Advanced Deep Learning Systems Report

Ajanthan Kanagasabapathy, Mengjie Zhang, Tom Zhou, William Ho

CID: 02017931, 02041373, 02021540, 02030870

Link to GitHub repository

1 Introduction

Knowledge distillation (KD) is a model compression technique that tries to transfer knowledge from a well-performing model (referred to as the teacher model) to a smaller, untrained model (referred to as the student model) through the student learning the teacher model's internal hidden representations and outputs. This smaller model will learn a more condensed representation of the teacher model so that its performance will still be close to the teacher model, while being an overall smaller model with fewer parameters. This optimization approach can improve inference speed and efficiency, and is especially useful when trying to deploy a well-performed model in a computationally limited environment.

For our project, we have chosen to optimize several language models with our KD pipeline. The performance of the student models will be evaluated against a collection of sequence classification tasks as benchmarks. The idea of our project stemmed from a research paper that introduced a novel way of performing KD with the aid of neural architecture search (NAS) with reinforcement learning (RL).[1] In order to discover the best student model architectures, we have designed an automated optimization process for KD. This report details the design choices and experiments, including their respective outcome and trade-offs, as well as the characteristics of our optimization pipeline.

2 Experiment design

2.1 General aim

The purpose of our project is to design an automated pipeline that performs KD using consumer-grade hardware such as the Nvidia RTX series. In the KD pipeline, we created the search space for the student model configuration based on the typical teacher model architectures. The KD process uses masked language modelling as training to make the training process model-agnostic. NAS was done with the generated search space to find the best performing configuration of the student models.

The teacher models we will be using come from the transformer library. The library offers a great extent of flexibility over the manipulation and construction of different models, as all the models from Hugging Face can be easily integrated with PyTorch's optimization tools. The model sizes of our choice will not exceed 110 megabytes, which enables our pipeline to be deployed on most computer hardware. Moreover, this allows us to keep up with the performance baseline obtained in the paper whilst using much less memory storage. The details of the model selections are documented in the following subsections.

2.2 Evaluation process

To set our goals, we first selected a variety of language tasks that will demonstrate the effectiveness of our optimization pipeline. Then we selected datasets that were suitable for training for KD. And finally, we chose from the Hugging Face library a range of language models that has good performance at completing those language tasks while being able to be trained in a reasonable time to be used as teacher models.

In the next stage, we used the KD pipeline to select the best student models from each teacher model. To do this, we use either the Optuna NAS pipeline or a pipeline based on reinforcement learning to evaluate the performance of each model and their candidate architectures using a mini-KD training process. The mini-KD uses only a subset of the original dataset, which makes it more time efficient in selecting the best performing student models. To maximize the expected performance of the trained models, we only use the global best student model from all the Optuna trials or RL episodes of a training session. We performed parameter tuning for each training session to determine the optimal number of episodes, number of candidates, number of mini-KD epochs, and pool size.

Finally, we evaluate the performance of the best student models from each teacher model using an evaluation function. To compare the models optimized using our KD pipeline with models trained using traditional methods, we first evaluate the performance of all the teacher models that were used for KD as a baseline. Then, we evaluate the performance of student models and compare their performance with the corresponding teacher models to evaluate how well KD is able to train the student models and analyze any discrepancies in performance scores.

2.3 Dataset choices

According to the referenced paper, the models are distilled using a subset of 7M sentences from the multilingual CC100 dataset [2], [3]. This data set contains sentences from more than 100 different languages. However, due to limited computational resources and time constraints, we initially chose to use a subset of 10,000 English sentences from the CC100 dataset - `xu-song/cc100-samples`.

Within the dataset, we chose to use English only, as multilingual models such as XLM-RoBERTa were bigger than English-only models like BERT, both training and evaluation of multilingual models were more challenging.

Later in the project, once the KD pipeline had been implemented and tested, we opted to use a subset of 100,000 lines from the same corpus used to pre-train the original BERT model - a combination of English Wikipedia and BookCorpus data [4]. This dataset is considerably larger than the previously used CC100 subset, leading to an improved performance of the student models and yielding more representative results. Although this change increased training times, the benefits in model performance justified the added computational cost.

