# A Computational Study of Matrix Decomposition Methods for Compression of Pre-trained Transformers

**Viktoria Chekalina**[1*] **Daniil Moskovskiy**[1*] **Sergey Pletenev**[1,2*]
**Mikhail Seleznyov**[1] **Sergey Zagoruyko**[3] **Alexander Panchenko**[1,4]

[1]Skolkovo Institute of Science and Technology, [2]HSE University,
[3]MTS AI, [4]Artificial Intelligence Research Institute
{s.pletenev, d.moskovskiy, v.chekalina, a.panchenko}@skol.tech

## Abstract

Transformer-based models have significantly advanced the field of Natural Language Processing. However, their large size and computational complexity present challenges. As a result, there is considerable interest in developing approaches to compress these models without compromising their performance on specific tasks. This paper presents a comparative study of low-rank matrix and tensor factorization techniques for compressing Transformer-based models. Specifically, we apply Singular Value Decomposition (SVD) and Tensor Train Matrix (TTM) decomposition to represent the fully connected layers in a compressed form. Following Hsu et al. (2022), we extend the FWSVD approach by adding Fisher information to the TTM decomposition and present a novel method called FWTTM.

Our experimental results indicate that the efficiency of these methods varies with the compression level. Notably, integrating Fisher information to align task and decomposition objectives enhances the performance of factorized with TTM transformer-based models and encoder-decoders.

## 1 Introduction

In recent years, the field of Natural Language Processing has made significant progress with the development of large pre-trained language models such as BERT (Devlin et al., 2019). While these models have achieved state-of-the-art performance on various tasks, their size and computational requirements make them challenging to deploy in resource-constrained environments. As a result, there has been growing interest in developing techniques to compress these models while maintaining their performance.

Language models often lose their capacity which leads to worse quality at somewhat significant compression levels. In this case, the models are fine-tuned until a certain quality is achieved on the task. However, fine-tuning is also resource-intensive, even for a compressed model. To make it more efficient, we use the alignment of the low-rank compression objective and the task objective as introduced by Hsu et al. (2022). This makes the compressed model more consistent with further fine-tuning.

One approach to model compression is to apply matrix factorization techniques to the heaviest part of the Transformer – fully-connected layers (see Table 2). The most popular and simplest choice is to use the SVD to reduce the number of parameters while retaining the model's expressive power.

Applying SVD to a matrix can decrease its expressibility (Yang et al., 2018). However, additional techniques are employed to ensure a satisfactory quality of the resultant model. Hsu et al. (2022) introduce the Fisher Weighted SVD (FWSVD) approach, which considers the significance of each parameter for the model's performance during the compression process based on gradient values.

Another method for compressing large language models is Tensor-train matrix decomposition, or simply TTM (Oseledets et al., 2011). TTM transforms a weight matrix into a high-order tensor, which is then expressed as a product of lower-dimensional objects. In this study, we expand the application of the Fisher Weighted SVD (FWSVD) approach to TTM, creating a novel approach called FWTTM.

Our contributions can be summarized as follows:

- We extend the previous work by Hsu et al. (2022) and incorporate weighting based on Fisher information inside the TTM decomposition (we denote this approach as FWTTM).
- We provide a comprehensive analysis of the performance of the BERT model compressed with SVD, TTM, FWSVD, and FWTTM on various ranks on tasks of GLUE bench-

*Equal contribution.

mark (Wang et al., 2019) and the BART model (Lewis et al., 2020) on the sequence-to-sequence tasks of text summarization and text detoxification.

- We provide an implementation of the studied methods widely applicable to pre-trained Transformer models, such as those at the Huggingface repository.[1]

## 2 Related Work

This section reviews methods related to model size reduction. It contains Knowledge distillation, Quantization, Pruning, and low-rank Approximation techniques.

The first approach, Knowledge distillation (KD), learns a student model with a smaller parameter budget guided by a larger trainer model. These methods can also transfer knowledge from a large teacher model to a smaller student model (Hinton et al., 2015). KD can improve the generalization performance of the student model and reduce its size and computational cost (Jiao et al., 2020). We use DistilBERT (Sanh et al., 2019) – a distilled version of the BERT model as one of the strong baselines in our work.

Pruning is another powerful technique to reduce the number of deep neural network parameters. The goal of neural network pruning is to identify and remove unimportant connections to reduce the model size without affecting network accuracy. Movement pruning (Sanh et al., 2020) is an efficent approach for pruning unstructured networks. This method gives high sparsity in the model while preserving the original quality score. On the other hand, such models will show effectiveness only with specialized hardware and may not give any benefits to standardized devices such as GPUs.

Block pruning is another effective method for reducing the number of deep neural network parameters. This approach involves removing entire blocks of unimportant connections rather than individual connections. This can result in a more structured and efficient network architecture. One example of block pruning is filter pruning, where entire filters in a convolutional neural network are removed (Li et al., 2017). Another example is channel pruning, where entire channels are removed from the network (He et al., 2017). As opposed to movement pruning, this approach encourages pruning that can be optimized on dense hardware.

[1]github.com/s-nlp/compression

| Layer/Model | BERT | | BART | |
|---|---|---|---|---|
| Full model | 109 M | 100% | 140 M | 100% |
| Fully connected layers | 57 M | 52% | 84 M | 60% |
| Embedding layers | 24 M | 22% | 38 M | 27% |
| Attention heads | 28 M | 26% | 23 M | 16% |

Table 1: Number of parameters for different layers in various Transformer architectures.

| BERT | | | BART | | |
|---|---|---|---|---|---|
| Compr. Rate | SVD | TTM | Compr. Rate | SVD | TTM |
| 48% (53 M) | 6 | 10 | 60% (83 M) | 10 | 10 |
| 63% (69 M) | 183 | 60 | 74% (102 M) | 210 | 64 |
| 95% (102 M) | 534 | 110 | 90% (125 M) | 460 | 96 |

Table 2: Ranks for different compression approaches. The values in parentheses in the "Compr. Rate" column represent the fraction of weights are left compared to original uncompressed model. The values in the "SVD" and "TTM" columns are ranks.

The quantization approach enables the reduction of the model size without compromising the parameter count, achieved by reducing the number of bits allocated to each parameter. The concept of quantization-aware training, which involves training the model with the reduced weights, came from general deep learning (Hawks et al., 2021) to transformer-based encoders (Wang et al., 2022).

Low-rank approximation techniques provide an alternative way to achieve model compression. One such technique is SVD, which has been successfully applied to compress various components of neural networks, such as word embeddings (Lan et al., 2020), attention matrices (Michel et al., 2019), and transformer layers (Hu et al., 2021). Another approximation technique is TTM, which decomposes high-order tensors into a sequence of low-order tensors (Oseledets et al., 2011). TTM has been employed for compressing word embeddings (Hrinchuk et al., 2020), CNNs (Garipov et al., 2016), and even vision transformers (Minh et al., 2022).

## 3 Low-rank Compression Methods

In this section, we describe the low-rank approximation methods used in our computational study to compress feedforward layers of Transformers: SVD, TTM, FWSVD, and our proposed FWTTM, a novel approach.

### 3.1 Singular Value Decomposition (SVD)

We compress the initial model by replacing fully-connected layers with their SVD analogs.

Assuming that $W$ is a layer weight matrix, we define SVD as follows: $\mathcal{W} = U\Sigma V^T$. Then we use truncated products of it $U_r = U[:, : r], \Sigma_r = \Sigma[: r, : r], V_r = V[:, : r]$ to define weights for two sequential linear layers, with which we will replace the current:

$$\mathcal{W}_2 = U_r \sqrt{\Sigma_r} \tag{1}$$
$$\mathcal{W}_1 = \sqrt{\Sigma_r} V_r^T \tag{2}$$

As a result, we get an approximation of linear matrix $\mathcal{W} \approx \mathcal{W}_2 \mathcal{W}_1$ and an approximation of the initial layer $Y \approx X \mathcal{W}_1^T \mathcal{W}_2^T + b$.

If $\mathcal{W}$ has $n_{in}, n_{out}$ shape, the number of parameters in the layer before compression is $n_{in} \times n_{out}$; after representation by truncated SVD, it is $r \times (n_{in} + n_{out})$.

### 3.2 Tensor Train Matrix Decomposition (TTM)

In the TTM decomposition as defined by Oseledets et al. (2011) the matrix $\mathcal{W} \in \mathbb{R}^{I \times J}$ is represented in $2D$-order tensor $\mathcal{T} \in \mathbb{R}^{I_1 \times J_1 \times \cdots \times I_D \times J_D}$ in TTM format, where $I = \prod_{k=1}^{D} I_k, J = \prod_{k=1}^{D} J_k$. In other words, each element of $\mathcal{T}$ is computed as

$$\mathcal{T}_{i_1,j_1,\ldots,i_D,j_D} = \sum_{\substack{r_0,r_1,\\ \ldots,r_{D-1},r_D}} \mathcal{G}^1_{r_0,i_1,j_1,r_1} \cdots \mathcal{G}^D_{r_{D-1},i_D,j_D,r_D}, \tag{3}$$

where $\mathcal{G}^d \in \mathbb{R}^{R_{d-1} \times I_d \times J_d \times R_d}$, $d = \overline{1, D-1}$ are *core tensors (cores)* of TTM decomposition, vector $(R_0, \ldots, R_D)$ is called TTM ranks. Note that $R_0 = R_D = 1$ and $R = max(R_0, \ldots, R_D)$.

The compression rate in a TTM layer, with respect to the number of parameters is defined as follows

$$\text{c\_rate} = \frac{R(I_1 J_1 + I_D J_D) + R^2 \sum_{d=2}^{D-1} I_d J_d}{\prod_{k=1}^{D} I_k J_k} \tag{4}$$

We use our own implementation of this type of layer based on the algorithm by Oseledets et al. (2011).

