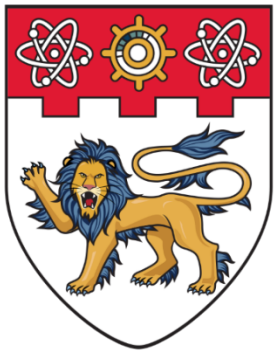

NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

SC4001 – Neural Networks and Deep Learning

Group Report

Done by:

Name	Matriculation Number
Huang Yongjian	U2320879D
Syed Ali Redha Alsagoff	U2340588F

Table of Contents

Introduction

Text sentiment analysis is a classic problem in Natural Language Processing (NLP) and has wide ranging applications in market research, opinion mining and customer feedback. In this project, we are primarily working with the IMDB dataset, a binary sentiment classification dataset consisting of 25,000 examples each of positive and negative movie reviews [1]. We conducted an empirical analysis on the model performance between different Transformer architectures such as encoder-only models (BERT, DistilBERT, ELECTRA, DeBERTa-v3 and Sentence-BERT) and decoder-only models (LLaMA and GPT-2) to determine which architecture is best-suited for different use-cases. In addition, we present the results of different parameter-efficient finetuning methods including LoRA, QLoRA and Prompt Tuning for BERT and DeBERTa-v3 for different sample sizes to test for sample size efficiency, training and inference time and memory overhead. To deal with limited data, we experimented with different data augmentation techniques and few-shot learning models. We also conducted further analysis for zero-shot and few-shot prompts for decoder-only models. Finally, we will present the results for tackling domain adaptation for encoder-only model finetuned on IMDB dataset, including ablation studies on component-based model freezing and a novel adversarial-finetuning method on the SST-2, a similar binary sentiment classification dataset [2].

Review of Existing Techniques

Traditionally, Recurrent Neural Network (RNN) have been widely used to model sequential data like text due to the inherent design of hidden states evolving over time to capture sequence dependencies. However, RNNs struggle to learn long-term dependency between different tokens due to the vanishing gradient problem arising from backpropagation through time. Furthermore, the sequential nature of RNN is difficult to parallelise, resulting in slower training and inference time that linearly scales with input size. To deal with this, Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) introduced gated mechanism in RNN to regulate the flow of information in the network. This helps facilitate more effective gradient propagation across longer sequences. However, these gated neural networks are still limited by being uni-directional in nature and only considers past context when predicting the next token, resulting in the loss of important information from future context. This is particularly relevant in natural language, where the choice of words used is often influenced by both the preceding and succeeding words. While Bidirectional Long Short-Term Memory (BiLSTM) tries to address this limitation by processing the sequence from left-to-right and right-to-left and concatenating the final hidden states [3], the learnt representation from a simple concatenation operation may not fully capture the complex inter-token dependencies.

To overcome these issues, the Transformer architecture was introduced as a recurrence-free approach with the novel inclusion of multi-headed self-attention, which allows input tokens to attend to each other simultaneously regardless of their relative position [4]. This enables the model to learn bidirectional context more efficiently. Furthermore, the computation of self-attention involves mainly matrix multiplication, which

is easily parallelizable with the advent of modern GPU acceleration. As such, the Transformer architecture outperforms RNN-based models in both training efficiency and the modelling of long-term dependencies.

There are three types of Transformer architectures: Encoder-only, Decoder-only and Encoder-Decoder. We will be focusing on the Encoder-only and Decoder-only architectures for our project.

