SIMON FRASER UNIVERSITY

ENSC 861 FINAL PROJECT

Efficient Storage of Fine-Tuned Transformer Models via Data-Free Delta Compression

Author:
MohammadReza SADEGHIAN

Supervisor:
Dr. Jie LIANG

August 11, 2025

Abstract

Efficient Storage of Fine-Tuned Transformer Models via Data-Free Delta Compression

by MohammadReza SADEGHIAN

With the pre-train and fine-tune paradigm becoming mainstream for Large Language Models (LLMs), the high storage and deployment costs of maintaining numerous task-specific models pose a significant challenge. Delta compression methods aim to mitigate this by storing only the difference (delta) between a pre-trained base model and its fine-tuned counterpart. This project begins by implementing Delta-DCT, a data-free compression method that leverages the Discrete Cosine Transform (DCT). As a primary contribution, I propose and develop a novel alternative based on the Discrete Wavelet Transform (DWT) and other JPEG-inspired methods. My experimental results on language and vision transformer models indicate that the proposed DWT-based approach shows significant promise. Under certain configurations, it improves upon the original DCT method by achieving a higher compression ratio with comparable or even superior accuracy. This work highlights the potential of DWT as an effective and powerful alternative for data-free delta parameter compression.

Contents

Introduction

1.1 The Problem and Its Significance

In recent years, Large Language Models (LLMs) and foundation models based on the Transformer architecture have revolutionized the field of artificial intelligence Huang et al., 2025. The dominant paradigm for leveraging these models involves two stages: a large-scale pre-training phase on internet-scale data, followed by a fine-tuning phase to adapt the model to specific tasks or user needs Liu et al., 2024. This approach has endowed models with unprecedented flexibility, enabling them to excel in diverse applications ranging from instruction following and question answering to specialized professional domains Ping et al., 2024.

This success paints an exciting vision of a future where millions of unique, personalized models are served concurrently. However, realizing this vision is hindered by two fundamental challenges:

1. **High Storage Cost:** Each fine-tuned model is a full, large-sized copy of the base model's parameters. Storing thousands or millions of these multi-gigabyte models requires enormous storage capacity and presents significant management overhead Liu et al., 2024.

2. **High Serving Cost:** Concurrently serving multiple distinct models is prohibitively expensive, as each model demands dedicated GPU memory. The process of loading different models from disk into GPU memory is also time-consuming, leading to increased latency for end-users Liu et al., 2024.

To overcome these obstacles, the concept of **Delta Compression** has emerged. The core idea is to store only a single, high-precision copy of the pre-trained base model and, for each task, store only the compressed *difference* or *delta* of the parameters (i.e., $W_{\text{finetune}} - W_{\text{base}}$) Huang et al., 2025. Given that the vast majority of a model's knowledge is acquired during pre-training, it is hypothesized that these delta parameters contain significant redundancy and are thus highly compressible Liu et al., 2024. Successfully compressing this delta can dramatically reduce storage and serving costs, paving the way for the widespread deployment of customized AI models.

1.2 Background and Literature Review

The idea of delta compression has garnered significant research interest, leading to several distinct approaches. Early attempts focused on low-rank approximations like Singular Value Decomposition (SVD), but studies revealed that the deltas from full fine-tuning are often high-rank, limiting the effectiveness of this method alone Liu et al., 2024; Ping et al., 2024. More recent and prominent approaches include BitDelta and Delta-CoMe, which represent two different philosophies in delta compression.

BitDelta: Simplicity and Speed with 1-bit Quantization. The BitDelta method, proposed by Liu et al. Liu et al., 2024, introduces a simple yet remarkably effective technique. *Its strengths and methodology* lie in its aggressive quantization of the delta parameters to a mere 1-bit. This is achieved by taking the sign of each weight in the delta matrix and applying a single, high-precision scaling factor across the entire layer Liu et al., 2024. This factor is initialized to minimize the L_2 error and is further refined using a brief knowledge distillation step Liu et al., 2024. The primary advantages of BitDelta are its computational simplicity, extremely high compression ratio (>10x), and significant impact on reducing GPU memory and multi-tenant serving latency Liu et al., 2024. It performs particularly well on chat-based fine-tunes Liu et al., 2024. *Its limitations*, however, stem from this same simplicity. The uniform 1-bit quantization is an aggressive, lossy approach that fails to differentiate between parameters of varying importance. As noted by competitors, this method can severely degrade the performance of models fine-tuned for specialized tasks like mathematics or code generation, where numerical precision is critical Ping et al., 2024.

