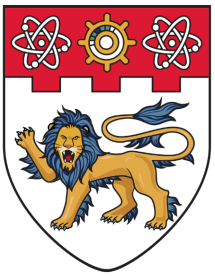

NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

SC4001: Neural Network and Deep Learning

Group Project Report

No.	Name	Matriculation Number
1	Nadya Yuki Wangsajaya	U2320059K
2	Shiu Lok Chun, Wesley	U2321673A

Group Project Report **1**
1. Introduction **3**
2. Capabilities to Evaluate **3**
3. Performance of Different Types of Fine-tuning **3**
3.1. Review of Existing Techniques 3
3.1.1. Low-Rank Adaptation (LoRA) 3
3.1.2. Prefix-tuning 4
3.1.3. Infused Adapter by Inhibiting and Amplifying Inner Activations (IA3) 4
3.2. Methodology 4
3.3. Results 5
3.4. Discussion 5
4. Comparison of Different Transformer Architectures **5**
4.1. Review of Existing Techniques 5
4.1.1. BERT 5
4.1.2. GPT-2 6
4.1.3. BART 6
4.2. Methodology 6
4.3. Results 7
4.4. Discussion 7
5. Comparison of Different Types of BERT Models **7**
5.1. Review of Existing Techniques 7
5.1.1. ALBERT 7
5.1.2. RoBERTa 8
5.2. Methodology 8
5.3. Results 8
5.4. Discussion 9
6. Domain Adaptation **9**
6.1. Review of Existing Techniques 9
6.2. Methodology 9
6.3. Results 9
6.4. Discussion 10
7. Data Augmentation **10**
7.1. Review of Existing Techniques 10
7.2. Methodology 10
7.3. Results 10
7.4. Discussion 10
8. Hyperparameter Tuning **11**
8.1. Review of Existing Techniques 11
8.2. Methodology 11
8.3. Results 11
8.4. Discussion 12
9. Conclusion **12**
10. References **13**

1. Introduction

Our team selected Task C: Sentiment Analysis (SA) using the IMDB movie review dataset, where the model is tasked to predict the sentiment of a given movie review. As the reviews are written by humans, they are littered with various literary techniques, such as sarcasm and irony, which are challenging to discern for a straightforward neural network.

In this report, our team worked closely with transformer models, which is the SOTA deep learning architecture for natural language processing (NLP). We experimented with different transformer architectures, BERT models, and fine-tuning methods. We also dabbled in domain adaptation problems, data augmentation tasks and hyperparameter search for fine-tuning. In the end, we aim to provide a comprehensive report on the transformer's performance in text sentiment analysis tasks.

2. Capabilities to Evaluate

We intend to understand the overarching capabilities of transformer models in text sentiment analysis. In particular, we study the following capabilities:

- **Performance of different types of fine-tuning.** Between full fine-tuning, LoRA, prefix-tuning, and IA3, which one results in the most performant BERT model?
- **Comparison of various transformer architectures.** In text sentiment analysis tasks, which one performs the best, encoder-only, decoder-only or encoder-decoder model?
- **Comparison of various BERT models.** Between BERT, RoBERTa, and ALBERT, which one is the most suitable in text sentiment analysis tasks?
- **Domain adaptation.** How can I adapt a model pre-trained on a general dataset to our specific IMDB dataset?
- **Data augmentation.** How can I deal with small dataset availability?
- **Hyperparameter tuning.** How do I find the best hyperparameter for fine-tuning models?

3. Performance of Different Types of Fine-tuning

3.1. Review of Existing Techniques

In order for pre-trained models to be functional in text sentiment analysis tasks, we have to fine-tune it to a subset of the IMDB dataset. There are many approaches to fine-tuning models, such as (1) unsupervised fine-tuning, where the language model is trained on an unlabelled dataset, (2) supervised fine-tuning (SFT), where it is trained on datasets with labels, (3) reinforcement learning from human feedback (RLHF), in which human feedback is used for optimization [1]. We found the most suitable method to be SFT, as our IMDB dataset provides a labelled training split.

In terms of parameter update, there are a few options: (1) full fine-tuning, which updates the entire parameter of the pre-trained model, and (2) parameter efficient fine-tuning (PEFT) [2], which updates only a small portion of the model parameter, reducing the computational costs incurred by full fine-tuning while maintaining similar performance. Truthfully, for our use case, since we mostly dealt with small-sized BERT models (100 - 200M parameters), the computational cost of fully fine-tuning the model is small and therefore, it is theoretically the best method to create the most performant model. However, we are curious about how it compares to PEFT, and so we present results comparing full fine-tuning with three different PEFT methods [4][5][6].

3.1.1. Low-Rank Adaptation (LoRA)

One of the most popular PEFT methods, widely used in diffusion models, but also has language model applications [4][7][8]. The creators of LoRA hypothesize that updates in weights during pre-trained model fine-tuning have low intrinsic rank (i.e. the parameters only need to be adjusted slightly). Through Singular Value Decomposition (SVD), we can therefore decompose this low-rank update matrix $\triangle W$ to be the product of 2 low-rank matrices A and B.

