Best Way to Squeeze: A Comparison of Model Compression Techniques in Natural Language Processing

João Carlos Lopes Antunes
joao.carlos.antunes@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

October 2021

Abstract

Current research in natural language processing shows a growing number of models extensively trained with large computational budgets, pursuing the goal of outperforming other state-of-the-art models in test-set performance scores. However, with such computationally demanding requirements, training these models often requires several hours, days, or worse. Furthermore, the sheer complexity and resources required to evaluate such models prevents them from being deployed in devices with strict resource and response latency limitations. We focus our attention on the effort required to train and evaluate such high performing models. We analyze several of the latest proposed models, gather information about their computational budgets, datasets used and model complexity, and apply state-of-the-art model compression techniques to create compact versions of those models. We then evaluate whether the trade-off between model performance and budget is worthwhile, in terms of evaluation efficiency, model simplicity and environmental footprint.

Keywords: machine learning, neural networks, natural language processing, model compression, model evaluation, environmental footprint

1. Introduction

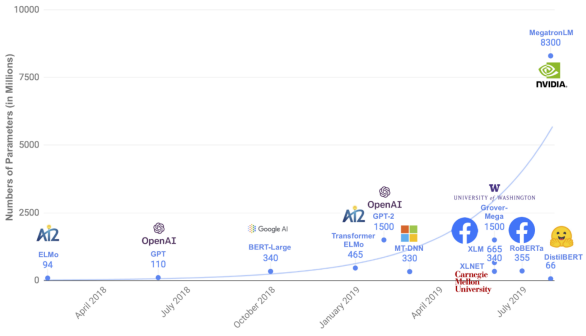


Figure 1: Number of parameters of some of the larger NLP models released between 2018 and 2019. Source: [17]

The field of machine learning has brought a revolution in performing human-like tasks through computing, especially with the research and development in neural network technology. This is especially true for recent models which are overwhelmingly based on the Transformer [25], a neural network architecture that has revolutionized Natural Language Processing (NLP) with its well-performing generalization to several different NLP tasks. This architecture favors bulky models containing a big number of parameters – model weights and coefficients that change and learn with the training data – which require large amounts of computational resources to both train and run; such models include *BERT* [3] (up to 340 million parameters), *GPT-2* [16] (up to 1.5 billion[1] parameters), *Megatron-LM* [19] (8.3 billion parameters), *Turing-NLG* [13] (17 billion parameters), *GPT-3* [1] (a staggering 175 billion parameters), among many others. As mentioned by Microsoft in their *Turing-NLG* blog post [13], *"larger natural language models lead to better results"*.

The issue with such high requirements for high-performing language models becomes more apparent once the scale of the hardware is taken in consideration: fitting a model with a billion parameters in a single advanced data-center GPU is impossible. *Model parallelism* circumvents this issue by partitioning large models over several GPUs; this same parallelism technique was used to train the 8.3 billion parameters of the *Megatron-LM* model over 512 GPUs [19]. However, even with parallelism, the *memory consumption* remains a major problem for less advanced hardware used outside of data-centers, especially if we consider devices with low-end hardware (usually for portability, cost, and

[1] Following the U.S. convention of a billion (10^9).

Benchmark	Error Rate	Polynomial Cost		Exponential Cost	
		CO_2	$	CO_2	$
SQuAD 1.1	**4,621%**	10^4	10^5	10^5	10^5
	2%	10^7	10^7	10^{15}	10^{15}
	1%	10^{10}	10^{10}	10^{32}	10^{32}
CoLLN 2003	**6,5%**	10^5	10^5	10^5	10^5
	2%	10^{35}	10^{35}	10^{73}	10^{74}
	1%	10^{53}	10^{53}	10^{173}	10^{173}
WMT 2014 (EN-FR)	**54,4%**	10^4	10^4	10^4	10^4
	30%	10^{15}	10^{15}	10^{22}	10^{22}
	10%	10^{35}	10^{35}	10^{99}	10^{100}

Table 1: Implications of achieving performance benchmarks on carbon emissions (lbs) and economic costs ($USD) from deep learning in NLP (namely, question answering, named entity recognition and machine translation) based on projections from polynomial and exponential models (the current state-of-the-art score for each benchmark is displayed in **bold text**). The carbon emissions and economic costs of computing power usage are calculated using the conversions from [20]. Adapted from source: [22]

power consumption reasons) such as smartphones.

Another concern with executing such large models is the *time* they take to infer a result – the latency of the model – which is heavily dependent on the environment where the model is deployed. Benchmarking tests [10] have shown that the usage of a GPU provides substantially lower latency than the usage of a CPU for model inference, which impacts time-sensitive tasks such as autocompleting word typing. For devices with hardware constraints, such as smartphones, this effectively leads to an undesirable trade-off between model accuracy and response latency.

