Reducing Energy Consumption in AI Inference via Model Compression

Ravi Shah
The University of Texas at Austin
ravishah@utexas.edu

Armaan Hirani
The University of Texas at Austin
hirani@utexas.edu

***Abstract*—As deep learning systems continue to scale, their increasing energy demands, particularly during the inference phase, pose a growing challenge to sustainability and deployment at scale. In this study, we analyze the impacts of model compression techniques on reducing energy consumption. Specifically, we consider three techniques: quantization, pruning, and knowledge distillation. We benchmark results of these techniques on a Vision Transformer (ViT) model on the CIFAR-100 dataset. We measure energy consumption using Intel's RAPL and NVIDIA's NVML tools on CPU and GPU hardware. Further, we propose an energy-efficient AI inference pipeline that utilizes structural pruning, knowledge distillation, and quantization of weights to a uint8 data type. Our results demonstrate that compressing neural networks before inference can save significant energy with a minimal loss in accuracy. More broadly, this study highlights the potential impact that model compression can have on green AI deployment. The implementation code is available at https://github.com/RaviShah1/energy-efficient-ai-inference.**

***Index Terms*—energy efficiency, quantization, pruning, distillation, computer vision, machine learning**

I. Introduction

As the need for deep learning enabled systems becomes more and more widespread, SaaS (software as a service) providers such as OpenAI, Google, and Microsoft, in conjunction with state entities, have invested hundreds of millions of dollars in AI research and infrastructure development. A significant portion of this funding is dedicated solely to energy production to power data centers responsible for their required computation.

Over time, these neural networks have progressed to become more accurate and capable, and consequently have become structurally larger and computationally complex. As a result, the computational cost associated with these networks have become a significant challenge for energy efficiency. Within the process of creating, training, and deploying a deep learning model (inference), only the latter is done continuously, leading inference to be the main source of energy usage in its life cycle. An example of this is a study conducted by AWS, which claims that over 90% of total energy consumption occurs during inference [1].

Despite these growing demands, measuring the actual energy consumption and usage of AI systems in a production or deployed environment is not a straightforward process. The equipment needed to accurately do so is very scarcely available outside of advanced research laboratories or very niche specialized deployment instances. As an alternative, researchers tend to turn toward measuring run time performance and model execution time as an alternative method for energy usage. While not the most accurate measurement, this method allows for relative comparisons across models and configurations and has in turn become a widely accepted proxy for such.

The need to create a reduction in energy consumption during the inference stage for deep neural network models has led to multiple innovations with promising results. The underlying idea for the most prominent tend to fall under the same umbrella: the overall reduction of unneeded consumption, in turn reducing energy consumption. While this has been the driving goal while developing these techniques, the issue of decreasing energy consumption while preserving accuracy and performance has become more and more prevalent. Through this search, three techniques have become widespread in their implementation: quantization, pruning, and distillation.

Quantization involves reducing the precision of the values used in a network's computation. These weights are most commonly 32-bit floating point integers, and are converted to lower precision weights such as 8-bit integers. Doing this, models become smaller and faster through a reduction in required floating point operations. These operations using less bits than their former, are less of a burden on memory and computational hardware, especially when they are supported by specialized hardware.

Pruning is a technique that focuses on removing parts of the model that are insignificant to the computation process and the associated output. This can range from individual weights to larger neurons and their incident edges. There are two main types of pruning, structural and unstructural. Structured pruning removes larger layers or channels, and the latter removes individual weights. By pruning a model's weights by importance, the size of a model can be drastically reduced while still maintaining its accuracy. These pruned networks in total carry a smaller footprint, and also have a decreased power draw.

Distillation transfers the predictive power of a large, pre trained model, commonly known as the "teacher", to a smaller model, referred to as the "student". The student is trained by tuning the logits layer of the student to match what would be derived in the teacher model, ultimately producing an output identical or extremely similar to the teacher model. This technique allows the student model to perform remarkably similarly to the teacher model, while being much smaller than the teacher model in terms of parameter count and energy usage.

