Midterm Report
Neural Networks at a Fraction: Table Structure Recognition

Aritra Mukhopadhyay [1] [2] [3]

Abstract

Our work aims to deploy large transformer networks for Table Structure Recognition (TSR) tasks on mobile devices. To reduce model size, we employ Quaternions to develop transformers from scratch. Quantization techniques further minimize model size while preserving accuracy. We explore the ONNX format for seamless mobile integration.

A key experiment compares 'Prune-Finetune' and 'Finetune-Prune' approaches, involving finetuning and pruning pretrained models, offering vital optimization insights. Our goal is to enable efficient deployment of robust TSR models on resource-constrained mobile devices, with our work contributing significantly to this endeavor.

1. Insightful analysis of 2-3 related papers

1.1. Attention Is All You Need

The paper **"Attention Is All You Need"** by Vaswani et al (Vaswani et al., 2023). is a seminal work that introduced the Transformer architecture, revolutionizing natural language processing and machine learning. At its core, the Transformer relies on self-attention mechanisms to process input sequences efficiently.

The self-attention mechanism is the backbone of the Transformer and is mathematically defined as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V$$

Here, Q, K, and V represent query, key, and value matrices, respectively, and d_k is the dimension of the key vectors. This equation calculates a weighted sum of values, where each position's weight depends on its relevance to all other positions.

A key innovation introduced by the paper is multi-head attention, which allows the model to attend to different parts of the input sequence simultaneously. This enhances its ability to capture complex dependencies in the data.

The Transformer architecture introduced in 'Attention Is All You Need' comprises an encoder and a decoder. Each component has multiple layers, incorporating self-attention mechanisms and feedforward neural networks. The encoder captures input sequence information effectively, while the decoder ensures causal generation by masking self-attention. This design enables the Transformer's success in diverse sequence-to-sequence tasks like machine translation and language modeling.

The Transformer architecture's impact extends beyond natural language processing, influencing various fields due to its efficient handling of long-range dependencies and parallelization capabilities. This paper's elegant mathematical formulation and the self-attention mechanism's versatility have shaped the modern landscape of deep learning for sequences.

1.2. End-to-End Object Detection with Transformers

The paper **"End-to-End Object Detection with Transformers"** from Facebook AI (Carion et al., 2020) DETR, a groundbreaking method that re-imagines object detection as a direct set prediction problem. This innovative approach streamlines the detection pipeline by eliminating the need for hand-designed components like non-maximum suppression or anchor generation, typically used to encode prior knowledge about the task.

DETR's core framework combines a transformer encoder-decoder architecture with a set-based global loss, ensuring unique predictions through bipartite matching. The method leverages a fixed set of learned object queries to reason about object relations and global image context simultaneously. By directly outputting the final set of predictions in parallel, DETR simplifies the model's conceptual complexity.

[*]Equal contribution [1]Company Name, Location, Country [2]Department of XXX, University of YYY, Location, Country [3]School of ZZZ, Institute of WWW, Location, Country. Correspondence to: Firstname1 Lastname1 <first1.last1@xxx.edu>, Firstname2 Lastname2 <first2.last2@www.uk>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

Notably, DETR achieves remarkable accuracy and competitive runtime performance on the demanding COCO object detection dataset, comparing favorably to the well-established Faster RCNN baseline. Despite its slower inference speed, the gains in accuracy make it a compelling choice for object detection tasks. Furthermore, DETR's versatility extends to panoptic segmentation in a unified manner, outperforming other approaches in this regard.

In summary, ”End-to-End Object Detection with Transformers” introduces DETR as a transformative approach to object detection. By replacing conventional components with a transformer-based architecture, it achieves remarkable accuracy and simplifies the model design, demonstrating the power of leveraging transformers in computer vision tasks.

1.3. Grits: Grid table similarity metric for table structure recognition

In the paper ”GriTS: Grid Table Similarity Metric for Table Structure Recognition” authored by Brandon Smock et al (Smock et al., 2023), the authors introduce a novel metric, GriTS, for evaluating Table Structure Recognition (TSR) systems. Unlike previous metrics, GriTS evaluates the correctness of predicted tables directly as matrices, preserving their natural form.