Additionally, *monetary cost* and *environmental footprint* are often overlooked yet important costs. A recent review [18] estimates that training a model with 1.5 billion parameters costs around $80k[2], scaling up to $1.6m with proper hyperparameter tuning; these figures act as a paywall for developing new models, as not many research labs can afford costs of this magnitude. However, we are reaching a point where not even those who can make such high monetary investments are willing to do it: the authors of the *GPT-3* model found a mistake when implementing the system, but decided against fixing it since *"due to the cost of training it was not feasible to retrain the model"* [1]. There is also a considerable environmental cost in training models: a recent study [20] reported that a fully trained *BERT* model emits as much CO_2 as a trans-American flight from New York to San Francisco. Not only that, but future models may bring much more drastic effects to the environment; using projections based on current model endeavors necessary to hit state-of-the-art benchmark results (shown in Table 1), we can expect both the environmental and economical cost of training a deep learning model for NLP usage to be higher in the order of a few magnitudes at best, or hundreds of magnitudes at worst, not to mention the necessary computational requirements to achieve such high energy expenditure.

As an effort to tackle the issues explained before, we propose a comparison of three different compression techniques – *knowledge distillation*, *weight pruning* and *model quantization* – when applied on three different NLP tasks – *sentiment analysis*, *named entity recognition* and *dialog-driven sentence generation* – and compare the resulting compressed models, determining whether the trade-off between performance and resource usage is worth it. We also propose combining two of the compression techniques, quantization and pruning, to compare the resulting compressed language models against the separately compressed models for both techniques. We propose a well-detailed evaluation of every model: we display the expected validation performance for all evaluation metrics, the final size of the model, average training and inference time, computing infrastructure used, model hyperparameters and dataset splits, as well as a link to model implementation code. We verify the inversely proportional correlation between execution latency and the size and complexity of the compressed model. Furthermore, we propose a brief comparison between uncompressed and compressed models when running in low-end hardware by testing the performance of quantized models in a *Raspberry Pi*. Additionally, we compare the training process in terms of training time and power usage, as well as estimate CO_2 emissions for the same model to be trained in a data center, to understand whether the reduction in model size and complexity translates to a decrease in the computational budget required to further train and fine-tune the model, as well as a decrease in the environmental footprint of the overall training process.

2. Related Work

2.1. Better model evaluation and result reporting

An important but often overlooked part of comparing the performance of language models is the way the experiment details and consequent results are reported. Papers often only report their best accuracy values, but omit important details such as the time spent training the model, or any finely tuned

[2]Costs denoted in U.S. dollars

hyperparameter values. Not only that, but displaying the best accuracy values does not confidently portray the performance of a model since accuracy values can often vary depending on several testing factors, such as random weight initialization. *Expected validation performance* [4] is calculated via the expected accuracy value obtained from validation data, given a resource budget to train and evaluate n models. This expected value takes in consideration every accuracy value obtained during model development and testing, instead of simply reflecting the best observed value after n evaluations, thus obtaining a more accurate reflection of the overall performance of the model. Additionally, by displaying the performance of the model as a function of computational budget, we can estimate the resources required to attain a certain expected performance value; this can be useful in scenarios where the budget provided for training a model is limited, or if there is a minimum accuracy value required: the estimated budget for a given performance value can be calculated, and unnecessary budget expenses are avoided by only spending resources below the estimated value.

2.2. Compressing BERT

For the past couple of years, NLP research has thoroughly taken advantage of model development based on large pre-trained models, refining and fine-tuning them into performing different tasks, with one of the most widely used models as the starting point being *BERT*. Despite being a very versatile and effective model, *BERT* still has a large memory footprint and requires heavy computing during inference; therefore, model compression has been an especially important field of study for any *BERT*-based models.

Q8BERT [26] is a quantized version of *BERT* that achieves a 4× smaller memory footprint while only losing less than 1% accuracy relative to the original model; this was achieved by quantizing all weights within the Embedding and Fully Connected layers – which contain over 99% of the weights present in *BERT* – to 8bit values. The weight quantization was done during training, using a technique called *quantization-aware training*: during model fine-tuning, fake quantization [9] is used to simulate the value errors obtained when rounding down floating-point numbers. These values are then back-propagated to the model, which ends up learning how to overcome quantization errors.

DistilBERT [17] was developed by teaching a smaller version of *BERT*, reducing the amount of layers from the regular 12 to just 6, using the available pre-trained one as a teacher. It manages to retain 97% of the accuracy from the original model, while being 40% smaller and 60% faster. Due to the common dimensionality between the teacher and the student models, the small model was able to be initialized by directly taking layers out of the teacher model. The authors also refer to the orthogonality of other model compression techniques relative to knowledge distillation; with additional pruning and quantization, *DistilBERT* could be compacted even further, but not without some expected losses in performance. Additionally, the authors studied the performance of the compact model on mobile devices; two smartphone applications were built for question answering, one using $BERT_{BASE}$ and the other *DistilBERT*, and both were deployed on an *iPhone 7 Plus*. The average inference time was measured, with *DistilBERT* being 71% faster than the base model.

