0.30	57.40 ±0.34	61.31 ±0.29
	BERT-Tiny-FiT	60.97 ±0.65	59.13 ±0.81	60.03 ±0.12
	BERT-L-2-FE	74.46 ±0.02	57.04 ±0.24	64.60 ±0.28
	BERT-L-2-FiT	67.87 ±0.81	65.23 ±0.60	66.52 ±0.21
	BERT-L-12-FE	67.10 ±0.04	56.02 ±0.30	61.05 ±0.17
	BERT-L-12-FiT	66.08 ±0.68	63.78 ±1.08	64.89 ±0.23
	BERT-FE	71.25 ±0.78	61.21 ±0.07	65.84 ±0.32
	BERT-FiT	71.74 ±0.80	**69.75** ±0.39	70.72 ±0.31
	RoBERTa-FE	69.67 ±0.25	64.08 ±0.26	66.76 ±0.15
	RoBERTa-FiT	**73.78** ±1.06	69.26 ±2.15	**71.74** ±0.62
	SVM (Zha and Li, 2018)	-	-	62.9
BGC_EN	GloVe-FE	81.90 ±0.34	50.98 ±0.65	62.84 ±0.40
	GloVe-FiT	82.46 ±0.19	53.49 ±0.21	64.89 ±0.12
	BERT-Tiny-FE	80.64 ±0.09	55.62 ±0.28	65.83 ±0.20
	BERT-Tiny-FiT	71.50 ±0.33	70.69 ±0.08	71.10 ±0.19
	BERT-L-2-FE	**85.31** ±0.08	61.61 ±0.02	71.55 ±0.01
	BERT-L-2-FiT	78.14 ±1.29	74.56 ±0.99	76.30 ±0.11
	BERT-L-12-FE	82.69 ±0.12	54.89 ±0.11	65.98 ±0.12
	BERT-L-12-FiT	72.75 ±0.21	74.16 ±0.20	73.44 ±0.17
	BERT-FE	76.87 ±0.44	70.18 ±0.27	73.37 ±0.05
	BERT-FiT	77.54 ±0.23	78.64 ±0.3	78.09 ±0.17
	RoBERTa-FE	73.72 ±0.31	71.35 ±0.34	72.52 ±0.03
	RoBERTa-FiT	78.41 ±0.37	**79.36** ±0.32	**78.88** ±0.07
	Caps. Network (Aly et al., 2019)	77.21 ±0.54	71.73 ±0.63	74.37 ±0.35

Table 2: Comparison between feature extraction (FE) and our baselines on multi-label datasets using Precision/Recall/F1 (%). We include the best published result on the respective dataset for context.

References

Jader Abreu, Luis Fred, David Macêdo, and Cleber Zanchettin. 2019. Hierarchical attentional hybrid neural networks for document classification. *CoRR*, abs/1901.06610.

Rami Aly, Steffen Remus, and Chris Biemann. 2019. Hierarchical Multi-label Classification of Text with Capsule Networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 323–330, Florence, Italy. Association for Computational Linguistics.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174, Brussels, Belgium. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pre-training text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

W. R. Hersh, C. Buckley, T. J. Leone, , and D. H. Hickam. 1994. Ohsumed: An interactive retrieval evaluation and new large test collection for research. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*. ACM Press.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339.

David Lewis, Yiming Yang, Tony Russell-Rose, and Fan Li. 2004. RCV1: A New Benchmark Collection for Text Categorization Research. *Journal of Machine Learning Research*, 5:361–397.

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

Ankit Pal, Muru Selvakumar, and Malaikannan Sankarasubbu. 2020. Magnet: Multi-label text classification using attention-based graph neural network. *Proceedings of the 12th International Conference on Agents and Artificial Intelligence*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 7–14, Florence, Italy. Association for Computational Linguistics.

Anthony Rios and Ramakanth Kavuluru. 2018. Few-Shot and Zero-Shot Multi-Label Learning for Structured Label Spaces. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3132–3142, Brussels, Belgium. Association for Computational Linguistics.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108.

Harish Tayyar Madabushi and Mark Lee. 2016. High accuracy rule-based question classification using question syntax and semantics. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1220–1230, Osaka, Japan. The COLING 2016 Organizing Committee.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6861–6871. PMLR.