2.4 Model choices

Several language models were selected as candidate teacher models for the KD pipeline. RoBERTa-base was included because of its architectural similarity to XLM-RoBERTa (model used in [1]), but without the added multilingual capabilities - which were not relevant to this project, as training was conducted exclusively on English data. Alongside RoBERTa, the base versions of BERT and ALBERT were also selected, given their strong performance and established reputations.

BERT, in particular, was a valuable choice as a teacher model due to the availability of widely recognized, high-performing distilled variants, including those released by Google [5], as well as models such as TinyBERT [6] and DistilBERT [7]. This allowed for direct comparisons between student models produced through our KD pipeline and existing distilled models, enabling an evaluation of both architectural differences and the overall effectiveness of our approach.

2.5 NAS strategies

During the NAS search where we attempt to find optimal student architectures by performing mini-KD, we choose between using an RL based LSTM controller as described in the paper, or using Optuna with the objective of maximizing the reward of mini-KD students.

2.6 Performance metrics

2.6.1 Loss and latency

For every KD episode, we select a few candidates for KD training. For each candidate, we trained a student candidate with a mini-KD trainer for a few epochs. The latency is evaluated at the end of the training. Finally, the reward for that candidate is calculated using the latency and the loss in the last epoch. This allows us to compare each candidate's performance for each model category.

2.7 Evaluation task selection

We evaluate the performance using the General Language Understanding Evaluation (GLUE) benchmark [8]. This benchmark contains a series of tasks with datasets that are used to evaluate the linguistic performance of models. The GLUE tasks are model-agnostic, so different kinds of language model can go through our KD pipeline and be evaluated against the same metric. We set up the evaluation method for the GLUE tasks to follow closely to the ones used in the original evaluation for teacher models, using the epochs, learning rates, and batch sizes specified in their documentation. This allows the scores of the student models to be compared fairly with those of the teacher models.

Different tasks in the GLUE benchmark have different difficulties both in terms of complexity and the time required to complete the evaluation. For example, CoLA is a syntactic metric that doesn't transfer well from teacher to student, as seen from many of the results. Tasks like MNLI and QNLI have large datasets that could take hours to evaluate. Therefore, datapoints for those tasks could be limited.

3 Results analysis

3.1 KD training performance overview

The performance of our KD training can be summarized in a plot of inference time vs. average GLUE benchmark scores. With KD, the student model should have lower latency/inference time while still retaining a high benchmark score. We compared our NAS optimized models and baseline models (models with same architecture as existing distilled models, but trained under our KD pipeline) against results from the teacher models and existing distilled models.

The results show that our KD pipeline is capable of producing student models that reduce the inference time by more than 3x. However, there's still a performance gap compared to existing distilled models such as DistillBERT and TinyBERT. This is likely due to our dataset being smaller than the ones used for those models, and some distilled models have task-specific training and optimizations that would further improve performance compared to our generalized method.

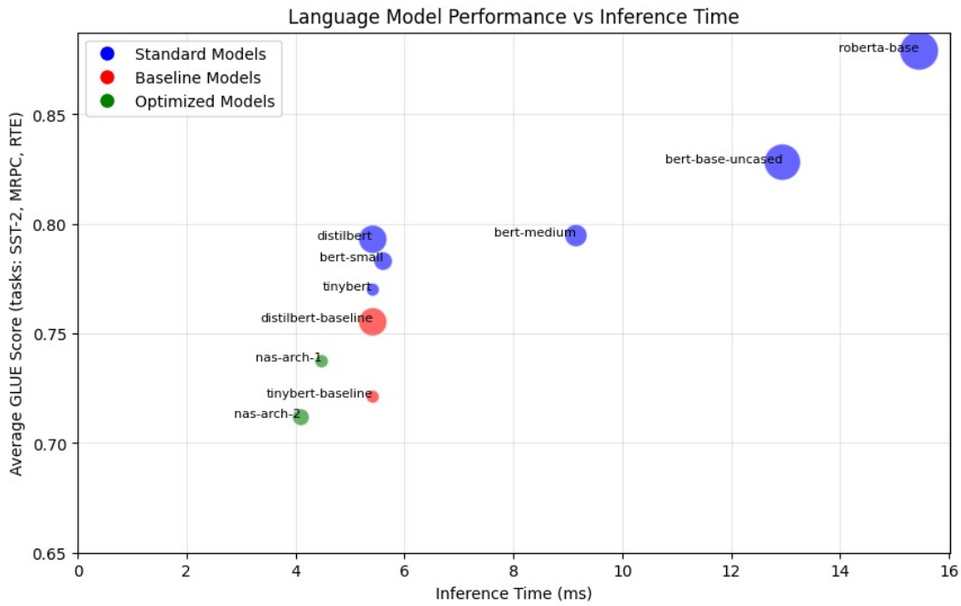


Figure 1: Performance summary for the chosen models, bubble size indicates total parameter size