2.3. Lottery Ticket Hypothesis in NLP

In recent years the NLP community focused on building larger Transformer models, as stated before. Concurrently, the computer vision community of researchers explored the Lottery Ticket Hypothesis [6], which states that *"dense, randomly-initialized, feed-forward networks contain subnetworks (winning tickets) that – when trained in isolation – reach test accuracy comparable to the original network in a similar number of iterations"*; simply put, conventional pruning techniques can unveil smaller neural networks (mentioned as subnetworks) which can be trained to reach performances similar to the parent network.

Based on this formulation, several concurrent studies [2, 7, 15] focused on applying the same line of thought to *BERT*. The results demonstrate that the Lottery Ticket Hypothesis holds true for NLP, and valuable conclusions arrived from this research:

- Matching "winning ticket" subnetworks can be found between varying values of sparsity, from as low as 40% to as high as 90%. This means that, for specific tasks, a model consisting of roughly one tenth of the pre-trained *BERT* model can hit similar performances;
- These subnetworks can be found with no extra training required, by directly pruning the pre-trained *BERT* model with no need for any fine-tuning beforehand. The pruned model can still reach a similar performance to the full model;
- For most models fine-tuned to accomplish downstream NLP tasks, the subnetworks found appear to be specific for that specific task, and are unable to be transferred to other tasks.

One of the conclusions opened up new possibilities: "winning tickets" found at 70% sparsity using the task originally used for pre-training *BERT* (masked language modeling) were shown to be universal, showing that learning can be transferred

Consumption	CO_2e (kg)
Air travel, 1 person, NY↔SF	900
Human life, avg, 1 year	5,000
American life, avg, 1 year	16,400
Car, avg incl. fuel, 1 lifetime	57,153
Training one model (GPU)	
NLP pipeline (parsing, SRL)	18
w/ tuning & experiments	35,592
Transformer (big)	87
w/ neural arch. search	284,019

Table 2: Common CO_2 emissions from regular human activity, compared against the training of common NLP models. Source: [20] (converted from lbs to kg)

to other downstream tasks while maintaining accuracy. The resulting implication of a pre-trained *BERT* model properly pruned down to nearly a third of its size still being able to accomplish similar accuracy values when fine-tuned to a downstream NLP task is an incredible breakthrough, especially for low-end or otherwise budget-restricted hardware. Given the positive results, proper pruning after the initial training could be seen as a second stage of the *BERT* pre-training process in the future.

2.4. Energy consumption and Carbon footprint

With larger and better performing models, comes greater resource expenditure. While most NLP models from a decade ago could be properly trained on end-user laptops and other common hardware, training a state-of-the-art model nowadays requires dedicated hardware with several GPUs or TPUs, even if the goal is just to fine-tune an already existing pre-trained model. With this in mind, and considering that properly training and validating a model requires many executions to experiment with different architectures and hyperparameter configurations, the energy consumption is an important (but often forgotten) evaluation detail when training a model. Reporting this detail would allow for cost-benefit analysis between different models, especially when the model is meant to be retrained for downstream usage, such as fine-tuning for a new NLP task.

The amount of energy consumed by a model during the entire training process is not only important in terms of monetary cost, but also due to the effect it has on the environment. The European Environment Agency has reported that, on average, for every kilowatt produced per hour, the equivalent of $275g$ of CO_2 is released to the environment as greenhouse gases [5]. However, since the most popular cloud compute service – Amazon Web Services – is hosted in the United States, it is more reasonable to consider the value reported by the U.S. Environmental Protection Agency (EPA) as the average greenhouse gas emission for model energy consumption: $947lbs$ per megawatt-hour [24], or $430g$ per kilowatt-hour. Considering that data centers spend a substantial amount of energy maintaining servers up and running, and taking in consideration the impact of greenhouse gas emission on the environment, reporting energy consumption is an ever-increasing necessity for any trained models nowadays.

3. Experiments

In this section, we describe our approach to evaluate the effects of model compression, and detail our experiments on applying the compression techniques described to different NLP tasks.

3.1. Training Setup

In order to study model compression and its effect on current language models, we picked three fundamental and widely used tasks in NLP: text classification (in particular, *sentiment analysis*), sequence labeling (specifically, *named entity recognition*) and *sentence generation*. Across every model trained, we used the PyTorch framework [14]. For the models themselves, we picked $BERT_{BASE}$ [3] as the pre-trained base model to accomplish the sentiment analysis[3] and named entity recognition[4] NLP tasks, and $GPT\text{-}2_{Small}$ [16] for conversation-driven sentence generation[5].

To obtain the baseline from which we can compare the compressed models, we will fine-tune the pre-trained model for the task at hand.

To distill knowledge, we will create a small student model using only the first k Transformer layers from the pre-trained model ($BERT_k$). Afterwards, the previously fine-tuned uncompressed model (used as baseline) will act as the teacher model and perform *patient knowledge distillation* (PKD) [21].