For Encoder-only variant, the Bidirectional Encoder Representations from Transformers (BERT) was a breakthrough in NLP by stacking multiple Encoder layers from the Transformer Architecture. After pretraining on large text corpus on Masked Language Modelling (MLM) and Next Sequence Prediction (NSP) tasks, BERT is able to learn meaningful bidirectional context in language and achieved state-of-the-art performance in various NLP tasks after supervised finetuning on downstream datasets [5]. Many of the subsequent breakthroughs in model architecture for NLP are similarly variants of the BERT, which we will be testing for our sentiment analysis task. DistilBERT is a lightweight version of BERT that utilises knowledge distillation during pretraining to reduce model size by 40%, while retaining 97% of its language understanding capabilities and being 60% faster [6]. ELECTRA replaced the MLM pretraining in BERT with a more sample-efficient pretraining task called Replaced Token Detection (RTD) involving a discriminator network predicting the tokens which have been replaced by samples drawn from a generator network [7]. DeBERTa introduced disentangled attention where each word is represented by its positional and word embeddings, with the attention weights being computed using the disentangled matrices [8]. DeBERTa-v3 improves the model performance further by adopting the ELECTRA-style RTD pretraining with Gradient Disentangled Embedding Sharing [9]. SentenceBERT extends BERT with a Siamese and triplet network structure during training (typically finetuned on sentence-pair task) to produce semantically meaningful sentence embeddings [10].

For Decoder-only variants, GPT-2 is a causal language model with 1.5B parameter that is trained on a large corpus known as WebText without explicit supervision on next-token prediction [11]. LLaMA is a family of open-sourced language models that are trained on publicly available dataset and had achieve state-of-the-art performances with fewer parameters compared to other models [12].

Apart from Transformer-based models, we also further experimented with xLSTM, which introduces exponential gating with memory mixing and new memory structure to achieve comparable performances to Transformer on language modelling tasks [13]. For few-shot learning experiment, we also used Sentence Transformer Finetuning (SetFit), an efficient prompt-free framework that utilises contrastive learning to finetune Sentence Transformer without requiring large annotated datasets [14].

Experiments, Results & Discussions

We present our experiment results trained using a single A100 GPU node, using a mix of Google Colab notebooks and NSCC jobs to improve time effiency. Training time varies from 30 minutes to 4 hours.

Experiment 1: Comparing Base models on sentiment analysis tasks

Methodology

We tested a variety of models, comprising of transformer and LSTM architectures. For our transformer models, we compared BERT, DistilBERT, ELECTRA, DeBERTa-v3, GPT-2 and LLaMA 3.2 1B, with last two finetuned on a sequence classification head.

Furthermore, we wanted to test whether testing few-shot using a classification head would yield better results. We hypothesise this to work well since for causal tasks, few-shot prompting yields better results. We believe this experiment would be insightful and worth going into since we rarely see few-shot prompting used this way for a non-causal objective. The training examples for the prompts were randomly sampled from the rest of the training set, and the models were finetuned on few-shot prompts for the classification task for fair comparison.

Moreover, we experimented with causal methods for different-sized LLaMA models from 1B to 8B, outlined in Table 3.2, 3.3, 3.4 for experiment 3. We attempted this for GPT-2 as well, but since it had not undergone instruction tuning like the LLaMA models, it was not feasible to get a good sentiment analysis pipeline from it, as it didn't follow the instructions from the prompt well. For all the encoder-only models, we used Optuna, a Bayesian Optimisation framework for hyperparameter tuning [15] for number of epochs, batch-size, learning rate and weight-decay (only for BERT and ELECTRA) over 5 trials. For our LSTM models, we compared the performance of BiLSTM and xLSTM, trained over 20 epochs with max sequence length of 256, batch size of 32 and embedding dimension of 128. We repeat use of these optimal parameters in later experiments due to computational and time constraints.

Results & Discussion

Model	Accuracy	F1
BERT	94.1%	94.2%
DistilBERT	93.1%	93.2%
ELECTRA	95.6%	95.6%
DeBERTa-v3	**96.1%**	**96.2%**

Model	Accuracy	F1
GPT-2	93.9%	94.0%
LLaMA (zero-shot)	83.3%	81.1%
LLaMA (5-shot)	92.5%	92.6%

Model	Accuracy
BiLSTM	85.1%
xLSTM	84.4%

Tables 1.1 (left), 1.2 (middle) and 1.3 (right): Scores for sentiment analysis for BERT models, Decoder-Only models, and LSTM models respectively

As we can see above, DistilBERT was the worst BERT model, as expected since it is simply a distilled version of BERT and would thus be the only model that perform worse than BERT. Our best model is DeBERTa-v3, and we hypothesise it is due to its robust disentangled attention that allows it to gain better contextual understanding of its input sequences, allowing it to better predict sentiments.