Taken together, these techniques represent a shift in how artificial intelligence systems are implemented and deployed. By compressing the size of models, reducing their parameter counts and FLOPs, these methods directly correlate to a decrease in the number of computations required. This sequentially speeds up inference time and lowers power draw per prediction, and culminates in a reduced energy consumption. As energy efficiency becomes as important as model accuracy and latency, compression techniques like such are a key factor in driving the next generation of model design as well as the long term sustainability and scalability of deep neural networks.

II. Background and Related Works

A. Energy Measurement Techniques

Measuring CPU energy can be especially difficult due to a lack of standard tools. Intel's Running Average Power Limit (RAPL) interface works as an on-chip energy measurement tool. Specifically, RAPL utilizes activity counters and calibrated weights to estimate DRAM energy, memory power, and capping [2].

NVIDIA provides tools for monitoring the energy consumption of their GPUs. Specifically, nvidia-smi and its underlying NVIDIA Management Library (NVML) provide measurements for real-time power usage. These tools consider power draws, power limits, and thermal constraints for accurate energy profiling. Additionally, NVML provides users with an easy to use API for querying power usage [3].

B. Quantization

Quantization is a model compression technique that reduces the precision of the weights. By default, most neural networks use 32-bit floating point (FP32) numbers. However, quantization converts weights to lower-precision types such as unsigned 8-bit integers. By using fewer bits, the model requires less memory and processes inputs significantly faster [4].

C. Pruning

The idea of pruning was originally presented as *The Lottery Ticket Hypothesis*. This suggested that within a dense neural network, there exists a sparse sub-network (*winning ticket*) that could be trained in isolation to achieve similar accuracy to the fully dense model. This work suggests the *winning tickets* both retain accuracy better and generalize better than the fully dense model. This work is the inspiration for pruning, the idea of removing unnecessary parameters and components from a large model to create a smaller pruned model [5].

The idea of pruning has been used not only to make models generalize better but also as a form of model compression. Compressed, smaller models are often more energy efficient. One comprehensive study categorized multiple pruning strategies and analyzed the accuracy and energy consumption of each technique. First, this study points out that there is an assortment of different pruning strategies. Second, the study highlights the lack of standard evaluation metrics for evaluating the quality of Green AI models. Third, the study emphasizes that there is a lack of pruning strategies tailored for custom model architectures [6].

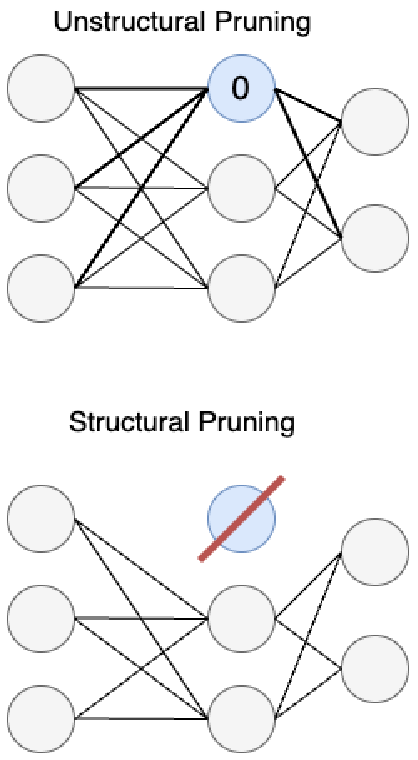

Fig. 1. Unstructured pruning masks out weights by zeroing them out. Structured pruning removes the node and incident edges entirely.

Pruning can be unstructural or structural. Unstructured pruning masks out individual weights by setting their values to zero without modifying the model's structure. This is the type of pruning built into PyTorch's standard pruning functions [7]. In contrast, structural pruning physically removes entire components such as channels from the model. This modifies the structure of the model and reduces the model's computational load. Hence, structural pruning can be used as a form of model compression. For example, Dependency Graph (DepGraph) is a framework that identifies dependencies between model layers by constructing a graph of the model inter and intra layer dependencies. DepGraph can then be used to perform group-level pruning without breaking the model architecture [8]. A visualization of the difference between unstructural and structural pruning can be found in Fig. 1.

D. Distillation

The concept of knowledge distillation was originally formalized by Hinton et al. Distillation is a training technique to enable a small model (student) to mimic the output distributions of a larger model (teacher). Formally, distillation minimizes a distillation loss which is a combination of a soft target loss from the teacher and a hard target loss from the ground truth.