Delta-CoMe: High Fidelity via SVD-based Mixed-Precision. To address the limitations of simpler methods, the Delta-CoMe approach was proposed, offering a more sophisticated and precise solution Ping et al., 2024. *Its strengths and methodology* are inspired by the long-tail distribution of singular values in the delta matrix. Delta-CoMe first decomposes the delta using SVD and then applies a **mixed-precision quantization** strategy Ping et al., 2024. Singular vectors corresponding to larger (more important) singular values are quantized with higher precision (e.g., 8-bit or 3-bit), while less critical vectors receive fewer bits (e.g., 2-bit) Ping et al., 2024. This hybrid low-rank and low-bit approach allows it to preserve vital information with high fidelity. Consequently, Delta-CoMe achieves near-lossless performance across a broader range of tasks, including the specialized math and code domains where BitDelta struggles Ping et al., 2024. *Its limitations* are the trade-off for its high performance. The method is computationally more intensive due to the SVD step. Furthermore, it relies on the GPTQ algorithm for quantization, which requires a **calibration dataset** Ping et al., 2024. This dependency makes the method not entirely "data-free" and reliant on representative data to achieve its optimal performance.

Delta-DCT: The Data-Free Middle Ground. The baseline method for my project, Delta-DCT, carves out a middle ground Huang et al., 2025. Inspired by JPEG image compression, it avoids the complexity of SVD by operating on small patches of weights. It uses a simple L2-norm for importance scoring and applies the computationally efficient Discrete Cosine Transform (DCT). Its key advantage is being a completely data-free and training-free method, making it more straightforward to deploy than Delta-CoMe, yet more nuanced than BitDelta Huang et al., 2025.

1.3 My Project's Goals and Contributions

This project begins with a faithful implementation of the Delta-DCT method to serve as a strong baseline Huang et al., 2025. However, the primary goal is to innovate upon this foundation by proposing and evaluating a novel extension that yields superior performance.

The central contribution of this work is the introduction and analysis of the **Discrete Wavelet Transform (DWT)** as a more efficient alternative to DCT for this compression task. I will demonstrate through a series of comprehensive experiments that my proposed DWT-based framework, particularly when combined with a selective coefficient-keeping strategy, achieves a **significantly higher compression ratio** than the DCT baseline while maintaining, and in some cases exceeding, the original model's accuracy. This report will provide evidence that DWT is not just a viable alternative, but a more powerful and flexible tool for data-free delta compression.

Project Description

2.1 Methodology

2.1.1 The Baseline Method: Delta Compression with DCT

The foundation of this project is the implementation of the Delta-DCT method, a data-free delta compression algorithm introduced by Huang et al. Huang et al., 2025. This approach draws inspiration from the classic JPEG image compression pipeline to efficiently compress the delta parameters of fine-tuned models. The entire process is training-free and requires no calibration data. My implementation of this baseline method follows a sequence of steps for each targeted layer.

Step 1: Delta Parameter Calculation

The process begins by identifying the changes introduced during fine-tuning. For each corresponding weight tensor in the pre-trained model (W_{base}) and the fine-tuned model (W_{finetune}), the delta parameter matrix (Δ) is calculated through element-wise subtraction:

$$\Delta = W_{\text{finetune}} - W_{\text{base}}$$

This Δ matrix represents the complete information added during the fine-tuning process and is the target for our compression.

Step 2: Patchlization

Similar to how JPEG divides an image into blocks, the 2D delta matrix Δ is partitioned into smaller, non-overlapping square patches. A configurable patch size, $p \times p$ (e.g., 16×16), is used. If the dimensions of the delta matrix are not perfectly divisible by the patch size, zero-padding is applied to ensure complete coverage.

Step 3: Importance Assessment

Not all changes made during fine-tuning are equally important. To prioritize the more significant modifications, an importance score is calculated for each patch using its Frobenius (L_2) norm. The rationale is that patches with a larger norm contain larger magnitude changes, indicating a greater contribution to the model's performance on the target task.

$$I_k = ||P_k||_F = \sqrt{\sum_{i=1}^{p}\sum_{j=1}^{p}(P_{k_{ij}})^2}$$

Step 4: Mixed-Precision Bit-width Allocation

Based on the calculated importance scores, a mixed-precision quantization strategy is employed. Patches are sorted by their scores, and a pre-defined bit allocation strategy is applied. **A range of strategies were explored in my experiments, from simple 50/50 splits (e.g., 50% of patches at 2 bits, 50% at 0 bits) to more granular triple-precision allocations (e.g., assigning 4-bit, 2-bit, and**

0-bit precision to different tiers of patches). This ensures that the most significant changes are stored with higher fidelity.