LLaMA 1B was the worst model among the decoder-only models, and GPT-2 was the best decoder-only model. This is, at first glance counter-intuitive, but it makes sense when we realised the model's pre-training objective is not mainly context understanding, but next-token prediction. GPT-2, being a smaller model, would thus be less rigid in its re-learning. We can also see that our few-shot hypothesis was proven right, with the LLaMA 1B few-shot model improving accuracy by almost 10%.

Transformer models were also mostly better than LSTMs across the board, and BiLSTM was a slightly better model than xLSTM.

Experiment 2: Comparison between different Parameter-Efficient Finetuning (PEFT) methods

Methodology

We compared three different PEFT methods: Low-Rank Adaptation (LoRA), Quantized Low-Rank Adaptation (QLoRA) and Prompt Tuning. LoRA is able to achieve efficient finetuning by injecting small trainable low-rank matrices into the attention layers while freezing the original model weight [16]. QLoRA reduces the memory footprint of LoRA further with the adoption of 4-bit quantization [17]. Prompt tuning (PT) is another PEFT method that introduces learnable soft prompt embeddings that are prepended to the input while keeping the entire model weight frozen [18].

We present the results for the three different PEFT methods and supervised finetuning (SFT) for BERT and DeBERTa-v3 models across varying levels of data availability. For brevity, we will name the different data availability as the following: Full (25,000 train + 25,000 test examples), Medium (5,000 train + 5,000 test examples) and Small (500 train + 500 test examples). QLoRA result will only be shown for BERT as there is well documented issue for mixed precision training for DeBERTa-v3 (explained in the Colab notebook in more details). We will be comparing the results with the following metrics: accuracy, F1, training time, inference time, percentage of trainable parameters, memory footprint and peak memory usage. The following results are obtained using the following tuned hyperparameters: BERT (learning rate: 3e-5, batch size: 16, epochs: 3, weight decay: 0.1) and DeBERTa-v3 (learning rate: 2e-5, batch size: 32, epochs: 2).

Results & Discussion

Model	Accuracy	F1
BERT (Full, SFT)	94.1%	94.2%
BERT (Full, LoRA)	91.4%	91.5%
BERT (Full, QLoRA)	90.9%	91.1%
BERT (Full, PT)	72.5%	73.5%
BERT (Medium, SFT)	92.5%	92.6%
BERT (Medium, LoRA)	84.1%	85.0%
BERT (Medium, QLoRA)	81.8%	83.1%
BERT (Medium, PT)	50.1%	0.7%
BERT (Small, SFT)	87.8%	87.7%
BERT (Small, LoRA)	51.4%	54.9%
BERT (Small, QLoRA)	60.6%	55.1%
BERT (Small, PT)	60.4%	61.6%

Model	Accuracy	F1
DeBERTa-v3 (Full, SFT)	96.1%	96.2%
DeBERTa-v3 (Full, LoRA)	93.8%	93.9%
DeBERTa-v3 (Full, PT)	50.2%	0.7%
DeBERTa-v3 (Medium, SFT)	95.0%	95.1%
DeBERTa-v3 (Medium, LoRA)	54.7%	58.5%
DeBERTa-v3 (Medium, PT)	50.3%	2.9%
DeBERTa-v3 (Small, SFT)	85.0%	82.5%
DeBERTa-v3 (Small, LoRA)	49.0%	65.8%
DeBERTa-v3 (Small, PT)	51.2%	1.6%

Tables 2.1 (left) and 2.2 (right): Model comparison across different data availability

Generally, model performance is ranked as follows: SFT > LoRA > QLoRA > PT. Both models perform better with more training data and it is interesting to note that all PEFT methods suffer greatly from model degradation in small and medium data scenarios. As PEFT involves only a small percentage of trainable parameters, we hypothesise that larger gradient updates are less likely to cause learning instabilities and will enable better generalisation and sample efficiency in limited data scenarios. To test this, we experimented with higher learning rate at 1e-3 and were able to obtain greatly improved results in Table 2.3. Note that additionally for PT, we also increased the number of epochs to 15 and batch size to 32 in line with the author's suggestion due to the randomly initialised nature of the prompt embedding, which will need more aggressive tuning to learn to encode useful representation [18]. This explains why PT performs very badly under the original settings.