The paper's main contribution lies in the generalization of the NP-hard two-dimensional largest common substructure (2D-LCS) problem to the 2D most similar substructures (2D-MSS) problem. The authors propose a polynomial-time heuristic for solving this problem, providing both upper and lower bounds on the true similarity between matrices. Empirical evaluations on a real-world dataset demonstrate that the practical difference between these bounds is negligible. Additionally, GriTS outperforms alternative metrics, showcasing the superiority of matrix similarity in TSR performance evaluation.

Moreover, GriTS unifies three subtasks within TSR, namely cell topology recognition, cell location recognition, and cell content recognition, simplifying evaluation and enabling more meaningful comparisons across different TSR approaches. This paper presents a significant advancement in the evaluation of TSR systems, offering a versatile and robust metric for assessing their performance.

2. Experiments

2.1. Converting the Models to ONNX

Converting deep learning models to the Open Neural Network Exchange (ONNX) format is a pivotal step in our work. ONNX serves as an open, widely supported standard for representing deep learning models. This format enjoys backing from major players in the field, including Microsoft, Facebook, and Amazon. The significance of this conversion lies in the goal of deploying our Table Structure Recognition (TSR) model on mobile devices. ONNX's cross-framework compatibility makes it a natural choice for ensuring the seamless integration of our model on a variety of platforms.

However, this process presented some significant setbacks. Notably, we observed that the ONNX version of our model performed significantly slower, nearly twice as slow as its PyTorch counterpart, even when executed on the same GPU. This slowdown raised concerns about the practicality of deploying our Table Structure Recognition (TSR) model on resource-constrained mobile devices.

Another unexpected challenge we encountered was related to the model size. Despite the PyTorch version of the quaternion model being one-fourth the size of the real model, the ONNX versions of both the real and quaternion models turned out to be of the same size. This discrepancy puzzled us and prompted further investigations into optimizing the ONNX conversion process to address these performance and size-related setbacks.

2.2. Quantization of the Models

Quantization, a key optimization technique, reduces model size without significant performance loss by decreasing weight and activation precision. Our study aims to strike the right balance between size reduction and performance preservation, crucial for efficient TSR model deployment on resource-limited devices.

Quantization plays a crucial role in model optimization, but it presented its own set of challenges during our work. Firstly, we faced difficulties in loading the quantized models after saving them. These issues are essential for practical deployment and need to be resolved for a seamless user experience.

Additionally, we discovered that quantization is layer-specific and does not seamlessly work with custom layers. Given that our quaternion models rely on custom layers, we had to implement these layers ourselves as custom components. Unfortunately, this led to quantization not functioning as intended for our quaternion models, creating obstacles in achieving optimal model size reduction.

These setbacks encouraged us to explore alternative techniques and custom solutions to address the challenges posed by quantization and to ensure that our models are well-optimized for deployment. We will work on this next.

2.3. Implementation of Transformer from Scratch

This endeavor necessitates the development of a custom Transformer architecture, encompassing both real and quaternion variants. The rationale behind this effort is twofold: firstly, to facilitate the creation of quaternion layers from the ground up, and secondly, to acquire comprehensive insights into the architecture of the real Transformer.

To achieve this objective, we have meticulously implemented a comprehensive set of components. These components encompass the real and quaternion versions of Self-Attention, both with and without associated weights, as well as positional encoding, encoder layers, decoder layers, and the overarching Transformer layer. Our implementations adhere closely to the principles and structures outlined in the seminal ”Attention Is All You Need” (Vaswani et al., 2023) paper and course materials, ensuring a rigorous foundation for our project.

While the work remains ongoing, as rigorous testing of the code is yet to be completed, the initial results are promising. However, the true evaluation of the implemented layers will only be ascertained through their integration into the DETR (DEtection TRansformer) model. This systematic approach ensures that our contributions align with established standards and serve as a robust foundation for the subsequent stages of our research. (Code will be attached later.)

