
Fine-tuning and Quantization of Dinov2 Model Project

Final Report

Zeqi Jia, Tianyi Ren, Di Wen, Furi Xiang
Carnegie Mellon University
Pittsburgh, PA 15213

1 Overview and Context

1.1 Abstract

Our final report focuses on enhancing the efficiency of deep learning models for medical imaging classification by leveraging a self-supervised Vision Transformer. Specifically, we train the DINOv2 model with an appended classification head on the HAM10000 skin lesion dataset, addressing the challenge of reducing the computational footprint of high-performing vision models without significantly sacrificing accuracy. To achieve this, we apply Quantization-Aware Training (QAT) to last few layers of the DINOv2 backbone and the classifier layer, which substantially reduces the model size and inference cost while maintaining competitive test accuracy.

1.2 Motivation

Medical Image Classification is a critical task in medical image analysis that involves categorizing medical images, such as X-rays, magnetic resonance imaging, and CT images, into distinct groups based on image type or the presence of specific structures and diseases. The goal is to take advantage of computer algorithms to automatically identify and classify these images based on their content, thereby aiding in diagnosis, treatment planning, and disease monitoring. Recent advances in deep learning and computer vision have provided promising solutions for medical image analysis, offering the potential to significantly improve healthcare and patient outcomes. However, the prevailing paradigm in training deep learning models requires large quantities of labeled training data, a process that is both time-consuming and cost-prohibitive, particularly in the medical domain, where expert annotation is essential. Our work is motivated by the need to develop resource-efficient models that can deliver high accuracy with reduced data dependency, making advanced medical diagnostic tools more accessible and practical in real-world healthcare settings.

1.3 Objective

Our objective is to develop a more efficient and resource-friendly model for medical image classification by using the DINOv2_vits14 architecture trained on the HAM10000 dataset. We aim to balance model size, inference speed, and classification accuracy by incorporating Quantization Aware Training (QAT) into our workflow and fine-tuning only the last few layers of the network while freezing the majority of the pre-trained layers. Our experiments have shown that quantization can substantially reduce the memory footprint of the model parameters by a factor of 4 and accelerate test time, although with a slightly decrease in test accuracy by less than one percent. In particular, the testing time for the quantized model is significantly shorter compared to that of the original full-precision model.

In addition, we conducted a comprehensive evaluation of both the full-precision and quantized models under various testing scenarios. Our evaluation compared metrics such as classification accuracy, inference time per image, and overall usage of computational resources. These experiments provided

concrete evidence of the trade-offs between model efficiency and performance, guiding further improvements toward a deployable, resource-constrained solution for medical image classification.

2 Related work & background

The most relevant prior works related to this project are primarily DINOv2 and Partially Binarized Large Language Models.

2.1 Literature review

DINO Self-supervised learning (SSL) has gained significant attention as an alternative to supervised learning, particularly in Vision Transformers (ViTs). The paper *Emerging Properties in Self-Supervised Vision Transformers* (Caron et al., ICCV 2021)[6] introduces DINO, a self-supervised learning method that leverages knowledge distillation without labels. This article discusses the application of self-supervised learning in Vision Transformer, focusing on the performance improvement and feature discovery of the self-supervised learning method DINO for ViT models. The key contribution of DINO can be divided into following three aspects: (1) self-Supervised learning in ViTs, (2) knowledge distillation without labels, and (3) computational efficiency and viability for model compression.

- **Self-Supervised learning in ViTs:** Self-supervised ViT features explicitly encode object boundaries and semantic segmentation information that is not apparent in supervised ViTs or CNNs. The learned features perform well in k-Nearest Neighbors (k-NN) classification, achieving 78.3% top-1 accuracy on ImageNet using ViT-S/8. The method employs a momentum encoder and multi-crop training, which are key factors in preventing pattern collapse and improving feature quality.
- **Knowledge distillation without labels:** DINO utilizes a self-distillation framework in which a network of students learns from a network of teachers who are exponential moving averages (EMAs) of students. Optimization uses centering and sharpening strategies to prevent trivial solutions. Unlike traditional knowledge distillation using pre-trained teachers, DINO dynamically builds teacher networks during training.
- **Computational efficiency and viability for model compression:** Compared to traditional supervised ViT, DINO achieves highly competitive accuracy while reducing computational cost, which makes it an excellent candidate for model quantization and compression.

