Extreme Fine-tuning: A Novel and Fast Fine-tuning Approach for Text Classification

Boonnithi Jiaramaneepinit[1], **Thodsaporn Chay-intr**[1], **Kotaro Funakoshi**[2] and **Manabu Okumura**[2]
[1]School of Engineering, Tokyo Institute of Technology
[2]Institute of Innovative Research, Tokyo Institute of Technology
{jiara.boon, chayintr, funakoshi, oku}@lr.pi.titech.ac.jp

Abstract

Although fine-tuning a pre-trained model with a conventional approach has shown to be effective in various downstream tasks, previous work has used only backpropagation to fine-tune the model, which causes a massive amount of computational resources and time. We propose Extreme Fine-Tuning (EFT), a novel approach for fine-tuning a pre-trained model effectively and efficiently. EFT uses backpropagation for a brief fine-tuning and an iterative extreme learning machine for training a classifier. We applied EFT to four text classification datasets, MELD, IEMOCAP, IMDb, and AG News, and compared its performance with state-of-the-art (SOTA) approaches. The results indicate that EFT noticeably outperformed the other approaches in training-time measurement with comparable model performance. We will release our code at https://github.com/up-33/extreme-fine-tuning.

1 Introduction

Artificial neural networks (ANNs) have been successfully applied to many tasks such as natural language processing (NLP) (Vaswani et al., 2017; Otter et al., 2019; Li et al., 2022) and computer vision (Minaee et al., 2020). One of the essential components under their hood is backpropagation (BP), a gradient-descent-based learning algorithm. The BP became the conventional approach to train an ANN model due to its ability to learn sophisticated patterns from a large amount of data (Schmidhuber, 2015). However, training a model on large-scale data from scratch requires massive computational resources (Conneau and Lample, 2019; Zhuang et al., 2021).

Pre-trained models (PTMs), e.g., bidirectional encoder representations from transformers (BERT) (Devlin et al., 2019), have been proposed to acquire a huge amount of general knowledge from large-scale data. Adding a fully connected (FC) layer as the last layer of the pre-trained model enables fine-tuning to specific tasks. In fine-tuning, the layer can be modified for specific tasks. Its parameters are adjusted to minimize task-specific loss for accurate prediction (Howard and Ruder, 2018). However, it usually applies iterative weight updates through the BP, which consumes unnecessary computational resources, particularly for large and deep ANNs (Sun et al., 2017).

Apart from training the FC layer through the BP, an extreme learning machine (ELM), an ANN training framework, was proposed by Huang et al. (2004) to accelerate the training of a single hidden layer feedforward neural network (SLFN). While the BP offers flexibility, the ELM provides simpler and more computationally efficient solutions. It calculates weights based on another arbitrary weights, making it faster than the BP (Huang et al., 2012). Variants such as a constrained ELM (CELM) (Zhu et al., 2014) and an iterative ELM (I-ELM) (Jiaramaneepinit and Watchareeruetai, 2018) have been also developed to improve performance and reduce memory consumption.

To the best of our knowledge, most studies have used only the BP to transfer or fine-tune a PTM to a specific task (Devlin et al., 2019; Liu et al., 2019; Kim and Vossen, 2021; Song et al., 2022; Lee and Lee, 2022; Shen et al., 2021; Heinsen, 2022; Bingyu and Arefyev, 2022). This makes the training process consume an enormous amount of computational resources and time. Thus, we propose Extreme Fine-Tuning (EFT), a novel fine-tuning approach that keeps up model performance and improves training efficiency. EFT utilizes the BP and I-ELM to speed up fine-tuning, improving training performance comparing with other fine-tuning approaches. We conducted experiments for measuring model performance and training efficiency through text classification tasks. Our contributions are as follows: 1) We propose EFT that speeds up the conventional way to fine-tune PTMs by adopting I-ELM to replace a BP-based FC layer for text classification tasks. 2) Applying EFT to four datasets for text classification produced comparable results compared with the previous studies (Kim and Vossen, 2021; Song et al., 2022; Lee and Lee, 2022; Shen et al., 2021; Heinsen, 2022; Bingyu and Arefyev, 2022). Nonetheless, EFT requires noticeably less training time.

2 Related Work

2.1 Models Trained with Backpropagation

Most studies integrated PTMs and various ANNs, and relied on the BP (Song et al., 2022; Lee and Lee, 2022), emphasizing performance rather than training efficiency. This makes more parameters need to be adjusted, thus increasing model size (Yu et al., 2022a,b) and affecting training time.

Even though various speed-up techniques have been proposed to address the time-consuming aspect of the BP, prior studies have primarily focused on development- or precision-oriented optimization for training or inference through the BP (Guo et al., 2019; Yang et al., 2022; Zaiem et al., 2023).

2.2 Extreme Learning Machines

ELMs are frameworks for training an ANN. They were proposed to be an alternative way to train a SLFN (Huang et al., 2004). ELMs solved model parameters in one-shot calculation based on Moore-Penrose inverse, that makes the training faster than the BP. The ELM procedure is described in Appendix A. However, the ELM has several drawbacks, such as out-of-memory issues and prioritizing hard-to-predict instances. To address these problems, I-ELM was proposed to enable iteratively training instances (Jiaramaneepinit and Watchareeruetai, 2018). Model parameters are stored and calculated by dividing a dataset into batches, instead of the whole dataset at once. The procedure to train ANNs with I-ELM is fully explained in Appendix B.

3 Extreme Fine-tuning

EFT incorporates the BP followed by I-ELM to speed up the training of a network. We first build a BP-based feature extractor by fine-tuning a PTM, e.g., BERT or RoBERTa (Liu et al., 2019), and removing its BP-based classifier, enabling the extraction of prior knowledge and an overview of the input data. We then use the output from the feature extractor to build an I-ELM-based classifier for final output inference. We show our EFT procedure in Algorithm 1, given S instances of training data.