During backpropagation, only low-rank matrices A and B are updated, while the pre-trained layer W is frozen. After training is done, we merge W with A and B using the formula $W_{\text{merged}} = W + \frac{\alpha}{r}BA$, where α is a hyperparameter and r is the rank (usually between 1 - 8).

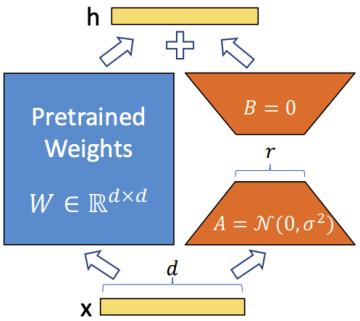

Figure 1. LoRA, only A and B are updated during backpropagation

In Figure 1, with full fine-tuning, we would need to train $d \times d$ parameters, while using LoRA, we only need to train $2 \times r \times d$. With LoRA, we reduce training space and time.

3.1.2. Prefix-tuning

A type of prompt tuning, a trainable module called 'prefix', is added to each transformer layer (specifically prepended to the K and V sequence of the attention mechanism) in the pre-trained large language model [5][9][10]. The pre-trained weights are frozen, changes are only made to the prefix.

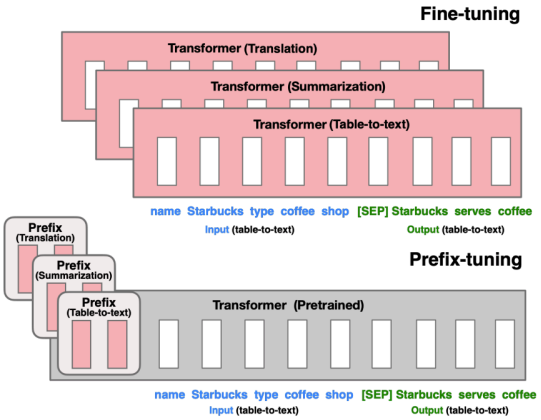

Figure 2. Prefix-tuning, only the prepended prefix (in pink) are updated

Since the length of prefixes, p, is a small hyperparameter, prefix-tuning also aims to reduce training space and time (Figure 2).

3.1.3. Infused Adapter by Inhibiting and Amplifying Inner Activations (IA3)

It is an extremely lightweight PEFT method which rescales the activations within the attention and feedforward modules of a transformer [6][11]. The rescaling is based on a trainable learned vector fine-tuned during training, while pre-trained weights are frozen. As such, updates are only made to the small learned vectors, whose size is equivalent to the summation of all hidden layer sizes.

3.2. Methodology

The fixed hyperparameter configuration for the model comparisons are as follows: **learning rate:** 2e-5, **batch size:** 16, **weight decay:** 0.01, **warmup rate:** 0.0, **patience:** 3. For LoRA: **rank:** 8, **alpha:** 32, **lora dropout:** 0.1 (to reduce overfitting). For prefix-tuning: **prefix length / number of virtual tokens:** 20. We also do not apply prefix-tuning to the last layer of the model. For IA3: **target**

modules: [q, k, v]. We used the Adam optimizer and we set a maximum of 5 epochs. We used BERT-base as our model.

3.3. Results

	Trainable parameter / %	Accuracy	F1 score	Training speed / min
Full fine-tuning	100	**0.930**	**0.931**	56
LoRA	0.40	0.912	0.914	44
Prefix-tuning	12.0	0.905	0.907	47
IA3	0.07	0.727	0.728	**34**

Table 1. Accuracy, F1 score and training speed of BERT

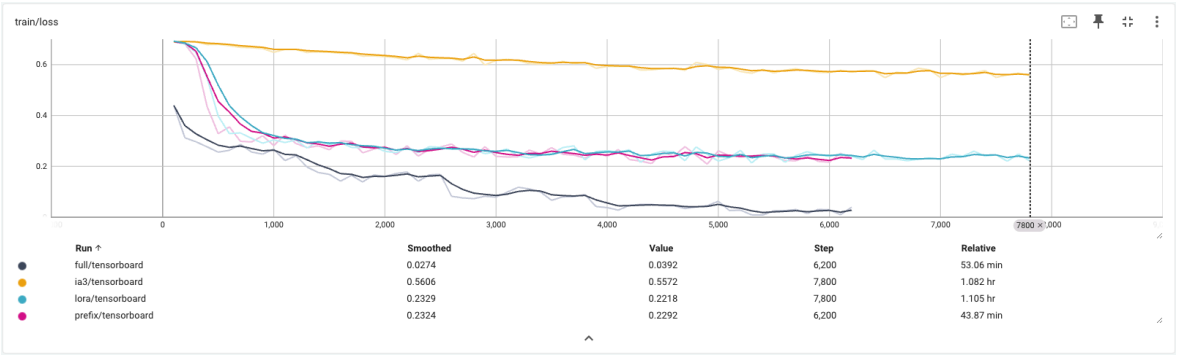

Figure 3. Training loss of different fine-tuning methods

As expected, full fine-tuning is strictly the best method in terms of accuracy and F1 score. LoRA and prefix-tuning also achieves respectable results as they slashed 15 - 20% of training time while maintaining good accuracy and F1 score. Most surprisingly, IA3 performs worst in terms of accuracy and F1 score, despite its quick training speed.