3.2 LSTM controller vs Optuna

To compare the NAS search processes, we run the workflows with the same mini-KD (4 epochs, 30% proxy dataset), and equivalent search parameters of 100 Optuna trials and LSTM with 15 episodes of 8 candidates, so that the total number of training epochs are similar, at around 400.

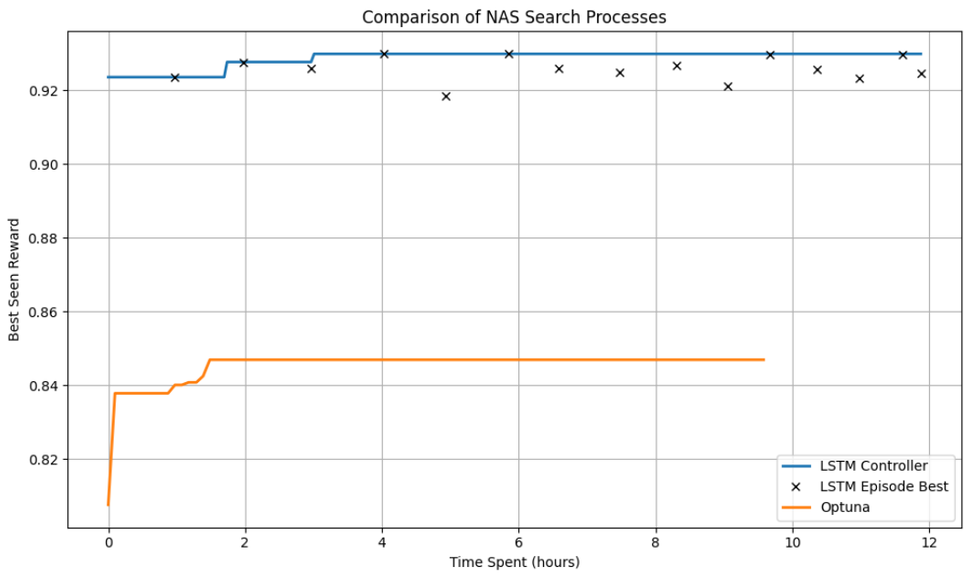
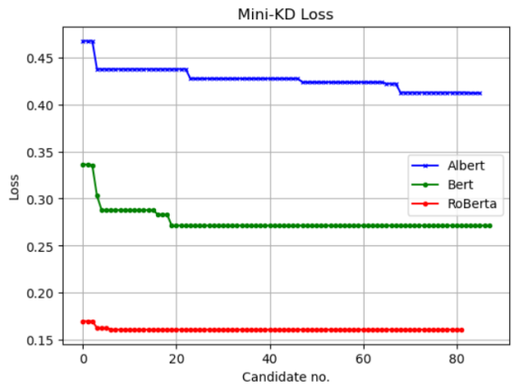


Figure 2: Comparison of Optuna and LSTM controller NAS workflow using BERT-base teacher and previous defined dataset. Running on a single L40s GPU.

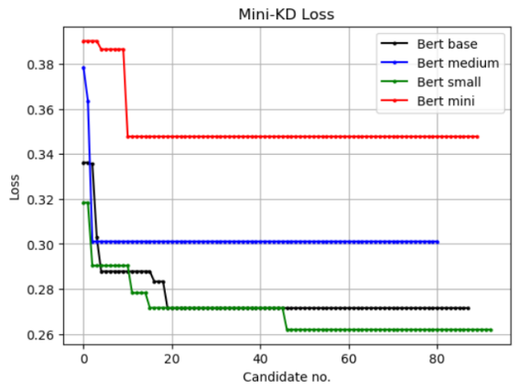
We observe that the RL-based LSTM search flow identified higher-reward student architectures, likely due to its increased randomness in sampling the search space. Despite various optimizations, such as caching the results of previously evaluated student candidates to avoid redundant training, the LSTM based search requires more time to reach superior results. This is primarily because the early stages of the search were dominated by randomly generated candidates, while the controller was still learning from each episode's outcomes. It is reasonable to expect that this approach would yield even greater benefits when applied to larger and more complex search spaces.