Model	Accuracy	F1
BERT (Medium, LoRA, lr: 1e-3)	91.4%	91.7%
BERT (Medium, QLoRA, lr: 1e-3)	91.8%	92.1%
BERT (Medium, PT, lr: 1e-3, 15 epochs)	82.9%	83.1%
BERT (Small, LoRA, lr: 1e-3)	84.0%	83.8%
BERT (Small, QLoRA, lr: 1e-3)	86.0%	86.4%
BERT (Small, PT, lr: 1e-3, 15 epochs)	68.8%	66.4%
DeBERTa-v3 (Medium, LoRA, lr: 1e-3)	94.9%	95.1%
DeBERTa-v3 (Medium, PT, lr: 1e-3, 15 epochs)	94.4%	94.4%
DeBERTa-v3 (Small, LoRA, lr: 1e-3)	83.2%	80.4%
DeBERTa-v3 (Small, PT, lr: 1e-3, 15 epochs)	49.2%	66.0%

Table 2.3 Model performance comparison after adjusting learning rate

Next, we report the training and inference time for full data scenario in Table 2.4. We can see that PT has the best training and inference time, followed by LoRA, SFT and QLoRA. However, we note that PT generally does not perform well with few epochs and increasing the number of epochs to improve performance will drastically increase training time. As such, for scenarios where training time is crucial, LoRA will be the most suitable. For scenarios where inference time is crucial, SFT and LoRA are excellent choices.

Model	Training time (s)	Inference time (s)
BERT (Full, SFT)	1433	155
BERT (Full, LoRA)	1093	165
BERT (Full, QLoRA)	1594	171
BERT (Full, PT)	1054	167
DeBERTa-v3 (Full, SFT)	1533	273
DeBERTa-v3 (Full, LoRA)	1290	263
DeBERTa-v3 (Full, PT)	764	125

Table 2.4 Time comparison between different finetuning experiments

Lastly, we present the results in Table 2.5 for number of training parameters, memory footprint and peak memory usage of BERT model trained in full data scenario.

Model	Training params %	Memory footprint (MB)	Peak memory (MB)
BERT (Full, SFT)	100	417.7	7435
BERT (Full, LoRA)	0.40	419.3	5429
BERT (Full, QLoRA)	0.40	133.9	2082
BERT (Full, PT)	0.55	417.7	4532

Table 2.5 Memory comparison between finetuning experiments

This is where PEFT shines. PEFT reduces training parameters significantly, with QLoRA reducing the both the model size and peak memory usage during training by more than two-thirds compared to SFT. The memory efficiency of QLoRA make it most suitable for low-memory systems, while maintaining decent performance.

Experiment 3: Comparing few-shot learning methods for limited data cases

Methodology

To simulate a learning scenario with limited data, each model is trained on 16 training examples per sentiment and evaluated on 3,000 unseen examples. Some of the tested models leverage the sentence embeddings generated by the Sentence Transformer all-MiniLM-L6-v2. To test if the embedding generated can encode useful sentence-level information, we conducted a simple linear probing experiment using logistic regression trained on these embeddings and obtained an accuracy score of 81.1%, validating the embeddings' effectiveness.

We experimented with the following models: K-Means Classification on sentence embedding, Nearest Neighbour Classification (1-NN) on sentence embedding, 1-NN on sentence embedding and data augmentation (back-translation, deletion, synonym replacement and mixed), BERT SFT, BERT SFT with data augmentation (back-translation) and SetFit. For the data augmentation, 2 augmented examples are created per sample.