### 3.3 Fisher Weighted SVD (FWSVD)

Hsu et al. (2022) propose injecting the Fisher information into decomposition algorithms to minimize the gap between decomposition and task-oriented objectives. Fisher information determines the importance of each parameter for predictions in a given task (Bishop and Nasrabadi, 2007). We follow the approach introduced by Hsu et al. (2022) and approximate the Fisher matrix using dataset $\mathcal{D} = \{d_1, \ldots, d_{|\mathcal{D}|}\}$, for each weight matrix $\mathcal{W} \in \mathbb{R}^{I \times J}$:

$$\begin{aligned} \mathcal{I}_{\mathcal{W}} &= \mathbb{E}\left[\left(\frac{\partial}{\partial \mathcal{W}} \log p(\mathcal{D}|\mathcal{W})\right)^2\right] \\ \mathcal{I}_{\mathcal{W}} &\approx \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \left(\frac{\partial}{\partial \mathcal{W}} \mathcal{L}(d_i; \mathcal{W})\right)^2 \end{aligned} \tag{5}$$

Having this, ideally, we would want to solve weighted low-rank approximation:

$$\left|\left|\sqrt{\mathcal{I}_{\mathcal{W}}} * (\mathcal{W} - \hat{\mathcal{W}})\right|\right|^2 \to \min_{\operatorname{rank} \hat{\mathcal{W}} = r} \tag{6}$$

Unfortunately, this problem does not have a closed-form solution. Therefore, Hsu et al. (2022) propose to sum Fisher matrix by rows and solve low-rank approximation with row-wise weighting, which can be done using SVD:

$$\begin{aligned} \tilde{\mathcal{I}}_{\mathcal{W}} &= \operatorname{diag}\left(\mathcal{I}_{\mathcal{W}} \cdot \mathbf{1}\right), \\ \hat{\mathcal{W}} &= \tilde{\mathcal{I}}_{\mathcal{W}} \mathcal{W} = USV^T, \end{aligned} \tag{7}$$

where $\mathbf{1} = (1, \ldots, 1) \in \mathbb{R}^{J \times 1}$, diag - diagonal matrix with size $\mathcal{I} \times \mathcal{I}$.

The resulted weighted factors for initial matrix $\mathcal{W} \approx \hat{U}\hat{S}\hat{V}^T$ are computed as follows:

$$\hat{U} = \tilde{\mathcal{I}}_{\mathcal{W}}^{-1} U, \hat{S} = S, \hat{V} = V. \tag{8}$$

As a result, we get low-rank approximations, which account for parameter importances for the target task.

### 3.4 Fisher Weighted TTM (FWTTM)

To apply the Fisher matrix to the TTM algorithm, we represent equation 8 in the following format:

$$\begin{aligned} Y &= X \left(\tilde{\mathcal{I}}_{\mathcal{W}} + \lambda \mathbf{1}\right)^{-1} \left(\tilde{\mathcal{I}}_{\mathcal{W}} + \lambda \mathbf{1}\right) \mathcal{W} + b \\ Y &= \hat{X}\, \hat{\mathcal{W}} + b \\ Y &\approx \hat{X} \text{ TTM}\left(\hat{\mathcal{W}}\right) + b \end{aligned} \tag{9}$$

We add regularization term $\lambda$ in order to improve numerical stability. Our numerical experiments demonstrated the efficiency of such regularization though the selection of the best suitable $\lambda$ value is an additional process. We discuss the process of selection in the Appendix C. The algorithm consists of the following steps:

| FC Layers | Regular FC | TTM-10 | SVD-6 |
|---|---|---|---|
| Inference time (ms) | 37.8 | 63.8 | 27.6 |
| Power ($10^{-5}$ kWh) | 1.7 | 2.11 | 1.4 |

Table 3: Average inference time (in ms) and power consumption for a BERT model with regular fully-connected, SVD-6, and TTM-10 layers. The batch size is equal to 1.

1. Compute the Fisher matrix $\mathcal{I}_{\mathcal{W}}$ for the original layer matrix $\mathcal{W}$.
2. Apply the $\mathcal{I}_{\mathcal{W}}$ to $\mathcal{W}$ and do original TTM decomposition on $\hat{\mathcal{W}}$.
3. For each forward step of the FWTTM algorithm, we use the $\tilde{\mathcal{I}}_{\mathcal{W}}^{-1}$ on $X$ to inverse matrix from the previous step.

## 4 Transformer Compression Setup

This section describes our setup for compressing Transformer models using low-rank approximation approaches. We focus on two methods: TTM and SVD, *with* and *without* Fisher information. We aim to reduce the number of parameters in the model while maintaining its performance. Furthermore, we assume we can access the task-oriented model-tuning process. We use the information obtained within this process to improve the quality of the compression and thus speed up the tuning by the desired values.

### 4.1 Baselines

We compare our compressed model to the model obtained by Distillation (Jiao et al., 2020), Block Pruning (Sanh et al., 2020) and inference of the original model with floating-point precision equal to 16. Note that both Distillation and Block Pruning are training-aware methods. It means they require fine-tuning for the desired task so we can use it only in the Double-train pipeline.

For mixed precision training and evaluation, we use the FP16 library, which is built-in in PyTorch (Paszke et al., 2019). We set the optimization level to 01 and patched all torch functions and tensor methods, except those that benefit from FP32 precision (softmax, etc.) For the two analyzed models, we obtained a compression up to 52% for the BERT model and 54% for the BART model. However, since the tables show compression of the models relative to the number of their parameters, but FP16 quantization keeps the number of parameters the same, we indicate the actual number of parameters with a dagger (†).

| Memory (MB) / Layers | Regular FC | TTM-10 | SVD-6 |
|---|---|---|---|
| Model + input | 418.7 | 204.6 | 204.9 |
| After Forward | 1069.5 | 840.1 | 851.9 |
| After Backward | 869.3 | 433.6 | 429.9 |
| Peak usage | 1101.1 | 901.8 | 865.7 |

Table 4: Memory usage for BERT with regular fully-connected, SVD-6, and TTM-10 layers on an NVIDIA A40 GPU. The measurements are for one forward-backward operation with a batch size of 1.

### 4.2 Experimental setup

In this study, we evaluate the performance of four proposed methods based on the BERT and BART models. We apply three different compression ratios to these models and present the resulting ranks in Table 2. We also assess the performance of the compressed BERT model on nine natural language understanding tasks, including language acceptability, sentiment analysis, paraphrasing, and natural language inference. The compressed BART model is evaluated on text summarization and detoxification tasks.

Moreover, we run two setups for compressing and evaluating models on the GLUE, ParaDetox (Logacheva et al., 2022) and XSUM (Narayan et al., 2018) datasets:

- Single-train. We fine-tune a model for each task, compress it and measure performance.
- Double-train. We follow the same steps as for the Single-train and fine-tune the compressed model again on the same task.

### 4.3 Selection of hyperparameters

The proposed layer structure assumes two sets of hyperparameters - TTM *cores shapes* and *ranks* for both TTM and SVD.

For the maximum compression rate in TTM, non-rank shapes of cores should be as close to each other as possible. We choose $I_k \cdot J_k$ so that they are equal and approximately equal to $(I \cdot J)^{1/D}$. As a cores, we take objects with sizes $[1 \times 8 \times 12 \times R]$, $[R \times 12 \times 16 \times R]$, $[R \times 8 \times 16 \times 1]$.

Ranks $r$ for truncation in SVD and $R$ for TTM are selected based on the desired compression level.

| Method | C. Rate | AVG | STSB | CoLA | MNLI | MRCP | QNLI | QQP | RTE | SST2 | WNLI |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Full | 100 % | 0.79 | 0.88 | 0.57 | 0.84 | 0.90 | 0.91 | 0.87 | 0.67 | 0.92 | 0.54 |
| DistilBERT | 61 % | 0.76 | 0.87 | 0.51 | 0.82 | 0.87 | 0.89 | **0.88** | 0.59 | 0.91 | 0.48 |
| FP16 eval. | 100%$^\dagger$ | 0.78 | 0.88 | 0.55 | 0.83 | 0.88 | 0.90 | **0.88** | 0.67 | 0.91 | 0.48 |
| Block Pruning (75%) | 61% | 0.72 | 0.85 | 0.24 | 0.83 | 0.83 | 0.86 | 0.87 | 0.52 | 0.88 | 0.56 |
| SVD | | 0.37 | 0.24 | 0.00 | 0.36 | 0.20 | 0.50 | **0.47** | 0.48 | 0.52 | 0.51 |
| FWSVD | | 0.38 | 0.25 | 0.00 | 0.33 | **0.39** | 0.50 | 0.40 | 0.49 | 0.51 | **0.56** |
| TTM | 49 % | 0.40 | 0.38 | 0.00 | 0.37 | 0.20 | 0.53 | 0.42 | **0.50** | 0.69 | 0.51 |
| FWTTM | | **0.44** | **0.58** | 0.01 | **0.37** | 0.26 | 0.56 | 0.42 | **0.50** | **0.70** | 0.51 |
| SVD | | 0.45 | 0.63 | 0.01 | 0.36 | 0.22 | 0.51 | 0.54 | 0.54 | 0.78 | 0.48 |
| FWSVD | | **0.55** | 0.54 | **0.07** | **0.52** | **0.55** | 0.62 | **0.70** | **0.58** | **0.79** | **0.55** |
| TTM | 63 % | 0.44 | 0.65 | 0.01 | 0.40 | 0.16 | 0.54 | 0.52 | 0.48 | 0.74 | 0.48 |
| FWTTM | | 0.47 | **0.71** | 0.00 | 0.43 | 0.18 | **0.64** | 0.56 | 0.47 | 0.72 | 0.49 |
| SVD | | 0.70 | 0.81 | 0.26 | 0.82 | 0.69 | 0.88 | 0.87 | 0.53 | 0.90 | 0.53 |
| FWSVD | | **0.78** | **0.88** | 0.55 | **0.84** | 0.87 | **0.90** | **0.88** | 0.64 | **0.92** | **0.55** |
| TTM | 95 % | 0.76 | 0.87 | 0.52 | 0.79 | 0.86 | 0.87 | 0.86 | 0.65 | 0.91 | 0.48 |
| FWTTM | | 0.77 | **0.88** | **0.56** | 0.83 | **0.88** | **0.90** | **0.88** | **0.66** | **0.92** | 0.46 |

Table 5: Results of different types of compression of BERT for experiment with task-oriented fine-tuning and further compression (Single-train). The best results at each model size are in **bold**, best overall results are underlined. Standard deviations are included in Table 9 in Appendix.

## 5 Computational performance and Energy Metrics of Training Loops

In this section, we evaluate the memory requirements for a single training loop of the BERT model using TTM- and SVD-based linear layers with ranks 10 and 6, respectively (refer to Table 2). As illustrated in Table 4, utilizing compressed representations considerably decreases the memory needed for training. Notably, for a given compression rate, the memory requirement remains consistent regardless of the compression method—both TTM and SVD-based models consume similar memory amounts.

We assess electricity consumption with the Eco2AI library[2] (Budennyy et al., 2023). As shown in Table 3, models with SVD layers achieve the best results in terms of power efficiency. In contrast, TTM consumes more electricity and takes longer, even when compared to the original model. We hypothesize this increased consumption might result from the sequential multiplication of multiple cores during the forward function execution.

## 6 Experiments with NLU tasks

For experiments we use `bert-base-uncased` checkpoint[3] from the Hugging Face (Wolf et al., 2019) model hub.