Following the Lottery Ticket Hypothesis shown in Section 2.3, we will be pruning the pre-trained model directly by gradually applying *iterative magnitude pruning* (IMP), and then continue training to recover pruning accuracy losses; from here, the model can be fine-tuned according to the desired task.

To perform model quantization, we will use *quantization-aware training* (QAT) (detailed in Section 2.2) during fine-tuning of the pre-trained model for the given task. Additionally, taking advan-

[3] https://github.com/Uziskull/lightning-text-classification

[4] https://github.com/Uziskull/BERT-NER

[5] https://github.com/Uziskull/lightning-convai

tage of the compatibility between both compression techniques, we will be using *quantization-aware training* on the previously pruned model during fine-tuning of the task.

3.1.1 Datasets

To train the language models fine-tuned on the sentiment analysis task, we used the *IMDB review* dataset [11] – a group of positive and negative *IMDB* movie reviews. For the named entity recognition task, we made use of the *CoNLL-2003* dataset [23] – a collection of phrases where every word has a corresponding part-of-speech tag, syntactic tag and named entity tag. To train and fine-tune the *GPT-2* based conversational model on the task of sentence generation, we used the *Persona-Chat* dataset [27], which is a crowd-sourced collection of over 160 thousand utterances between pairs of personas. Additionally, to prune both *BERT* and *GPT-2* based models, we need to pre-train the models on the NLP tasks of masked language modeling and causal language modeling, respectively; we made use of the *WikiText* corpus [12], a bundle of several million tokens extracted from verified articles on *Wikipedia*.

3.1.2 Evaluation

We followed a quantitative approach to evaluation, focusing on comparing the original models to their compressed counterparts. Following a set of guidelines for best practices [4], for every model trained and evaluated, we reported a description of the computing infrastructure used during training, the average runtime for each approach, the details of dataset splits, the corresponding validation performance for each reported test result, a link to implemented code, the response time of the model during execution, and the size of the model.

Additionally, we used heuristics related to energy consumption and environmental footprint [20] to compare the budget required to train a model against its compressed alternative. For every model trained, the average GPU power draw was obtained to calculate the power consumption of the model, which will then be used to estimate the amount of CO_2 produced (as explained in Section 2.4). While it is not the only component consuming energy, we will only be focusing on the GPU power draw as it is the main power funnel when training a model.

3.2. Testing Details

We set up a controlled testing environment by maintaining the same testing hardware and hyperparameters across models for the same tasks; all of these testing configurations are presented in Table 3. The framework used to train the models accurately took note of the evaluation details, such as the time it took to train and evaluate each model and the metrics measured during inference; the average GPU power draw for every model trained was queried from the NVIDIA System Management Interface (`nvidia-smi`[6]) in parallel to the model training itself, which gathered and registered the current power draw of the GPU being used, with an interval of 5 seconds.

	Sentiment Analysis	Named Entity Recognition	Sentence Generation
GPU	1080 Ti (11GB)	TITAN X (12GB)	1080 Ti (11GB)
Data Splits (tr./dev/test)	50/25/25	70/15/15	99/0.5/0.5
Epochs	5	5	3
Batch Size (train/test)	8/8	32/8	2/1
Learning Rate	3e-5, 1e-4, 3e-4, 5e-5		6.25e-5, 5e-5, 1e-4

Table 3: General training details for every task, including hardware and hyperparameter configurations. All settings remained the same across every model trained in said task, whether compressed or not.

Every model run was made deterministic by setting a fixed seed for randomness, which prevented variations in weight initialization and data order across runs, thus limiting the variable testing details to just the different learning rates and the different compression techniques applied. This decision meant that our fine-tuned models did not obtain the best results reported by the authors of the different models, however this was outside the scope of our work since our focus was set on the effects of compression techniques themselves.

3.3. Knowledge Distillation

To compress each task-specific model, we designated the fine-tuned baseline model as the teacher and created a smaller version of the model to be used as the student; following the authors of the paper regarding PKD [21], the network architecture of the smaller model is identical to the base model used for the task, but the number of hidden layers was cut in half and only the first six layers of pre-trained weights were used for initialization. To train the student models, we followed the PKD-Skip strategy outlined by the authors.

To transfer the knowledge from the teacher model to the target student model, we took the entire training dataset used for the student model and

[6] `https://developer.nvidia.com/nvidia-system-management-interface`

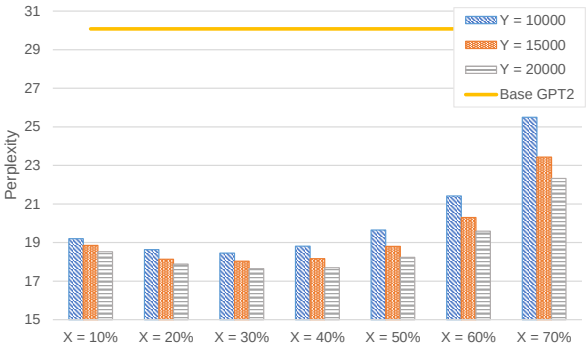


Figure 2: Experiment results for *GPT-2* models after applying iterative magnitude pruning to $X\%$ sparsity, by pruning 10% of the weights every Y training steps. The models were trained on the *WikiText-103* dataset. The perplexity of the base *GPT-2* model is displayed for comparison.

ran it through the teacher model; the resulting output from the intermediate and final layers was the knowledge set to be distilled to the student model. We were unable to perform the same teacher knowledge gathering for the sentence generation task since the resulting dataset with extra knowledge was too large to be able to be loaded during student training. To get around this issue, we ran the teacher model during the student training to get the necessary resulting values on-the-fly; while the models were successfully distilled, the resulting overloading of the GPU had some drawbacks related to training time and power spent.