Datasets	Type	|Class|	Avg. Len.	Train	Test
MELD	Emotion	7	8	9,989	2,610
IEMOCAP	Emotion	6	22	4,778	1,622
IMDb	Sentiment	2	292	25,000	25,000
AG News	Topic	4	44	120,000	7,600

Table 1: Statistics of four datasets

Algorithm 1 EFT procedure to fine-tune a model

```
1: Initialize feature extractor $P_f$ using a PTM.
2: Add fully connected layer $FC_\alpha$ as the last layer to feature extractor $P_f$ for mapping output representations to labels.
3: Unfreeze all model parameters.
4: Fine-tune the model using BP for $k$ epochs to obtain fine-tuned BP-based classifier $FC_\alpha{}'$ and fine-tuned feature extractor $P_f{}'$.
5: Remove $FC_\alpha{}'$, retaining $P_f{}'$.
6: Construct I-ELM-based classifier $FC_z$ with arbitrary orthogonal matrix $\mathbf{W}$ and bias vector $\mathbf{b}$.
7: Calculate hidden layer $\mathbf{H}$ of I-ELM where $\mathbf{H} \in \mathbb{R}^{S \times nh}$ with $nh$ hidden nodes.
8: Calculate $\mathbf{\Lambda}$ and $\mathbf{\Gamma}$ of I-ELM using $\mathbf{H}$ where $\mathbf{\Lambda} \in \mathbb{R}^{r \times nh}$, $\mathbf{\Gamma} \in \mathbb{R}^{S \times nh}$, and $r$ is the number of classes or nodes in the output layer.
9: Calculate output weight matrix $\mathbf{U}$ of I-ELM using $\mathbf{\Lambda}$ and $\mathbf{\Gamma}$ where $\mathbf{U} \in \mathbb{R}^{r \times nh}$.
10: if performance or iteration criteria is acceptable then
11:     Go to Step 17.
12: else
13:     Identify misclassified instances $\mathbf{X}_w$ by feedforwarding $\mathbf{X}$ through the model.
14:     Update $\mathbf{\Lambda}$ and $\mathbf{\Gamma}$ with $\mathbf{X}_w$.
15:     Go back to Step 9.
16: end if
17: Integrate $FC_z$ to $P_f{}'$ for final output inference.
```

Our EFT-based model is represented in Figure 1. The purpose of Step 4 is to let the feature extractor get familiar with the task and the whole dataset. Note that, during the k epochs in this step, using a non-optimal learning rate helps the feature extractor, fine-tuned with the BP-based classifier, learn overall features faster without focusing on optimal network weights.

4 Experiments

4.1 Datasets

We used four datasets, MELD, IEMOCAP, IMDb, and AG News, to compare EFT and prior studies. Table 1 lists their statistics. Their details are described in Appendix C. While we followed the official data split for MELD and IEMOCAP, we split 10% of the training set for validation for IMDb and AG News, which have no validation set.

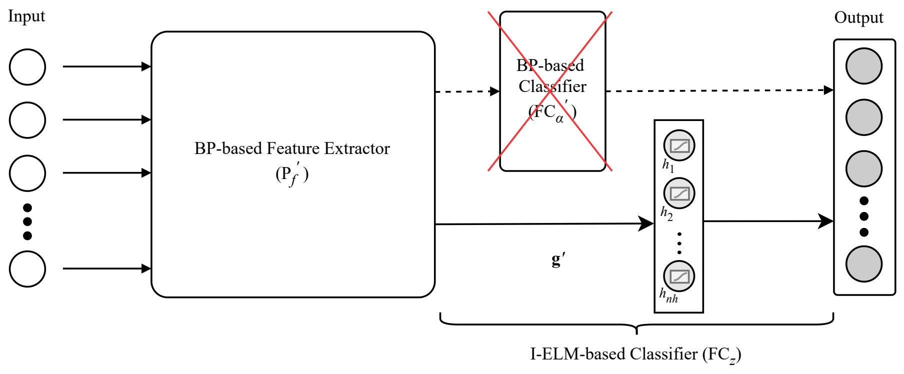

Figure 1: Overview of our EFT-based model after integrating the I-ELM-based classifier

4.2 Setups

Hyperparameters, including the number of backpropagation epochs k for all models and the number of hidden nodes nh for our model, were tuned based on validating scores. We first increased k and stopped when the validating score starts to drop. We then replaced a classifier with I-ELM. Next, we increased nh and stopped when the validating score starts to drop. The resultant values of k and nh for each model are described in Appendix D. During I-ELM training, we randomly generated an orthogonal matrix for an arbitrary weight. Thereafter, we trained a model using I-ELM with only one iteration. All models were trained on a single machine with a Ryzen 9 3900X CPU, a Geforce RTX 3090 GPU, and 64GB of DDR4 RAM. Appendix E further describes the environment setups.

4.3 Evaluation Metrics

We compared the model performance and efficiency with current SOTA models across the four datasets. The performance was measured using weighted-F1 (WF1), accuracy (ACC), or error, while the efficiency was assessed by training time (TTM). These values were the average of three runs. We showed both reproduced and reported (*) scores. The training time of baselines was measured based on the duration of feedforwarding and BP, while the training time of EFT was measured by calculating the duration between Steps 4 and 9, including BP (Steps 4 to 5), feedforwarding for I-ELM (Steps 6 to 7), and weight calculation of I-ELM (Steps 8 to 9). These are further described in Appendix F.

For a fair comparison, we used the PTM in the compared models for our model. We also tried the experiments with RoBERTa-large, a large PLM, when the compared models did not utilize it. We performed a significance test using paired bootstrap resampling (Koehn, 2004). Since MELD and IEMOCAP consist of sequences of utterances, we additionally introduced variations that incorporated 128 tokens of past and future utterances into the input data for MELD and 128 tokens of past utterances into the input data for IEMOCAP.

4.4 Results

In Table 2, our EFT-RoBERTa-large and EFT⋆-RoBERTa-large stood out to be the most efficient, taking only 5 and 25 minutes, respectively, on MELD, while the performances in WF1 are comparable to the best baseline model. This suggests that EFT offers a promising trade-off between the performance and the efficiency on MELD.

Models	PTM	k	WF1*	WF1	TTM
(Kim and Vossen, 2021)⋆		5	65.61	64.76	190
(Song et al., 2022)		5	66.50	<u>65.63</u>	46
(Lee and Lee, 2022)	RoBERTa large	10	66.52	63.60	285
EFT [Ours]		3	-	‡64.76	**5**
EFT⋆ [Ours]		3	-	‡**65.82**	**25**

Table 2: Comparison of weighted-F1 (WF1) and training time in minutes (TTM) on MELD. Scores with an asterisk (*) are reported scores. ⋆ indicates the model was trained with surrounding utterances (Kim and Vossen, 2021). ‡ indicates the model's WF1 scores are **comparable** to the best baseline model with the same PTM (underlined) ($p<0.05$).