[2]github.com/sb-ai-lab/Eco2AI
[3]huggingface.co/bert-base-uncased

### 6.1 Experimental settings

We perform experiments on the GLUE benchmark using the evaluation script and metrics provided by Hugging Face library[4]. Additionally, we run our experiments with five different random seeds and report the average performance across runs to ensure the robustness of our results.

### 6.2 Results

We report the evaluation results of the BERT model on the GLUE benchmark using different compression methods in Tables 5 and 6, for Single- and Double-train setups, respectively. We also show the extended results across multiple random seeds with standard deviations included in Tables 9, 10. We observe that the overall absolute scores for the Single-train setup are remarkably lower than those for the Double-train setup, as previously reported in research. This is because, after changing the low-rank-based compression structure of the layers, additional fine-tuning is required to regain an acceptable performance level. In this paper, we present both setups because the Single-train setup can be used for large models that do not fit into the memory of available GPUs (as fine-tuning usually requires double the model size to store gradients).

Experiments show that TTM decomposition outperforms SVD at low ranks (i.e., high compression levels), while SVD performs better at higher ranks. This difference is more pronounced for Single-train

[4]github.com/huggingface/transformers/text-classification

| Method | C. Rate | AVG | STSB | CoLA | MNLI | MRCP | QNLI | QQP | RTE | SST2 | WNLI |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Full (109 mln.) | 100 % | 0.79 | 0.88 | 0.57 | 0.84 | 0.90 | 0.91 | 0.87 | 0.67 | 0.92 | 0.54 |
| DistilBERT | 61% | 0.76 | 0.87 | 0.51 | 0.82 | 0.87 | 0.89 | 0.88 | 0.59 | 0.91 | 0.48 |
| FP16 eval. | 100%$^\dagger$ | 0.78 | 0.88 | 0.55 | 0.83 | 0.88 | 0.90 | 0.88 | 0.67 | 0.91 | 0.48 |
| Block Pruning (75%) | 61% | 0.72 | 0.85 | 0.24 | 0.83 | 0.83 | 0.86 | 0.87 | 0.52 | 0.88 | 0.56 |
| SVD | 49% | 0.68 | **0.83** | 0.00 | **0.79** | 0.79 | **0.85** | **0.87** | 0.59 | **0.87** | 0.49 |
| FWSVD | | 0.68 | 0.82 | 0.04 | **0.79** | 0.79 | **0.85** | **0.87** | 0.56 | 0.86 | **0.54** |
| TTM | | **0.69** | **0.83** | **0.15** | 0.78 | **0.81** | 0.84 | **0.87** | **0.60** | 0.86 | 0.43 |
| FWTTM | | 0.68 | **0.83** | **0.14** | 0.78 | **0.81** | 0.84 | **0.87** | **0.60** | 0.86 | 0.41 |
| SVD | 63% | 0.75 | 0.86 | 0.43 | **0.83** | 0.84 | **0.89** | **0.88** | 0.64 | **0.90** | 0.50 |
| FWSVD | | **0.77** | **0.87** | **0.47** | **0.83** | **0.85** | **0.89** | **0.88** | **0.65** | **0.90** | **0.56** |
| TTM | | 0.70 | 0.85 | 0.10 | 0.81 | 0.81 | 0.86 | **0.88** | 0.61 | 0.88 | 0.49 |
| FWTTM | | 0.70 | 0.85 | 0.15 | 0.82 | 0.82 | 0.86 | 0.86 | 0.62 | 0.89 | 0.46 |
| SVD | 95% | 0.78 | **0.89** | **0.56** | **0.84** | **0.88** | **0.91** | **0.89** | 0.68 | **0.91** | 0.44 |
| FWSVD | | **0.79** | **0.89** | **0.56** | **0.84** | **0.88** | 0.90 | **0.89** | **0.69** | **0.91** | **0.51** |
| TTM | | 0.77 | 0.88 | 0.52 | 0.83 | 0.83 | 0.89 | 0.88 | 0.68 | 0.90 | **0.51** |
| FWTTM | | 0.78 | **0.89** | 0.54 | 0.83 | **0.88** | 0.90 | **0.89** | 0.67 | **0.91** | 0.49 |

Table 6: Results of different types of compression of BERT for experiments with task-oriented fine-tuning, compression, and further fine-tuning (Double-train). The best results at each model size are in **bold**, best overall results are underlined. Standard deviations are included in Table 10 in the Appendix.

than for Double-train.

Incorporating Fisher information consistently improves SVD and noticeably improves TTM at high ranks without degrading its performance at other ranks. TTM performs poorly on some tasks, such as CoLA, but better on others, such as STSB (see Figure 3). Low-rank compression methods, especially FWSVD, outperform fine-tuned baseline models of approximately the same size at medium compression rates.

We also observe that for Double-train, FWSVD at 63% compression rate performs comparably or better than Distillation and Pruning baselines and could be combined with FP-16 quantization.

## 7 Experiments with sequence-to-sequence models

We test different layer compression methods on the encoder-decoder model BART (Lewis et al., 2020) on two sequence-to-sequence tasks. Namely, we test different compression methods on a subtask of textual style transfer - text detoxification and the task of text summarization. In our experiments, we use `bart-base` checkpoint from the HuggingFace (Wolf et al., 2019) model hub. Examples of models generation are shown in the Appendix B

### 7.1 Text Summarization

In our text summarization experiments, we use the XSUM dataset (Narayan et al., 2018), which contains news articles and their corresponding single-sentence summaries. We aim to train BART to generate accurate and concise summaries of the input articles.

We evaluate the performance of models using the **ROUGE** metrics (Lin, 2004): we use **ROUGE-1** and **ROUGE-2** to measure the overlap between the generated and reference summaries at the unigram and bigram levels and **ROUGE-L** to measure the longest common subsequence between the generated and reference summaries.

### 7.2 Results

We present the results for summarization on the XSUM dataset in Table 8. In the Single-train setup, FWSVD outperforms other methods across different compression levels for all metrics. However, in the Double-train pipeline, FWTTM emerges as the top performer at low ranks, while TTM excels only in the **ROUGE-2** metric at high ranks. In terms of the remaining metrics, SVD demonstrates superior performance. Notably, the tensor and matrix compression techniques used in the Double-train setup exhibit improved results compared to the baselines.

As the number of ranks increases, the quality of the model does not experience linear grow, but gains quality at certain ranks. Lower ranks elicit hallucinations while on the medium compression levels the models tend to simply copy the input text. Finally, at high ranks, the model restores the ability to summarization. This is only observed for the Single-train approach.

| Pipeline | | Single-train | | | | Double-train | | | |
|---|---|---|---|---|---|---|---|---|---|
| Method | C. Rate | STA | SIM | FL | J | STA | SIM | FL | J |
| `bart-base` | 100 % | - | - | - | - | 0.89 | 0.60 | 0.82 | 0.44 |
| FP16 eval. | 100%$^{\dagger}$ | | - | - | - | 0.89 | 0.60 | 0.82 | 0.44 |
| Block Pruning (95%) | 63% | - | - | - | - | *0.92* | *0.34* | *0.30* | *0.12* |
| Block Pruning (65%) | 74% | - | - | - | - | 0.82 | 0.60 | 0.73 | 0.36 |
| SVD | 60% | *0.97* | *0.18* | *0.10* | *0.01* | 0.75 | **0.59** | 0.65 | 0.28 |
| FWSVD | | *0.32* | *0.46* | *0.58* | *0.07* | **0.78** | **0.59** | 0.68 | **0.30** |
| TTM | | *0.97* | *0.19* | *0.16* | *0.03* | 0.74 | 0.58 | 0.64 | 0.27 |
| FWTTM | | *0.82* | *0.17* | *0.14* | *0.01* | 0.77 | **0.59** | **0.69** | **0.30** |
| SVD | 74% | *0.85* | *0.21* | *0.14* | *0.03* | 0.82 | 0.60 | 0.77 | 0.38 |
| FWSVD | | *0.32* | *0.46* | *0.58* | *0.07* | **0.87** | 0.61 | **0.80** | **0.42** |
| TTM | | *0.99* | *0.17* | *0.06* | *0.01* | 0.82 | 0.61 | 0.75 | 0.37 |
| FWTTM | | *0.97* | *0.17* | *0.45* | *0.08* | 0.84 | **0.62** | 0.76 | 0.38 |
| SVD | 90% | **0.85** | 0.42 | 0.72 | 0.25 | 0.86 | 0.61 | 0.81 | **0.43** |
| FWSVD | | 0.70 | 0.64 | **0.82** | **0.35** | **0.87** | 0.61 | 0.81 | **0.43** |
| TTM | | 0.49 | 0.60 | 0.71 | 0.18 | 0.86 | 0.61 | 0.80 | 0.41 |
| FWTTM | | 0.44 | **0.68** | 0.78 | 0.21 | 0.86 | **0.62** | **0.82** | **0.43** |

Table 7: Results of different types of BART compression for detoxification experiments with task-oriented fine-tuning, compression, and further fine-tuning (Single-train and Double-train). The best results at each model size are in **bold**, best overall results are underlined. *Italic* results represent senseless model outputs.

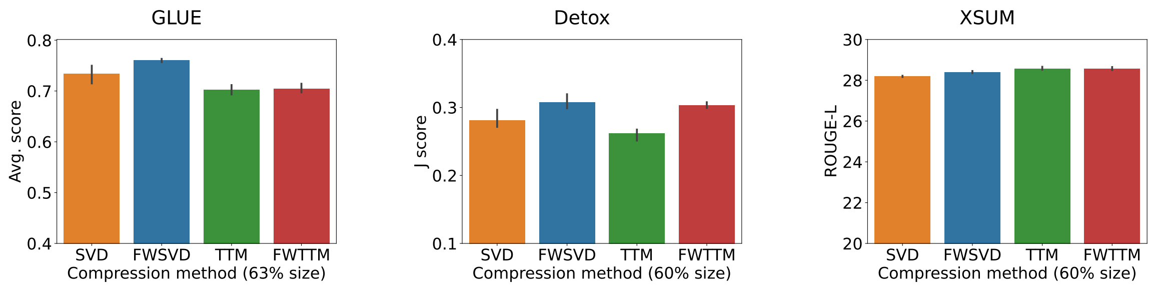

Figure 1: Comparison of compression methods for GLUE, detox, XSUM with Double-train setup.

### 7.3 Text Detoxification

Text detoxification aims to rewrite a sentence in a rude form into a neutrally formulated sentence while preserving its meaning.

We use the parallel dataset ParaDetox (Logacheva et al., 2022) in our experiments. This dataset contains pairs of sentences in rude and neutral forms, which allows us to train text detoxification models in a way similar to neural machine translation. The scale of the dataset also makes training faster and more convenient. We follow the evaluation pipeline presented by Logacheva et al. (2022) and measure the performance of our models using three metrics: **STA** (style transfer accuracy), **SIM** (similarity), and **FL** (fluency of the generated text). **STA** measures how well the model transfers the style of the input sentence from rude to neutral. **SIM** measures how similar the meaning of the generated sentence is to the input sentence. **FL** measures how fluent and natural the generated sentence is.