Results & Discussion

Model	Acc
K-Means	50.4%
1-NN	59.2%
1-NN w back-translation	61.2%
1-NN w synonym replacement	58.2%
1-NN w deletion	58.9%
1-NN w mixed aug	60.5%
BERT	49.7%
BERT w back-translation	52.6%
SetFit	64.3%

Table 3.1 Model comparisons for few-shot learning

SetFit has the best performance followed by 1-NN with back-translation. This implies that SetFit helps in few-shot learning and augmenting dataset with back-translation can further improve model performance under limited data scenario. This makes sense given that SetFit trains on a contrastive learning objective that pulls the embedding space of samples with same labels together and maximises differences between those with different labels. Back-translation is also more effective than other data augmentation as it is not a naïve word-for-word replacement strategy and can capture the semantic meaning of anchor examples while providing lexical diversity to promote model generalisation in limited data settings. As a result, a naïve 1-NN method showed most improvement when the anchor data is augmented with back-translation.

Methodology

Another method that we could consider, if the number of training samples is extremely small (less than 100), is make use of Large Language Models (LLMs). While LLMs are not particularly trained for sentiment analysis, they were trained on a much larger corpus of data compared to other models and learnt more robust representations words and world knowledge, which allows them to generalise very well to tasks. Moreover, upscaling them to the order of billions of parameters have resulted in emergent behavior such as In-Context Learning (ICL) [19]. Recent works have shown that ICL has even greater potential than finetuning for aligning model parameters for implicit patterns [20], which may apply to sentiment analysis. We thus compared the ICL capabilities of 3 LLMs: LLaMA 1B, 3B, and 8B. For fair comparison, we kept models from the same general series to reduce potential bias from different types of pre-training strategies used. We tested these models under the following set of shots: {0,1,3,5,10,20} to measure their performance. This allows us to fairly measure how scaling up model parameters and shot examples help to improve performance for sentiment analysis.

As for our experimental set-up, we used the test split of the IMDB dataset to test the models and randomly sampled the examples from the train set to be used for the few-shot examples. After getting the examples, the model was given the following system prompt: "You are a sentiment analysis expert. Given a movie

review, classify the sentiment as either positive or negative. Only output 'positive' or 'negative' without any other text or explanation.". However, we found that the models tend to output their answer as the first word, followed by a justification, despite the instructions. Thus, we simply just sampled the first word of the generated answer. In the case of an invalid but partially matched answer like “ive” or “ative”, we re-prompted the model again up to 2 times, with the system prompt: "You are a sentiment analysis expert. Given a movie review, classify the sentiment as either positive or negative. CRITICAL INSTRUCTION: Your response MUST START with EXACTLY the word 'positive' or the word 'negative'. The first word of your response must be either 'positive' or 'negative', representing the answer.". If the model still generates an invalid answer after 3 tries, we count it as a wrong answer. We also present the statistics for ‘corrected’ accuracy, which only considers cases when the sampled output is either ‘positive’ or ‘negative’ and excludes results that are marked as wrong due to formatting issues. Tables 3.2, 3.3 and 3.4 show our results.

Results & Discussion

Shots	Accuracy	Corrected accuracy	Avg tries
0	87.7%	91.6%	1.086
1	87.0%	90.8%	1.085
3	83.2%	86.9%	1.086
5	85.4%	89.2%	1.086
10	87.8%	91.7%	1.086
20	0.0%	0.0%	3.000

Shots	Accuracy	Corrected acc.	Avg tries
0	88.8%	92.8%	1.103
1	87.0%	91.4%	1.098
3	87.2%	91.3%	1.095
5	89.4%	84.2%	1.103
10	89.1%	93.7%	1.101
20	0.0%	0.0%	3.000

