CSC413 Research Project

Yuqi Chen, Yifang Zhang, Michelle Lin, Yuanfan Chen

April 2024

1 Introduction

Nowadays, there are various machine learning and deep neural network models being developed every day and many of them are being used in the industry. Thus, we should not only consider accuracy, but also efficiency, both computationally and storage wise. Fine-tuning a large model is computational expensive. As claimed by (Liu et al., 2024), fine tuning does not add much information to these huge models. Furthermore, fine-tuned models are expensive to store and run due to the large number of parameters involved in such big models. Hence, an efficient process of fine-tuning and inference is needed. This is where the "compressed" parameters come in. Some methods can be used to reduce the size of parameters of the fine-tuned models. A problem we need to think about is how can we reduce the size without hurting the performance of the model. As proposed by (Liu et al., 2024), they use a method called BitDelta to decompose the weights of the fine-tuned model. However, in their code, the reduction is restricted to with a causal language modeling head and cannot be used in other models. Hence, it is worth extending the idea to other models. Also, in their code, they only compress the linear layers in the transformer block, and we want to extend that to the full model. The original code can only be used for CausalLM from hugging face and only compressed projection in MLP layers and attention layers on Linux. We aim to build a pipeline that is able to compress even more on a larger variety of models on all of Linux, mac os, and Windows. Our pipeline takes in the pre-trained model, fine-tuned model and dataset to finetune on. Then, we use compression techniques to compress the fine-tuned model, and finally return the compressed parameters, i.e. bit data, to users. In this case, the memory used is reduced since we need to store and compute less variables. Also the GPU usage is reduced since we now perform less computation. Hence, the whole pipeline is a very valuable tool to be used in the industry.

2 Project Outline Figure

Below is a figure demonstrating our pipeline, see appendix for a more detailed demonstration:

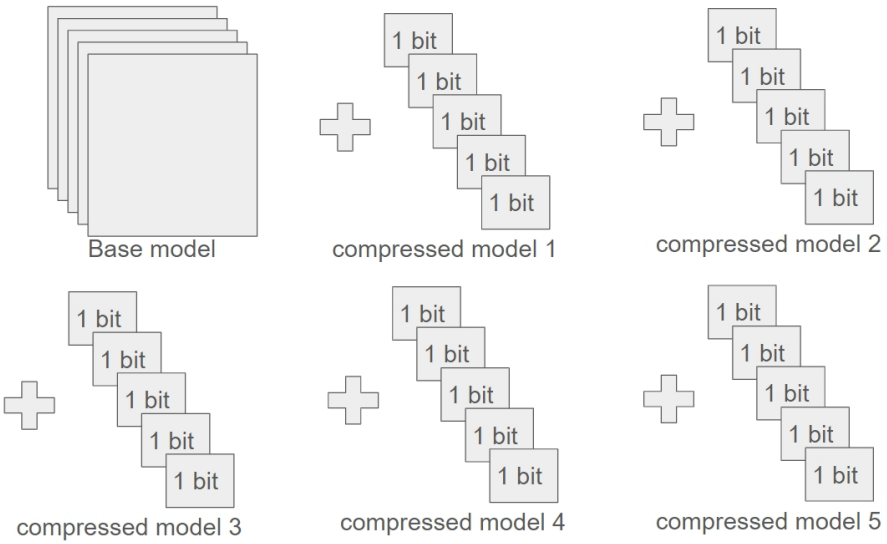

Figure 1: parameters for 5 fine-tuned model (compressed)

3 Background and Related Work

3.1 Model Compression

Not all parameters in a neural network contribute equally and are important (LeCun et al, 1990), thus making it possible to compress models. Model Compression is the process of reducing a model's size - number of parameters - to create a much more efficient model in both inference time and storage space required, while maintaining the accuracy of the model. This is important for real-world applications of models which are constrained due to several factors, including the memory and computational resources availability. Currently, there are many different methods of model compression - including knowledge distillation, quantization, pruning and more.

3.2 Quantization

Quantization is another model compression technique in which the number of bits required to store weights is reduced. They can be represented as 16 bit, 8 bit and all the way down to 1 bit - a binary representation, which was employed by BitDelta - a delta compression technique which is discussed below.

3.3 BitDelta

BitDelta is a delta compression technique and aims to quantize a model's weight delta - the additional change in parameters between the base model and fine-

tuned model - and represent it as 1 bit. This technique is composed of two main steps - 1. representing the delta of the fine-tuned parameters as a scalar times a binary representation and 2. fine-tuning this representation via distillation (Liu et al , 2024). This technique has several implications including a drastic computation time speed up of approximately 10x that of prior techniques. It was also found to be successful in reducing GPU memory requirements for models (Liu et al , 2024).