### 7.4 Results

We depict the results of our experiments with compression in Table 7 and provide the extended version of the table with variations included in Table 11. Text generation also preserves the trend shown in language comprehension. However, unlike GLUE, in the task of detoxification, all the compressed models in the Single-train pipeline are hallucinating and generating senseless tokens at low and medium ranks. We depict these results with *italic*.

Overall, FWSVD shows the best results at most of the compression leves in all the metrics. Both SVD and FWSVD, however, demonstrate comparable performance at low and high ranks with FWSVD being slightly better than others in **J** metric at 74% compression rate. Another important notion is that FWTTM is *almost always better*

| Pipeline | | Single-train | | | Double-train | | |
|---|---|---|---|---|---|---|---|
| Metric | | ROUGE | | | ROUGE | | |
| Method | C. Rate | 1 | 2 | L | 1 | 2 | L |
| `bart-base` | 100% | 42.4 | 19.6 | 34.5 | 42.4 | 19.6 | 34.5 |
| FP16 eval. | 100% | 32.8 | 11.0 | 25.5 | 32.8 | 11.0 | 25.5 |
| Block Pruning (95%) | 63% | - | - | - | 23.4 | 5.7 | 18.8 |
| Block Pruning (65%) | 74% | - | - | - | 34.6 | 12.2 | 27.9 |
| SVD | 60% | *6.3* | ***0.5*** | *5.2* | 35.6 | 13.4 | 28.2 |
| FWSVD | 60% | ***8.1*** | ***0.5*** | ***6.8*** | 35.5 | 13.6 | 28.4 |
| TTM | 60% | *4.2* | *0.2* | *3.7* | **36.1** | **13.9** | **28.6** |
| FWTTM | 60% | *5.0* | *0.2* | *4.2* | 36.0 | 13.8 | **28.6** |
| SVD | 74% | *8.1* | *0.5* | *6.9* | 40.2 | 17.4 | 32.4 |
| FWSVD | 74% | **21.2** | **4.4** | **16.5** | **40.6** | **17.8** | **32.9** |
| TTM | 74% | *6.0* | *0.4* | *4.9* | 39.3 | 16.7 | 31.5 |
| FWTTM | 74% | *7.3* | *0.4* | *5.9* | 39.6 | 16.9 | 31.8 |
| SVD | 90% | 30.8 | 9.9 | 23.8 | **41.6** | **18.8** | **33.8** |
| FWSVD | 90% | **39.4** | **16.5** | **31.7** | **41.6** | **18.8** | **33.8** |
| TTM | 90% | 27.2 | 7.1 | 20.5 | 41.3 | 18.6 | 33.5 |
| FWTTM | 90% | 29.0 | 8.0 | 21.8 | 41.5 | **18.8** | 33.6 |

Table 8: Results of different types of compression of BART for experiments on XSUM dataset with task-oriented fine-tuning and further compression (Single-train and Double-train). The best results at each model size are in **bold**, best overall results are underlined. *Italic* results represent senseless model outputs.

than TTM, same applied for the pair of SVD and FWSVD. Therefore, it is essential to highlight that the Fisher information acquired for the language modeling task also contributes to improvements in metrics that are not directly related to LM (such as **STA** and **FL**); moreover, they are obtained by the auxiliary neural network model.

For the task of text detoxification compression has also more inconsistent results. With double-train approach quality of the models does grow linearly with ranks. But without additional finetuning models leads to generate random texts up to 90% compression rate (see Table 7) The exception is the FWSVD method, which at least restores the ability to generate meaningful text on 75% compress rate. A high STA score should not be misleading, the score itself is based on retrieving toxicity in the text, and sentences of random tokens are not toxic. This is also evident in the other metrics, which show close to zero scores.

## 8 Conclusion

In the proposed work, we explore the Transformer compression techniques that involve low-rank and tensor decomposition of its most heavy part – fully-connected layer that takes 50-60% of all the parameters depending on the exact model. We test the performance of compressed BERT and BART models on natural language understanding and generation tasks and compare the suggested compression with other popular approaches, such as pruning, distillation, and quantization. Furthermore, for the first time, we adapt the method proposed by Hsu et al. (2022) to TTM decomposition by incorporating Fisher information, which measures the importance of individual layer parameters concerning the training objectives.

Our experiments, summarized in Figure 1, show that incorporating Fisher information (FW* models) consistently improves the quality of the compression for both SVD and TTM. When addressing the NLU and the sequence-to-sequence problems, TTM and FWTTM approaches exhibit superior quality at lower ranks. Moreover, FWTTM shows comparable performance to FWSVD at 95% compression rate. In generating non-toxic text, FWTTM outperforms other methods across most metrics. However, a notable difference in metrics STA and FL distinguishes it from SIM behavior. As the compression ratio decreases, the performance gap between the methods diminishes. At a medium compression level, baseline approaches based on training-aware distillation, pruning, and quantization, demonstrate inferior or comparable performance to the respective in each group decomposition-based methods.

## 9 Limitations

In this section, we discuss the limitations of this work. First, the TTM algorithm requires more hyperparameters: core ranks, core shapes and so on. Moreover, despite being better at low and medium ranks, TTM and FWTTM have a noticeable drawback. As it can be seen from Table 3, forward passes for models, compressed with TTM and FWTTM, are slower than for SVD and even the original fully-connected layer. This is due to the fact that both in TTM and FWTTM contain more tensors, and actual multiplications require more time and power.

## 10 Potential Risks and Ethical Considerations

On the one hand, training large models from scratch can harm the environment due to resource wastage. However, our proposed model mitigates this issue by significantly reducing the number of parameters and floating point operations required for learning and fine-tuning, thereby minimizing resource consumption. This approach can potentially reduce the cost of training models to achieve desired performance levels in the future.

Additionally, it is important to highlight our commitment to ethical practices. We strictly adhere to the principle of not using private data or data that contains confidential, harmful, or discriminatory information. Instead, we rely on publicly available pre-trained models and datasets from the Hugging Face repository.

By acknowledging these potential risks and upholding ethical standards, we aim to ensure responsible and sustainable research practices.

## 11 Acknowledgements

The work was supported by the Center in the field of Artificial Intelligence in the direction of optimizing management decisions to reduce the carbon footprint on the basis of the Skolkovo Institute of Science and Technology under Contract No. 70-2021-00145/10841 dated 02.11.2021.

## References

Christopher M. Bishop and Nasser M. Nasrabadi. 2007. *Pattern Recognition and Machine Learning*. *J. Electronic Imaging*, 16(4):049901.

SA Budennyy, VD Lazarev, NN Zakharenko, AN Korovin, OA Plosskaya, DV Dimitrov, VS Akhripkin, IV Pavlov, IV Oseledets, IS Barsola, et al. 2023. Eco2ai: carbon emissions tracking of machine learning models as the first step towards sustainable ai. In *Doklady Mathematics*, pages 1–11. Springer.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Timur Garipov, Dmitry Podoprikhin, Alexander Novikov, and Dmitry P. Vetrov. 2016. Ultimate tensorization: compressing convolutional and FC layers alike. *CoRR*, abs/1611.03214.

Benjamin Hawks, Javier M. Duarte, Nicholas J. Fraser, Alessandro Pappalardo, Nhan Tran, and Yaman Umuroglu. 2021. Ps and qs: Quantization-aware pruning for efficient low latency neural network inference. *Frontiers Artif. Intell.*, 4:676564.

Yihui He, Xiangyu Zhang, and Jian Sun. 2017. Channel pruning for accelerating very deep neural networks. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 1398–1406. IEEE Computer Society.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

Oleksii Hrinchuk, Valentin Khrulkov, Leyla Mirvakhabova, Elena D. Orlova, and Ivan V. Oseledets. 2020. Tensorized embedding layers. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 4847–4860. Association for Computational Linguistics.

Yen-Chang Hsu, Ting Hua, Sungen Chang, Qian Lou, Yilin Shen, and Hongxia Jin. 2022. Language model compression with weighted low-rank factorization. In *The Tenth International Conference on Learning Representations, ICLR*

*2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Peng Hu, Xi Peng, Hongyuan Zhu, Mohamed M. Sabry Aly, and Jie Lin. 2021. OPQ: compressing deep neural networks with one-shot pruning-quantization. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 7780–7788. AAAI Press.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. Tinybert: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 4163–4174. Association for Computational Linguistics.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7871–7880. Association for Computational Linguistics.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. 2017. Pruning filters for efficient convnets. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Varvara Logacheva, Daryna Dementieva, Sergey Ustyantsev, Daniil Moskovskiy, David Dale, Irina Krotova, Nikita Semenov, and Alexander Panchenko. 2022. Paradetox: Detoxification with parallel data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 6804–6818. Association for Computational Linguistics.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 14014–14024.

Hoang Pham Minh, Nguyen Nguyen Xuan, and Tran Thai Son. 2022. Tt-vit: Vision transformer compression using tensor-train decomposition. In *Computational Collective Intelligence - 14th International Conference, ICCCI 2022, Hammamet, Tunisia, September 28-30, 2022, Proceedings*, volume 13501 of *Lecture Notes in Computer Science*, pages 755–767. Springer.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 1797–1807. Association for Computational Linguistics.

Ivan V. Oseledets, Eugene E. Tyrtyshnikov, and Nickolai Zamarashkin. 2011. Tensor-train ranks for matrices and their inverses. *Comput. Methods Appl. Math.*, 11(3):394–403.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108.