Step 5: DCT and Quantization

This is the core compression step. Each patch is transformed to the frequency domain using the 2D Discrete Cosine Transform (DCT). After the transform, the resulting DCT coefficients are quantized using a simple linear scheme based on the allocated bit-width b. This involves normalizing the coefficients to the range $[0, 1]$ and then scaling them to integers in the range $[0, 2^b - 1]$. The minimum and maximum values for each patch are stored as metadata for decompression.

Scope of Compression

It is critical to note that this compression methodology is not applied to all parameters. The process selectively targets the large, two-dimensional weight matrices (e.g., linear layers in Transformer blocks). Smaller or specialized layers (e.g., embeddings, normalization layers, and classification heads) are excluded to preserve model integrity.

2.1.2 My Innovations and Extensions

While the baseline Delta-DCT method provides a solid foundation, my primary goal was to move beyond a simple implementation and explore potential improvements. My research process was guided by a deeper investigation into the image compression analogy, leading to a series of innovations designed to test new hypotheses and potentially create a more effective compression framework.

Integrating a Standard JPEG Quantization Table

The baseline method uses a linear quantization scheme. My first hypothesis was that a non-uniform approach, similar to the standard JPEG quantization table Wallace, 1991, could be more effective. The rationale is that, like in images, high-frequency changes in delta parameters might be less critical than low-frequency ones. I implemented this by integrating the standard JPEG luminance table (resized via interpolation for different patch sizes) into the pipeline.

Exploring Post-Transform Importance Scoring

A second exploratory innovation involved altering the order of operations. The baseline method calculates importance scores **before** the DCT. I hypothesized that calculating importance **after** the transform—based on the energy of the low-frequency DCT coefficients—might provide a more accurate measure of a patch's significance. This alternative workflow was also implemented and evaluated.

Adopting the Discrete Wavelet Transform (DWT)

The central innovation of my project was the proposal to replace DCT with the Discrete Wavelet Transform (DWT), the core technology of the more advanced JPEG2000 standard Taubman and Marcellin, 2002. The theoretical advantages of DWT, such as its superior energy compaction and multi-resolution analysis, formed a strong motivation Antonini et al., 1992. A key feature of my DWT implementation is the flexibility to use different coefficient-keeping strategies to find an optimal balance between accuracy and compression:

- **all:** Retains all four sub-bands (LL, LH, HL, HH) for the highest fidelity.

- **ll_lh_hl:** Retains the approximation and primary horizontal/vertical details, discarding the diagonal (HH) band.
- **ll_only:** The most aggressive strategy, retaining only the coarse approximation (LL band).

As the Results chapter will show, this DWT-based approach proved to be the most successful of the innovations, justifying a more in-depth analysis.

2.2 Experimental Setup

This section details the models, datasets, and parameters used to evaluate the implemented delta compression methods. All experiments were conducted on a personal laptop, a MacBook M3 Pro. This hardware choice, while capable, presented computational constraints that precluded the use of very large-scale Large Language Models (LLMs). Consequently, my selection of models was deliberately focused on smaller, yet widely-used, architectures from both the vision and language domains that could be feasibly processed.

Furthermore, while I developed a complete fine-tuning pipeline as part of this project's codebase, the process of fine-tuning models from scratch proved to be excessively time-consuming on the available hardware. To maximize efficiency and focus resources on the core research objective—the analysis of compression algorithms—I opted to use publicly available models that were already fine-tuned on standard benchmark datasets. These models were sourced from the Hugging Face Hub, allowing for a more extensive exploration of the compression techniques themselves.

2.2.1 Models and Datasets

I selected four models, covering two distinct domains: text classification and image classification. For each, a standard pre-trained base model and a corresponding fine-tuned version were used.