However, some limitations that we noticed was that the code implemented for compression technique was hard-coded structures that they support and thus, potentially lacks in generalizability to the models that they can intake - the implementation only accepts certain types of models, requiring a specific structure.

We hope to extend on BitDelta in our project through the creation of a more generalized compressor, using the BitDelta technique that is able to intake many different types of models.

4 Data Processing and Cleaning

Since our goal is to compress the fine-tuned model, we will work on the database they originally fine-tuned on and we don't need to additionally clean the dataset. We use the 'load_dataset' function from hugging face. This allows us to load any dataset from the hub. However, if we want to load datasets from other places, more functions need to be added to handle this. As for tokenizing the data, we need to pass in some parameters of the dataset, like keys. Since we want to build a pipeline that can be used for more tasks, we need to handle datasets with different keys or structures. Hence, we need to do some extra work so that we can tokenize the dataset automatically. For now, our pipeline only supports the GLUE (General Language Understanding Evaluation) datasets which is used to train and evaluate natural language understanding systems. This benchmark contains 9 different datasets that require different tasks to be performed - including single sentence tasks, similarity detection and more. There are several subsets inside GLUE, each of them have different keys (Wang et al, 2019). Thus, in order to automatically load given any subset from GLUE, we create a mapping between keys and names for each subset. Given the subset's name, it will find the keys for the tokenizer to tokenize the dataset.

5 Model Architecture

The architecture of the resulting model from our compression process would be mostly the same as the input fine tuned model, however, with small modifications. The overall process of our compression is:

1. Make a copy of the chosen fine tuned model

2. For each submodule in each module of the copied fine-tuned model that has 'lin' in its name, we will replace it with a new module named BinaryDiff. In that module, we would have a bitdelta matrix and a coefficient, which are computed from the weights of the corresponding modules in the original fine-tuned model and base model.

3. After finishing replacing all the submodules, we will only save the weights of those newly created BinaryDiff modules, as our compressed model. In order to load our compressed model, we will first load the original base model. Then, replace the weights in the model by the weights from the BinaryDiff modules we saved.

4. This architecture is the same as the one introduced in the original paper. Below is an illustration of the compression process from the paper.

This architecture is the same as the one introduced in the original paper. See Figure 5 in Appendix for an illustration of the compression process from the paper.

One major difference in our model from BitDelta's is the layers that are compressed. In the original implementation made by Liu et al (2024), we figured that their implementation was hard coded. They can only compress the projection submodules, which must have substring "proj" in its name, of the modules that must have substrings "mlp" or "self_attn" in their names. This appears to be a necessary design as there are no easy, and general ways to modify a module's weight. However, this also makes it hard for us to make modifications to the existing code to apply it to different models, including DistiledBertModel.

What we did was, instead of compressing the projection submodules, we chose to compress all linear submodules (which are of the "Linear" module). Essentially, this is the same approach as the one in the paper, but we took a different implementation. For each linear layer, we conduct the following computation:

$$\mathbf{W} = \mathbf{W}_{\text{finetune}} - \mathbf{W}_{\text{base}}$$

$$\mathbf{W}_{\text{new}} = \begin{cases} +1(\text{ if } W_{i,j} > 0) \\ -1(\text{ if } W_{i,j} \leq 0) \end{cases}$$

$$\alpha = \frac{1}{nm} \sum_{ij} |W_{ij}|$$

Where $\mathbf{W}_{\text{finetune}}, \mathbf{W}_{\text{base}} \in \mathbb{R}^{n \times m}$.

6 Experiment

6.1 Baseline

We chose sets of fine-tuned models on distilbert-base-uncased, fnet-base, albert-large, bert-large-uncased, and bert-large-cased as the baseline models. We will compare the accuracies as well as size of the models before and after compression to evaluate the performance.

We chose these models because they are fine-tuned on the GLUE datasets and their accuracies vary from 0.6 - 0.9, hence we can not only see the performance, but also how the original performance might affect the compressed model's performance.

6.2 Quantitative Result

Figure 2: Table Showing the test accuracies obtained for the Original and Compressed Models as well as the original and compressed model sizes.