Table 3 presents a comparison of models in terms of WF1 and training time on IEMOCAP. Our EFT⋆-RoBERTa-large achieved the highest WF1 of 69.44 with the training time of only 46 minutes. We observed the improvement in WF1 and TTM over the baseline models.

The IMDb results in Table 4 show that EFT-RoBERTa-base and EFT-RoBERTa-large achieved accuracies of 95.26 and 96.15, respectively, which

Model	PTM	k	WF1*	WF1	TTM
(Kim and Vossen, 2021)⋆	RoBERTa large	5	68.57	67.21	360
(Lee and Lee, 2022)		10	66.61	65.79	220
EFT [Ours]		6	-	53.43	**6**
EFT⋆ [Ours]		6	-	‡**69.44**	**46**

Table 3: Comparison of weighted-F1 (WF1) and training time in minutes (TTM) on IEMOCAP. The notations are the same as in Table 2.

are comparable to the baselines. They significantly outperformed the baselines in training-time efficiency, taking only 15 and 69 minutes, respectively.

Model	PTM	k	ACC*	ACC	TTM
(Bingyu and Arefyev, 2022)	RoBERTa base	10	95.79	95.74	78
EFT [Ours]		1	-	‡95.26	**15**
(Heinsen, 2022)	RoBERTa large	10	96.20	**96.36**	295
EFT [Ours]		2	-	‡96.15	**69**

Table 4: Comparison of model accuracy (ACC) and training time in minutes (TTM) on IMDb. The notations are the same as in Table 2.

Table 5 shows the results in terms of error and training time on AG News. EFT-RoBERTa-large achieved the low error of 4.79 and the training time of 111 minutes. To compare EFT-BERT-base with the BERT-base-based baseline model, we achieved the error of 5.77 with the training time of only 30 minutes.

Model	PTM	k	Error*	Error	TTM
(Sun et al., 2020)	BERT base	3	4.80	**4.68**	549※+196
EFT [Ours]		1	-	‡5.77	**30**
EFT [Ours]	RoBERTa large	3	-	‡4.79	**111**

Table 5: Comparison of model error and training time in minutes (TTM) on AG News. The notations are the same as in Table 2. The symbol ※ denotes the training time for pre-training on an RTX Titan due to the specific model setup. Since there is no RoBERTa-large-based baseline model, the BERT-base-based baseline model was used to compare to EFT-RoBERTa-large for ‡.

Table 6 presents training time in EFT focusing on a single training epoch for traditional BP and I-ELM. The training time was measured in minutes.

TTM of 1 BP epoch: This column refers to the training time in minutes needed to complete a single BP epoch using the given model. All parameters were unfrozen to be able to be updated during the training. For instance, on the MELD dataset, the EFT-RoBERTa-large model requires 1 minute for one BP epoch of updating the whole model.

TTM of 1 I-ELM epoch: This column provides the training time in minutes that I-ELM needs to finish one epoch. Note that parameters from the BP step were frozen, and only parameters in an I-ELM classifier was updated. For instance, on the MELD dataset, 0.5 minutes (30 seconds) are needed for one I-ELM epoch to finish training of a classifier for EFT-RoBERTa-large.

Dataset	Model	TTM of 1 BP epoch	TTM of 1 I-ELM epoch
MELD	EFT-RoBERTa-large	1	0.5
	EFT⋆-RoBERTa-large	7	3
IEMOCAP	EFT-RoBERTa-large	0.5	0.25
	EFT⋆-RoBERTa-large	7	2.5
IMDb	EFT-RoBERTa-base	8	4
	EFT-RoBERTa-large	27	13
AG News	EFT-BERT-base	15	11
	EFT-RoBERTa-large	32	14

Table 6: Comparative overview of training time for models trained with EFT, focusing on a single BP or I-ELM epoch. ⋆ indicates the model was trained with surrounding utterances.

Table 7 shows the estimated numbers of floating-point operations (FLOPs) of the breakdown of the EFT procedure, focusing on the amount of computation. The FLOPs numbers of feedforward and BP were estimated with a FLOP profiler from DeepSpeed (Rasley et al., 2020).

BP-based feature extractor: This column provides the number of FLOPs used for fine-tuning the feature extractor during k epochs of BP.

BP-based classifier: This column provides the number of FLOPs used for fine-tuning the classifier during k epochs of BP.

Feedforward for I-ELM: This column provides the number of FLOPs used for computing the inputs of I-ELM by feedforwarding inputs through the BP-based feature extractor.

I-ELM-based classifier: This column provides the number of FLOPs used for computing I-ELM. This includes the calculation of matrices $\mathbf{H}$ and $\hat{\mathbf{U}}$, which contain matrix inverse operation. See Appendix G for the FLOP estimation of the matrix inverse operation.

In summary, Table 6 offers training efficiency and computation amount of EFT on different datasets. We could find that the training time for one I-ELM epoch is consistently approximately a half of the time required for one BP epoch across different datasets and models. In addition, Table 7 offers computation amount of EFT. By analyzing the table, we could find that the FLOPs used for the I-ELM-based classifier is significantly lower than the FLOPs used for a BP-based classifier, except

Dataset	Model	Estimated FLOPs			
		BP-based features extractor	BP-based classifier	Feedforward for I-ELM	I-ELM-based classifier
MELD	EFT-RoBERTa-large	1,166.57T	126.55G	391.27T	3.28G
	EFT⋆-RoBERTa-large	13,179.98T	126.55G	4398.65T	3.28G
IEMOCAP	EFT-RoBERTa-large	979.82T	120.95G	162.45T	3.64G
	EFT⋆-RoBERTa-large	16,065.33T	120.95G	2,678.19T	3.64G
IMDb	EFT-RoBERTa-base	4,349.81T	53.22G	4,347.75T	18.19G
	EFT-RoBERTa-large	28,031.60T	189.11G	14,006.80T	8.08G
AG News	EFT-BERT-base	4,377.90T	1.33G	4,371.37T	241.00G
	EFT-RoBERTa-large	29,467.68T	1364.26G	9,822.83T	38.47G

Table 7: Comparative overview of estimated numbers of floating-point operations (FLOPs) of the breakdown of the EFT procedure.

for EFT-BERT-base on AG News. The differences of training time and numbers of FLOPs play a crucial role in understanding why the proposed EFT achieves superior training speed.