Vision Models

- **Vision Transformer (ViT):** I used the ViT architecture introduced by Dosovitskiy et al., 2020.
  - *Pre-trained Base:* `google/vit-base-patch16-224-in21k`
  - *Fine-tuned Version:* `nateraw/vit-base-patch16-224-cifar10`
  - *Dataset:* The model was evaluated on the **CIFAR-10** dataset Krizhevsky, 2009.
- **Swin Transformer:** I also included the Swin Transformer, a hierarchical vision transformer Liu et al., 2021.
  - *Pre-trained Base:* `microsoft/swin-tiny-patch4-window7-224`
  - *Fine-tuned Version:* `rs127/swin-tiny-patch4-window7-224-finetuned-cifar10`
  - *Dataset:* This model was also evaluated on the **CIFAR-10** dataset Krizhevsky, 2009.

Language Models

- **RoBERTa:** I selected RoBERTa, a robustly optimized BERT pre-training approach Liu et al., 2019.
  - *Pre-trained Base:* `roberta-base`
  - *Fine-tuned Version:* `textattack/roberta-base-SST-2`

  - *Dataset:* Evaluation was performed on the Stanford Sentiment Treebank (SST-2) task from the **GLUE benchmark** Wang et al., 2018.

- **DistilBERT:** As a lighter-weight alternative, I used DistilBERT, a distilled version of BERT Sanh et al., 2019.
  - *Pre-trained Base:* `distilbert-base-uncased`
  - *Fine-tuned Version:* `distilbert-base-uncased-finetuned-sst-2-english`
  - *Dataset:* This model was also evaluated on the **GLUE SST-2** dataset Wang et al., 2018.

While all four models were tested to ensure the generalizability of the findings, for conciseness in this report, the subsequent results and analysis will primarily focus on RoBERTa and ViT as representative examples of language and vision domains, respectively.

2.2.2 Compression Parameters and Evaluation Metrics

Across the experiments, I explored a range of compression hyperparameters. The search space included:

- **Patch Sizes:** 8, 16, 32, and 64.
- **Bit Allocation Strategies:** Four distinct strategies were evaluated to assess the trade-off between precision and compression:
  1. A simple 50/50 split with 2-bit precision: `[(2, 0.5), (0, 0.5)]`
  2. A simple 50/50 split with 4-bit precision: `[(4, 0.5), (0, 0.5)]`
  3. A triple-precision strategy with lower fidelity: `[(2, 0.34), (1, 0.33), (0, 0.33)]`
  4. A higher-fidelity triple-precision strategy: `[(4, 0.34), (2, 0.33), (0, 0.33)]`

The primary metric for evaluating model performance post-compression was **Accuracy**. This was compared against the accuracy of the original, uncompressed fine-tuned model to calculate the performance degradation, or "accuracy drop." The storage efficiency was measured by the **Compression Ratio**.

2.3 Implementations

To translate the theoretical methods into a functional system, I developed a modular codebase primarily using Python 3. This section outlines the key libraries, the architecture of the code, and the specific logic used for the implementation.

2.3.1 Tools and Libraries

The implementation relies on several key open-source libraries to build a robust and efficient pipeline. **PyTorch** was used for all tensor operations and neural network manipulations. The **Hugging Face Transformers** library provided the essential infrastructure for loading both pre-trained and fine-tuned models, as well as their corresponding tokenizers and image processors. For the core mathematical transforms, I utilized **SciPy**'s ‘fftpack‘ module for the Discrete Cosine Transform (DCT) and the **PyWavelets** library for the Discrete Wavelet Transform (DWT).

2.3.2 Code Architecture and Workflow

The project's architecture was designed to be modular and maintainable, with a clear separation of concerns. The core compression and decompression logic was encapsulated in `compression.py` and `decompression.py`, respectively. These modules contain the functions responsible for patchlization, importance scoring, bit allocation, and the transform-quantize-dequantize cycle for both DCT and DWT.

An orchestrator script, `pipeline.py`, was created to manage the end-to-end workflow. The `compress_model` function in this script applies the chosen compression algorithm to suitable layers, while the `decompress_model` function reconstructs the model by applying the inverse operations. This entire experimental process was managed by `runner.py` and an interactive Jupyter Notebook, `model_evaluation.ipynb`, which used configuration dictionaries to systematically run experiments with different models, patch sizes, and compression strategies.

It is important to note that while the precision of the quantization is determined by the bit-strategy (e.g., 2-bit or 4-bit), the quantized integer values are ultimately stored in `torch.int8` tensors. This is because `int8` is the smallest standard byte-addressable data type available for general tensor operations in PyTorch.