Distillation has shown small models can preserve the accuracy of larger models [9].

Further, another study has investigated the impact of knowledge distillation on energy consumption and runtime in natural language processing (NLP) models. This study found that distilled models consistently outperformed larger models in terms of energy efficiency while maintaining most of the accuracy [10].

E. Similar Studies

Álvarez et al. consider rising energy demands associated with deploying large machine learning models, emphasizing the inference stage as the primary consumer of energy within ML system deployment and life cycle. The group resorts to using computational indicators of inference time and GPU usage as approximations for total energy used. They cover topics such as dynamic quantization, PyTorch's torch.compile, and both global and local pruning to maximize model efficiency through controlled testing on 42 image classification models sourced via Hugging Face. They found that dynamic quantization was particularly useful for reducing inference energy usage without diminishing accuracy. Global pruning, another method test, tends to introduce an extremely high optimization overhead even when reducing computational complexity. They emphasize a balance between energy efficiency, computational performance, and model accuracy [11].

Sreenivas et al. performed a study by NVIDIA presenting the Minitron approach for compressing large language models (LLMs). This study utilized a combination of structural pruning and knowledge distillation. They consider two pruning strategies: depth pruning (removes full layers) and width pruning (trims components such as neurons, attention heads, etc). They found that width pruning outperformed depth pruning. Additionally, they use a teacher correction phase for fine-tuning the teacher model on the distillation dataset. Overall, they demonstrate an effective pipeline for the model compression of LLMs [12].

Goel et al. provided a survey of techniques for low power deep learning. Specifically, they investigated parameter quantization and pruning, compressed convolutional filters and matrix factorization, network architecture search, and distillation. They found that combining multiple of these techniques could lead to significant model compression that resulted in improved energy efficiency [13].

III. Methodology

A. System Setup and Energy Tracking

We ran our experiments on a Quadro P2000 GPU and an Intel Xeon E3-1270 v6 CPU. We measured GPU energy consumption using the NVIDIA Management Library (NVML) [3]. We measured the CPU energy consumption using Intel's Running Average Power Limit (RAPL) [2]. Notably, the CPU measurements seemed to vary depending on the time of day we ran the code, leading to inconsistencies. This is likely due to changes in temperature at the location the machine was stored. Hence we only compare CPU measurements from code that was run around the same time, and we tend to use GPU energy measurements as a more reliable gauge. For increased efficiency of training models, we use a NVIDIA A100 GPU on Google Colab, and due to system requirements, we do not have energy measurements from model training.

B. Experiment Setup

For all experiments, we use a Vision Image Transformer (ViT) [14] model. Prior to experiments, the model had been trained on the CIFAR-100 dataset [15]. To understand the effects that applying the techniques had on performance, we evaluate the ViT model on the CIFAR-100 test set in all experiments. For each experiment, before measuring the energy of running the code on the test set, we warm up the processors by running five batches of dummy input through the model. This reduces the bias from start-up overhead in the energy measurements. After warming up the processors, we run the model in the inference phase with batch size eight across the 10,000 images in the test set. For each experiment, we track the Top-1 accuracy, Top-5 accuracy, time in seconds of the run, total GPU energy consumption of the run, total CPU energy consumption of the run, and model metadata such as the number of parameters and floating point operations (FLOPs) of the model. We divide the time and energy consumption measurements by the number of samples to achieve the per sample measurements.

C. Quantization Setup

The model was initially trained using FP32 (32-bit floating point) weights. We perform post-training quantization to evaluate the energy savings and performance changes. Specifically, we use the quanto Python library to modify the data types of the weights. We benchmark the results of quantizing the weights to 8-bit floats and 8-bit integers. For consistent measurements, we ran the evaluation for each data type five times taking the mean and standard deviation across the trials.

D. Pruning Setup

Unstructurual pruning was performed using PyTorch's built-in pruning functions [7]. We prune all layers excluding attention layers and the output layer. Specifically, this prunes a certain percentage of channels by zeroing out the ones with the lowest L1-norm.