3.4. Discussion

Looking at the training progress (Figure 3), IA3 loss decreases very slowly compared to LoRA and prefix-tuning. This might be due to hyperparameter mismatch, where the target modules or learning rate that we chose is not suitable. Additionally, it is important to note that IA3 only changes a fraction of parameters (0.07%) compared to LoRA and prefix-tuning (0.4% and 12% respectively) and full fine-tuning (100%). As such, it is expected that it would perform worse compared to the other three. The hypothesis that PEFT helps save time and compute is corroborated by our results, where PEFT methods saved at least 15% of training time, showing its potential to greatly reduce overhead when training larger models. Since the best performing fine-tuning method is full fine-tuning, we will use full fine-tuning for the rest of the experiments.

4. Comparison of Different Transformer Architectures

4.1. Review of Existing Techniques

Transformer architectures for NLP tasks are typically categorised into three groups: encoder-only, decoder-only, and encoder-decoder. As their name suggests, each architecture consists of either an encoder, a decoder, or both. In this section, we examine and compare these three architectures.

4.1.1. BERT

BERT, which stands for Bidirectional Encoder Representations from Transformers, is an encoder-only model that is known for its ability to read both left and right context simultaneously. BERT is typically

used for text classification tasks such as sentiment analysis, due to its ability to understand full sentences from its bidirectional architecture [11].

BERT is pre-trained on two self-supervised tasks: **Masked Language Modelling** (MLM) and **Next Sentence Prediction** (NSP). In MLM, some words are randomly replaced with a [MASK] token, and the model is trained to predict the original word that was masked based on the context of the remaining words in the sentence. In NSP, the model is fed pairs of sentences, and learns to predict whether the second sentence follows the first in the original text. In both tasks, cross-entropy loss is used during training.

Following pre-training, BERT is fine-tuned for downstream tasks by replacing the MLM and NSP heads with a task-specific classification head, typically a dense layer. For sentiment analysis, the classification head predicts whether the text expresses a positive or negative sentiment, and cross entropy is used during training.

4.1.2. GPT-2

GPT-2 [12], also known as Generative Pre-trained Transformer 2, is a decoder-only model that generates texts in an autoregressive manner. GPT-2 is trained using causal masking to predict the next token given all previous tokens. Since GPT-2 processes inputs from left to right, it is naturally suited for text generation.

Nevertheless, GPT-2 can be finetuned for classification tasks like sentiment analysis [13]. The input text is passed through the pre-trained decoder, and the final token's hidden state is treated as the representation of the whole sequence. A classification head is appended to the final token's hidden state to predict the sentiment of the entire sequence. Cross-entropy loss is used for training.

4.1.3. BART

BART [14], or Bidirectional and AutoRegressive Transformers, is an encoder-decoder model that combines the strength of both BERT and GPT-2. The encoder is fully bidirectional, allowing it to understand an entire input, while the decoder generates texts token by token in an autoregressive manner. As such, BART is known as a sequence-to-sequence transformer model, since it transforms one piece of text into another.

BART is a denoising autoencoder, and is pre-trained by corrupting input text and training the model to reconstruct the original. The original BART paper uses the following five noising techniques: **Token Masking** identical to that in BERT, **Token Deletion** where random tokens from the input are deleted, **Text Infilling** where spans of texts are replaced with a single [MASK] token, **Sentence Permutation** where sentences are shuffled in a random order, and **Document Rotation** where a token is chosen randomly, and the document is rotated to begin in that order.

BART can also be adapted to sequence classification tasks like sentiment analysis. The same input text is fed into both the encoder and decoder, with a special classification token appended to the end of the decoder input. The decoder then performs cross attention using the encoder's output and uses the final decoder token's hidden state to make classification.

4.2. Methodology

We use pre-trained BERT, GPT-2 and BART models taken from HuggingFace and perform full fine-tuning for sentiment analysis on the IMDB dataset. The fixed hyperparameter configuration for the model comparisons are as follows: **learning rate:** 2e-5, **batch size:** 16, **weight decay:** 0.01, **patience:** 3. We split the "train" dataset into 80% training and 20% validation set, before running on 50 epochs with early-stopping, while using Adam optimizer.

4.3. Results

	Parameters no.	Accuracy	F1 score	Best epoch	Train samples per second
BERT	110M	0.931	0.931	4	341
GPT-2	117M	0.932	0.931	2	**433**
BART	139M	**0.943**	**0.944**	2	370

Table 2. Comparison of different transformer architecture

Despite being the most natural choice for classification tasks, BERT fared the worst in terms of accuracy, F1 score and training speed. Our results from BERT agree with other studies [15]. GPT-2 performed near-identical in terms of accuracy and F1 score to BERT, but delivered a 27% higher training throughput in terms of samples processed per second, while converging earlier. BART outperformed BERT and GPT-2 in accuracy and F1 score, while demonstrating decent training speed by converging as quickly as GPT-2, and delivering 8.5% higher training throughput than BERT.

4.4. Discussion

GPT-2 and BART's performances demonstrate the capabilities of a decoder in sentiment analysis, despite the fact that classification is not their natural task. Furthermore, the higher training throughputs of GPT-2 and BART indicate that BERT's fully bidirectional self-attention could be more expensive compared to causal attention [16].