4.5 Discussion

We evaluated our EFT upon text classification tasks, emphasizing the effects of different dataset characteristics and model architectures. We considered several criteria for dataset characteristics, such as the dataset size, the task type variation, and the number of classes. We found that EFT works well with any selected datasets. To demonstrate the effectiveness across model architectures, we fine-tuned both BERT and RoBERTa with different sizes. We found that our EFT can improve the training time for any PTMs, while it maintains the performance. On average, EFT reduces fine-tuning time by 74.82% when compared to the best-performing baseline models. From our analysis, we also found that one I-ELM epoch takes approximately a half of one BP epoch, emphasizing the efficiency of EFT. This enables EFT to train the models in a fast manner. The numbers of FLOPs are also significantly lower in most of the cases, except for EFT-BERT-base on AG News. This is possibly due to the hyperparameter tuning, which leads to the classification of two classes with an excessively low BP epoch number (1 BPE) and an excessively high I-ELM hidden node number (1,000 I-ELM hidden nodes).

The superiority of I-ELM over BP lies in its adoption of a one-shot calculation of ELM, which directly obtains weights without the need of computing losses or errors. On the other hand, BP, relying on gradient descent, has significant computational overhead by feeding errors backward through the model layers. EFT capitalizes on this efficiency contrast. It outperforms baseline fine-tuning strategies by leveraging its efficiency through a unique combination of just 1 to 6 epochs of BP and 1 epoch of I-ELM, ensuring fast model training. In contrast, traditional baselines employ 5 to 10 epochs of BP for PTMs, introducing a time-intensive process.

5 Conclusion

We proposed EFT, a novel approach for fine-tuning a pre-trained model effectively and efficiently. We showed our EFT demonstrates shorter training time with competitive performance than the current SOTA models. These results highlight the potential of EFT as promising options for various NLP tasks, offering a favorable balance between model performance and efficiency. The potential of EFT is not limited to text classification but also extends to other classification and even non-classification tasks, such as generation tasks. It would be interesting to delve into the capability of EFT in such other fields.

6 Limitations

While EFT offers advantages in training efficiency, there are several limitations that should be taken into consideration.

Applicability to Specific Tasks: The effectiveness of EFT may vary depending on tasks and dataset characteristics. In our study, we evaluated EFT on four different text classification datasets. Further research might be required to investigate the effectiveness of EFT across a wider range of tasks.

Optimal Configuration of EFT: The performance and efficiency of EFT may be sensitive to the configuration, including the BP epoch number, BP learning rate, activation function, and number of hidden nodes in I-ELM. Determining the optimal configuration requires careful experimentation and tuning. Our study provides a baseline configuration for EFT, but further investigation might be needed to explore its sensitivity.

Model Size of EFT: The size of models trained with EFT increases depending on the number of hidden nodes of I-ELM. Our observations revealed that excessive increase of the hidden nodes results in overfitting issues.

Acknowledgements

Our gratitude extended to MEXT and Shippusu (Thailand) Co., Ltd. for supporting us. Special thanks should go to Sakulthip Rassameecharern-tham for the encouragement and several forms of support.

References

Zhang Bingyu and Nikolay Arefyev. 2022. The document vectors using cosine similarity revisited. In *Proceedings of the Third Workshop on Insights from Negative Results in NLP*, pages 129–133, Dublin, Ireland. Association for Computational Linguistics.

Carlos Busso, Murtaza Bulut, Chi-Chun Lee, Ebrahim (Abe) Kazemzadeh, Emily Mower Provost, Samuel Kim, Jeannette N. Chang, Sungbok Lee, and Shrikanth S. Narayanan. 2008. Iemocap: interactive emotional dyadic motion capture database. *Language Resources and Evaluation*, 42:335–359.

Alexis Conneau and Guillaume Lample. 2019. *Cross-Lingual Language Model Pretraining*. Curran Associates Inc., Red Hook, NY, USA.

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.

William Falcon and The PyTorch Lightning team. 2019. PyTorch Lightning.

Yunhui Guo, Honghui Shi, Abhishek Kumar, Kristen Grauman, Tajana Rosing, and Rogerio Feris. 2019. Spottune: Transfer learning through adaptive fine-tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Franz A. Heinsen. 2022. An algorithm for routing vectors in sequences.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Chao-Chun Hsu, Sheng-Yeh Chen, Chuan-Chun Kuo, Ting-Hao Huang, and Lun-Wei Ku. 2018. Emotion-Lines: An emotion corpus of multi-party conversations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Guang-Bin Huang, Hongming Zhou, Xiaojian Ding, and Rui Zhang. 2012. Extreme learning machine for regression and multiclass classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(2):513–529.

Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. 2004. Extreme learning machine: a new learning scheme of feedforward neural networks. In *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541)*, volume 2, pages 985–990 vol.2.

Boonnithi Jiaramaneepinit and Ukrit Watchareeruetai. 2018. Iterative extreme learning machine. In *2018 22nd International Computer Science and Engineering Conference (ICSEC)*, pages 1–6.

Taewoon Kim and Piek Vossen. 2021. Emoberta: Speaker-aware emotion recognition in conversation with roberta. *CoRR*, abs/2108.12009.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395, Barcelona, Spain. Association for Computational Linguistics.

Joosung Lee and Wooin Lee. 2022. CoMPM: Context modeling with speaker's pre-trained memory tracking for emotion recognition in conversation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5669–5679, Seattle, United States. Association for Computational Linguistics.

Qian Li, Hao Peng, Jianxin Li, Congying Xia, Renyu Yang, Lichao Sun, Philip S. Yu, and Lifang He. 2022. A survey on text classification: From traditional to deep learning. *ACM Transactions on Intelligent Systems and Technology*, 13(2).

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. 2020. Image segmentation using deep learning: A survey.

Daniel W. Otter, Julian R. Medina, and Jugal K. Kalita. 2019. A survey of the usages of deep learning in natural language processing.

Soujanya Poria, Devamanyu Hazarika, Navonil Majumder, Gautam Naik, Erik Cambria, and Rada Mihalcea. 2019. MELD: A multimodal multi-party dataset for emotion recognition in conversations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 527–536, Florence, Italy. Association for Computational Linguistics.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, page 3505–3506, New York, NY, USA. Association for Computing Machinery.

Jürgen Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117.

Weizhou Shen, Siyue Wu, Yunyi Yang, and Xiaojun Quan. 2021. Directed acyclic graph network for conversational emotion recognition. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1551–1560, Online. Association for Computational Linguistics.