Structural pruning was implemented using DepGraph [8] via the Torch-Pruning library which constructs a dependency graph. We applied channel pruning to all layers in the Vision Transformer model except the output classifier. We remove the least important channels based on the L1-norm. Based on our preliminary experiments, the attention blocks typically contained important information. Thus, we also include an architecture-aware structural pruner that prunes all layers excluding the output classifier and all attention layers.

E. Distillation and Full Pipeline

For all model training configurations, we trained for 10 epochs with a batch size of 16. We trained with an AdamW

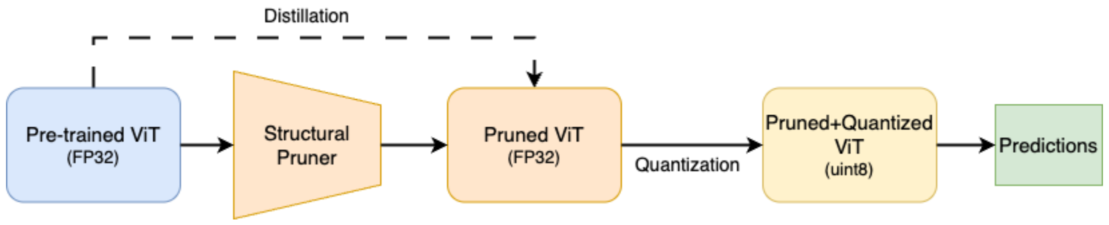

Fig. 2. The full pipeline: the pre-trained ViT is pruned, the pruned model is retrained using distillation with the pre-trained model as the teacher and the pruned model as the student, the student model is then quantized and used to make predictions.

optimizer [16] such that the learning rate is initially set to 3e-5 and scheduled with a CosineAnnealingLR scheduler [17]. We trained on the CIFAR-100 training set and split the set into a training set (90%) and a validation set (10%) using a random split. We save the model weights during training and select the weights with the highest validation Top-1 accuracy. We maintain two training pipelines. The first pipeline trains/fine tunes a model to optimize for minimizing cross-entropy loss between the predictions and ground truth labels. The second pipeline is a knowledge distillation pipeline. The pipeline takes a teacher model and a student model. Specifically, the distillation loss combines two components: the cross-entropy loss with the true labels (hard targets), and a soft target loss that encourages the student's output distribution to match the softened teacher output. Hence, for distillation we minimize:

$$\mathcal{L}_{\text{distill}} = \alpha \cdot T^2 \cdot \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{C} \left(\tilde{y}_{ij}^{(T)} \cdot \log \frac{\tilde{y}_{ij}^{(T)}}{p_{ij}^{(T)}} \right) + (1-\alpha) \cdot \mathcal{L}_{\text{CE}}(p_i, y_i) \tag{1}$$

where α is the weight used to balance the losses, T is the temperature used to soften the logits, N and C are the batch size and number of classes respectively, $\tilde{y}_{ij}^{(T)}$ and $p_{ij}^{(T)}$ are the softmax-normalized outputs of the teacher and student models respectively, computed using temperature T, and $\mathcal{L}_{\text{CE}}$ is a cross entropy loss function [9].

Fig. 2 shows our proposed full energy efficient pipeline. First, we prune 10% of all layers from the pre-trained ViT using structural pruning. Then, we distill the knowledge from the pre-trained ViT into the pruned ViT with an α parameter of 0.25. Next, we perform quantization on the newly distilled model to transform its weights to uint8 data types. Finally, we use the quantized model to make predictions.

IV. Experimental Results

A. Quantization

We compare three models in this section: baseline (FP32), a FP8 quantized version of the baseline, a uint8 quantized version of the baseline. As shown in Table I, all three models have roughly the same accuracy. In fact, the uint8 quantized model even had a slight improvement in accuracy compared to the FP32 baseline (0.9153 compared to 0.9148). It is unclear whether this improvement in accuracy is a result of noise or an actual result of the model generalizing better with reduced precision. Regardless, quantization had effectively no impact on model accuracy in any experiment.

As shown in Table I and Fig. 3, there were noticeable changes in speed and energy consumption during the inference phase. The speed and energy consumption of FP8 model and uint8 model are similar; however, overall, the uint8 model had slight improvements in performance compared FP8 model. Specifically, the uint8 model achieved a 17.1% speedup compared to the FP32 model and a 16.6% reduction in total energy consumption.