Nonetheless, encoder-only models are proven to generally outperform decoder-only models [17]. BERT's underperformance in our experiment could be attributed to its smaller model size, as a larger number of model parameters are known to increase model performance in NLP [18]. Additionally, BERT may be sub-optimal and inefficient compared to other encoder-only models, which we will be examining in the next section.

5. Comparison of Different Types of BERT Models

5.1. Review of Existing Techniques

BERT was introduced to the world in 2018, and marked a huge leap in NLP performance on many benchmarks. Just a year later, Google and Facebook released their own variations of BERT, named ALBERT and RoBERTa respectively, which they claim to be an improvement from the original BERT.

5.1.1. ALBERT

ALBERT, which stands for A Lite BERT, claims to reduce memory usage and training time while maintaining or even improving BERT's performance [19]. Scaling model parameters has shown to improve performance in NLP, but solely relying on scaling model size is not feasible given limitations in resources like data, time, storage and energy [18]. ALBERT's ability to reduce model size while retaining performance is achieved by two parameter reduction techniques.

Firstly, **factorised embedding parameterisation** decouples WordPiece embedding size E from hidden layer size H. The WordPiece embeddings carry context-independent representations, while hidden-layer embeddings carry context-dependent representations, thus H should naturally be larger than E. However, BERT ties E with H (i.e. $E \equiv H$), which is suboptimal as increasing H necessitates E to increase as well, which in turn increases the size of the embedding matrix, which has size $V \times E$, where V represents the vocabulary size. ALBERT fixes this inefficiency by projecting the tokens into a lower-dimensional embedding space of size E, before projecting it into the hidden space of size H. As such, the embedding parameters are reduced from $O(V \times H)$ to $O(V \times E + E \times H)$, which results in a significant parameter saving when $H \gg E$.

In BERT, each transformer layer has its own distinct parameters, making the model very large, especially as the number of layers increases. ALBERT counters this by implementing **cross-layer parameter sharing**, where the same set of weights for the attention head and feed-forward network is reused for each layer. This dramatically reduces the number of parameters in the model, and allows stacking of more layers without compromising model size. Furthermore, the weight-sharing stabilises the network during training and allows for faster training.

In addition to parameter reduction, the ALBERT paper ascertains that BERT's NSP is ineffective due to its lack of difficulty compared to MLM [19]. NSP trains a model to predict whether two sentences follow each other, combining topic prediction and coherence prediction. However, topic prediction is significantly easier and tends to dominate the task, while coherence is more challenging and not sufficiently learned through NSP. Thus, ALBERT improves on NSP by introducing **Sentence Order Prediction** (SOP), a similar technique that removes topic prediction and focuses on coherence prediction. In SOP, a positive example is two sentences in the correct order, while a negative example is the same two sentences, but in reversed order. This is contrary to NSP, where a negative example will likely contain two sentences of different topics.

5.1.2. RoBERTa

RoBERTa, which stands for Robustly Optimized BERT Approach, retains the same model architecture as BERT, but makes several key improvements in how BERT is pre-trained [20]. The RoBERTa paper claims that BERT is significantly undertrained, motivating the development of RoBERTa.

Compared to BERT, RoBERTa is trained on longer input sequences, using larger batch sizes, over a greater volume of data, and for a longer duration. This minimises undertraining, and allows for improvement in generalisation. Similar to ALBERT, RoBERTa identifies NSP as ineffective and even detrimental for downstream performances [20][19]. However, unlike ALBERT which replaces NSP with another task, RoBERTa completely removes NSP from pre-training.

While BERT uses static masking, RoBERTa improves on this by employing dynamic masking. In BERT, masks are applied during preprocessing, thus a certain sentence will always be masked in the same way during training. In contrast, RoBERTa randomly masks tokens at runtime, meaning that the same sentence can be masked differently in each training epoch. This allows the model to learn richer representations because it sees more diverse masking patterns.

5.2. Methodology

We compare these three models based on their MLM accuracy before fine-tuning. We use pre-trained BERT, GPT-2 and BART models taken from HuggingFace and perform full fine-tuning for sentiment analysis on the IMDB dataset. The fixed hyperparameter configuration for the model comparisons are as follows: **learning rate:** 2e-5, **batch size:** 16, **weight decay:** 0.01, **patience:** 3. We split the "train" dataset into 80% training and 20% validation set, before running on 50 epochs with early-stopping, while using Adam optimizer. Note that SA refers to Sentiment Analysis in the table below.

5.3. Results

	Parameters no.	MLM accuracy	SA accuracy	SA F1 score
BERT	110M	0.573	0.931	0.931
ALBERT	**11.8M**	0.168	0.933	0.934
RoBERTa	125M	**0.655**	**0.945**	**0.945**

Table 3. Comparison of different BERT models

RoBERTa beats BERT in MLM accuracy, while ALBERT performs extremely poorly in MLM accuracy. In terms of sentiment analysis accuracy and F1 score, ALBERT slightly edges BERT out despite being about 10.7% smaller in terms of number of parameters. However, RoBERTa is the clear winner not just amongst BERT and ALBERT, but also GPT-2 and BART from section 4.