2.3.3 Selective Layer Compression

A key aspect of the implementation is that compression is not applied uniformly to all model parameters. Instead, a filtering logic was implemented within the `compress_model` pipeline to selectively target the most suitable layers for compression. The primary targets were the large, two-dimensional weight matrices that constitute the majority of a model's size, such as those in attention and feed-forward layers.

Conversely, several types of layers were explicitly excluded from compression to preserve model integrity and performance:

- **Non-2D and Small Layers:** The patching process is designed for 2D matrices, so layers with other tensor shapes were skipped. Additionally, any layer with fewer parameters than a single patch was excluded.
- **Critical Functional Layers:** Layers identified by keywords such as `classifier`, `pooler`, `head`, and `embeddings` were bypassed. These layers are often critical for the model's final output or input processing and are generally not suitable for this compression method.

For any layer not selected for compression, its original delta parameter was stored without modification. This selective strategy focuses the compression effort where it is most impactful, maximizing storage savings while minimizing the risk of performance degradation.

2.4 Results

Full, detailed tables for all experiments and models are available for review in the accompanying Jupyter Notebook, `model_evaluation.ipynb`.

2.4.1 Baseline Performance of the Delta-DCT Method

First, to establish a strong performance baseline for the standard Delta-DCT method, I conducted a comprehensive hyperparameter search. This search was performed for each model on a set of patch sizes and bit allocation strategies to find the optimal configuration. The experiments consistently revealed that for the DCT method, the more nuanced triple-precision strategies yielded the best results, often preserving or even slightly improving upon the original model's accuracy.

For conciseness, Table 2.1 summarizes only the optimal configuration found for the two representative models, which serves as the benchmark against which my innovations are measured. The results indicate that while DCT is effective at preserving accuracy, its ability to compress is limited, with the compression ratio consistently remaining between 3.5x and 4.0x.

TABLE 2.1: Optimal Baseline Performance using the Delta-DCT Method.

Model	**Optimal DCT Config**	**Accuracy (Change)**	**Comp. Ratio**
RoBERTa-SST2	32px, Triple-Precision with lower fidelity	**95.00% (+0.50%)**	3.89x
ViT-CIFAR10	32px, Triple-Precision with lower fidelity	99.00% (-0.50%)	3.84x

2.4.2 Analysis of Initial Innovations

Following the establishment of the baseline, I conducted a broad exploratory analysis of all proposed innovations to identify the most promising avenue for in-depth study. This involved testing a JPEG quantization table, the use of post-transform importance scoring, and a comprehensive initial evaluation of the DWT method with its three primary coefficient-keeping strategies.

The results, summarized for the RoBERTa and ViT models in Table 2.2, were highly conclusive. The application of the standard JPEG quantization table was found to be detrimental, causing a significant accuracy drop with no compression benefit. Similarly, the post-transform importance scoring method failed to yield any significant improvement over the baseline, resulting in nearly identical performance.

In contrast, the Discrete Wavelet Transform (DWT) showed significant and varied potential. While the 'all' strategy preserved accuracy, it did not improve the compression ratio over the baseline. The `ll_only` strategy offered a massive compression boost but at the cost of an unacceptable accuracy drop. The `ll_lh_hl` strategy, however, consistently emerged as the most effective, providing a superior compression ratio while maintaining very high accuracy. Based on these clear findings, the DWT `ll_lh_hl` method was selected as the sole focus for the subsequent in-depth analysis.

TABLE 2.2: Performance of All Initial Innovations on Representative Models.

Model	**Method / DWT Strategy**	**Accuracy (Drop)**	**Comp. Ratio**
RoBERTa-SST2	DCT (Baseline)	95.00% (+0.50%)	3.89x
	DCT + Quant. Table	91.00% (-3.50%)	3.89x
	DCT + Post-Transform	94.85% (-0.15%)	3.89x
	DWT (`all`)	92.50% (-2.00%)	3.80x
	DWT (`ll_lh_hl`)	92.50% (-2.00%)	5.03x
	DWT (`ll_only`)	68.00% (-26.50%)	14.12x
ViT-CIFAR10	DCT (Baseline)	99.00% (-0.50%)	3.84x
	DCT + Quant. Table	94.00% (-5.50%)	3.84x
	DCT + Post-Transform	98.90% (-0.60%)	3.84x
	DWT (`all`)	98.00% (-1.50%)	3.75x
	DWT (`ll_lh_hl`)	99.50% (0.00%)	4.94x
	DWT (`ll_only`)	96.00% (-3.50%)	13.35x