3.3 KD loss performance

In this section, we present the results for distillation performance for the ALBERT, BERT and RoBERTa models. For each NAS episode, we trained using the mini-KD function a sampled candidate for a few epochs. In the following plots, we use the mini-KD loss of the last epoch as the candidate loss. The plot shows how NAS efficiently selects the better candidate models using the past candidate model and the best candidate models encountered to decrease training loss over the optimization process.



(a) Average distillation loss vs candidate samples.



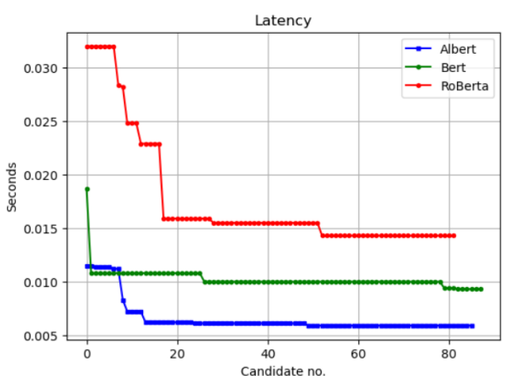
(b) Average mini KD loss per model vs candidate samples.

Figure 3: We plot the minimum achieves loss across all the KD training epochs. This smooths out the noisy training curve. We can see that in general, the RoBerta model achieved the lowest loss. Loss stops decreasing from up to 30 candidate samples.

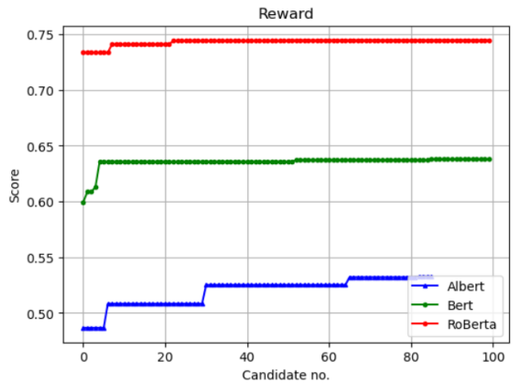
3.4 Time and performance comparison

To avoid large latency overhead, we restricted the student model parameters in the search space to be not greater than it's teacher model's parameters to balance between performance and throughput. Moreover, the reward function applies weighting to both latency and loss, meaning the models were distilled while satisfying both timing and loss constraints. From the plots, our KD pipeline is effective in training the student models as the MSE for all three models decreased to around ∼0.015 after 20 RL episodes. It can also be observed that the KD pipeline was successful in discovering the optimal architectures to achieve the maximum possible reward. In terms of performance trade-off, BERT achieves the highest reward but also has the highest latency. To avoid overfitting, we fine-tuned the parameter to stop training at the point where the reward stops increasing.

3.4.1 Hugging Face variant base models (ALBERT, BERT and RoBERTa)



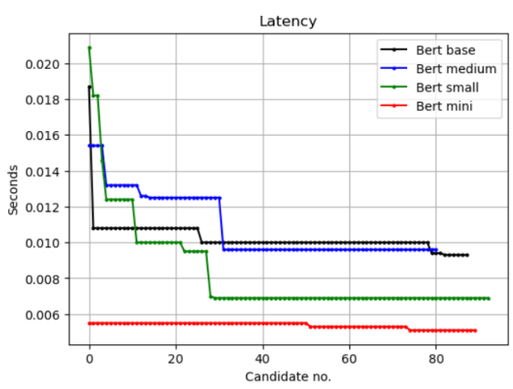
(a) Average distillation loss vs candidate samples.



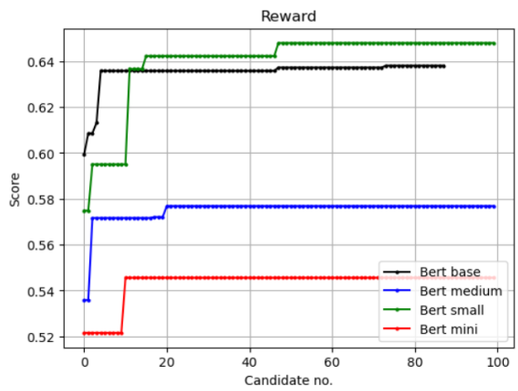
(b) Average mini KD loss per model vs candidate no.