Base model name	**Fine-tuned model name**	**DATASET**	**ACCURACY**		**SIZE**	
			Original	Compressed	Original	Compressed
distilbert-base-uncased	avneet/distilbert-base-uncased-finetuned-sst2	glue-sst2	0.9151	0.8865	267MB	140MB
distilbert-base-uncased	BlitherBoom/distilbert-base-uncased-finetuned-sst2	glue-sst2	0.9094	0.9071	267MB	140MB
fnet-base	gchhablani/fnet-base-finetuned-rte	glue-rte	0.6282	0.5993	331MB	158MB
bert-large-uncased	TehranNLP-org/bert-large-sst2	glue-sst2	0.9255	0.9094	1340MB	431MB
bert-large-cased	Cheng98/bert-large-sst2	glue-sst2	0.9255	0.9037	1334MB	424MB
albert-large	anirudh21/albert-large-v2-finetuned-rte	glue-rte	0.5487	0.5451	708MB	326MB

To test our compression on the models, we based the performance of our models on 2 main factors - the **accuracy** of the original fine-tuned model vs the compressed model and the **size** of the original model and our compressed model, an indicator of our compression rate. In order to test the accuracies of the original model and that compressed, the following datasets which are a part of the GLUE benchmark were used:

- **RTE** - Recognizing Textual Entailment dataset contains textual entailment challenges obtained from news and Wikipedia text.

- **SST-2** - Stanford Sentiment Treebank contains fully labeled parse trees and allows for sentiment analysis. There are 11,855 single sentences in the dataset.

6.3 Qualitative Result

As from Table above, we can tell that the compression rate is around 50% - 70% , which effectively saves space. Since we compressed every linear layer, the more linear layers the model has, the higher the compression rate is. Also, as shown in table 1, the accuracy difference is within 3% for all of them. Even though the accuracies drop a little for the compressed models, compared to the compression rate and the size of the models, this is acceptable. Especially, when the base model is large, this technique can be useful since it greatly compresses the size, while not hurting the performance too much.

7 Discussion

7.1 Results

As of currently, our model retains relatively satisfactory performance on the tested models and datasets using the selected GLUE benchmark tasks and datasets - as mentioned in the Qualitative Analysis, the compressed model achieves an at most 3% accuracy difference between the original fine-tuned model test accuracy and that of the compressed. Furthermore, we achieved a compression rate of almost 70%. These results are remarkable as they were achieved with only compressing the linear layers of the model's network without further fine-tuning after compression. This has the potential to greatly reduce the memory required to store the weights (our largest compression rate on tested fine-tuned models was 0.68). Furthermore, with our current basic implementation, we have achieved accuracy levels comparable to the original. Due to the time limitation of this project, we were not able to further fine-tune the compressed model, but adding a fine-tuning step, for example, via distillation will have the potential to close the gap between the original fine-tuned model's accuracy and that of the compressed.

7.2 Difficulty and Limitation

One biggest challenge of this project is the usage of the triton[1] package in the original implementation. Triton was not designed for operating systems other than linux, and due to this limitation, the code provided by the authors of our paper is almost impossible to be deployed on our local machines and we needed to rebuild from scratch ourselves.

Another major challenge is generalizing the compression process. Due to the fact that models have very different underlying architectures, it is really hard to complete a generalized compression tool that can be used on all types of model.

In order to lower the complexity of our project we finally decided to apply the compression process to only ModelForSequenceClassification. Although the scale of our project is simplified compared to the paper's, our results indicate good performance, even with this simplification. We were able to compress the size to half of the original size, while maintaining a good fidelity.

Finally, the scale distillation approach in our chosen paper is not easy to implement from scratch. While it is also hard to make modifications to the existing code by the authors (due to the triton package), we had to use 1-bit compression without distillation in order to get some test results before the deadline.

7.3 Future:

Based on what we have, we can extend the automatic pipeline to other models in hugging face. Adding a data handler for different datasets, we will have a more generous pipeline. As mentioned earlier, the original implementation of BitDelta also further improves their compressed models through fine-tuning the delta representation with distillation - a process where the information from the larger model is transferred to a smaller model through training the smaller model on the output of the larger model. This can also be explored in our implementation in the future to further improve accuracy of our compressed model.

8 Ethical Consideration

Our project is based on compressing a given fine-tuned model using the provided dataset, without any additional processing of the dataset. If there is a potential ethical problem in the dataset or the model is biased, we cannot avoid them. For example, unfortunately, through our process, we are not able to mitigate ethical issues that are originally present - if the dataset does not cover the whole population or is biased sampled, or if the pre-trained model already contains gender bias, that will be learnt by the model.