3.4. Pruning

To prune every task-specific model, we first took the pre-trained base model for each task and trained it further while applying IMP [2], which prunes the attention heads of the model over several training epochs. The original code supplied by the paper authors only worked for *BERT* models, thus some changes were made to the code to also accept and prune *GPT-2*. As previously mentioned, we decided to use the *WikiText* [12] corpus for the required extra training on the pre-trained base models. There are two *WikiText* corpora available for usage, *WikiText-2* and *WikiText-103*; we conducted a small comparison between both on a pre-trained *BERT* by applying IMP to 70% sparsity with a 10% pruning step every Y iterations, varying between three different numbers of iterations. Our results show that models trained and pruned using the *WikiText-103* corpus always obtained higher accuracy and lower perplexity scores, so we decided to use that corpus for the remainder of the models trained during IMP.

To apply IMP, we followed the observed universality found for "winning tickets" (as detailed in Section 2.3) and pruned *BERT* to 70% sparsity with a 10% pruning step every Y iterations, varying the number of iterations to find which model would display better results overall. Since the research conducted on the Lottery Ticket Hypothesis was done on *BERT* based models, the universality of task transferal observed at 70% sparsity could not be directly inferred to the *GPT-2* model; as such, we decided to conduct some extra testing to observe what the ideal sparsity should be by applying IMP to the pre-trained *GPT-2* model, pruning the model to $X\%$ sparsity with a 10% pruning step every Y iterations, varying both the number of iterations and the sparsity itself (between 10% and 70%). The results (displayed in Figure 2) show a trend throughout all the different sparsity percentage models: the resulting perplexity is lowered until the model becomes 30% sparse, then it steadily begins to rise as the sparsity grows. As such, we chose 30% sparsity as a valid sparsity percentage for pruning *GPT-2*, and focused on comparing the results from the different training iterations used when performing IMP at that sparsity.

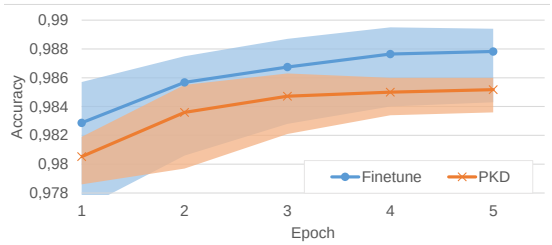
With both base models successfully pruned, we decided to use the perplexity metric to choose which model was better, since it is an evaluation method intrinsic to the model itself and does not directly evaluate the task at hand; this resulted in the two chosen models being *BERT* pruned to 70% sparsity with a 10% pruning step every 15000 iterations, and *GPT-2* pruned to 30% sparsity with a 10% pruning step every 20000 iterations. These models were then fine-tuned in the same manner as the baseline models.

3.5. Quantization

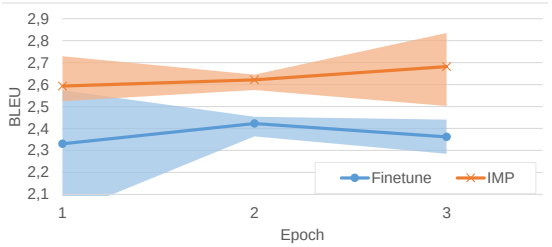
Our initial approach to apply and study quantization was to make usage of quantization-aware training; however, at the time of writing, PyTorch does not directly support applying QAT to models that require embedding layers, thus ruling out QAT on Transformer architecture models such as *BERT* and *GPT-2*. While solutions to this problem exist, all of them require fundamental changes to the architecture of the model, which in turn make QAT a non-versatile compression technique to apply and place it outside of the scope of this work; with this in mind, we decided to study dynamic quantization (DQ) instead.

To quantize every model, we applied DQ to the fine-tuned models used as baseline with no additional training time, since it is a post-training compression technique; the weights of the models were quantized to 8bit (using the *QNNPACK*[7] backend). *GPT-2* based models make use of one-dimensional convolutions over incoming data (Conv1D layers),

[7] `https://github.com/pytorch/QNNPACK`



(a) Accuracy for distilled models (PKD) fine-tuned in named entity recognition



(b) BLEU scores for pruned models (IMP) fine-tuned in sentence generation

	Model Size (MB)	Inf. Time CPU (s)	Inf. Time GPU (s)	Accuracy	F1
Baseline	440	10.000	0.003	0.938	0.937
DQ	175	4.850	0.286	0.934	0.927

(c) Experiment results for quantized models (DQ) fine-tuned in sentiment analysis

Figure 3: Excerpt of compressed model results compared against the respective baseline models.

which are not supported by the DQ backend. In order to be able to quantize *GPT-2*, every Conv1D layer had to be swapped with Linear layers after loading the model; while this change allowed for *GPT-2* models to be quantized, it had some negative effects on the performance of those models.