Figure 4: Latency and reward for all the sampled candidates.

3.4.2 Bert family models (BERT medium, small and mini)



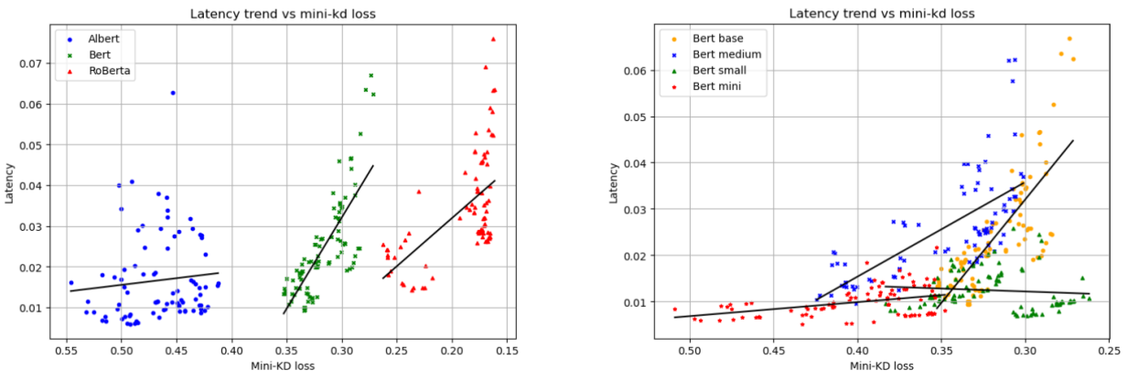
(a) Average distillation loss vs candidate samples.



(b) Average mini KD loss per model vs candidate samples.

Figure 5: We performed the same testing to the BERT family models

3.5 Loss-latency trade-off



(a) Average distillation loss vs mini-KD epochs

(b) Average mini KD loss per model vs mini-KD epochs

Figure 6: Training loss vs model latency for ALBERT, BERT and RoBERTa (left) and for BERT, BERT medium, BERT small and BERT mini (right). Interestingly, latency increases as loss decreases for all the models but BERT small. Among them, the ALBERT model has the largest latency. For the same latency, the BERT and RoBERTa models achieved a much smaller loss compared to the ALBERT model. This suggests that the BERT and RoBERTa models are likely to be more suitable for the KD task and have a higher probability of performing better in the GLUE score evaluation.

3.6 NAS student architectures

After a complete NAS run, with default parameters, we obtain the top 3 reward architectures recommended by the process, ($\text{NAS}_{\text{Arch1}}$, $\text{NAS}_{\text{Arch2}}$, $\text{NAS}_{\text{Arch3}}$), with different latency budgets.

Model	Distillation Loss↓	CPU Latency Speed Up↑	Reward↑
$\text{Random}_{\text{Seed1}}$	0.284	5.8x	0.665
$\text{Random}_{\text{Seed2}}$	0.236	2.0x	0.659
$\text{Random}_{\text{Seed3}}$	0.289	4.6x	0.654
$\text{NAS}_{\text{Arch1}}$	0.217	5.2x	0.923
$\text{NAS}_{\text{Arch2}}$	0.188	4.6x	0.904
$\text{NAS}_{\text{Arch3}}$	0.159	2.9x	0.864

Table 1: Random Sampling vs NAS. Mini-KD metrics shown for 3 random architectures and NAS recommended.

We clearly see that the final recommended architectures for NAS have greater rewards, and so the NAS process is working correctly.