Since we are compressing the model using only 1-bit each matrix and rely on the parameters of the pre-trained model, the results given by the compressed fine-tuned model might be more similar to the result from the pre-trained model. Thus, it might either introduce biases into the model or amplify the biases.

Another impact is on the environment. For a very general pre-trained model, we need significantly less parameters than for fine-tuned models and thus, require less memory resources. Hence, since compression results in a model with significantly less parameters, it has the potential to greatly reduce the amount of memory required to store the weights and computational resources required at inference, which saves power. Hence, it will do less harm to the environment.

9 Bibliography

A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman, "GLUE: A MULTI-TASK BENCHMARK AND ANALYSIS PLATFORM FOR NATURAL LANGUAGE UNDERSTAND- ING." Accessed: Apr. 19, 2024. [Online]. Available: https://arxiv.org/pdf/1804.07461v3.pdf

G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," arXiv:1503.02531 [cs, stat], Mar. 2015, Available: https://arxiv.org/abs/1503.02531

J. Liu et al., "BitDelta: Your Fine-Tune May Only Be Worth One Bit," arXiv (Cornell University), Feb. 2024, doi: https://doi.org/10.48550/arxiv.2402.10193.

L. Cun, Y. Le Cun, J. Denker, and S. Sol1a, "Optimal Brain Damage." Accessed: Jan. 23, 2024. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/1989/file/6c9882bbac1c7093bd25041881277658-Paper.pdf

"nyu-mll/glue · Datasets at Hugging Face," huggingface.co. https://huggingface.co/datasets/nyu-mll/glue/viewer/wnli (accessed Apr. 19, 2024).

"avneet/distilbert-base-uncased-finetuned-sst2 · Hugging Face," huggingface.co, Jan. 30, 2024. https://huggingface.co/avneet/distilbert-base-uncased-finetuned-sst2 (accessed Apr. 19, 2024).

"AI Risk Database," airisk.io. https://airisk.io/model/overview/pkg%3Ahuggingface%2FBlitherBoom%2Fdistilbert-base- uncased-finetuned-sst2%40fd8c99d079b82c49da4aaa9df203afe6af576e94 (accessed Apr. 19, 2024).

"gchhablani/fnet-base-finetuned-rte · Hugging Face," huggingface.co, Jan. 30, 2024. https://huggingface.co/gchhablani/fnet-base-finetuned-rte (accessed Apr. 19, 2024).

"TehranNLP-org/bert-large-sst2 · Hugging Face," huggingface.co, Jan. 04, 2024. https://huggingface.co/TehranNLP-org/bert-large-sst2 (accessed Apr. 19, 2024).

"Cheng98/bert-large-sst2 · Hugging Face," huggingface.co, Jan. 30, 2024. https://huggingface.co/Cheng98/bert-large-sst2 (accessed Apr. 19, 2024).

"anirudh21/albert-large-v2-finetuned-rte · Hugging Face," huggingface.co, Jan. 30, 2024. https://huggingface.co/anirudh21/albert-large-v2-finetuned-rte (accessed Apr. 19, 2024).

10 Appendix

10.1 Github Repo Link

https://github.com/yi1z/bitdelta/tree/deploy

10.2 Contribution

Yifang Zhang: Mainly focused on understanding the approaches in paper and the original implementation, designing generalized implementation which can work across system, and make initial testing scripts for selected model pairs and dataset. Wrote model architecture and difficulty and limitation sections of the report.

Michelle Lin: For this project, I mainly contributed to the writing of this paper and conducting literature review. Specifically, I wrote the Related Works section and Results in Discussion. Additionally, I contributed in editing the paper overall and organising the outputs generated. Unfortunately, I was not able to participate much in the coding aspect of this project due to time constraints, but have taken a look through the code and what has been done thus far.

Yuqi Chen: I brought this topic with my teammates initially. As for the report, I'm responsible for writing the introduction, data processing and cleaning, result analysis, ethical consideration, and making the graph demonstration. I also helped with the code pipeline by making it more automatically and conducted experiments.

ALL: We all participated in meetings where we met to discuss the project and progress updates.