Xiaohui Song, Liangjun Zang, Rong Zhang, Songlin Hu, and Longtao Huang. 2022. Emotionflow: Capture the dialogue level emotion transitions. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8542–8546.

Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. 2017. Revisiting unreasonable effectiveness of data in deep learning era. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 843–852.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2020. How to fine-tune bert for text classification?

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Yoonseok Yang, Kyu Seok Kim, Minsam Kim, and Juneyoung Park. 2022. GRAM: Fast Fine-tuning of Pre-trained Language Models for Content-based Collaborative Filtering. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–851, Seattle, United States. Association for Computational Linguistics.

Youngwoo Yoo and Se-Young Oh. 2016. Fast training of convolutional neural network classifiers through extreme learning machines. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 1702–1708.

Tan Yu, Hongliang Fei, and Ping Li. 2022a. Cross-probe bert for fast cross-modal search. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 2178–2183, New York, NY, USA. Association for Computing Machinery.

Wenhao Yu, Chenguang Zhu, Zaitang Li, Zhiting Hu, Qingyun Wang, Heng Ji, and Meng Jiang. 2022b. A survey of knowledge-enhanced text generation. *ACM Comput. Surv.*, 54(11s).

Salah Zaiem, Robin Algayres, Titouan Parcollet, Slim Essid, and Mirco Ravanelli. 2023. Fine-tuning strategies for faster inference using speech self-supervised models: a comparative study. In *ICASSP 2023 - International Conference on Acoustics, Speech, and Signal Processing*, Rhodes, Greece.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Wentao Zhu, Jun Miao, and Laiyun Qing. 2014. Constrained extreme learning machine: A novel highly discriminative random feedforward neural network. In *2014 International Joint Conference on Neural Networks (IJCNN)*, pages 800–807.

Liu Zhuang, Lin Wayne, Shi Ya, and Zhao Jun. 2021. A robustly optimized BERT pre-training approach with post-training. In *Proceedings of the 20th Chinese National Conference on Computational Linguistics*, pages 1218–1227, Huhhot, China. Chinese Information Processing Society of China.

A Procedure for Extreme Learning Machines

ELMs train a network in the following three steps:

1. Generate an arbitrary input weight matrix $\mathbf{W}_{N \times D}$, which connects the input layer of D nodes to the hidden layer of N nodes, and an arbitrary bias vector $\mathbf{b}_{N \times 1}$.

2. Calculate a matrix $\mathbf{H}$ of the response hidden layer as

$$\mathbf{H} = \sigma\left(\mathbf{WX} + \mathbf{b}\right), \tag{1}$$

where $\mathbf{X}_{D\times S} = [\mathbf{x}_1\ \mathbf{x}_2\ \cdots\ \mathbf{x}_S]$ is a matrix of S training instances and $\sigma(\cdot)$ is an activation function.

3. Calculate an output weight matrix $\mathbf{U}_{C\times N}$ as shown in Eq. (2):

$$\hat{\mathbf{U}} = \mathbf{T}\mathbf{H}^{\dagger}, \tag{2}$$

where $\mathbf{H}^{\dagger}$ is the Moore-Penrose inverse of the corresponding matrix $\mathbf{H}$, with which the output weight matrix $\mathbf{U}$ connects the hidden layer to the output layer of C nodes. They project the matrix $\mathbf{H}$ to a target matrix $\mathbf{T}_{C\times S} = [\mathbf{t}_1\ \mathbf{t}_2\ \cdots\ \mathbf{t}_S]$, corresponding to the training matrix $\mathbf{X}$, as shown in Eq. (3):

$$\mathbf{T} = \mathbf{U}\mathbf{H}. \tag{3}$$

Moreover, another ELM variation that incorporates L2 regularization was proposed (Huang et al., 2012). It calculates a solution $\hat{\mathbf{U}}$ using the following equation Eq. (4):

$$\hat{\mathbf{U}} = \mathbf{T}\mathbf{H}^{\top}(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^{\top})^{-1}, \tag{4}$$

where $\mathbf{I}$ is an identity matrix and C is a parameter used for restricting the effect of L2 regularization. The equation can be re-written as

$$\hat{\mathbf{U}} = \mathbf{\Lambda}(\frac{\mathbf{I}}{C} + \mathbf{\Gamma})^{-1}, \tag{5}$$

$$\begin{aligned}\mathbf{\Lambda} &= \mathbf{T}\mathbf{H}^{\top},\\ &= [\mathbf{t}_1\ \mathbf{t}_2\ \cdots\ \mathbf{t}_S][\mathbf{h}_1\ \mathbf{h}_2\ \cdots\ \mathbf{h}_S]^{\top},\\ &= \sum_{s=1}^{S}\mathbf{t}_s\mathbf{h}_s^{\top},\end{aligned} \tag{6}$$

$$\begin{aligned}\mathbf{\Gamma} &= \mathbf{H}\mathbf{H}^{\top},\\ &= [\mathbf{h}_1\ \mathbf{h}_2\ \cdots\ \mathbf{h}_S][\mathbf{h}_1\ \mathbf{h}_2\ \cdots\ \mathbf{h}_S]^{\top},\\ &= \sum_{s=1}^{S}\mathbf{h}_s\mathbf{h}_s^{\top},\end{aligned} \tag{7}$$

where Eqs. (6) and (7) can be calculated as the sum of products (Yoo and Oh, 2016).

B Procedure for an Iterative Extreme Learning Machine

The I-ELM procedure is as follows:

1. Given training data with feature inputs $\mathbf{X}$ and target outputs $\mathbf{T}$ of S instances, generate an arbitrary input weight matrix $\mathbf{W}$ and a bias vector $\mathbf{b}$.

2. Calculate the hidden layer $\mathbf{H}$ using Eq. (1).

3. Calculate the matrices $\mathbf{\Lambda}$ and $\mathbf{\Gamma}$ using Eqs. (6) and (7), then store them into the memory.

4. Calculate the output weight matrix $\mathbf{U}$ using Eq. (5).

5. Feedforward the matrix $\mathbf{X}$ through the model.

6. In accordance with the target output $\mathbf{T}$, identify misclassified instances $\mathbf{X}_w$. If the performance of the current model is acceptable, end the algorithm. Otherwise; go to the next step.