5.4. Discussion

ALBERT's shockingly low MLM accuracy suggests that ALBERT performs poorly at token-level predictions, since ALBERT has a much smaller embedding size (128), compared to BERT's embedding size of 768. Although ALBERT is pre-trained on the same datasets as BERT, namely the BookCorpus [21] and English Wikipedia datasets, the reduced embedding dimensionality inherently limits its representational capacity, resulting in lower precision for token-level modelling and reduced generalisability in MLM tasks. Prior work [22] reports ALBERT's MLM accuracy at approximately 0.540, which supports our hypothesis. The fact that our results fall significantly below this reported value further suggests a lack of generalisation, while BERT achieves an MLM accuracy that exceeds 0.540 in our experiment. However, this is not a huge issue for sentiment analysis, as ALBERT outperforms BERT in sentiment analysis accuracy and F1 score. This is extremely impressive given 89.3% reduction in model size, and shows that ALBERT effectively sacrifices token-level prediction for parameter efficiency, while still retaining sentence-level understanding for classification tasks.

RoBERTa being the most performant model in MLM and sentiment analysis shows that it is truly a robust improvement over BERT. Furthermore, RoBERTa outperforms GPT-2 and BART, demonstrating the prowess of encoder-only models in classification tasks like sentiment analysis.

6. Domain Adaptation

6.1. Review of Existing Techniques

In NLP, domain adaptation refers to the process of fine-tuning a pre-trained language model on domain-specific data, typically using transfer learning, to improve performance on tasks within that domain [23]. Fine-tuning a pre-trained model like BERT on MLM is shown to allow the model to adapt to the target domain distribution effectively and efficiently, resulting in better performance in downstream tasks [24].

6.2. Methodology

We aim to compare the performance of BERT with and without fine-tuning MLM on our domain of IMDB movie reviews. We fine-tune the pre-trained BERT model taken from HuggingFace on MLM using the unsupervised split of the IMDB dataset, and we name this fine-tuned model **BERT-Pro**. We then perform full fine-tuning of BERT and BERT-Pro for sentiment analysis on the IMDB dataset. The fixed hyperparameter configuration for the model comparisons are as follows: **learning rate:** 2e-5, **batch size:** 16, **weight decay:** 0.01, **patience:** 3. We split the "train" dataset into 80% training and 20% validation set, before running on 50 epochs with early-stopping, while using Adam optimizer.

6.3. Results

	MLM accuracy	SA accuracy	SA F1 score
BERT	0.573	0.931	0.931
BERT-Pro	**0.650**	**0.947**	**0.947**

Table 4. Comparison of different original and domain-adapted BERT

As expected, BERT-Pro outperforms BERT in all metrics, and achieves extremely impressive sentimental analysis accuracy and F1 score, exceeding our previous best set by RoBERTa.

6.4. Discussion

The performance of BERT-Pro highlights the importance of domain adaptation in NLP, demonstrating that a "sub-optimal" model like BERT can outperform RoBERTa when adapted to the target domain. When the source pre-training data and target domain are misaligned, performance often degrades: a phenomenon known as domain shift [25]. Fine-tuning BERT on the IMDB unsupervised dataset using the MLM objective helps align the training distribution with the target distribution, thereby mitigating the effects of domain shift.

7. Data Augmentation

7.1. Review of Existing Techniques

Data augmentation in NLP is slightly more tricky than in Computer Vision, where we could simply rotate or crop the image. Back-translation is one possible data augmentation technique for monolingual data, and the idea is to translate texts from the target language to a source language, then back to the target language [26]. Two machine translation models will be required: one to translate from target to source, and another from source to target. These machine translation models are trained using parallel data in a supervised manner, which consists of pairs of texts that are translations of each other in two different languages [27].

7.2. Methodology

We use MarianMT [28] to translate the train split of the IMDB dataset from English to German, then back to English, since this pair of languages have been proven effective for back-translation [26]. We combined the original dataset with the augmented dataset to form a combined dataset, then fine-tune BERT using the combined dataset. We experiment with two data-splitting strategies of the training and validation set: (1) randomly splitting the combined dataset, and (2) ensuring consistent splits between the original and augmented data (i.e. original data and its generated augmented data are always placed in the same split). The fixed hyperparameter configuration for the model comparisons are as follows: **learning rate:** 2e-5, **batch size:** 16, **weight decay:** 0.01, **patience:** 3. We split the "train" dataset into 80% training and 20% validation set, before running on 50 epochs with early-stopping, while using Adam optimizer.

7.3. Results

	Accuracy	F1 score	Best epoch
BERT fine-tuned on original dataset (25k)	**0.931**	**0.931**	4
BERT fine-tuned on combined dataset with random splitting (50k)	0.928	0.929	16
BERT fine-tuned on combined dataset with consistent splitting (50k)	0.926	0.928	**1**

Table 5. Comparison of BERT models with data augmentation

Surprisingly, contrary to what we initially expected, the augmented data from back-translation not only did not improve performance, but degraded it. Fine-tuning on the combined dataset with random splitting also caused the model to converge much later than usual, while fine-tuning on the combined dataset with consistent splitting allowed for less epochs before convergence

7.4. Discussion

The poor performance is likely attributed to the **translationese effect,** which describes the structural differences between translated and non-translated texts [29]. Back-translation is particularly susceptible to this effect: while it can serve as an effective data augmentation technique when the source sentences are translationese, it tends to underperform when the source sentences are natural text [30][31]. Given that the IMDB dataset consists of highly informal and naturally written movie

reviews, the translationese effect could be amplified, potentially leading to a domain shift away from the original target distribution.