10.3 Figures

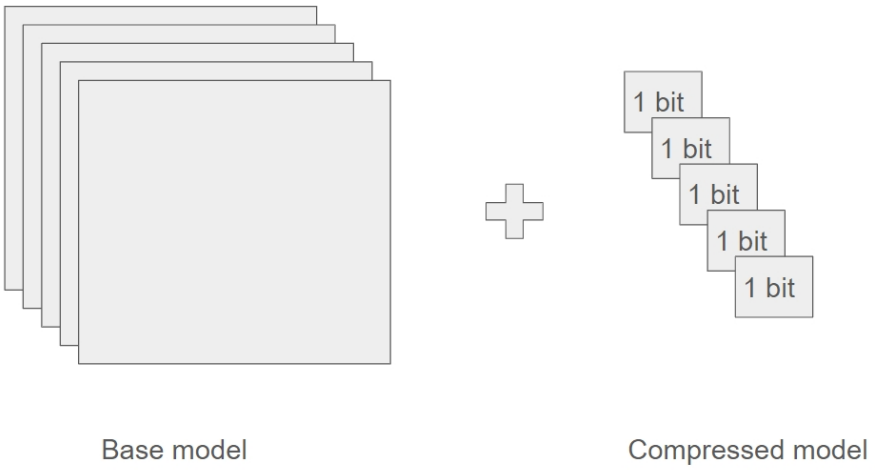

Figure 3: parameters for 1 fine-tuned model (compressed)

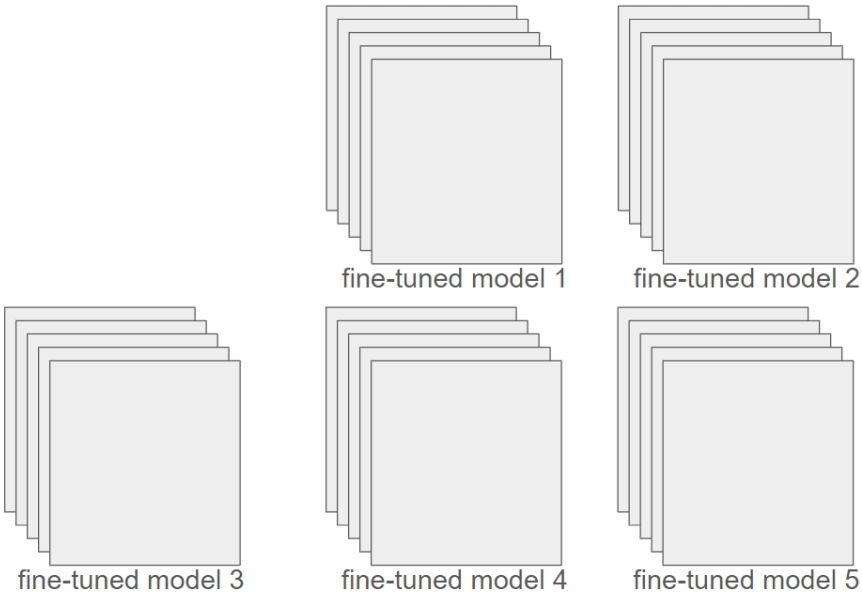

Figure 4: parameters for 5 fine-tuned model (original)

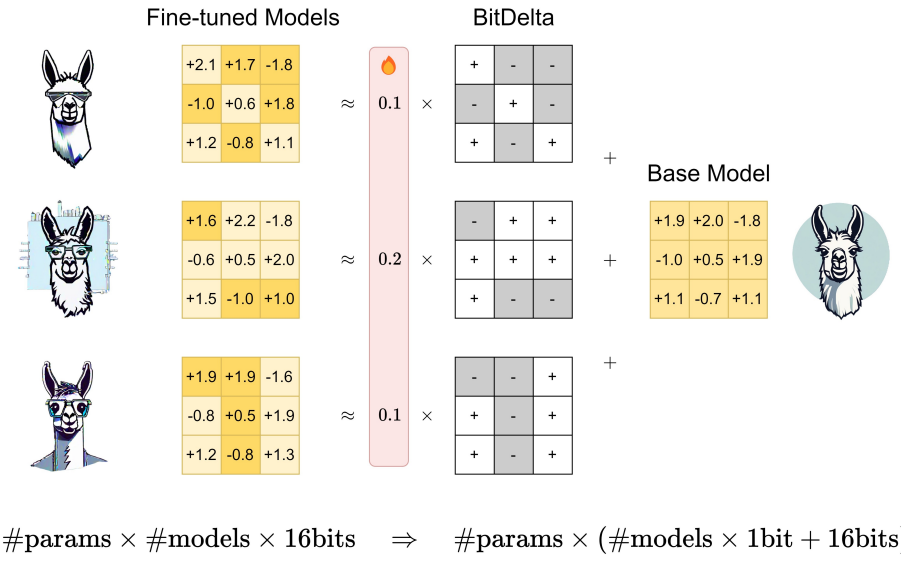

Figure 5: the BitDelta model architecture (Liu et al, 2024) and which our model is based on