B. Pruning

Initially, we tested unstructural pruning that masked a certain percentage of the models weights to have a value of zero. Unstructural pruning did not achieve any model compression as it retrained the same number of parameters and FLOPs. Our experiments confirmed the energy consumption of the unstructurally pruned model was identical to that of the baseline model. Further, the pruning resulted in significant drops in accuracy. Overall, unstructured pruning proved to be completely ineffective.

Next, we tried structurally pruning all modules of the network. While structural pruning achieved significant improvements in energy efficiency, the model experienced a significant decline in accuracy. In fact, pruning 10% of each layer caused the model to lose nearly 90% of its accuracy rendering it useless.

Next, we tried a different variation of structural pruning. This time we prune all layers besides the output classifier and attention blocks. For every layer, we pruned a percentage of its parameters (pruning ratio). The results of this technique can be found in Table II. Similar to quantization, pruning was able to save a significant amount of energy. Unlike quantization, this save energy came at the cost of some accuracy. As shown in Fig. 3, the model's size (parameters and FLOPs) reduces in a linear way as the pruning ratio increases. Additionally, this causes the energy consumption to also have a decreasing linear relationship to the pruning ratio. However, as the pruning ratio increases, the accuracy of the model decreases exponentially. Notably, a pruning ratio of 0.10 results in a sub 90% accuracy and a pruning ratio of 0.30 results in a sub 50% accuracy model.

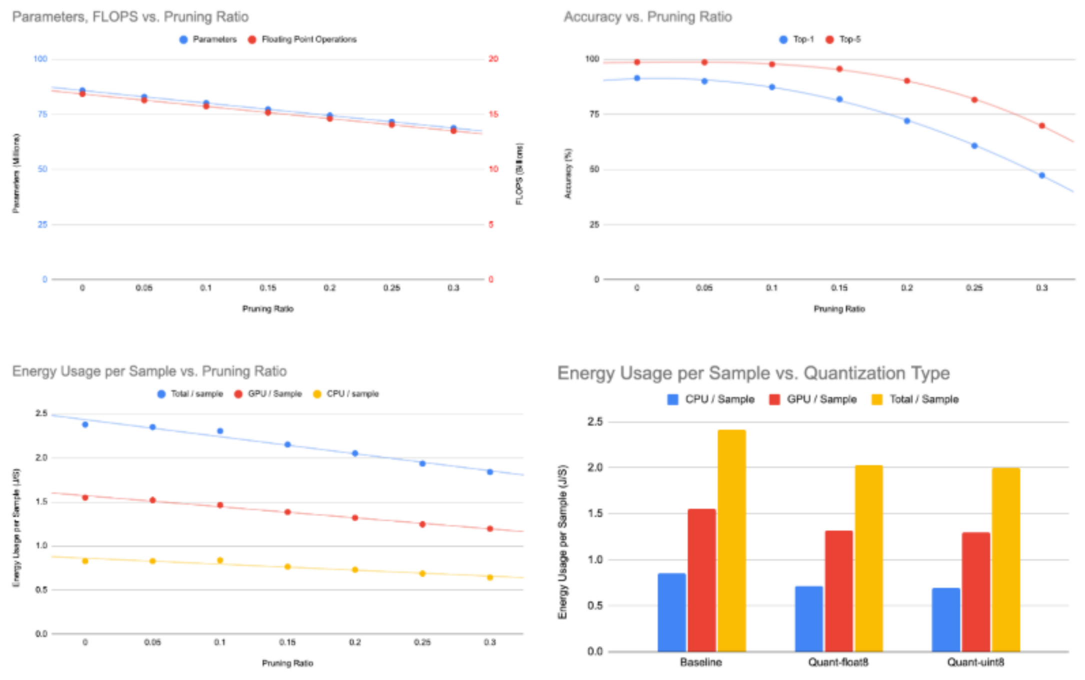

Fig. 3. Results Figure

TABLE I
PERFORMANCE AND ENERGY METRICS FOR BASELINE AND QUANTIZED MODELS. ALL VALUES ARE REPORTED AS MEAN ± STANDARD DEVIATION.