Shots	Accuracy	Corrected accuracy	Avg tries
0	88.8%	92.5%	1.085
1	87.1%	91.0%	1.086
3	88.4%	92.4%	1.087
5	90.6%	94.8%	1.090
10	90.7%	94.8%	1.090
20	0.0%	0.0%	3.000

Tables 3.2 (left), 3.3 (middle), 3.4 (right): Few-shot performance for LLaMA 1B, 3B and 8B respectively

As we can see above, in general, increasing the number of examples does improve results for the larger 3B and 8B models slightly, and does not really affect 1B much. This makes sense, since the emergent ICL ability of LLMs is due to its scale, and thus scaling up further would improve the LLM’s ability for ICL. We can also see that the accuracy generally increases with the number of parameters.

We can also see 20-shot prompts led a 0% accuracy for all the models, due to incorrect formatting. This can be attributed to the smaller LLMs having lower learning capacity, especially for ICL. We hypothesise that this is due to contextual drift, where the model overfits to the context window, that it forgets the proper formatting. Due to computational constraints, we could not test this on larger models. From our debugging scripts (that were removed to reduce excessive clutter), we found that most of the rejected answers were ‘ositive’ and ‘egative’, both of which could imply the intended answer. As such, we did an ablation study on the 0% accuracy cases by allowing these partially matched answers (barring the case of ‘tive’) to be considered correct and report our findings in Table 3.5.

Model	Accuracy	Corrected accuracy	Avg tries
1B (20-shot)	75.8%	83.8%	1.19
3B (20-shot)	85.4%	95.4%	1.21
8B (20-shot)	86.3%	95.5%	1.20

Table 3.5: Ablation study on easing restrictions for model output acceptance

Based on the above, while it significantly improved results, retry-rates is still relatively higher than results from 10-shot and below, showing that this relaxation of the accepted result is still insufficient. However, corrected accuracy rates have improved for 3B and 8B, implying that more shots for ICL does indeed improve performance, but at a trade-off with coherency after a certain threshold.

Experiment 4: Comparing Different Finetuning methods for Domain Adaptation

Methodology

The use of supervised finetuning is a common approach for dealing with domain shifts and adaptation. However, there is a limited number of specialised finetuning methods proposed for this issue. As such, we propose the use of Domain Adversarial Finetuning (DAFT), inspired by Domain Adversarial Training of Neural Networks (DANNs) [22], a technique used to learn domain-invariant representations. DANNs opt for a dual training objective of domain confusion and label discrimination, to extract features that are domain-invariant for classification. Unlike traditional DANNs that train from scratch, DAFT applies this adversarial principle only to the finetuning stage of pretrained language models. By inserting a domain classifier with gradient reversal during backpropagation, we only adapt necessary components of the model while preserving the knowledge encoded in the pretrained weights. We hypothesise this allows for more efficient transfer learning between domains with minimal computational overhead compared to full adversarial training from scratch.

For the experimental set-up, we used our BERT model finetuned on the IMDB dataset in experiment 1 as baseline 1. For baseline 2, we finetuned BERT on the SST-2 dataset. To test for domain adaption using SFT, we conducted an ablation study for transfer learning in baseline 1 on the SST-2 dataset with component-based model freezing acting as a form of regularisation. This also efficiently saves on compute by finetuning on less parameters.

We used only 3 or 5 epochs for all experiments, depending on when convergence occurred. The parameters at the best epoch were used for testing. For the component-based model freezing, we tested on the following three components: head-only, body-only and head + last 2 layers. We used the same hyperparameters that were optimal for BERT found in experiment 1.

For DAFT, we used the following hyperparameters: lambda=0.1, batch size=16, epochs=3, learning rate=3e-5, weight decay=0.01. We chose these parameters from best parameters found finetuning the BERT model earlier and parameters recommended from the DANN paper. There was a dimension

mismatch between IMDB and SST-2 for the domain discrimination task, so we opted to test 2 different strategies for the model: truncating to 128 tokens and dynamically padding them to 512 tokens.