The high epochs needed for convergence during the fine-tuning using the combined dataset with random splitting may be explained by the lack of independence between the training and validation sets. We hypothesise that random splitting allows pairs of original and augmented texts to appear in both sets simultaneously, resulting in data leakage. As a result, the validation set no longer provides an unbiased estimate of the true generalisation loss. Consequently, the validation loss may continue to decrease even after the model has overfitted, since the validation samples are more similar to those in the training set. This aligns with prior research suggesting that random data splitting may lead to overly optimistic evaluation metrics [32].

In contrast, the fine-tuning with the combined dataset using consistent splitting ensures that the training and validation set are independent of each other, and thus did not exhibit this issue of data leakage, resulting in convergence at epoch 1 instead of 16, supporting our hypothesis. The larger dataset size also allowed for more training steps per epoch, allowing our BERT model to converge in less epochs. Nevertheless, performance remained limited, likely due to the translationese effect discussed above.

8. Hyperparameter Tuning

8.1. Review of Existing Techniques

While fine-tuning, there are multiple hyperparameters involved: *learning rate, batch size, weight decay* (regularization), and *warmup rate*. To find the best possible hyperparameter values, we conducted hyperparameter tuning using the Optuna library [33].

Mathematically, Optuna optimizes the hyperparameter using Bayesian methods, specifically using the Tree-Structured Parzen Estimator (TPE) [34]. The statistics behind this method is rather complex, but essentially, the algorithm first collects results from randomly chosen hyperparameter values. From these results, the algorithm utilizes its 'splitting algorithm' to split the observed results into good trials and bad trials. Using Parzen Estimator, the algorithm then models the distribution of each of these trials, where $l(x)$ is the distribution of good trials and $g(x)$ is the distribution of bad trials. TPE then chooses the next $x = \text{argmax}\frac{l(x)}{g(x)}$. As such, hyperparameters are tuned smartly, instead of using brute-force methods such as grid-search.

8.2. Methodology

For our hyperparameter tune experimentations, we search through these spaces using the aforementioned TPE: **learning rate:** [1e-5, 5e-5], **batch size:** 8, 16, 32, **weight decay:** [0.01, 0.1], **warmup ratio:** [0.0, 0.1]. We set the number of trials to 20, with 5 epochs trained per trial. The model we used is the BERT-base model.

8.3. Results

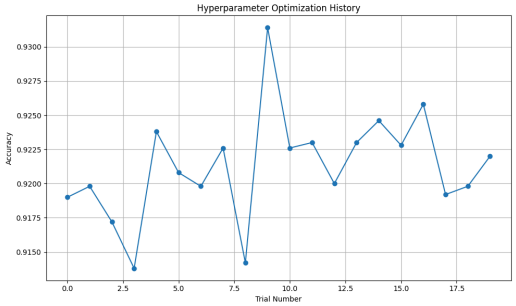

Figure 4. Accuracy across 20 trials

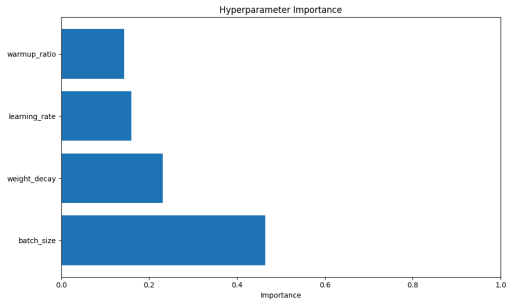

Figure 5. batch_size is the most important parameter

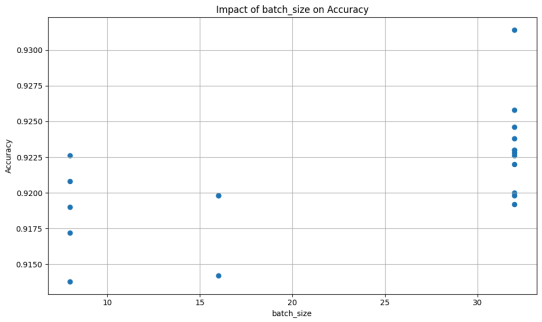

Figure 6. Higher batch_size of 32 performs best

Out of 20 trials, we found trial 10 with the following configurations to be the best for BERT (Figure 4). It uses this configuration: **learning rate:** 3.5e-5, **batch size:** 32, **weight decay:** 0.082, **warmup ratio:** 0.035. Using these settings, we get accuracy of **0.931** and F1 score of **0.931**. With optuna's get_param_importances() function, which utilizes ANOVA framework [35], we discovered that the most important hyperparameter is batch size (Figure 5). Further investigation reveals that batch size of 32 leads to a higher accuracy compared to 8 and 16 (Figure 6).