7. Update the matrices $\mathbf{\Lambda}$ and $\mathbf{\Gamma}$ as follows:

$$\mathbf{\Lambda} \leftarrow \mathbf{\Lambda} + \mathbf{T}_w\mathbf{H}_w^{\top}, \tag{8}$$

$$\mathbf{\Gamma} \leftarrow \mathbf{\Gamma} + \mathbf{H}_w\mathbf{H}_w^{\top}, \tag{9}$$

   where $\mathbf{H}_w = f(\mathbf{W}\mathbf{X}_w + \mathbf{b})$ is the output of the hidden layer given the feature input matrix $\mathbf{X}_w$, and $\mathbf{T}_w$ is the target output matrix corresponding to $\mathbf{X}_w$.

8. Go back to step 4.

C Datasets in Detail

Multimodal Emotion Lines Dataset (MELD): A dataset of conversations between two people annotated with seven emotions, i.e., anger, disgust, sadness, joy, neutral, surprise, and fear. It consists of 12,599 instances. Each instance includes a speaker name, an utterance, and an emotion. We followed the official data split from Poria et al. (2019); Hsu et al. (2018).

Interactive Emotional Dyadic Motion Capture (IEMOCAP): A dataset of dyadic conversations recorded and annotated for emotional information (Busso et al., 2008). It has been widely used for developing models for downstream tasks. It contains transcripts, audio, and video data from scripted- and improvised conversations among 10 actors (5 men and 5 women).

Internet Movie Database (IMDb): A dataset of movie reviews used for sentiment analysis and other NLP tasks (Maas et al., 2011). It contains 25,000 reviews each for training and testing data, 50,000 reviews in total. Each review is annotated with either a positive or negative label. The reviews are written in English, and were collected from IMDb.[1] This dataset is widely used for training and testing models for sentiment analysis, text classification, and natural language understanding.

AG News: A dataset of news articles, which is well-balanced and contains contents from a variety of sources, created from the AG's corpus for topic classification tasks (Zhang et al., 2015). It contains approximately 120,000 news articles, consisting of four classes, i.e., world, sports, business, and science/technology, with 30,000 articles each.

D Hyperparameter Tuning

Hyperparameters were selected based on validating scores. Figures 2, 3, 4, 5, 6, 7, 8, 9, 10, and 11 show validating scores for k. Figures 12, 13, 14, 15, 16, 17, 18, 19, 20, and 21 show validating scores for nh. Furthermore, the learning rate was set to 1e-5 for MELD and IMDb, and 5e-6 for IEMOCAP and AG News.

E Environment Setup

The experiments were conducted with PyTorch Lightning (Falcon and The PyTorch Lightning team, 2019). The floating point precision was set to float32. Data loaders from Kim and Vossen (2021) were modified to be compatible with EFT.

F Training Time of EFT

The training time of EFT includes BP, feedforwarding for I-ELM, and weight calculation of I-ELM.

BP (Steps 4 to 5): We apply BP to make the feature extractor briefly adapts to a whole dataset. This takes only few epochs, comparing to the baselines.

Feedforward for I-ELM (Steps 6 to 7): We feedforward the inputs through the feature extractor for the hidden layer **H**.

Weight calculation of I-ELM (Steps 8 to 9): We calculate the output weight based on values from the hidden layer **H**.

G FLOP estimation of Matrix Inverse Operation

The FLOP estimation of matrix inverse operation was computed by the summation of FLOPs of LU decomposition and equation solving of $Ly = b$ and $Ux = y$. Given nh hidden nodes of I-ELM, the number of FLOPs for LU decomposition is approximately $2/3 \times nh^3$ (assuming a dense matrix), and the number of FLOPs for forward substitution (solving $Ly = b$) and back substitution (solving $Ux = y$) are each approximately nh^2.