2.4. Finetune Before Pruning or After Pruning

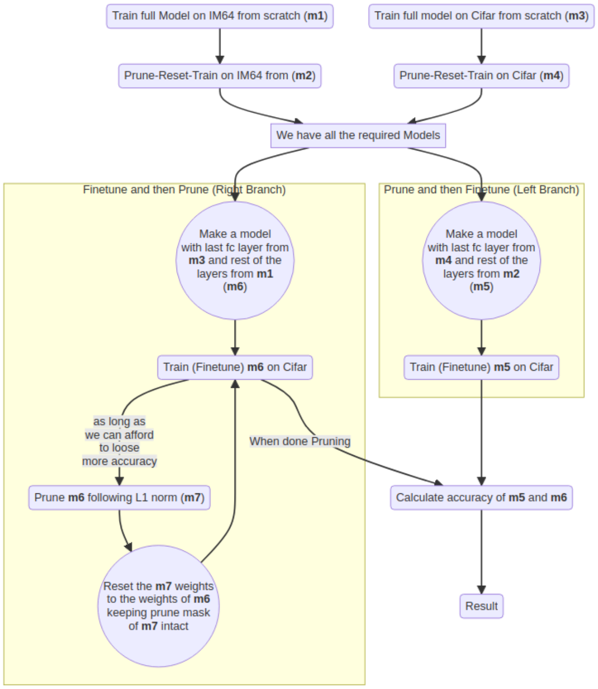
Train full Model on IM64 from scratch (m1)
Train full model on Cifar from scratch (m3)
Prune-Reset-Train on IM64 from (m2)
Prune-Reset-Train on Cifar (m4)
We have all the required Models
Finetune and then Prune (Right Branch)
Prune and then Finetune (Left Branch)
Make a model with last fc layer from m3 and rest of the layers from m1 (m6)
Make a model with last fc layer from m4 and rest of the layers from m2 (m5)
Train (Finetune) m6 on Cifar
Train (Finetune) m5 on Cifar
as long as we can afford to loose more accuracy
When done Pruning
Prune m6 following L1 norm (m7)
Calculate accuracy of m5 and m6
Reset the m7 weights to the weights of m6 keeping prune mask of m7 intact
Result

Figure 1. Flowchart explaining the experimental process of determining whether we should do finetuning before pruning or after pruning

The most challenging phase of our project involved a meticulous and intricate process, as illustrated in the provided flowchart 1. During this phase, we dealt with pretrained real and quaternion models, both pruned and unpruned, initially trained on the ImageNet 64x64 dataset.

Our primary task was to create hybrid models, a process vividly depicted in the diagram. We initiated this process by extracting the layers from these ImageNet 64x64 models and subsequently replacing their last layers with equivalently pruned and trained layers from the Cifar100 model. Let us call this action **’crafting’**. This action led to the formation of two distinct model branches, each following a unique trajectory.

In the left branch, we start with the crafted unpruned network (m6 as in diagram). We trained the model on Cifar100, pruned the model and calculated the pruning mask, reset the model to the weights of m6 keeping the mask intact, then training again. We continued this for 20 pruning iterations while keeping only 70% of the remaining weights after every pruning iteration.

On the right branch, we started with the pruned network and trained it (finetune) on the Cifar100 dataset.

The provided flowchart serves as a visual representation of the entire process, encapsulating the complexity and intricacies of this project phase.

3. Results

As of the current progress update, we have successfully executed all components for the real model, while the quaternion models remain under ongoing processing.

Initially, the unpruned real model exhibited an accuracy of approximately 65% on the CIFAR-100 dataset, while the pruned counterpart achieved approximately 64%. Subsequently, through a process of retraining on CIFAR-100, the accuracy of the unpruned model significantly improved to approximately 77%. In parallel, the pruned model also demonstrated notable enhancement, achieving an accuracy of 75.23% after undergoing retraining on the CIFAR-100 dataset.

References

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. End-to-end object detection with transformers, 2020.

Smock, B., Pesala, R., and Abraham, R. Grits: Grid table similarity metric for table structure recognition, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need, 2023.