Additionally, to evaluate the performance of compressed models on low-end hardware, we ran both the baseline and the quantized models on a *Raspberry Pi 4* (4GB RAM) to compare the inference time between the executed models, as well as between the models ran on the *Raspberry Pi* CPU versus the ones ran on the cloud computing environment with the dedicated GPU.

3.6. Quantization + Pruning

To test the combined effects of quantization and pruning, we applied DQ to the models previously pruned via IMP and evaluated the resulting models. Due to the nature of DQ, applying it to pruned models was a simple task, although the same workaround had to be done for the pruned *GPT-2* model. Furthermore, the quantized and pruned models were also ran on the *Raspberry Pi* to compare inference times.

4. Results

In this section, we review the results of the experiments described in Section 3 and establish comparisons to the baseline models in terms of performance and resource usage metrics.

4.1. Knowledge Distillation

For the sentiment analysis and named entity recognition tasks, PKD proved to be a very effective compression technique; none of the recorded metrics for the distilled models degraded more than 1% compared to the fine-tuned baseline. The expected performance ranges of the compressed model and the baseline overlap (as displayed in Figure 3(a)), implying that a well-trained distilled model could outperform the original uncompressed model, although this verification is outside the scope of this work. The small size of the distilled models (270MB – a 39% reduction) resulted in shorter training and inference times, with the sentiment analysis task achieving a 79% latency improvement over the baseline model. However, PKD was detrimental to the sentence generation task, with a severe 16.37% drop in the BLEU score compared to the baseline. We attribute this to the architecture of the student model being too compact to properly learn from the complex *GPT-2* teacher model, and believe that distillation could be viable for this task with a different student model architecture. Additionally, due to the distilling workaround described in Section 3.3, the training time of the model was extended by 52%, consequently increasing the power spent and estimated CO_2 emissions.

4.2. Pruning

Overall, IMP showed reasonable worsening of model quality for the sentiment analysis and named entity recognition tasks, with no major difference in training or inference time, and consequential power expenditure or CO_2 emissions. On the other hand, for sentence generation, the pruned models improved considerably compared to the baseline (shown in Figure 3(b)), with a substantial 13.56% increase of the BLEU score; additionally, the training and inference times for the pruned model are 4% and 9% smaller than the ones reported for the baseline, which we attribute to the more efficient computation of the complex convolution layers present in *GPT-2* models, given that 30% of the weights are set to zero.

Ultimately, no size improvement was obtained

Task	Metrics	Baseline	PKD	IMP	DQ	DQ + IMP
Sentiment Analysis	Accuracy	0.938	0.931 (↓ 0.76%)	0.874 (↓ 6.90%)	0.934 (↓ 0.51%)	0.883 (↓ 5.88%)
	F1	0.937	0.929 (↓ 0.82%)	0.883 (↓ 5.71%)	0.927 (↓ 1.00%)	0.881 (↓ 5.92%)
	Precision	0.951	0.946 (↓ 0.56%)	0.856 (↓ 10.06%)	**0.965 (↑ 1.40%)**	0.900 (↓ 5.38%)
	Recall	0.936	0.928 (↓ 0.85%)	0.933 (↓ 0.25%)	0.908 (↓ 2.94%)	0.884 (↓ 5.51%)
Named Entity Recogn.	Accuracy	0.988	0.985 (↓ 0.27%)	0.981 (↓ 0.68%)	0.979 (↓ 0.91%)	0.956 (↓ 3.19%)
	F1	0.941	0.928 (↓ 1.38%)	0.907 (↓ 3.53%)	0.894 (↓ 5.00%)	0.775 (↓ 17.66%)
	Precision	0.936	0.922 (↓ 1.54%)	0.903 (↓ 3.54%)	0.890 (↓ 4.99%)	0.798 (↓ 14.78%)
	Recall	0.945	0.934 (↓ 1.21%)	0.912 (↓ 3.53%)	0.898 (↓ 4.96%)	0.768 (↓ 18.72%)
Sentence Generation	BLEU	2.362	1.975 (↓ 16.37%)	**2.682 (↑ 13.56%)**	2.076 (↓ 12.10%)	**2.373 (↑ 0.50%)**
	TER	1.016	1.035 (↑ 1.86%)	1.024 (↑ 0.78%)	**1.016 (↓ 0.01%)**	**1.009 (↓ 0.68%)**
	BERTScore	0.852	0.849 (↓ 0.28%)	**0.854 (↑ 0.24%)**	0.849 (↓ 0.24%)	**0.852 (↑ 0.00%)**
	Hits@1	0.817	0.024 (↓ 97.10%)	0.812 (↓ 0.60%)	0.809 (↓ 0.96%)	0.803 (↓ 1.72%)
	Hits@5	0.977	0.140 (↓ 85.67%)	0.976 (↓ 0.10%)	0.974 (↓ 0.30%)	0.973 (↓ 0.39%)
	Hits@10	0.996	0.356 (↓ 64.24%)	0.996 (↓ 0.01%)	0.995 (↓ 0.10%)	0.995 (↓ 0.07%)