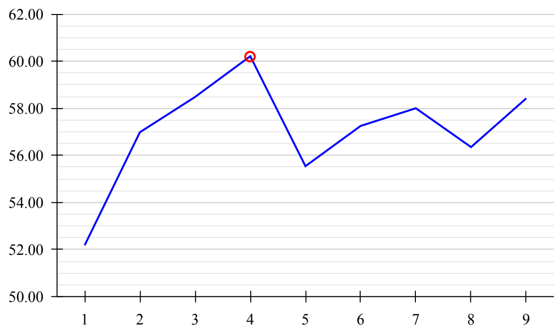

Figure 2: Validating WF1 of fine-tuned RoBERTa-base over k BPE on MELD

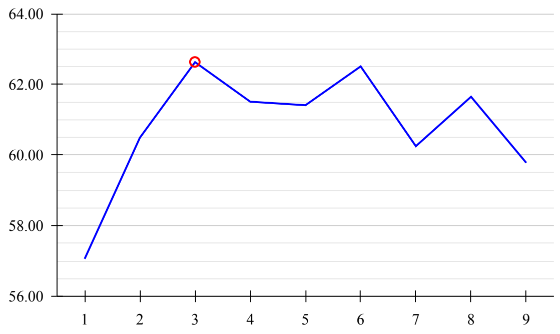

Figure 3: Validating WF1 of fine-tuned RoBERTa-large over k BPE on MELD

[1] https://www.imdb.com

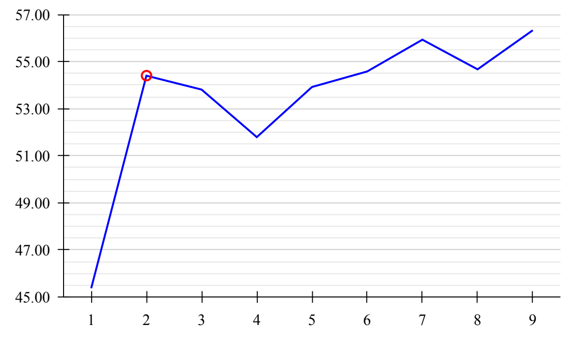

Figure 4: Validating WF1 of fine-tuned RoBERTa-base over k BPE on IEMOCAP

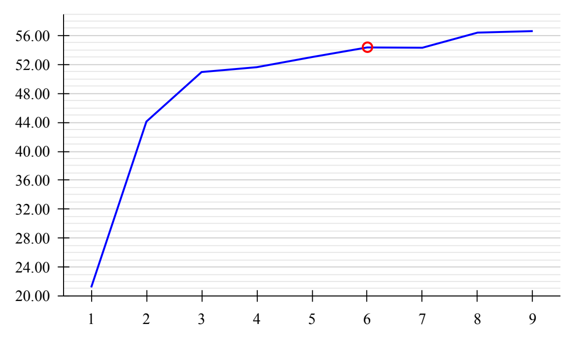

Figure 5: Validating WF1 of fine-tuned RoBERTa-large over k BPE on IEMOCAP

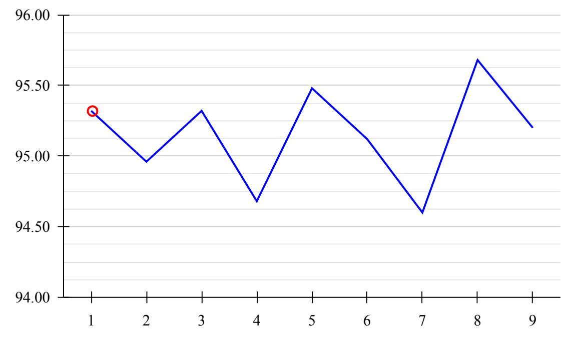

Figure 6: Validating WF1 of fine-tuned RoBERTa-base over k BPE on IMDb

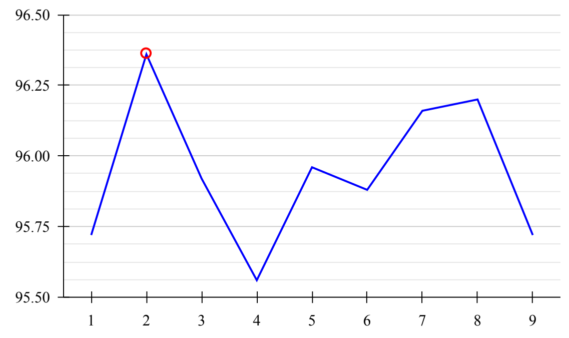

Figure 7: Validating WF1 of fine-tuned RoBERTa-large over k BPE on IMDb

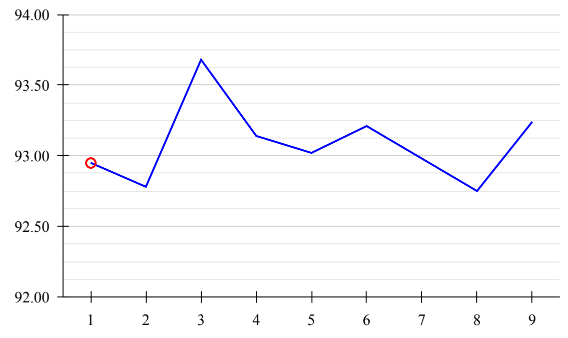

Figure 8: Validating WF1 of fine-tuned BERT-base-uncased over k BPE on AG News

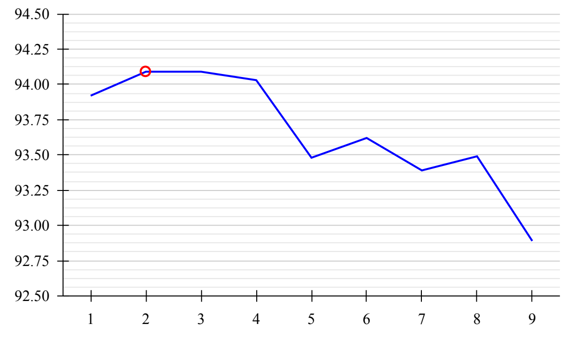

Figure 9: Validating WF1 of fine-tuned BERT-large-uncased over k BPE on AG News

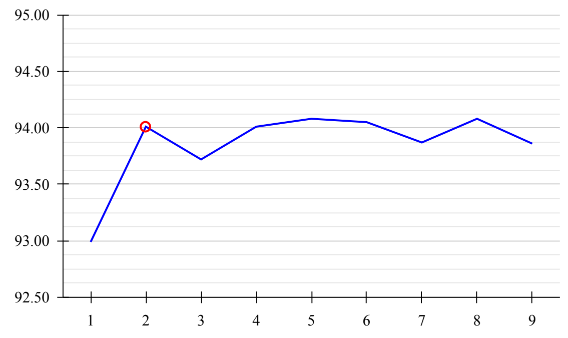

Figure 10: Validating WF1 of fine-tuned RoBERTa-base over k BPE on AG News

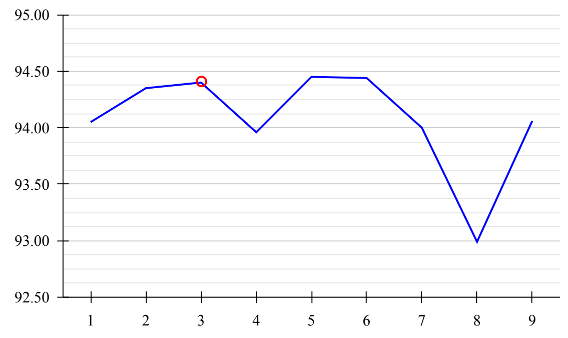

Figure 11: Validating WF1 of fine-tuned RoBERTa-large over k BPE on AG News

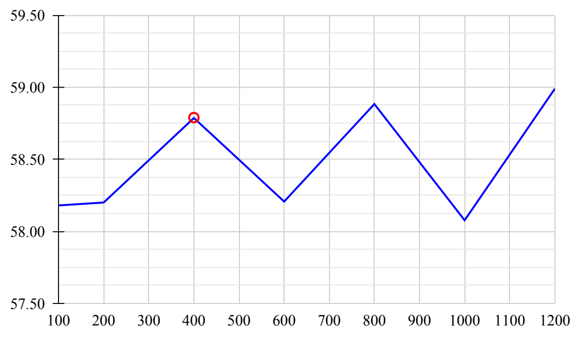

Figure 12: Validating WF1 of fine-tuned EFT-RoBERTa-base over nh I-ELM hidden nodes on MELD

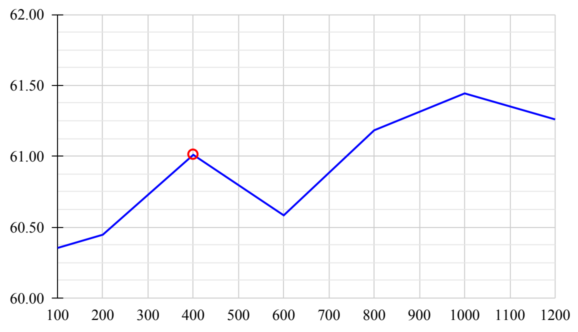

Figure 13: Validating WF1 of fine-tuned EFT-RoBERTa-large over nh I-ELM hidden nodes on MELD