Model	Top-1 Acc.	Top-5 Acc.	Time / Sample (s)	GPU E / Sample (J)	CPU E / Sample (J)	Total E / Sample (J)
Baseline	0.9148 ± 0.0000	0.9871 ± 0.0000	0.02533 ± 0.0000025	1.5507 ± 0.0020	0.8509 ± 0.0221	2.4016 ± 0.0240
Quant-float8	0.9145 ± 0.0000	0.9872 ± 0.0000	0.02116 ± 0.0000522	1.3139 ± 0.0056	0.7155 ± 0.0296	2.0294 ± 0.0275
Quant-uint8	0.9153 ± 0.0000	0.9871 ± 0.0000	0.02100 ± 0.0000043	1.3021 ± 0.0072	0.7001 ± 0.0028	2.0021 ± 0.0060

TABLE II
PERFORMANCE, EFFICIENCY, AND MODEL COMPLEXITY UNDER DIFFERENT PRUNING RATIOS r.

Pruning Ratio r	Top-1	Top-5	Time / Sample (s)	GPU / Sample (J)	CPU / Sample (J)	Total / Sample (J)	Params (M)	FLOPs (G)
0.00	0.9148	0.9871	0.02533	1.54981	0.82846	2.37827	85.88	16.86
0.05	0.9005	0.9864	0.02478	1.52083	0.82908	2.34991	83.00	16.29
0.10	0.8742	0.9774	0.02385	1.46415	0.83972	2.30387	80.19	15.74
0.15	0.8193	0.9565	0.02249	1.38544	0.76534	2.15078	77.32	15.17
0.20	0.7206	0.9023	0.02140	1.32006	0.73102	2.05108	74.51	14.62
0.25	0.6078	0.8162	0.02026	1.24561	0.68793	1.93354	71.71	14.07
0.30	0.4738	0.6988	0.01947	1.19599	0.64235	1.83834	68.83	13.50

TABLE III
PERFORMANCE AND EFFICIENCY METRICS ACROSS STAGES OF THE ENERGY-EFFICIENT INFERENCE PIPELINE.

Metric	Pre-trained ViT	Pruned (r=0.1)	+ Re-train (CE)	+ Distillation	+ Quantization (uint8)
Top-1 Accuracy	0.9148	0.0137	0.8803	0.8823	0.8828
Top-5 Accuracy	0.9871	0.0665	0.9823	0.9801	0.9803
Time / Sample (s)	0.02533	0.01977	0.01978	0.01982	0.01833
GPU Energy / Sample (J)	1.54947	1.21845	1.21813	1.22028	1.12374
CPU Energy / Sample (J)	0.90642	0.69557	0.69621	0.69703	0.62041
Total Energy / Sample (J)	2.45589	1.91402	1.91434	1.91731	1.74415
Params (M)	85.88	68.67	68.67	68.67	68.67
FLOPs (G)	16.86	13.47	13.47	13.47	13.47

C. Distillation and Full Pipeline

Now that we have verified the effectiveness of quantization and pruning, we merge these techniques in an inference pipeline (Fig. 2). Further, we incorporate knowledge distillation to recover the lost accuracy from pruning. The results of each stage of this pipeline are outlined in Table III. The original pre-trained model achieves 91.48% accuracy, consists of 85.88 M parameters, and requires 2.45 Joules per sample.

The next phase in the pipeline applied structural pruning. We prune 10% of all layers (including attention modules). We are okay with the lost accuracy from pruning attention blocks because we will re-train the model in future steps. This step reduces the accuracy by nearly 90%. However, the model is only 68.68 M parameters now. Thus, the pruning compressed the model by 20%. As a result, the model now only requires 1.91 Joules per sample (22% reduction in energy consumption).

Next, we re-train the model to recover the lost accuracy. We try two techniques here. First, as a control experiment, we re-train the model normally optimizing the model to minimize the cross-entropy loss between the model's predictions and the ground truth labels. Second, we re-train the model by distilling the knowledge from the original pre-trained model into the pruned model. Both training techniques significantly improve the accuracy. Across our experiments, we found that distillation typically produced slightly better Top-1 accuracy compared to standard training with cross-entropy loss. In this phase of the pipeline, the accuracy has increased to 88.23% while maintaining the same parameter count and energy efficiency as the pruned model before it was re-trained.