8.4. Discussion

Through hyperparameter tuning, we receive an accuracy score higher than our arbitrarily chosen hyperparameters. It is very surprising to see that batch size being the most important hyperparameter, and a higher batch size resulting in a higher accuracy. Usually, higher batch sizes end in worse results as the model converges on local minima instead [36]. However, a batch size that is too small also results in a more noisy training process, which impedes convergence [36]. As such, the lower accuracy at smaller batch sizes suggests that the model could not converge well within 5 epochs. We can conclude that a batch size of 32 is ideal for fine-tuning a model capped at a small number of epochs. It is important to note that we only searched batch sizes of 8, 16, and 32, due to time and compute considerations, as more options would necessitate a higher number of trials for a pattern to emerge. Further work should investigate the impact of even higher batch sizes, such as 128 and 512, on accuracy.

9. Conclusion

In conclusion, we found that full fine-tuning is the most suitable fine-tuning method due to the small IMDB dataset, where PEFT methods shine in larger dataset with greater compute demands. Additionally, despite GPT-2 and BART outperforming BERT on sentiment analysis on our dataset, we maintain that encoder-only models should be the architecture of choice in sentiment analysis, as BERT is an inefficient model amongst other encoder-only models. RoBERTa was able to demonstrate that it was superior to BERT, and outperformed GPT-2 and BART as well to reclaim the glory of encoder-only models. We then highlighted the importance of domain adaptation, by showing the significant improvements in model performance after fine-tuning BERT on the domain of IMDB movie reviews using the MLM objective. We also experimented with back-translation as a data augmentation technique to deal with small datasets, but found that back-translation was largely unsuitable for our IMDB dataset as it suffers from the translationese effect. Lastly, hyperparameter search using TPE reveals that higher batch size, such as 32, is important in ensuring higher accuracy, despite conventional wisdom.

On a personal note, this project was a truly enjoyable and highly educational experience for us. It challenged us to apply the knowledge gained throughout this module, while encouraging us to explore concepts beyond the classroom. The process inspired us to adopt a more analytical mindset in identifying optimal methodologies, and interpreting unexpected results with critical thought. Overall, we certainly gained a much deeper understanding of transformers, NLP, fine-tuning techniques and hyperparameter tuning which we greatly value as learning outcomes.

10. References

[1] “Illustrating Reinforcement Learning from Human Feedback (RLHF).” Accessed: Apr. 06, 2025. [Online]. Available: https://huggingface.co/blog/rlhf

[2] “PEFT.” Accessed: Apr. 09, 2025. [Online]. Available: https://huggingface.co/docs/peft/en/index

[3] 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.

[4] X. L. Li and P. Liang, “Prefix-Tuning: Optimizing Continuous Prompts for Generation,” Jan. 01, 2021, *arXiv*: arXiv:2101.00190. doi: 10.48550/arXiv.2101.00190.

[5] H. Liu *et al.*, “Few-Shot Parameter-Efficient Fine-Tuning is Better and Cheaper than In-Context Learning,” Aug. 26, 2022, *arXiv*: arXiv:2205.05638. doi: 10.48550/arXiv.2205.05638.

[6] “What is LoRA (Low-Rank Adaption)? | IBM.” Accessed: Apr. 08, 2025. [Online]. Available: https://www.ibm.com/think/topics/lora

[7] “Low Rank Adaptation: A Technical deep dive.” Accessed: Apr. 08, 2025. [Online]. Available: https://www.ml6.eu/blogpost/low-rank-adaptation-a-technical-deep-dive

[8] A. Razavi, “Understanding Prefix Tuning: A Novel Approach to Fine-Tuning Language Models,” Medium. Accessed: Apr. 06, 2025. [Online]. Available: https://medium.com/@razavipour6/understanding-prefix-tuning-a-novel-approach-to-fine-tuning-language-models-dc7dafeb32e4

[9] “Prefix Tuning vs. Fine-Tuning and other PEFT methods.” Accessed: Apr. 08, 2025. [Online]. Available: https://toloka.ai/blog/prefix-tuning-vs-fine-tuning/

[10] “IA3.” Accessed: Apr. 08, 2025. [Online]. Available: https://huggingface.co/docs/peft/en/conceptual_guides/ia3

[11] 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.

[12] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language Models are Unsupervised Multitask Learners”.

[13] D. E. Lee, “Fine-Tuning GPT-2 for Sentiment Analysis,” Medium. Accessed: Apr. 11, 2025. [Online]. Available: https://drlee.io/fine-tuning-gpt-2-for-sentiment-analysis-94ebdd7b5b24

[14] M. Lewis *et al.*, “BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension,” Oct. 29, 2019, *arXiv*: arXiv:1910.13461. doi: 10.48550/arXiv.1910.13461.

[15] S. Alaparthi and M. Mishra, “BERT: a sentiment analysis odyssey,” *J. Mark. Anal.*, vol. 9, no. 2, pp. 118–126, Jun. 2021, doi: 10.1057/s41270-021-00109-8.

[16] L. Yang, X. Zhou, J. Fan, X. Xie, and S. Zhu, “Can bidirectional encoder become the ultimate winner for downstream applications of foundation models?,” Nov. 27, 2024, *arXiv*: arXiv:2411.18021. doi: 10.48550/arXiv.2411.18021.