Table 4: Comparison between the results of all model compression techniques, and the variance percentage compared to the baseline results. The arrows represent whether the variance is positive or negative. **Filled cells** signify the best scores between the compression techniques, and **bold cells** signify that the score obtained for that compression technique is better than or equal to the baseline score.

since no method of sparse matrix representation would allow for a smaller sized weight storage without additional detrimental computational overhead [8]; additionally, PyTorch currently offers no way of saving or loading sparse weights.

4.3. Quantization

Unlike the other compression techniques, DQ is applied post-training; therefore, we cannot analyze the expected validation performance development over the training epochs, nor is there any training time or power usage to compare against. Observing the final results, we can see minor quality degradation on the *BERT* based models, with the task of sentiment analysis (displayed in Table 3(c)) only lowering its accuracy and F1 scores by 0.51% and 1.00%, respectively. For the sentence generation task, DQ showed lowered scores overall, which we attribute in part to the architectural change required for quantizing *GPT-2* based models, previously mentioned in Section 3.5. All models showed a reduction in model size, with *BERT* based models being compressed to 60% of the original model size and *GPT-2* based models ending up with a size of 280MB (57% of the starting size); this reduction is only reflected in the size loaded in memory, however, since there is no current way of saving or directly loading a quantized model in PyTorch.

Regarding the CPU testing done on the *Raspberry Pi*, there was a noticeable improvement on inference times, with 51.50%, 59.89% and 63.10% faster performance on the tasks of sentiment analysis, named entity recognition and sentence generation, respectively. However, this inference speedup is only seen within low-end CPU-optimized hardware; GPUs manage to execute integer weight operations faster than the *Raspberry Pi* and obtain lower inference times for both quantized and baseline models, with baseline models being several orders of magnitude faster.

4.4. Quantization + Pruning

Quantizing the already pruned models showed a minor overall improvement to the pruned sentiment analysis task, a major decline in quality in the pruned named entity recognition task, and a lower-than-expected degradation on the sentence generation task, all relative to the pruned models themselves. These changes were related to both the performance of the pruned models, as well as the influence of quantization on the models. While the performance of the model on CPU was still improved (much like the former quantization results show), the effects of quantizing the pruned models were somewhat inconsistent across tasks, further confirming that quantization should only be applied on a case-by-case basis.

4.5. Environmental Footprint

Overall, compression techniques that require lower training time spend less energy doing so, resulting in lower estimated CO_2 emissions. DQ was applied post-training, thus having a null CO_2 emission value. For the compression techniques applied pre-training, PKD was the one with lower energy expenditure and consequential environmental impact, due to the smaller model being able to be trained faster; this was not the case for the sentence generation task which, due to the workaround described in Section 3.3, spent the energy required to both train the student model and execute the teacher model. IMP had an ecological footprint similar to the base-

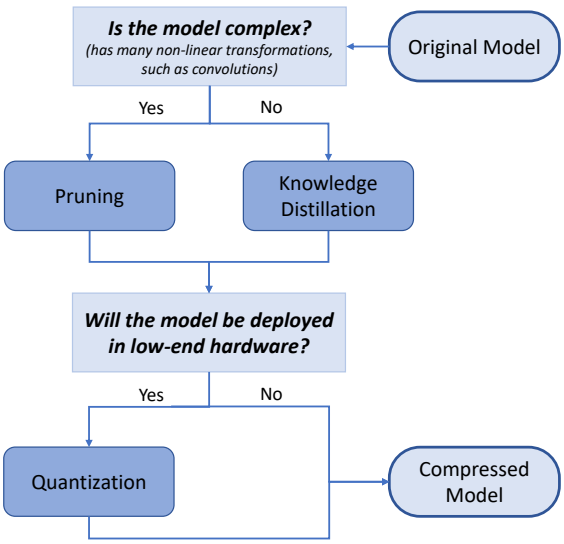


Figure 4: A simplified flowchart guide on how to best compress a NLP model, based on our findings.

line models, even showing smaller energy costs for the sentence generation task, but when accounting for the energy spent pruning the pre-trained models to the required sparsity before fine-tuning, the estimated CO_2 emissions become much higher, with increases as high as 342% (for the named entity recognition task).