Victor Sanh, Thomas Wolf, and Alexander M. Rush. 2020. Movement pruning: Adaptive sparsity by fine-tuning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Zheng Wang, Juncheng B. Li, Shuhui Qu, Florian Metze, and Emma Strubell. 2022. Squat: Sharpness- and quantization-aware training for BERT. *CoRR*, abs/2210.07171.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. 2018. Breaking the softmax bottleneck: A high-rank RNN language model. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

## A Extended Results

In this section we provide extended results for all the experiments with average scores and standard deviations over runs with 5 different random seeds. Note that for all the experiments with BERT and BART, for both pipelines, Single-train and Double-train, adding Fisher information improves the performance of FWTTM over the standard TTM approach.

| Method | Size | AVG | STSB | CoLA | MNLI | MRCP | QNLI | QQP | RTE | SST2 | WNLI |
|---|---|---|---|---|---|---|---|---|---|---|---|
| `bert-base` | 100% | 0.79 | 0.88 | 0.57 | 0.84 | 0.90 | 0.91 | 0.87 | 0.67 | 0.92 | 0.54 |
| DistilBERT | 61% | 0.76 | 0.87 | 0.51 | 0.82 | 0.87 | 0.89 | **0.88** | 0.59 | 0.91 | 0.48 |
| `bert-base` FP-16 | 100%† | 0.78 | 0.88 | 0.55 | 0.83 | 0.88 | 0.90 | **0.88** | 0.67 | 0.91 | 0.48 |
| Block Pruning (75%) | 61% | 0.72 | 0.85 | 0.24 | 0.83 | 0.83 | 0.86 | 0.87 | 0.52 | 0.88 | 0.56 |
| SVD | | $0.37 \pm 0.01$ | $0.24 \pm 0.12$ | $0.00 \pm 0.00$ | $0.36 \pm 0.01$ | $0.20 \pm 0.09$ | $0.50 \pm 0.02$ | $0.47 \pm 0.10$ | $0.48 \pm 0.01$ | $0.52 \pm 0.02$ | $0.51 \pm 0.07$ |
| FWSVD | | $0.38 \pm 0.03$ | $0.25 \pm 0.15$ | **$0.10 \pm 0.01$** | $0.33 \pm 0.01$ | $0.39 \pm 0.33$ | $0.50 \pm 0.01$ | $0.40 \pm 0.11$ | $0.49 \pm 0.03$ | $0.51 \pm 0.01$ | **$0.56 \pm 0.00$** |
| TTM | 49% | **$0.44 \pm 0.02$** | **$0.58 \pm 0.06$** | $0.02 \pm 0.03$ | $0.37 \pm 0.01$ | $0.25 \pm 0.22$ | **$0.56 \pm 0.01$** | $0.43 \pm 0.17$ | **$0.50 \pm 0.03$** | **$0.72 \pm 0.02$** | $0.51 \pm 0.07$ |
| FWTTM | | **$0.44 \pm 0.07$** | **$0.58 \pm 0.08$** | $0.01 \pm 0.02$ | **$0.37 \pm 0.00$** | $0.26 \pm 0.23$ | $0.56 \pm 0.02$ | $0.42 \pm 0.14$ | **$0.50 \pm 0.04$** | **$0.70 \pm 0.03$** | $0.51 \pm 0.07$ |
| SVD | | $0.45 \pm 0.02$ | $0.63 \pm 0.07$ | $0.01 \pm 0.02$ | $0.36 \pm 0.01$ | $0.22 \pm 0.11$ | $0.51 \pm 0.03$ | $0.54 \pm 0.06$ | $0.54 \pm 0.06$ | $0.78 \pm 0.03$ | $0.48 \pm 0.07$ |
| FWSVD | | **$0.55 \pm 0.03$** | $0.54 \pm 0.10$ | **$0.07 \pm 0.03$** | **$0.52 \pm 0.02$** | **$0.55 \pm 0.20$** | $0.62 \pm 0.02$ | **$0.70 \pm 0.07$** | **$0.58 \pm 0.05$** | **$0.79 \pm 0.05$** | **$0.55 \pm 0.02$** |
| TTM | 63% | $0.44 \pm 0.02$ | $0.65 \pm 0.03$ | $0.01 \pm 0.02$ | $0.40 \pm 0.02$ | $0.16 \pm 0.00$ | $0.54 \pm 0.06$ | $0.52 \pm 0.14$ | $0.48 \pm 0.00$ | $0.74 \pm 0.03$ | $0.48 \pm 0.08$ |
| FWTTM | | $0.47 \pm 0.05$ | **$0.71 \pm 0.02$** | $0.00 \pm 0.02$ | $0.43 \pm 0.02$ | $0.18 \pm 0.04$ | **$0.64 \pm 0.07$** | $0.56 \pm 0.14$ | $0.47 \pm 0.01$ | $0.72 \pm 0.06$ | $0.49 \pm 0.07$ |
| SVD | | $0.70 \pm 0.02$ | $0.81 \pm 0.02$ | $0.26 \pm 0.14$ | $0.82 \pm 0.00$ | $0.69 \pm 0.22$ | $0.88 \pm 0.00$ | $0.87 \pm 0.01$ | $0.53 \pm 0.06$ | $0.90 \pm 0.01$ | $0.53 \pm 0.08$ |
| FWSVD | | **$0.78 \pm 0.01$** | **$0.88 \pm 0.00$** | $0.55 \pm 0.02$ | **$0.84 \pm 0.00$** | $0.87 \pm 0.01$ | **$0.90 \pm 0.00$** | **$0.88 \pm 0.01$** | $0.64 \pm 0.01$ | **$0.92 \pm 0.01$** | **$0.55 \pm 0.04$** |
| TTM | 95% | $0.76 \pm 0.01$ | $0.87 \pm 0.00$ | $0.52 \pm 0.02$ | $0.79 \pm 0.00$ | $0.86 \pm 0.01$ | $0.87 \pm 0.01$ | $0.86 \pm 0.00$ | $0.65 \pm 0.01$ | $0.91 \pm 0.01$ | $0.48 \pm 0.01$ |
| FWTTM | | $0.77 \pm 0.02$ | **$0.88 \pm 0.00$** | **$0.56 \pm 0.02$** | $0.83 \pm 0.00$ | **$0.88 \pm 0.01$** | **$0.90 \pm 0.00$** | **$0.88 \pm 0.01$** | **$0.66 \pm 0.01$** | **$0.92 \pm 0.01$** | $0.46 \pm 0.10$ |

Table 9: Results of different types of compression of BERT for experiment with task-oriented fine-tuning and further compression (Single-train). The best results at each model size are in **bold**, best overall results are underlined.

| Method | Size | AVG | STSB | CoLA | MNLI | MRCP | QNLI | QQP | RTE | SST2 | WNLI |
|---|---|---|---|---|---|---|---|---|---|---|---|
| `bert-base` | 100% | **0.79** | 0.88 | **0.57** | **0.84** | **0.90** | **0.91** | 0.87 | 0.67 | **0.92** | 0.54 |
| DistilBERT | 61% | 0.76 | 0.87 | 0.51 | 0.82 | 0.87 | 0.89 | 0.88 | 0.59 | 0.91 | 0.48 |
| `bert-base` FP-16 | 100%† | 0.78 | 0.88 | 0.55 | 0.83 | 0.88 | 0.90 | 0.88 | 0.67 | 0.91 | 0.48 |
| B.P. (75%) | 61% | 0.72 | 0.85 | 0.24 | 0.83 | 0.83 | 0.86 | 0.87 | 0.52 | 0.88 | 0.56 |
| SVD | | $0.68 \pm 0.01$ | **$0.83 \pm 0.00$** | $0.00 \pm 0.01$ | **$0.79 \pm 0.01$** | $0.79 \pm 0.01$ | **$0.85 \pm 0.00$** | **$0.87 \pm 0.00$** | $0.59 \pm 0.01$ | **$0.87 \pm 0.00$** | $0.49 \pm 0.08$ |
| FWSVD | | $0.68 \pm 0.01$ | $0.82 \pm 0.01$ | $0.04 \pm 0.05$ | **$0.79$** $\pm 0.00$ | $0.79 \pm 0.00$ | **$0.85 \pm 0.01$** | **$0.87 \pm 0.00$** | $0.56 \pm 0.03$ | $0.86 \pm 0.00$ | **$0.54 \pm 0.06$** |
| TTM | 49% | **$0.69 \pm 0.01$** | **$0.83 \pm 0.00$** | **$0.15 \pm 0.01$** | $0.78 \pm 0.00$ | **$0.81 \pm 0.01$** | $0.84 \pm 0.00$ | **$0.87 \pm 0.00$** | **$0.60 \pm 0.01$** | $0.86 \pm 0.01$ | $0.43 \pm 0.08$ |
| FWTTM | | $0.68 \pm 0.01$ | **$0.83 \pm 0.00$** | **$0.14 \pm 0.02$** | $0.78 \pm 0.00$ | **$0.81 \pm 0.00$** | $0.84 \pm 0.00$ | **$0.87 \pm 0.00$** | **$0.60 \pm 0.01$** | $0.86 \pm 0.00$ | $0.41 \pm 0.09$ |
| SVD | | $0.75 \pm 0.02$ | $0.86 \pm 0.00$ | $0.43 \pm 0.02$ | **$0.83 \pm 0.00$** | $0.84 \pm 0.01$ | **$0.89 \pm 0.00$** | **$0.88 \pm 0.01$** | $0.64 \pm 0.02$ | **$0.90 \pm 0.01$** | $0.50 \pm 0.10$ |
| FWSVD | | **$0.77 \pm 0.00$** | **$0.87 \pm 0.00$** | **$0.47 \pm 0.02$** | **$0.83 \pm 0.00$** | **$0.85 \pm 0.01$** | **$0.89 \pm 0.01$** | **$0.88 \pm 0.01$** | **$0.65 \pm 0.01$** | **$0.90 \pm 0.01$** | **$0.56 \pm 0.01$** |
| TTM | 63% | $0.70 \pm 0.01$ | $0.85 \pm 0.00$ | $0.10 \pm 0.10$ | $0.81 \pm 0.00$ | $0.81 \pm 0.01$ | $0.86 \pm 0.00$ | **$0.88 \pm 0.01$** | $0.61 \pm 0.01$ | $0.88 \pm 0.00$ | $0.49 \pm 0.09$ |
| FWTTM | | $0.70 \pm 0.02$ | $0.85 \pm 0.00$ | $0.15 \pm 0.05$ | $0.82 \pm 0.00$ | $0.82 \pm 0.01$ | $0.86 \pm 0.00$ | **$0.88 \pm 0.01$** | $0.62 \pm 0.02$ | $0.89 \pm 0.01$ | $0.46 \pm 0.06$ |
| SVD | | $0.78 \pm 0.01$ | **$0.89 \pm 0.00$** | **$0.56 \pm 0.02$** | **$0.84 \pm 0.00$** | **$0.88 \pm 0.02$** | **$0.91 \pm 0.00$** | **$0.89 \pm 0.01$** | $0.68 \pm 0.01$ | $0.91 \pm 0.01$ | $0.44 \pm 0.08$ |
| FWSVD | | **$0.79 \pm 0.04$** | **$0.89 \pm 0.00$** | **$0.56 \pm 0.03$** | **$0.84 \pm 0.00$** | **$0.88 \pm 0.01$** | $0.90 \pm 0.00$ | **$0.89 \pm 0.01$** | **$0.69 \pm 0.01$** | $0.91 \pm 0.01$ | **$0.51 \pm 0.08$** |
| TTM | 95% | $0.77 \pm 0.05$ | $0.88 \pm 0.00$ | $0.52 \pm 0.03$ | $0.83 \pm 0.00$ | $0.83 \pm 0.06$ | $0.89 \pm 0.00$ | $0.88 \pm 0.00$ | $0.68 \pm 0.02$ | $0.90 \pm 0.00$ | **$0.51 \pm 0.07$** |
| FWTTM | | $0.78 \pm 0.02$ | **$0.89 \pm 0.00$** | $0.54 \pm 0.01$ | $0.83 \pm 0.00$ | **$0.88 \pm 0.01$** | $0.90 \pm 0.00$ | **$0.89 \pm 0.00$** | $0.67 \pm 0.04$ | **$0.91 \pm 0.01$** | $0.49 \pm 0.07$ |

Table 10: Results of different types of compression of BERT for experiments with task-oriented fine-tuning, compression, and further fine-tuning (Double-train). The best results at each model size are in **bold**, best overall results are underlined.