2.4.3 In-Depth Analysis of the DWT Method

The initial experiments identified the DWT method with the `ll_lh_hl` coefficient strategy as the most promising innovation. To better understand its characteristics, I conducted an in-depth analysis to evaluate the impact of its key hyperparameters. This analysis focused on patch sizes of 16, 32, and 64. An 8x8 patch size was initially tested but was excluded from the final results due to a disproportionately long execution time, making it impractical for larger models.

A surprising and significant finding was that, contrary to the DCT baseline, the DWT method consistently performed worse with the more complex triple-precision bit strategies. The simpler 50/50 split strategies delivered superior or equal accuracy in all tested scenarios. This suggests that DWT's superior energy compaction is so effective that a simple, aggressive quantization strategy is sufficient. Therefore, for clarity and to highlight the most effective configurations, Table 2.3 presents only the results from the 50/50 split strategies.

TABLE 2.3: In-Depth Analysis of the DWT (`ll_lh_hl`) Method

Model	Patch Size	Bit Strategy	Accuracy (Drop)	Comp. Ratio
RoBERTa-SST2	16	[(2, 0.5), (0, 0.5)]	93.50% (-1.00%)	4.56x
		[(4, 0.5), (0, 0.5)]	94.00% (-0.50%)	4.56x
	32	[(2, 0.5), (0, 0.5)]	93.50% (-1.00%)	5.03x
		[(4, 0.5), (0, 0.5)]	93.50% (-1.00%)	5.03x
	64	[(2, 0.5), (0, 0.5)]	93.50% (-1.00%)	5.16x
		[(4, 0.5), (0, 0.5)]	93.50% (-1.00%)	5.16x
ViT-CIFAR10	16	[(2, 0.5), (0, 0.5)]	99.50% (0.00%)	4.49x
		[(4, 0.5), (0, 0.5)]	99.50% (0.00%)	4.49x
	32	[(2, 0.5), (0, 0.5)]	99.50% (0.00%)	4.94x
		[(4, 0.5), (0, 0.5)]	99.50% (0.00%)	4.94x
	64	[(2, 0.5), (0, 0.5)]	99.50% (0.00%)	5.06x
		[(4, 0.5), (0, 0.5)]	99.50% (0.00%)	5.06x

The analysis of these results reveals two clear and important trends. First, for both models, **increasing the patch size leads to a direct and significant improvement in the compression ratio**. For instance, with the ViT model, moving from a patch size of 16 to 64 increased the compression ratio from 4.49x to 5.06x. This gain in efficiency came at a remarkably low cost: for ViT, there was zero accuracy loss, and for RoBERTa, the accuracy drop remained stable and minimal.

Second, the results confirm that for the DWT method, the higher-fidelity `[(4, 0.5), (0, 0.5)]` strategy provides little to no tangible benefit over the simpler `[(2, 0.5), (0, 0.5)]` strategy. This allows us to conclude that the optimal configuration for the DWT method involves using the largest feasible patch size with the simpler, lower-bitrate 2-bit allocation, as this combination provides the best compression without sacrificing performance.

2.4.4 Final Comparison: DWT vs. DCT

The final analysis directly compares the best-performing configuration of the proposed DWT method against the optimal DCT baseline for each representative model. This head-to-head comparison, presented in Table 2.4, serves to quantify the overall improvement provided by my innovation. The optimal DWT configuration was selected based on the findings from the in-depth analysis, prioritizing the highest compression ratio that could be achieved without a significant drop in accuracy.

TABLE 2.4: Final Comparison: Optimal DWT vs. Optimal DCT.

Model	**Method (Optimal Config)**	**Accuracy (Change)**	**Comp. Ratio**
RoBERTa-SST2	DCT (32px, Triple-Precision)	**95.00% (+0.50%)**	3.89x
	DWT (64px, 50/50 Split)	93.50% (-1.00%)	**5.16x**
ViT-CIFAR10	DCT (32px, Triple-Precision)	99.00% (-0.50%)	3.84x
	DWT (64px, 50/50 Split)	**99.50% (0.00%)**	**5.06x**

The results unequivocally demonstrate the superiority of the DWT-based approach. Across both models, the DWT method achieves a significantly higher compression ratio, consistently surpassing the 5.0x mark—a barrier the DCT method was unable to cross. For the ViT model, this superior compression was achieved with **zero accuracy loss**, making it a strictly better alternative. For the more sensitive RoBERTa model, the DWT method offered a massive **33% improvement in compression ratio** (5.16x vs 3.89x) at the cost of a minor 1.5% total accuracy difference, which represents a highly favorable trade-off.