3.7 GLUE score performance

3.7.1 BERT-base distilled student models

Model	**#Params**	**Speedup**	**SST-2** (67k)	**CoLA** (8.5k)	**STS-B** (5.7k)	**MRPC** (3.7k)	**RTE** (2.5k)
BERT-base	109M	1.0×	94.3	52.6	83.9	87.3	67.3
$\text{NAS}_{\text{Arch1}}$	15.6M	6.3×	86.1	14.5	17.5	70.1	53.0
$\text{NAS}_{\text{Arch2}}$	20.9M	5.4×	87.6	20.8	14.5	78.5	55.2
$\text{NAS}_{\text{Arch3}}$	52.8M	2.9×	89.2	27.4	18.9	82.4	57.3
6L-768H_4	52.2M	2.9×	88.4	29.5	21.3	81.1	57.0
DistilBERT_4	52.2M	3.0×	91.4	32.8	76.1	82.4	54.1
TinyBERT_4	14.5M	9.4×	91.9	35.2	81.5	85.4	62.1
$\text{Random}_{\text{Seed1}}$	20.4M	5.8×	81.7	0.0	6.7	81.2	51.9
$\text{Random}_{\text{Seed2}}$	52.8M	2.0×	80.7	6.9	12.2	77.7	53.1

Table 2: Comparison of distilled students. 6L-768H is the DistilBERT baseline.

3.7.2 Hugging Face variant base models (ALBERT, BERT and RoBERTa)

model	cola	mnli	mrpc	qnli	qqp	rte	sst2	wnli
$ALBERT_T$	61.22	35.60	65.71	60.00	63.57	70.00	80.00	50.00
$BERT_T$	60.49	81.95	89.45	87.50	83.00	68.00	91.00	56.34
$RoBERTa_T$	61.13	35.6	82.86	30.00	77.50	30.00	100.0	50.00

Table 3: The teacher models scores are used as the baseline. Note that the teacher models are trained using a smaller dataset therefore for a small number of tasks it's performing less well than the student model.

GLUE score of student models

Our NAS function already trains the models and there is an option to train the models further after NAS. We compare the performance of student models obtained with and without post-distillation training (PDT). The first row of each model is the model's performance score and the second row is the percentage of score achieved by the student model in comparison with the teacher model.

model	cola	mnli	mrpc	qnli	qqp	rte	sst2	wnli
$ALBERT_S$	9.70	32.95	70.69	**59.58**	78.19	51.26	80.62	53.52
% of teacher	15.84%	92.56%	**107.58%**	99.30%	**123.0%**	73.23%	**100.78%**	107.04%
$BERT_S$	**13.66**	**63.55**	72.25	58.70	**79.2**	**54.51**	**81.08**	54.93
% of teacher	**22.58%**	77.55%	64.63%	67.09%	95.42%	80.16%	89.10%	97.50%
$RoBERTa_S$	0.00	45.16	**72.75**	56.86	68.99	48.01	78.90	**56.34**
% of teacher	0%	**126.85%**	87.80%	**189.53%**	89.02%	**160.03%**	78.90%	**112.68%**

Table 4: GLUE scores for the student models without PDT

model	cola	mnli	mrpc	qnli	qqp	rte	sst2	wnli
$ALBERT_S$	15.66	69.17	68.70	60.21	79.09	48.74	82.45	52.11
% of teacher	25.58%	**194.30%**	**104.55%**	100.35%	**124.41%**	69.63%	**103.06%**	**104.22%**
$BERT_S$	**15.94**	**72.63**	68.93	**62.79**	**83.85**	**53.43**	**85.78**	**53.52**
% of teacher	**26.35%**	59.52%	77.06%	71.76%	101.02%	78.57%	94.26%	94.99%
$RoBERTa_S$	15.35	67.81	**69.87**	60.86	81.32	51.26	84.17	30.99
% of teacher	25.11%	190.48%	84.32%	**202.87%**	104.93%	**170.87%**	84.17%	61.98%

Table 5: GLUE scores for models with PDT.

It can be observed that the distilled student models was able to achieve 70-100% of its teacher model's performance. It can also be seen that PDT is effective in improving the GLUE scores.

3.7.3 BERT family models (BERT medium, small and mini)

GLUE score of teacher models

model	cola	mnli	mrpc	qnli	qqp	rte	sst2	wnli
$BERTMed_T$	42.44	80.00	84.11	86.16	83.40	63.18	88.07	56.34
$BERTS_T$	33.32	73.05	84.39	84.43	84.52	74.07	89.66	14.29
$BERTMini_T$	0.00	74.80	71.80	84.10	86.20	57.90	85.9	62.30

Table 6: GLUE score for teacher models. The table documents the performance of the teacher models as a baseline for comparison.