According to the Table 9, both TTM and FWTTM provide the best average score across all tasks. Moving on to the lower compression rates, FWTTM is outperforming both TTM and vanilla SVD approach, having 0.47 average GLUE score compared to 0.44 and 0.45 for TTM and SVD, respectively. Moreover, on the 95% compression rate despite being slightly worse than FWSVD on average, FWTTM demonstrates better scores for CoLA, the most challenging task from the whole dataset.

In the Double-train setup (Table 10), FWSVD performs better than others in terms of the average scores. FwTTM, however, shows comparable to FWSVD performance at 49% and 95% compression rates being 0.01 worse on average, but this difference is not statistically significant.

For the task of detoxification, in the Single-train setup, all the models fail to generate anything meaningful on all the compression rates up to 90% (see Table 11), the same is correct for summarization (Table 12). We provide generation examples for both detoxification and summarization in Tables 13 and 14, respectively.

In the Double-train pipeline FWTTM is the best by **SIM**, **FL** and **J** at 60% compression rate, and has the best overall **SIM** score at 90% comression rate.

| Pipeline | | Single-train | | | | Double-train | | | |
|---|---|---|---|---|---|---|---|---|---|
| Method | C. Rate | STA | SIM | FL | J | STA | SIM | FL | J |
| `bart-base` | 100 % | 0.89 | 0.60 | 0.82 | 0.44 | 0.89 | 0.60 | 0.82 | 0.44 |
| FP16 eval.$^\dagger$ | 100 % | 0.89 | 0.60 | 0.82 | 0.44 | 0.89 | 0.60 | 0.82 | 0.44 |
| B.P. (95%) | 63% | - | - | - | - | 0.92 | 0.34 | 0.30 | 0.12 |
| B.P. (65%) | 74% | - | - | - | - | 0.82 | 0.60 | 0.73 | 0.36 |
| SVD | 60% | *0.97 ± 0.04* | *0.18 ± 0.01* | *0.10 ± 0.05* | *0.01 ± 0.02* | 0.75 ± 0.01 | **0.59 ± 0.01** | 0.65 ± 0.01 | 0.28 ± 0.01 |
| FWSVD | | *0.32 ± 0.01* | *0.46 ± 0.01* | *0.58 ± 0.01* | *0.07 ± 0.01* | **0.78 ± 0.02** | **0.59 ± 0.01** | 0.68 ± 0.00 | **0.30 ± 0.01** |
| TTM | | *0.97 ± 0.04* | *0.19 ± 0.02* | *0.16 ± 0.04* | *0.03 ± 0.01* | 0.74 ± 0.02 | 0.58 ± 0.01 | 0.64 ± 0.02 | 0.27 ± 0.01 |
| FWTTM | | *0.82 ± 0.03* | *0.17 ± 0.02* | *0.14 ± 0.02* | *0.01 ± 0.01* | 0.77 ± 0.00 | **0.59 ± 0.00** | **0.69 ± 0.00** | **0.30 ± 0.00** |
| SVD | 74% | *0.85 ± 0.06* | *0.21 ± 0.01* | *0.14 ± 0.05* | *0.03 ± 0.01* | 0.82 ± 0.01 | 0.60 ± 0.01 | 0.77 ± 0.01 | 0.38 ± 0.01 |
| FWSVD | | *0.32 ± 0.01* | *0.46 ± 0.01* | *0.58 ± 0.01* | *0.07 ± 0.01* | **0.87 ± 0.00** | 0.61 ± 0.01 | **0.80 ± 0.01** | **0.42 ± 0.01** |
| TTM | | *0.99 ± 0.02* | *0.17 ± 0.01* | *0.06 ± 0.07* | *0.01 ± 0.01* | 0.82 ± 0.01 | 0.61 ± 0.01 | 0.75 ± 0.01 | 0.37 ± 0.01 |
| FWTTM | | *0.97 ± 0.01* | *0.17 ± 0.02* | *0.45 ± 0.00* | *0.08 ± 0.01* | 0.84 ± 0.01 | **0.62 ± 0.00** | 0.76 ± 0.01 | 0.38 ± 0.01 |
| SVD | 90% | **0.85 ± 0.01** | 0.42 ± 0.01 | 0.72 ± 0.01 | 0.25 ± 0.01 | 0.86 ± 0.00 | 0.61 ± 0.00 | 0.81 ± 0.00 | **0.43 ± 0.00** |
| FWSVD | | 0.70 ± 0.10 | 0.64 ± 0.01 | **0.82 ± 0.00** | **0.35 ± 0.01** | **0.87 ± 0.01** | 0.61 ± 0.00 | 0.81 ± 0.00 | **0.43 ± 0.00** |
| TTM | | 0.49 ± 0.17 | 0.60 ± 0.01 | 0.71 ± 0.00 | 0.18 ± 0.01 | 0.86 ± 0.01 | 0.61 ± 0.00 | 0.80 ± 0.01 | 0.41 ± 0.01 |
| FWTTM | | 0.44 ± 0.02 | **0.68 ± 0.01** | 0.78 ± 0.02 | 0.21 ± 0.01 | 0.86 ± 0.00 | **0.62 ± 0.00** | **0.82 ± 0.00** | **0.43 ± 0.00** |

Table 11: Results of different types of compression for the `bart-base` model for experiments with detoxification with task-oriented fine-tuning, compression, and further fine-tuning (Single-train and Double-train). The best results at each model size are in **bold**, best overall results are underlined. *Italic* results represent senseless model outputs.

| Pipeline | | Single-train | | | Double-train | | |
|---|---|---|---|---|---|---|---|
| Metric | | ROUGE | | | ROUGE | | |
| Method | C. Rate | 1 | 2 | L | 1 | 2 | L |
| `bart-base` | 100% | 42.4 | 19.6 | 34.5 | 42.4 | 19.6 | 34.5 |
| FP16 eval. | 100% | 32.8 | 11.0 | 25.5 | 32.8 | 11.0 | 25.5 |
| Block Pruning (95%) | 63% | - | - | - | 23.4 | 5.7 | 18.8 |
| Block Pruning (65%) | 74% | - | - | - | 34.6 | 12.2 | 27.9 |
| SVD | 60% | *6.3 ± 0.6* | *0.5 ±* **0.1** | 5.2 ± 0.4 | 35.6 ± 0.0 | 13.4 ± 0.0 | 28.2 ± 0.0 |
| FWSVD | | ***8.1 ± 2.3*** | ***0.5 ± 0.2*** | ***6.8 ± 1.7*** | 35.8 ± 0.1 | 13.6 ± 0.0 | 28.4 ± 0.1 |
| TTM | | *4.2 ± 0.4* | *0.2 ± 0.0* | *3.7 ± 0.4* | **36.1** ± 0.1 | **13.9 ± 0.1** | **28.6 ± 0.1** |
| FWTTM | | *5.0 ± 0.6* | *0.2 ± 0.0* | *4.2 ± 0.5* | 36.0 ± 0.1 | 13.8 ± 0.1 | **28.6 ± 0.1** |
| SVD | 74% | *8.1 ± 1.3* | *0.5 ± 0.2* | *6.9 ± 1.0* | 40.2 ± 0.1 | 17.4 ± 0.1 | 32.4 ± 0.0 |
| FWSVD | | **21.2 ± 0.9** | **4.4 ± 0.4** | **16.5 ± 0.6** | **40.6 ± 0.1** | **17.8 ± 0.1** | **32.9 ± 0.1** |
| TTM | | *6.0 ± 1.6* | *0.4 ± 0.2* | *4.9 ± 1.3* | 39.3 ± 0.1 | 16.7 ± 0.1 | 31.5 ± 0.1 |
| FWTTM | | *7.3 ± 1.7* | *0.4 ± 0.2* | *5.9 ± 1.2* | 39.6 ± 0.1 | 16.9 ± 0.0 | 31.8 ± 0.0 |
| SVD | 90% | 30.8 ± 0.9 | 9.9 ± 0.5 | 23.8 ± 0.8 | **41.6 ± 0.1** | **18.8 ± 0.0** | **33.8 ± 0.1** |
| FWSVD | | **39.4 ± 0.2** | **16.5 ± 0.1** | **31.7 ± 0.1** | **41.6 ± 0.1** | **18.8 ± 0.1** | **33.8 ± 0.1** |
| TTM | | 27.2 ± 0.3 | 7.1 ± 0.3 | 20.5 ± 0.4 | 41.3 ± 0.1 | 18.6 ± 0.1 | 33.5 ± 0.1 |
| FWTTM | | 29.0 ± 0.3 | 8.0 ± 0.2 | 21.8 ± 0.3 | 41.5 ± 0.1 | **18.8 ± 0.0** | 33.6 ± 0.1 |

Table 12: Results of different types of compression of BART for experiments on XSUM dataset with task-oriented fine-tuning and further compression (Single-train and Double-train). The best results at each model size are in **bold**, best overall results are underlined. *Italic* results represent senseless model outputs.

In summarization, FWSVD is a leader in both Single-train and Double-train pipelines. Moreover, FWSVD is the only method not to hallucinate at 74% compression rate in Single-train. FWTTM, however, consistently outperforms TTM by most metrics on all compression rates in the Double-train pipeline.