We used accuracy, F1 score and training time as a metric to compare the methods. Additionally, we also tested the models back on the IMDB dataset, to see how well the model generalised and how much the model reduced its original domain knowledge. We report our results in Tables 4.1, 4.2 and 4.3.

Results & Discussion

Model	Accuracy	F1
Baseline 1	86.9%	87.8%
Baseline 2	92.8%	92.4%
Head-only	87.0%	87.8%
Body-only	92.8%	92.9%
Head + last 2 layers	91.6%	91.7%
DAFT (truncated)	92.7%	92.7%
DAFT (dynamic padding)	**93.1%**	**93.2%**

Model	Accuracy	F1
Baseline 1	94.1%	94.2%
Baseline 2	90.4%	90.6%
Head-only	94.1%	93.9%
Body-only	92.9%	93.4%
Head + last 2 layers	92.8%	92.7%
DAFT (truncated)	89.3%	89.4%
DAFT (dynamic padding)	**93.9%**	**93.9%**

Model	Training time (s)
Head-only	672
Body-only	3670
Head + last 2 layers	1480
DAFT (truncated)	1425
DAFT (dynamic padding)	2574

Table 4.1 (left), 4.2 (middle): Results of DAFT and model component freezing on SST-2 and IMDB Dataset respectively. Table 4.3 (right): Training (Finetuning) time for each model

As we can see above, in terms of SFT, the body achieves the best average accuracy for both datasets. However, finetuning on the last 2 layers and the head shows close accuracy at a fraction of the training time, implying that it can be used under low resource constraints. For DAFT, it shows the best results with decent training time under the dynamic padding scheme. For the truncated scheme, while it yields decent results with shorter time, it does not retain its knowledge from the IMDB dataset well. This makes sense, since the IMDB dataset was the one being truncated for the experiment. This shows that DAFT with dynamic padding gives the best trade-off and confirms our hypothesis earlier. Truncation should only be used if the model is not going to be tested on the source domain again after finetuning, and the target domain is not truncated.

Conclusion & Recommendations

We have conducted a comprehensive study on the model performance of different architectures, different PEFT methods and their use-cases, model performance under limited data as well as the issue of domain adaptation. In summary, DeBERTa-v3 is the go-to model for best accuracy if time and compute is not the concern. PEFT like LoRA and QLoRA are apt when memory and compute are limited and LLMs perform the best in low-data scenarios given that they are pretrained on large corpus and can perform generally well in zero-shot and few-shot scenarios. Lastly, we found that DAFT with dynamic padding is indeed an efficient and effective way for domain adaptation and generalisability, but in scenarios where compute is low, basic finetuning on the last few layers and the head for transfer learning still yield decent results.

References

[1] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, “Learning Word Vectors for Sentiment Analysis,” in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, D. Lin, Y. Matsumoto, and R. Mihalcea, Eds., Portland, Oregon, USA: Association for Computational Linguistics, Jun. 2011, pp. 142–150. Accessed: Apr. 08, 2025. [Online]. Available: https://aclanthology.org/P11-1015/

[2] R. Socher *et al.*, “Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank,” in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, D. Yarowsky, T. Baldwin, A. Korhonen, K. Livescu, and S. Bethard, Eds., Seattle, Washington, USA: Association for Computational Linguistics, Oct. 2013, pp. 1631–1642. Accessed: Apr. 11, 2025. [Online]. Available: https://aclanthology.org/D13-1170/

[3] Z. Huang, W. Xu, and K. Yu, “Bidirectional LSTM-CRF Models for Sequence Tagging,” Aug. 09, 2015, *arXiv*: arXiv:1508.01991. doi: 10.48550/arXiv.1508.01991.

[4] A. Vaswani *et al.*, “Attention Is All You Need,” Aug. 02, 2023, *arXiv*: arXiv:1706.03762. doi: 10.48550/arXiv.1706.03762.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” May 24, 2019, *arXiv*: arXiv:1810.04805. doi: 10.48550/arXiv.1810.04805.

[6] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, “DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter,” Mar. 01, 2020, *arXiv*: arXiv:1910.01108. doi: 10.48550/arXiv.1910.01108.