GLUE score of student models without post-distillation training (PDT)

model	cola	mnli	mrpc	qnli	qqp	rte	sst2	wnli
$BERTMed_S$	8.83	53.96	69.26	**58.78**	70.28	48.01	79.24	**53.52**
% of teacher	20.81%	67.45%	82.34%	68.22%	84.27	%75.99%	89.97%	94.99%
$BERTS_S$	10.26	61.71	**72.78**	58.54	**75.87**	51.26	78.44	47.89
% of teacher	**30.79%**	**84.48%**	86.24%	**69.34%**	**89.77%**	69.20%	87.49%	**335.13%**
$BERTMini_S$	**11.9**	**62.01**	67.89	58.14	75.41	**54.15**	**79.36**	43.66
% of teacher	inf%	82.90%	**94.55%**	69.13%	87.48%	**93.52%**	**92.39%**	70.08%

GLUE score of student models with post-distillation training

model	cola	mnli	mrpc	qnli	qqp	rte	sst2	wnli
$BERTMed_S$	**14.91**	**67.34**	70.94	**61.23**	**81.12**	**52.35**	**83.6**	**39.44**
% of teacher	35.13%	84.18%	84.34%	71.07%	**97.27%**	82.86%	**94.92%**	70%
$BERTS_S$	12.25	64.15	69.73	60.31	78.93	51.26	80.96	35.21
% of teacher	**36.76%**	**87.82%**	82.63%	71.43%	93.39%	69.2%	90.3%	**246.4%**
$BERTMini_S$	11.79	64.26	**73.46**	60.79	78.52	51.26	81.08	22.54
% of teacher	inf%	85.91%	**102.31%**	**72.28%**	91.09%	**88.53%**	94.39%	36.18%

Table 7: In general, the smaller variants of BERT achieves a lower but acceptable score for the language tasks. Despite this, in some tasks they are able to achieve a GLUE score of > 0.7. Moreover, these variants uses much less memory and take less time to train. Hence they can be serve as good alternatives when the hardware budget is limited. In some cases, the student models are performing better than the teacher models. This serves as evidence that our model compression pipeline has improved the models' generalization.

4 Conclusion

In conclusion, our knowledge distillation optimization pipeline is effective in achieving distillation using its improved neural architecture search technique. It has been demonstrated to produce competitive results for a wide range of model choices and in optimizing the best models for the majority of the language tasks. It is able to accurately select the top performing models and distill them for optimal performance. The pipeline is fully automated and can be run on any consumer hardware with a training time of less than 3 hours, making it a powerful optimization tool for knowledge distillation tasks.

5 References

References

[1] A. Trivedi, T. Udagawa, M. Merler, R. Panda, Y. El-Kurdi, and B. Bhattacharjee, *Neural architecture search for effective teacher-student knowledge transfer in language models*, 2023. arXiv: 2303.09639 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2303.09639.

[2] A. Conneau, K. Khandelwal, N. Goyal, *et al.*, *Unsupervised cross-lingual representation learning at scale*, 2020. arXiv: 1911.02116 [cs.CL]. [Online]. Available: https://arxiv.org/abs/1911.02116.

[3] G. Wenzek, M.-A. Lachaux, A. Conneau, *et al.*, *Ccnet: Extracting high quality monolingual datasets from web crawl data*, 2019. arXiv: 1911.00359 [cs.CL]. [Online]. Available: https://arxiv.org/abs/1911.00359.

[4] Y. Zhu, R. Kiros, R. Zemel, *et al.*, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 19–27.

[5] I. Turc, M.-W. Chang, K. Lee, and K. Toutanova, "Well-read students learn better: On the importance of pre-training compact models," *arXiv preprint arXiv:1908.08962*, 2019.

[6] X. Jiao, Y. Yin, L. Shang, *et al.*, "TinyBERT: Distilling BERT for natural language understanding," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, T. Cohn, Y. He, and Y. Liu, Eds., Online: Association for Computational Linguistics, Nov. 2020, pp. 4163–4174. DOI: 10.18653/v1/2020.findings-emnlp.372. [Online]. Available: https://aclanthology.org/2020.findings-emnlp.372/.

[7] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: Smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[8] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, *Glue: A multi-task benchmark and analysis platform for natural language understanding*, 2019. arXiv: 1804.07461 [cs.CL]. [Online]. Available: https://arxiv.org/abs/1804.07461.