## B Generation examples for Sequence-to-Sequence Tasks

| Approach | C. Rate | Source (toxic) | Generated (neural) |
|---|---|---|---|
| Ground Truth | - | or are you pulling this out of your ass ? | Are you making things up out of thin air? |
| | | this piece of human garbage should rot in jail. | this criminal should be in jail |
| `bart-detox` | - | or are you pulling this out of your ass ? | or are you pulling this out of your behind?. |
| | | this piece of human garbage should rot in jail. | this piece of human should rot in jail. |
| SVD | 60% | or are you pulling this out of your ass ? | *worldly, you you you your your your life life life...The South South South south south south* |
| | | this piece of human garbage should rot in jail. | *worldly. garbage garbage garbage, human human human::: jail jail. jail. The prison.* |
| TTM | 60% | or are you pulling this out of your ass ? | *in the D D D M M M S S S N N N T T T Th Th Th* |
| | | this piece of human garbage should rot in jail. | *in the jail jail jail prison prison prison back back back D D D T T T M M* |
| FWSVD | 60% | or are you pulling this out of your ass ? | *absoror are you pull pull pull out out out of the ass ass ass???...* |
| | | this piece of human garbage should rot in jail. | *this piece piece piece human human human garbage garbage rot rot rot in jail jail jail prison jail jail* |
| FWTTM | 60% | or are you pulling this out of your ass ? | *in the D D D D D D D D D D D D D D D D D D D D* |
| | | this piece of human garbage should rot in jail. | *in the prison prison prison prison prison D D D D D D D D D D D D D D* |
| SVD | 74% | or are you pulling this out of your ass ? | *you are not out of the country. may may be be your out out out on out out* |
| | | this piece of human garbage should rot in jail. | *this garbage garbage garbage trash trash trash be jail jail. not jail jail jail prison. jail jail* |
| TTM | 74% | or are you pulling this out of your ass ? | *, is is just just just about about about the other other other line line line....* |
| | | this piece of human garbage should rot in jail. | *not not be be be in other other other line line line lines lines lines....* |
| FWSVD | 74% | or are you pulling this out of your ass ? | *or are you pulling this out out out of your ass?...,.. 3.* |
| | | this piece of human garbage should rot in jail. | *this piece of human garbage should rot rot rot in jail....,.. ).* |
| FWTTM | 74% | or are you pulling this out of your ass ? | *in the t t t t t t t t t t t t t t t t t t t t* |
| | | this piece of human garbage should rot in jail. | *to be be be be be be be be be be be be be be be be be be be be be be be be be be be be be be be be* |
| SVD | 90% | or are you pulling this out of your ass ? | or are you pulling this out of you?I don't know, I'm in a bad mood |
| | | this piece of human garbage should rot in jail. | This piece should rot in jail.It's not good to have it jailed. |
| TTM | 90% | or are you pulling this out of your ass ? | *or are you pulling this out of your ass?.* |
| | | this piece of human garbage should rot in jail. | *This piece of human garbage should rot in jail.* |
| FWSVD | 90% | or are you pulling this out of your ass ? | or are you pulling this out of your behind?. |
| | | this piece of human garbage should rot in jail. | this piece of human should rot in jail. |
| FWTTM | 90% | or are you pulling this out of your ass ? | Or are you pulling this out? |
| | | this piece of human garbage should rot in jail. | this piece of human garbage should rot in jail. |

Table 13: Detoxification examples for different `bart-base` compression ratios for fine-tuning and further compression (Single-train). Senseless outputs depicted with *italic*.

| **Approach** | **C. Rate** | **Examples** |
|---|---|---|
| Document | - | Four police officers were injured in the incident on Friday night. A man, aged 19, and a boy, aged 16, have been charged with six counts of aggravated vehicle taking. They are due to appear before Belfast Magistrates' Court on Monday. The 19-year-old man has also been charged with driving while disqualified and using a motor vehicle without insurance. |
| Summary | - | Two teenagers have been charged in connection with an incident in west Belfast in which a car collided with two police vehicles. |
| SVD | 60% | *in the the the 19 19— man man man woman woman woman man man in in in a a driving driving driving* |
| TTM | 60% | *in the the the police police police in a a a one one one the* |
| FWSVD | 60% | *in in in the the the as as as a a a in a a morning morning morning* |
| FWTTM | 60% | *in in in the the the police police police in a a a high high high High* |
| SVD | 74% | *a a " """"))))):):):: the the the range of range range range ranges ranges ranges between between between* |
| TTM | 74% | *...:::: T T T had had had been been a a a other other other have have have* |
| FWSVD | 74% | Two men have been charged with six counts of the aggravated vehicle taking. |
| FWTTM | 74% | *people people people, have have have been been been in a a political political political* |
| SVD | 90% | A man and a woman have been charged with a series of offences, including a charge of possession of a vehicle without a licence or licence licence. |
| TTM | 90% | Three men have been charged with offences relating relating to a police incident in north west west west. |
| FWSVD | 90% | Four people have been charged after a car was stolen during a police operation in Belfast. |
| FWTTM | 90% | Two men have been charged with offences of aggravated vehicle taking, following a police incident in Belfast. |

Table 14: Summarization examples for different `bart-base` compression ratios for fine-tuning and further compression (Single-train). Senseless outputs depicted with *italic*.

## C Experimental details

| Dataset | Learning Rate | Epochs | Batch Size | Weight Decay | Dropout | Max Length |
|---|---|---|---|---|---|---|
| GLUE (Single-train) | 5e-5 | 3 | 32 | 0.01 | 0.1 | 512 |
| GLUE (Double-train) | 2e-5 | 3 | 32 | 0.0 | 0.1 | 512 |
| XSUM (Single-train) | 8e-5 | 5 | 32 | 0.01 | 0.1 | 512 |
| XSUM (Double-train) | 3e-5 | 3 | 32 | 0.01 | 0.1 | 512 |
| ParaDetox (Single-train) | 5e-5 | 5 | 16 | 1e-5 | 0.1 | 512 |
| ParaDetox (Double-train) | 1e-5 | 3 | 16 | 1e-5 | 0.1 | 512 |

Table 15: Hyperparameters for the experiments.

This section provides additional experimental setups to supplement the main experimental section. We conducted experiments on 4 NVIDIA RTX 3090 GPUs, with 5 different random seeds per each experiment. We have optimized hyperparameters for each experiment individually for all the values specified on the corresponding row in table 15. The weight hyperparameter $\lambda$ is adjusted over the range of 0 to 0.1, using a step size of 0.005. For each step, three tasks were taken from GLUE: STSB, CoLA, and RTE and two tables were constructed for the Single-train and Double-train setups. We selected the best $\lambda$ value in terms of performance quality on these tasks. The final hyperparameter $\lambda = 0.06$ was used for all experiments.

## D Additional plots for all experimental setups

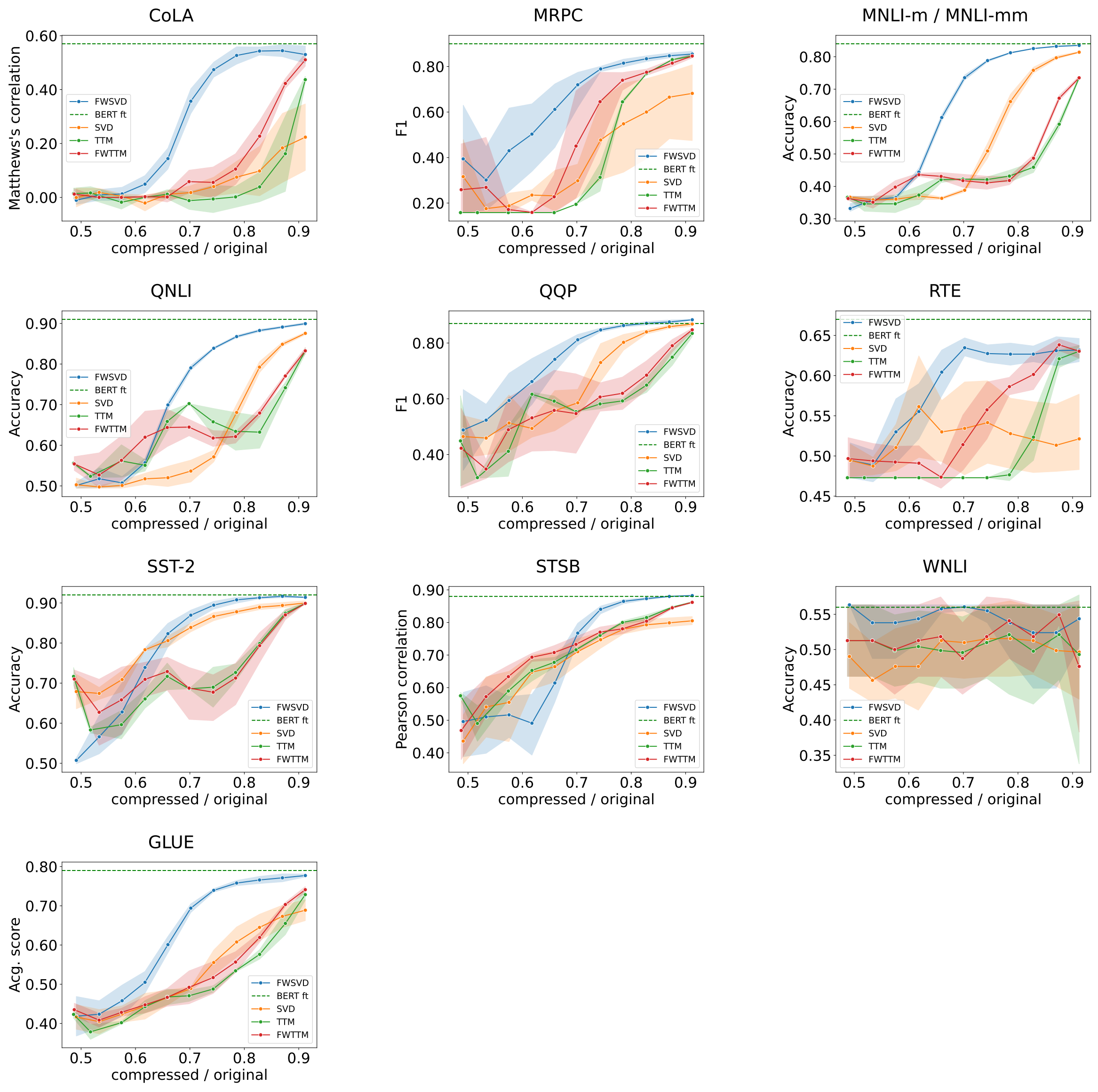

Figure 2: Performance of `bert-base-uncased` model with task-oriented fine-tuning and further compression (Single-train) on a GLUE benchmark.