Lin Xiao, Xin Huang, Boli Chen, and Liping Jing. 2019. Label-Specific Document Representation for Multi-Label Text Classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 466–475, Hong Kong, China. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 5754–5764.

Daochen Zha and Chenliang Li. 2018. Multi-label dataless text classification with topic modeling. *Knowledge and Information Systems*, 61(1):137–160.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 649–657.

A Appendix

Dataset	FE	FiT
AGNews	20	10
20NEWS	300	100
DBpedia	10	10
TREC-6	150	50
TREC-50	150	50
YELP	15	5
RCV1	50	20
BGC_EN	40	20
Ohsumed	300	80

Table A1: Number of epochs for every dataset.

Method	FE	FiT
GloVe	1643	3941
BERT-Tiny	693	1007
BERT-L-2	1371	3159
BERT-L-12	715	2419
BERT	1667	10319
RoBERTa	1729	10577

Table A2: Average memory usage during training in MiB.

	BERT		RoBERTa		GloVe		BERT-Tiny		BERT-L-2		BERT-L-12	
Dataset	**FE**	**FiT**	**FE**	**FiT**	**FE**	**FiT**	**FE**	**FiT**	**FE**	**FiT**	**FE**	**FiT**
AGNews	1.0	2.62	1.25	2.59	0.04	0.24	0.05	0.11	0.21	0.52	0.12	0.37
20NEWS	1.0	1.96	1.08	1.74	0.03	0.15	0.05	0.08	0.15	0.3	0.13	0.27
DBpedia	1.0	2.58	1.24	2.54	0.04	0.23	0.05	0.1	0.21	0.51	0.12	0.36
TREC-6	1.0	2.64	0.99	2.65	0.04	0.14	0.04	0.1	0.21	0.51	0.12	0.36
TREC-50	1.0	2.63	1.0	2.64	0.04	0.14	0.04	0.1	0.21	0.51	0.12	0.36
YELP	1.0	2.67	0.99	2.69	0.04	0.15	0.05	0.1	0.23	0.54	0.13	0.38
RCV1	1.0	1.55	0.79	1.32	0.05	0.18	0.06	0.08	0.19	0.3	0.12	0.2
Ohsumed	1.0	1.92	0.8	1.74	0.05	0.27	0.06	0.09	0.19	0.38	0.12	0.28
BGC_EN	1.0	2.11	1.08	1.97	0.05	0.27	0.06	0.1	0.2	0.42	0.12	0.29

Table A3: Average time in seconds per epoch measured as multiples of BERT-FE.

	BERT		RoBERTa		GloVe		BERT-Tiny		BERT-L-2		BERT-L-12	
Dataset	**FE**	**FiT**	**FE**	**FiT**	**FE**	**FiT**	**FE**	**FiT**	**FE**	**FiT**	**FE**	**FiT**
AGNews	4.32	5.62	5.43	5.58	0.17	0.51	0.78	0.90	3.65	4.51	2.08	3.17
20NEWS	10.08	6.86	11.33	6.11	0.34	0.50	1.93	0.85	5.46	3.32	3.58	2.42
DBpedia	10.51	26.74	12.91	26.43	0.41	2.39	2.04	3.35	8.76	16.02	5.07	11.20
TREC-6	1.50	1.32	1.49	1.32	0.19	0.29	0.12	0.08	0.62	0.41	0.35	0.29
TREC-50	1.50	1.32	1.50	1.32	0.19	0.29	0.12	0.08	0.62	0.41	0.35	0.29
YELP	17.65	15.74	17.51	15.82	3.61	6.10	4.30	4.02	7.99	6.39	7.07	6.72
RCV1	6.31	3.92	4.93	3.36	0.34	0.46	1.16	1.48	3.56	5.31	2.32	3.70
Ohsumed	5.58	3.00	4.58	2.71	0.25	0.40	0.94	0.34	3.06	1.44	1.88	1.02
BGC_EN	6.32	6.56	5.22	6.11	0.30	0.85	1.19	1.01	4.12	4.24	2.55	2.97

Table A4: Average total training time per dataset in hours.