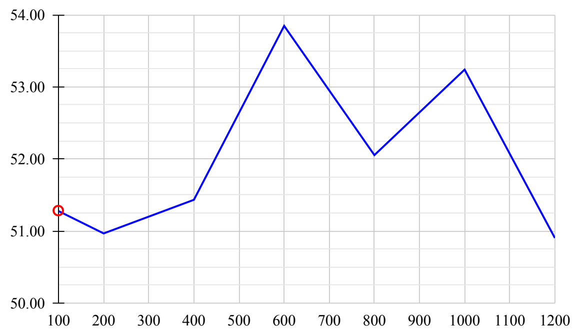

Figure 14: Validating WF1 of fine-tuned EFT-RoBERTa-base over nh I-ELM hidden nodes on IEMOCAP

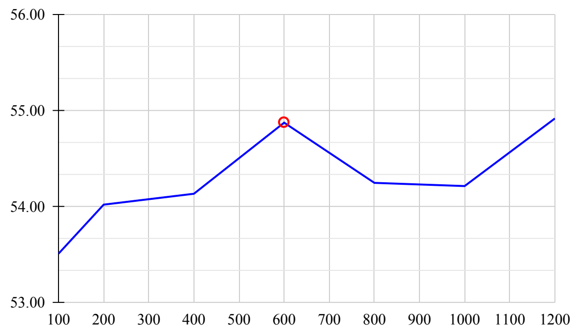

Figure 15: Validating WF1 of fine-tuned EFT-RoBERTa-large over nh I-ELM hidden nodes on IEMOCAP

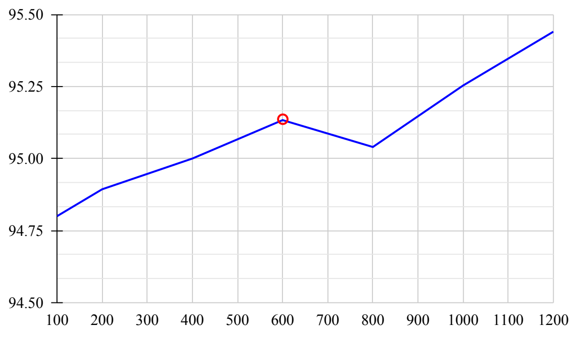

Figure 16: Validating WF1 of fine-tuned EFT-RoBERTa-base over nh I-ELM hidden nodes on IMDb

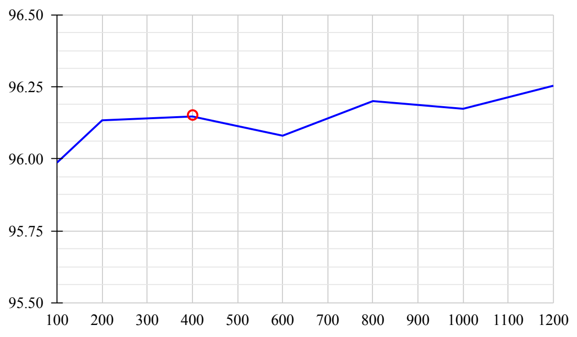

Figure 17: Validating WF1 of fine-tuned EFT-RoBERTa-large over nh I-ELM hidden nodes on IMDb

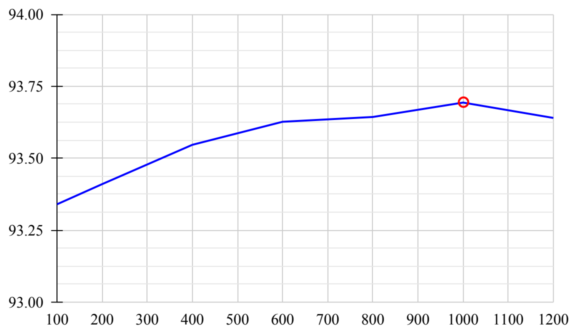

Figure 18: Validating WF1 of fine-tuned EFT-BERT-base-uncased over nh I-ELM hidden nodes on AG News

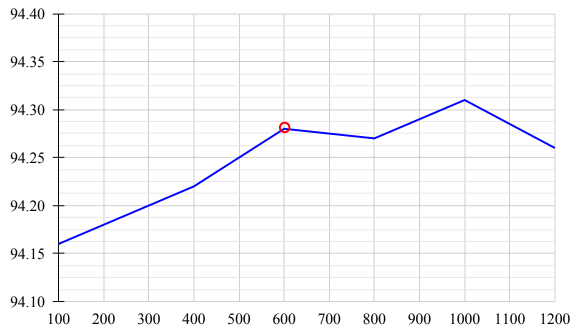

Figure 19: Validating WF1 of fine-tuned EFT-BERT-large-uncased over nh I-ELM hidden nodes on AG News

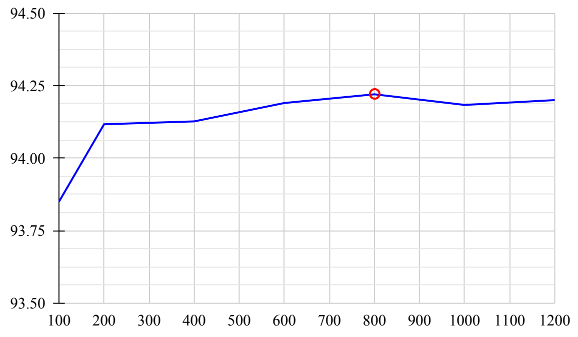

Figure 20: Validating WF1 of fine-tuned EFT-RoBERTa-base over nh I-ELM hidden nodes on AG News

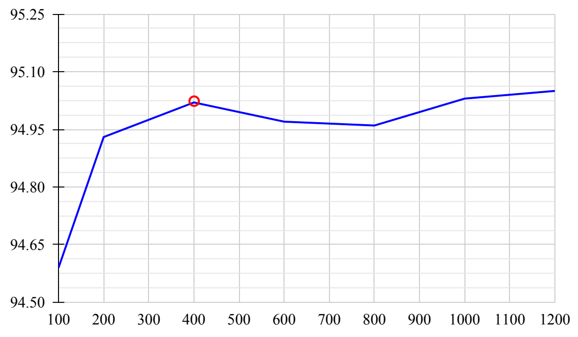

Figure 21: Validating WF1 of fine-tuned EFT-RoBERTa-large over nh I-ELM hidden nodes on AG News