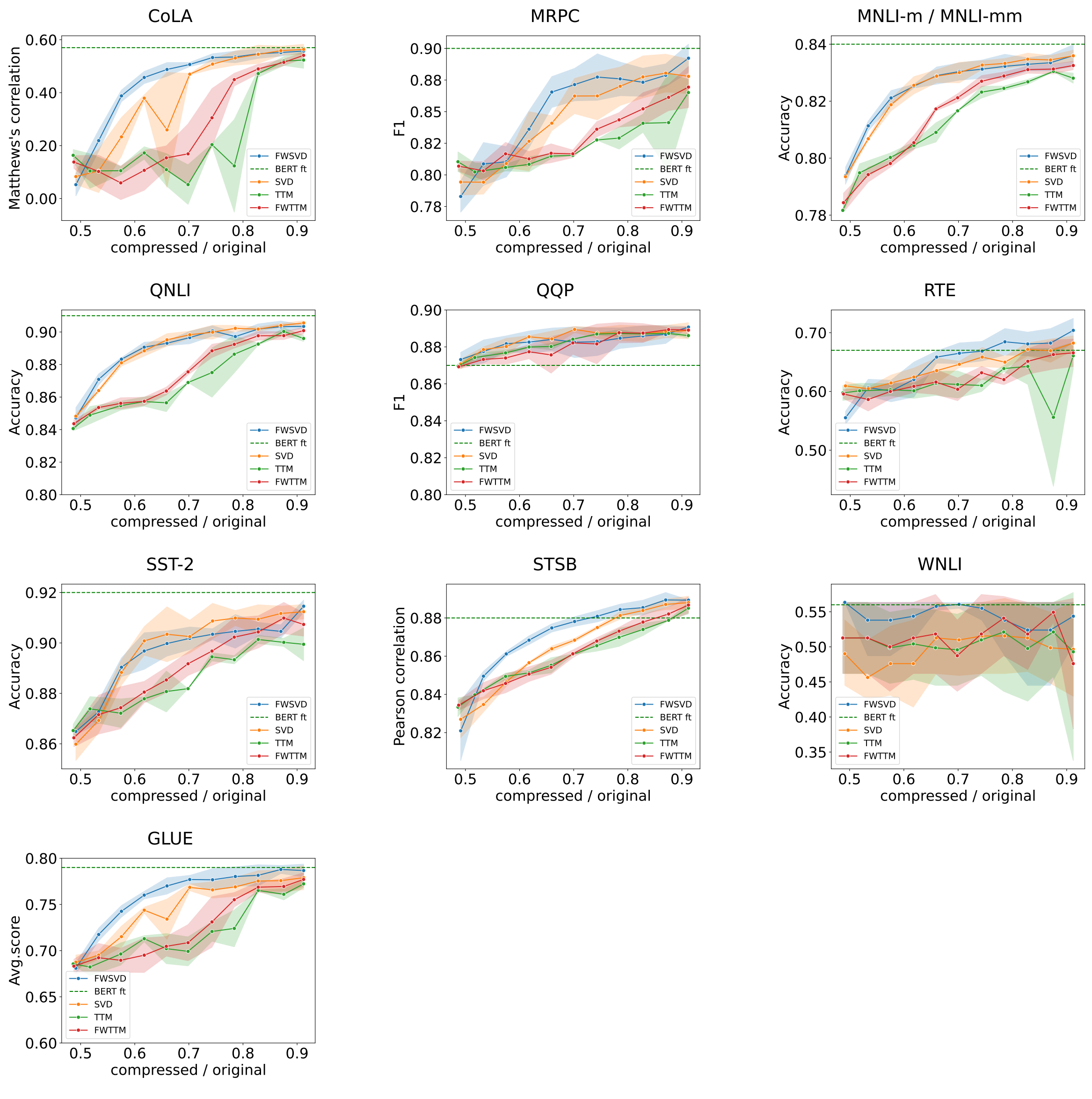

Figure 3: Performance of `bert-base-uncased` model with task-oriented fine-tuning, compression and further fine-tuning (Double-train) on a GLUE benchmark.

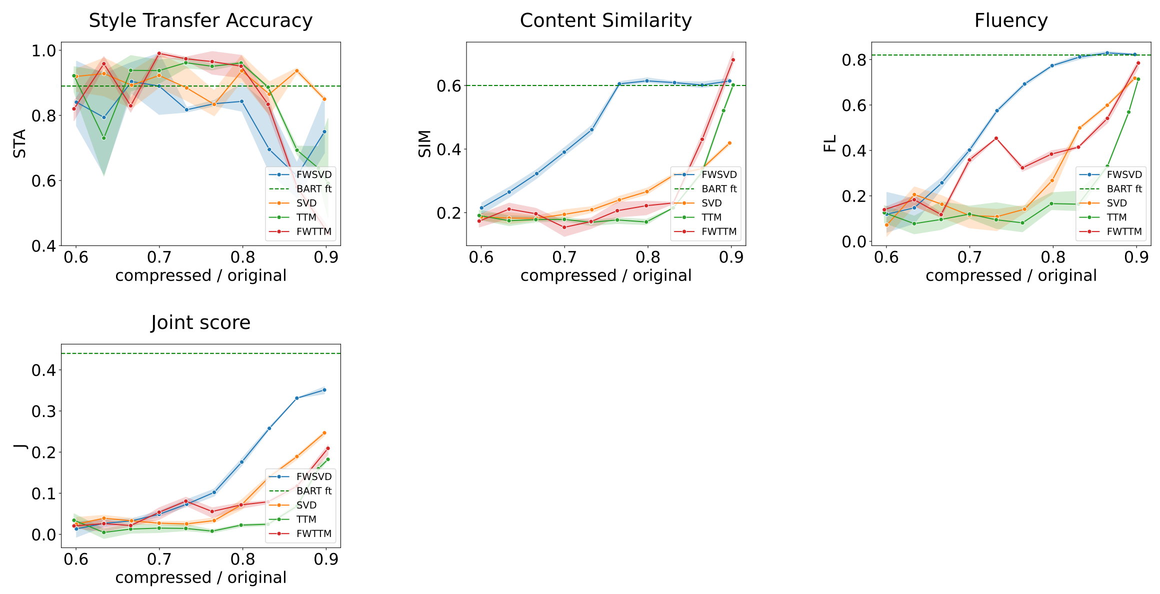

Figure 4: Performance of `bart-base` model with task-oriented fine-tuning and further compression (Single-train) on a text detoxification task.

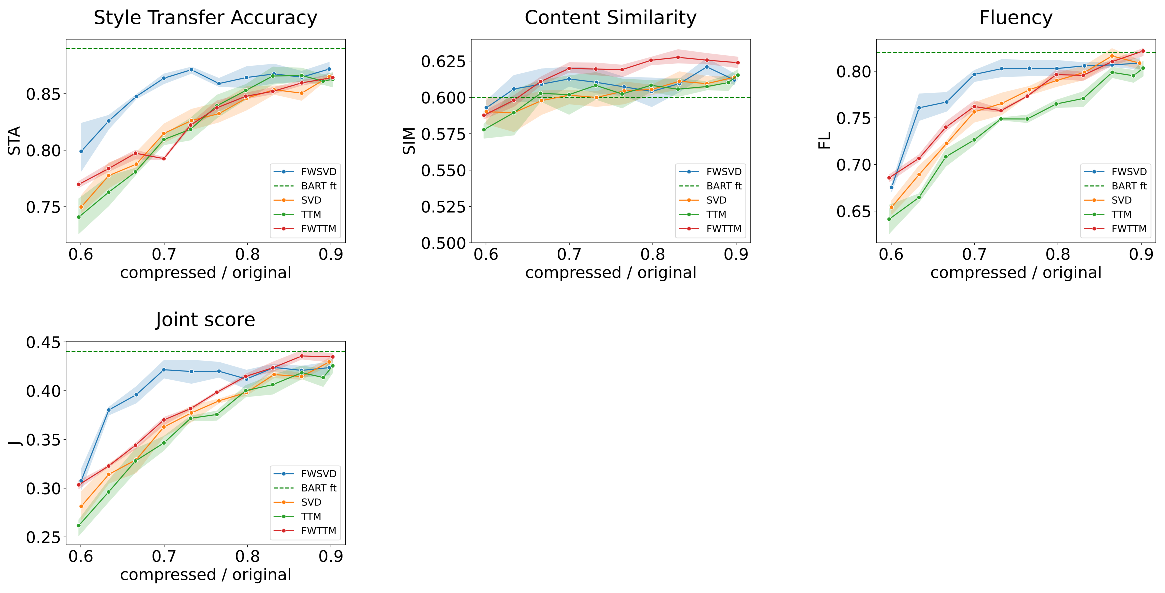

Figure 5: Performance of `bart-base` model with task-oriented fine-tuning, compression and further fine-tuning (Double-train) on a text detoxification task.

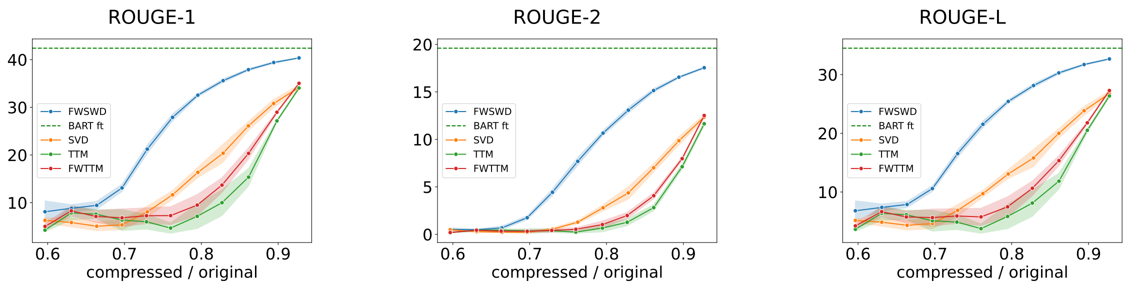

Figure 6: Performance of `bart-base` model with task-oriented fine-tuning and further compression (Single-train) on a XSUM benchmark.

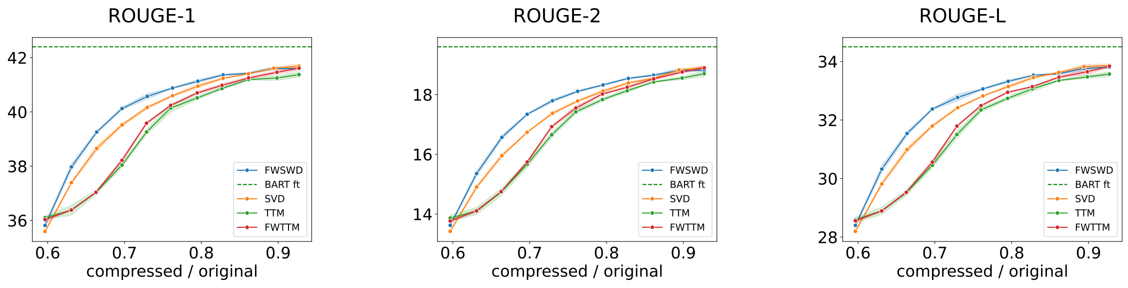

Figure 7: Performance of `bart-base` model with task-oriented fine-tuning, compression and further fine-tuning (Double-train) on a XSUM benchmark.