[17] A. Benayas, M. A. Sicilia, and M. Mora-Cantallops, “A comparative analysis of encoder only and decoder only models in intent classification and sentiment analysis: navigating the trade-offs in model size and performance,” *Lang. Resour. Eval.*, Dec. 2024, doi: 10.1007/s10579-024-09796-y.

[18] M. Treviso *et al.*, “Efficient Methods for Natural Language Processing: A Survey,” *Trans. Assoc. Comput. Linguist.*, vol. 11, pp. 826–860, Jul. 2023, doi: 10.1162/tacl_a_00577.

[19] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, “ALBERT: A Lite BERT for Self-supervised Learning of Language Representations,” Feb. 09, 2020, *arXiv*: arXiv:1909.11942. doi: 10.48550/arXiv.1909.11942.

[20] Y. Liu *et al.*, “RoBERTa: A Robustly Optimized BERT Pretraining Approach,” Jul. 26, 2019, *arXiv*: arXiv:1907.11692. doi: 10.48550/arXiv.1907.11692.

[21] Y. Zhu *et al.*, “Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books,” presented at the Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 19–27. Accessed: Apr. 11, 2025. [Online]. Available: https://www.cv-foundation.org/openaccess/content_iccv_2015/html/Zhu_Aligning_Books_and_ICCV_2015_paper.html

[22] S.-H. Tsang, “Review — ALBERT: A Lite BERT for Self-supervised Learning of Language Representations,” Medium. Accessed: Apr. 11, 2025. [Online]. Available: https://sh-tsang.medium.com/review-albert-a-lite-bert-for-self-supervised-learning-of-language-representations-14e1fcc05ba9

[23] “Fine-tuning a masked language model - Hugging Face LLM Course.” Accessed: Apr. 11, 2025. [Online]. Available: https://huggingface.co/learn/llm-course/en/chapter7/3

[24] C. Karouzos, G. Paraskevopoulos, and A. Potamianos, “UDALM: Unsupervised Domain Adaptation through Language Modeling,” Apr. 14, 2021, *arXiv*: arXiv:2104.07078. doi: 10.48550/arXiv.2104.07078.

[25] A. Ramponi and B. Plank, “Neural Unsupervised Domain Adaptation in NLP---A Survey,” Oct. 28, 2020, *arXiv*: arXiv:2006.00632. doi: 10.48550/arXiv.2006.00632.

[26] S. Edunov, M. Ott, M. Auli, and D. Grangier, “Understanding Back-Translation at Scale,” Oct. 03, 2018, *arXiv*: arXiv:1808.09381. doi: 10.48550/arXiv.1808.09381.

[27] J. Wieting, T. Berg-Kirkpatrick, K. Gimpel, and G. Neubig, “Beyond BLEU: Training Neural Machine Translation with Semantic Similarity,” Sep. 14, 2019, *arXiv*: arXiv:1909.06694. doi: 10.48550/arXiv.1909.06694.

[28] “MarianMT.” Accessed: Apr. 11, 2025. [Online]. Available: https://huggingface.co/docs/transformers/en/model_doc/marian

[29] V. Volansky, N. Ordan, and S. Wintner, “On the features of translationese,” *Digit. Scholarsh. Humanit.*, vol. 30, no. 1, pp. 98–118, Apr. 2015, doi: 10.1093/llc/fqt031.

[30] S. Edunov, M. Ott, M. Ranzato, and M. Auli, “On The Evaluation of Machine Translation Systems Trained With Back-Translation,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, Eds., Online: Association for Computational Linguistics, Jul. 2020, pp. 2836–2846. doi: 10.18653/v1/2020.acl-main.253.

[31] X.-P. Nguyen, S. Joty, K. Wu, and A. T. Aw, “Data Diversification: A Simple Strategy For Neural Machine Translation,” in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2020, pp. 10018–10029. Accessed: Apr. 11, 2025. [Online]. Available:

https://proceedings.neurips.cc/paper_files/paper/2020/hash/7221e5c8ec6b08ef6d3f9ff3ce6eb1d1-Abstract.html

[32] “Splitting data randomly can ruin your model | Data Science.” Accessed: Apr. 11, 2025. [Online]. Available: https://datascience.stanford.edu/news/splitting-data-randomly-can-ruin-your-model

[33] “Optuna - A hyperparameter optimization framework,” Optuna. Accessed: Apr. 11, 2025. [Online]. Available: https://optuna.org/

[34] S. Watanabe, “Tree-Structured Parzen Estimator: Understanding Its Algorithm Components and Their Roles for Better Empirical Performance,” May 26, 2023, *arXiv*: arXiv:2304.11127. doi: 10.48550/arXiv.2304.11127.

[35] F. Hutter, H. Hoos, and K. Leyton-Brown, “An Efficient Approach for Assessing Hyperparameter Importance,” in *Proceedings of the 31st International Conference on Machine Learning*, PMLR, Jan. 2014, pp. 754–762. Accessed: Apr. 10, 2025. [Online]. Available: https://proceedings.mlr.press/v32/hutter14.html

[36] D. Masters and C. Luschi, “Revisiting Small Batch Training for Deep Neural Networks,” Apr. 20, 2018, *arXiv*: arXiv:1804.07612. doi: 10.48550/arXiv.1804.07612.