[7] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, “ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators,” Mar. 23, 2020, *arXiv*: arXiv:2003.10555. doi: 10.48550/arXiv.2003.10555.

[8] P. He, X. Liu, J. Gao, and W. Chen, “DeBERTa: Decoding-enhanced BERT with Disentangled Attention,” Oct. 06, 2021, *arXiv*: arXiv:2006.03654. doi: 10.48550/arXiv.2006.03654.

[9] P. He, J. Gao, and W. Chen, “DeBERTaV3: Improving DeBERTa using ELECTRA-Style Pre-Training with Gradient-Disentangled Embedding Sharing,” Mar. 24, 2023, *arXiv*: arXiv:2111.09543. doi: 10.48550/arXiv.2111.09543.

[10] N. Reimers and I. Gurevych, “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks,” Aug. 27, 2019, *arXiv*: arXiv:1908.10084. doi: 10.48550/arXiv.1908.10084.

[11] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language Models are Unsupervised Multitask Learners,” 2019. Accessed: Apr. 11, 2025. [Online]. Available: https://www.semanticscholar.org/paper/Language-Models-are-Unsupervised-Multitask-Learners-Radford-Wu/9405cc0d6169988371b2755e573cc28650d14dfe

[12] H. Touvron *et al.*, “LLaMA: Open and Efficient Foundation Language Models,” Feb. 27, 2023, *arXiv*: arXiv:2302.13971. doi: 10.48550/arXiv.2302.13971.

[13] M. Beck *et al.*, “xLSTM: Extended Long Short-Term Memory,” Dec. 06, 2024, *arXiv*: arXiv:2405.04517. doi: 10.48550/arXiv.2405.04517.

[14] L. Tunstall *et al.*, “Efficient Few-Shot Learning Without Prompts,” Sep. 22, 2022, *arXiv*: arXiv:2209.11055. doi: 10.48550/arXiv.2209.11055.

[15] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, “Optuna: A Next-generation Hyperparameter Optimization Framework,” Jul. 25, 2019, *arXiv*: arXiv:1907.10902. doi: 10.48550/arXiv.1907.10902.

[16] E. J. Hu *et al.*, “LoRA: Low-Rank Adaptation of Large Language Models,” Oct. 16, 2021, *arXiv*: arXiv:2106.09685. doi: 10.48550/arXiv.2106.09685.

[17] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, “QLoRA: Efficient Finetuning of Quantized LLMs,” May 23, 2023, *arXiv*: arXiv:2305.14314. doi: 10.48550/arXiv.2305.14314.

[18] B. Lester, R. Al-Rfou, and N. Constant, “The Power of Scale for Parameter-Efficient Prompt Tuning,” Sep. 02, 2021, *arXiv*: arXiv:2104.08691. doi: 10.48550/arXiv.2104.08691.

[19] J. Wei *et al.*, “Emergent Abilities of Large Language Models”.

[20] Q. Yin *et al.*, “Deeper Insights Without Updates: The Power of In-Context Learning Over Fine-Tuning,” Oct. 07, 2024, *arXiv*: arXiv:2410.04691. doi: 10.48550/arXiv.2410.04691.

[21] B. Upadhayay, V. Behzadan, and A. Karbasi, “Cognitive Overload Attack:Prompt Injection for Long Context,” Oct. 15, 2024, *arXiv*: arXiv:2410.11272. doi: 10.48550/arXiv.2410.11272.

[22] Y. Ganin *et al.*, “Domain-Adversarial Training of Neural Networks,” May 26, 2016, *arXiv*: arXiv:1505.07818. doi: 10.48550/arXiv.1505.07818.

[23] J. Kirkpatrick *et al.*, “Overcoming catastrophic forgetting in neural networks,” *Proc. Natl. Acad. Sci. U.S.A.*, vol. 114, no. 13, pp. 3521–3526, Mar. 2017, doi: 10.1073/pnas.1611835114.