5. Conclusion

In this paper, we applied several common model compression techniques to NLP tasks and compared the trade-off between performance and computational resource usage. Based on our results, we outlined a general flowchart (displayed in Figure 4) for effectively compressing a language model; this flowchart is based on preliminary conclusions we drew from our study, and more research and experiments would have to be done in order to generalize this across all language models targeted for NLP usage. Furthermore, we conclude that the usage of model compression techniques that require extra training besides fine-tuning are noticeably detrimental to the ecological footprint of the model, and compression techniques that reduce the complexity and size of the model before training (or that require no additional training, in the case of post-training compression) should always be preferred.

For future work, we believe more extensive research could be done on each of the compression techniques, such as testing several student model architectures for knowledge distillation and experimenting with different compression schemes and bit depths for quantization. We also believe model compression should be fully integrated and properly implemented in popular model frameworks like PyTorch, instead of being available only as experimental features, to promote its usage across future language models and further reduce computational costs for training such complex models.

References

[1] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language Models are Few-Shot Learners. *arXiv:2005.14165 [cs]*, July 2020. arXiv: 2005.14165.

[2] T. Chen, J. Frankle, S. Chang, S. Liu, Y. Zhang, Z. Wang, and M. Carbin. The Lottery Ticket Hypothesis for Pre-trained BERT Networks. *arXiv:2007.12223 [cs, stat]*, Oct. 2020. arXiv: 2007.12223.

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[4] J. Dodge, S. Gururangan, D. Card, R. Schwartz, and N. A. Smith. Show Your Work: Improved Reporting of Experimental Results. *arXiv:1909.03004 [cs, stat]*, Sept. 2019. arXiv: 1909.03004.

[5] EEA. Greenhouse gas emission intensity of electricity generation — European Environment Agency.

[6] J. Frankle and M. Carbin. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. *arXiv:1803.03635 [cs]*, Mar. 2019. arXiv: 1803.03635.

[7] M. A. Gordon, K. Duh, and N. Andrews. Compressing BERT: Studying the Effects of Weight Pruning on Transfer Learning. *arXiv:2002.08307 [cs]*, May 2020. arXiv: 2002.08307.

[8] T. Hoefler, D. Alistarh, T. Ben-Nun, N. Dryden, and A. Peste. Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks. *arXiv:2102.00554 [cs]*, Jan. 2021. arXiv: 2102.00554.

[9] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2704–2713, June 2018. ISSN: 2575-7075.

[10] C. Kaiser. How much difference do GPUs make in model serving?, Jan. 2020.

[11] 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*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[12] S. Merity, C. Xiong, J. Bradbury, and R. Socher. Pointer Sentinel Mixture Models. *arXiv:1609.07843 [cs]*, Sept. 2016. arXiv: 1609.07843.

[13] Microsoft. Turing-NLG: A 17-billion-parameter language model by Microsoft, Feb. 2020.

[14] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[15] S. Prasanna, A. Rogers, and A. Rumshisky. When BERT Plays the Lottery, All Tickets Are Winning. *arXiv:2005.00561 [cs]*, Oct. 2020. arXiv: 2005.00561.

[16] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language Models are Unsupervised Multitask Learners, 2019.

[17] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv:1910.01108 [cs]*, Feb. 2020. arXiv: 1910.01108.

[18] O. Sharir, B. Peleg, and Y. Shoham. The Cost of Training NLP Models: A Concise Overview. *arXiv:2004.08900 [cs]*, Apr. 2020. arXiv: 2004.08900.

[19] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro. Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism. *arXiv:1909.08053 [cs]*, Mar. 2020. arXiv: 1909.08053.

[20] E. Strubell, A. Ganesh, and A. McCallum. Energy and Policy Considerations for Deep Learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy, July 2019. Association for Computational Linguistics.

[21] S. Sun, Y. Cheng, Z. Gan, and J. Liu. Patient Knowledge Distillation for BERT Model Compression. *arXiv:1908.09355 [cs]*, Aug. 2019. arXiv: 1908.09355.

[22] N. C. Thompson, K. Greenewald, K. Lee, and G. F. Manso. The Computational Limits of Deep Learning. *arXiv:2007.05558 [cs, stat]*, July 2020. arXiv: 2007.05558.

[23] E. F. Tjong Kim Sang and F. De Meulder. Introduction to the CoNLL-2003 shared task: language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 142–147, Edmonton, Canada, May 2003. Association for Computational Linguistics.

[24] O. US EPA. Emissions & Generation Resource Integrated Database (eGRID), July 2020.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is All you Need. *Advances in Neural Information Processing Systems*, 30:5998–6008, 2017.

[26] O. Zafrir, G. Boudoukh, P. Izsak, and M. Wasserblat. Q8BERT: Quantized 8Bit BERT. *arXiv:1910.06188 [cs]*, Oct. 2019. arXiv: 1910.06188.

[27] S. Zhang, E. Dinan, J. Urbanek, A. Szlam, D. Kiela, and J. Weston. Personalizing Dialogue Agents: I have a dog, do you have pets too? *arXiv:1801.07243 [cs]*, Sept. 2018. arXiv: 1801.07243.