In the final stage of the pipeline, we utilize quantization to further compress the model weights. Quantizing the model weight to the uint8 data type has no negative impact on accuracy. Furthermore, the quantization saves significant additional energy. The final stage of the pipeline achieves 88.28% accuracy (a 3.2% reduction from the original pre-trained model). The model now only consumes 1.74 Joules per sample (29% reduction in energy consumption). Notably, inference phase is also completed 27.6% faster.

V. Discussion

By far, quantization has proven to be the most simple, powerful, and robust technique to improve energy efficiency. Quantizing weights to the uint8 data type seems to achieve the best results. Not only does quantization typically result in no accuracy loss, but it also significantly reduces energy consumption. Interestingly, finding an effective quantization framework can be tricky. For example, the quantization functions built into the PyTorch library only support models running on CPU. Further, the many quantization libraries are specifically designed for large-language models. We found that the quanto library performs well and seems to generalize to any model.

Pruning a model effectively can be very difficult, especially if no additional training is applied to the pruned model. Often, pruning results in no improvement to energy efficiency. For example, unstructured pruning that masks out weights does not reduce the model's parameters or FLOPs. As a result no model compression occurs, and no improvements are made to speed nor energy efficiency. This is the case with PyTorch's built-in quantization library. In other cases, when structural pruning deletes weights, there is a risk of the model completely losing nearly all of its accuracy. This frequently occurred when we pruned a percentage of all layers. However, by considering model architecture, one can mitigate these issues. For example, the attention blocks contain a considerable amount of important information. Thus, avoiding pruning attention layers allowed the model to retain significantly higher accuracy. When pruning effectively, the size of the model (in parameters and FLOPs) and the energy consumption of the model seem to have a linear correlation with the pruning ratio. In contrast, the accuracy seem to follow more of an exponential trend with the pruning ratio. Hence, increasing the pruning ratio too rapidly can quickly deteriorate performance. Fortunately, re-training a pruned model to recover lost accuracy can be a powerful technique.

Distillation is an effective way to transfer knowledge from any larger teacher model to any smaller student model. Intuitively, the fully dense model (as a teacher) and the sparser pruned model (as a student) are good candidates for distillation. In our experiments, distillation can be useful for both recovering some lost accuracy and training a model from scratch. Fine-tuning and training the model without distillation produces good results as well. However, across our experiments, we notice that fine-tuning with distillation typically produced slightly better results. The effectiveness of distillation is also dependent on the quality of the training pipeline. For example, optimal hyper-parameters (e.g. batch size, learning rate, optimizer) would result in a better performance recovery. Additionally, other training techniques such as data augmentation could lead to improved performance. Overall, distillation is an effective technique to recover the accuracy of compressed models.

For the most part, we believe that RAPL and NVML provided aqequate data regarding energy consumption. Re-running the same experiment back-to-back produced results with an extremely low standard deviation. Thus, the tools were fairly consistent with their measurements. However, running the same experiment at different times in the day produced vastly different RAPL CPU measurements. We believe this is due to temperature changes at the location where the machine is kept. Hence, in order to compare results, we had to run all experiments within the same time frame. When interpreting results in the tables, values within the same table are comparable. However, comparing results between different tables may lead to a biased understanding of the CPU results.

Although achieving promising results, there remain several limitations in our proposals. First, while RAPL and NVML are useful and standard tools for measuring energy consumption, these tools are far from perfect. The overall system for energy profiling could be improved by utilizing additional tools to verify these energy measurements. For example, using a wall power meter could provide a more complete energy profile. Second, our study only evaluates the results of a single vision model on a single dataset. Since the code is generalizable, additional models and datasets can be tested for a more robust analysis. Third, the structural pruning algorithm could likely be enhanced. Specifically, by studying the model architecture in further detail, a more custom pruning algorithm could be designed. Fourth, the training pipeline for model distillation could be tuned, and additional training strategies such as image augmentation could be applied to further improve the model's accuracy. We hope to address these issues further in future works.