Furthermore, it is critical to reiterate that the DWT method achieved these superior results using a simpler and lower-bitrate 50/50 allocation strategy, whereas the DCT baseline required a more complex, lower-fidelity triple-precision strategy to remain competitive. This core finding—achieving better compression with a simpler strategy—provides strong evidence that DWT is a more efficient and powerful transform for this data-free delta compression task.

Conclusion

This project successfully explored the domain of data-free delta compression for fine-tuned models. Beginning with a faithful implementation of the DCT-based baseline method, Delta-DCT, I developed and proposed a novel compression framework centered around the Discrete Wavelet Transform (DWT). The primary objective was to investigate whether DWT, the core of the modern JPEG2000 standard, could offer a more efficient alternative to the JPEG-inspired DCT approach for compressing model parameters.

The experimental results decisively demonstrated the superiority of the proposed DWT method. My key finding is that the DWT framework, particularly when configured with a large patch size (64x64) and a selective coefficient strategy (`ll_lh_hl`), achieves a significantly higher compression ratio (consistently >5.0x) than the optimized DCT baseline (which was limited to <4.0x). Remarkably, this improved efficiency was achieved while using a simpler, lower-bitrate quantization strategy and, in the case of the ViT model, with zero loss in accuracy. This suggests that DWT's properties of superior energy compaction are highly effective for the task of delta parameter compression.

Despite these promising results, I acknowledge the limitations of this work. All experiments were conducted on a personal laptop, which restricted the analysis to smaller-scale models and prevented validation on the large LLMs where these techniques are most impactful. Furthermore, the reported compression ratios are based on the theoretical bit-precision of the quantized data; the implementation uses standard **int8** tensors for storage, and a true bit-packing mechanism would be required to realize these storage savings in practice.

Future work could extend this project in several exciting directions. First and foremost, validating the DWT method on larger models (7B and beyond) is a critical next step. Second, implementing a true bit-packing and unpacking pipeline would allow for the measurement of actual storage reduction and potential inference speedups. Finally, while this project used the simple `Haar` wavelet, exploring other wavelet families (e.g., Daubechies, Deep Learning Version) could potentially unlock an even better balance between compression and model fidelity.

Bibliography

Antonini, Marc et al. (1992). “Image coding using wavelet transform”. In: *IEEE Transactions on Image Processing* 1.2, pp. 205–220.

Dosovitskiy, Alexey et al. (2020). *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. arXiv: 2010.11929 [cs.CV].

Huang, Chenyu et al. (2025). *Seeing Delta Parameters as JPEG Images: Data-Free Delta Compression with Discrete Cosine Transform*. arXiv: 2503.06676 [cs.CV].

Krizhevsky, Alex (2009). *Learning Multiple Layers of Features from Tiny Images*. Tech. rep. University of Toronto.

Liu, James et al. (2024). *BitDelta: Your Fine-Tune May Only Be Worth One Bit*. arXiv: 2402.10193 [cs.LG].

Liu, Yinhan et al. (2019). *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. arXiv: 1907.11692 [cs.CL].

Liu, Ze et al. (2021). *Swin Transformer: Hierarchical Vision Transformer using Shifted Windows*. arXiv: 2103.14030 [cs.CV].

Ping, Bowen et al. (2024). *Delta-CoMe: Training-Free Delta-Compression with Mixed-Precision for Large Language Models*. arXiv: 2406.08903 [cs.CL].

Sanh, Victor et al. (2019). *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. arXiv: 1910.01108 [cs.CL].

Taubman, David S and Michael W Marcellin (2002). *JPEG2000: Image compression fundamentals, standards and practice*. Vol. 642. Kluwer academic publishers.

Wallace, Gregory K (1991). “The JPEG still picture compression standard”. In: *Communications of the ACM* 34.4, pp. 30–44.

Wang, Alex et al. (2018). *GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding*. arXiv: 1804.07461 [cs.CL].