VI. Conclusion

In this study, we proposed a pipeline for energy efficient AI inference. By combining quantization, pruning, and distillation, we were able to significantly compress a pre-trained ViT model while retaining most of its performance. Independently, we found that quantization is an extremely powerful and robust technique to reduce energy consumption without sacrificing noticeable accuracy. While unstructural pruning does not reduce energy consumption at all, structural pruning can lead to significant improvements. However, it is important to consider the model architecture when selecting the layers to prune in order to prevent extreme drops in accuracy. Finally, distillation is a powerful technique that can be used to transfer knowledge from a fully dense model to a pruned counterpart. Overall, this study has shown algorithmic model compression techniques are promising in terms of reducing energy consumption in neural networks.

References

[1] "Optimize AI/ML Workloads for Sustainability: Part 3, Deployment and Monitoring — AWS Architecture Blog," April 27, 2022. https://aws.amazon.com/blogs/architecture/optimize-ai-ml-workloads-for-sustainability-part-3-deployment-and-monitoring/.

[2] H. David, E. Gorbatov, U. R. Hanebutte, R. Khanna, and C. Le, "RAPL: Memory power estimation and capping," in 2010 ACM/IEEE International Symposium on Low-Power Electronics and Design (ISLPED), Aug. 2010, pp. 189–194. doi: 10.1145/1840845.1840883.

[3] NVIDIA System Management Interface Documentation — developer.nvidia.com. https://developer.download.nvidia.com/compute/DCGM/docs/nvidia-smi-367.38.pdf

[4] "Quantization — PyTorch 2.6 Documentation." https://pytorch.org/docs/stable/quantization.html.

[5] J. Frankle and M. Carbin, "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks," Mar. 04, 2019, arXiv: arXiv:1803.03635. doi: 10.48550/arXiv.1803.03635.

[6] J. Tmamna et al., "Pruning Deep Neural Networks for Green Energy-Efficient Models: A Survey," Cogn Comput, vol. 16, no. 6, pp. 2931–2952, Nov. 2024, doi: 10.1007/s12559-024-10313-0.

[7] "torch.nn.utils.prune.ln_structured — PyTorch 2.6 documentation." https://pytorch.org/docs/stable/generated/torch.nn.utils.prune.ln_structured.html

[8] G. Fang, X. Ma, M. Song, M. B. Mi, and X. Wang, "DepGraph: Towards Any Structural Pruning," Mar. 23, 2023, arXiv: arXiv:2301.12900. doi: 10.48550/arXiv.2301.12900.

[9] G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," Mar. 09, 2015, arXiv: arXiv:1503.02531. doi: 10.48550/arXiv.1503.02531.

[10] Y. Yuan et al., "The Impact of Knowledge Distillation on the Energy Consumption and Runtime Efficiency of NLP Models," in Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering - Software Engineering for AI, Lisbon Portugal: ACM, Apr. 2024, pp. 129–133. doi: 10.1145/3644815.3644966.

[11] A. G. Álvarez, J. Castaño, X. Franch, and S. Martínez-Fernández, "Impact of ML Optimization Tactics on Greener Pre-Trained ML Models," Sep. 19, 2024, arXiv: arXiv:2409.12878. doi: 10.48550/arXiv.2409.12878.

[12] S. T. Sreenivas et al., "LLM Pruning and Distillation in Practice: The Minitron Approach," Dec. 09, 2024, arXiv: arXiv:2408.11796. doi: 10.48550/arXiv.2408.11796.

[13] A. Goel, C. Tung, Y.-H. Lu, and G. K. Thiruvathukal, "A Survey of Methods for Low-Power Deep Learning and Computer Vision," Mar. 24, 2020, arXiv: arXiv:2003.11066. doi: 10.48550/arXiv.2003.11066.

[14] A. Dosovitskiy et al., "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," Jun. 03, 2021, arXiv: arXiv:2010.11929. doi: 10.48550/arXiv.2010.11929.

[15] "CIFAR-10 and CIFAR-100 datasets." Accessed: Apr. 17, 2025. [Online]. Available: https://www.cs.toronto.edu/ kriz/cifar.html

[16] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," arXiv, Mar. 05, 2019. doi: 10.48550/arXiv.1711.05101.

[17] I. Loshchilov and F. Hutter, "SGDR: Stochastic Gradient Descent with Warm Restarts," arXiv, Aug. 25, 2017. doi: 10.48550/arXiv.1608.03983.