DINOv2 This version introduces several key improvements over DINO to make it more powerful and efficient in self-supervised visual feature learning. According to the *DINOv2: Learning Robust Visual Features without Supervision* (Maxime et al., arXiv:2304.07193v2)[3], the main enhancements include: (1) scalability and stability of training, (2) improved feature quality and robustness, (3) higher modeling capabilities and knowledge distillation, (4) efficient training techniques, (5) enhanced evaluation and performance gains, and (6) better low-level visual features.

- **Scalability and stability of training:** Compared with the uncurated datasets commonly used in self-supervised learning, DINOv2 uses 142 million curated images for training, which dramatically improves the training scale and data quality. The optimized training pipeline not only converges faster and more stably, but also reduces memory usage by 2X compared to the previous method, while the improved Sinkhorn-Knopp centering and momentum scheduling further enhance the feature diffusion and optimization stability.
- **Improved feature quality and robustness:** By training on the collated LVD-142M dataset, DINOv2 significantly enhances the general-purpose feature extraction capability compared to previous self-supervised models that relied on uncollated web data. The improved data quality enables the model to generalize more effectively to a variety of visual tasks, including classification, segmentation, depth estimation, and retrieval. In addition, DINOv2 demonstrates a better patch-level representation, which enhances its performance in segmentation and dense prediction tasks, making it more suitable for a variety of computer vision applications.
- **Higher modeling capabilities and knowledge distillation:** DINOv2 trains a ViT model using 1B parameters and refines it into a smaller model that outperforms OpenCLIP and

other SSL methods. And the knowledge refined from the larger model is used to improve the generalization ability of the smaller model. Unlike previous methods, separate projection heads are used for different loss terms (DINO and iBOT) to avoid training instability.

- **Efficient training techniques:** DINOv2 improves training efficiency and computational resource utilization through several optimization techniques. The optimized FlashAttention provides faster computation and lower memory usage, making the attention mechanism more efficient. Meanwhile, Sequence Packing, an NLP technique, is introduced to reduce computational redundancy and improve parallel processing by concatenating multiple crops into a sequence. In addition, Fully Segmented Data Parallelism (FSDP) further optimizes the training process of large-scale models, allowing billion-parameter ViTs to be efficiently trained on multiple GPUs for better scalability and resource utilization.
- **Enhanced evaluation and performance gains:** DINOv2 performs well in standard benchmarks, not only outperforming previous self-supervised models such as iBOT and DINO, but also showing competitiveness against weakly supervised models such as OpenCLIP. In addition, in generalization tests such as ImageNet-A and ImageNet-Sketch, DINOv2 shows stronger robustness and better adaptation to different data distributions. Finally, DINOv2 has a strong out-of-the-box ability to achieve high levels of performance on a wide range of vision tasks even without additional fine-tuning.
- **Better low-level visual features:** Compared with previous SSL models, DINOv2 is able to capture richer pixel-level information, and thus performs better in segmentation and depth estimation tasks. Meanwhile, the introduction of KoLeo regularization further evens out the distribution of features and enhances the retrieval performance of the model. Furthermore, the higher resolution fine-tuning learning strategy with higher resolution (518×518) in the final stage of training significantly improves the performance of fine-grained classification and dense prediction tasks, making DINOv2 more generalizable to a wide range of visual tasks.

Partially Binarized Large Language Models (PB-LLM) is a novel approach to compressing LLMs through network binarization, a radical form of quantization that reduces model weights to 1 bit while preserving linguistic reasoning capabilities[4]. Network binarization had been an active field of research in computer vision based models, but by 2023 when PB-LLM was written, there had only been attempts in binarizing BERT-based language models with up to 110M parameters. PB-LLM aims to increase the parameter size of LLMs that can be binarized, addressing the bottleneck of LLM inference speed and reducing the model’s footprint.

- **Network Binarization:** The goal of binarization function is to represent floating point weights in the model into 1-bit binary values, which is achieved through a special set of forward and backward functions.

Forward function for network binarization:

$$\text{sign}(x) = \begin{cases} +1 & x \geq 0 \\ -1 & x < 0 \end{cases}$$

Backward function for network binarization:

$$\frac{\partial \mathcal{L}}{\partial x} = \begin{cases} \frac{\partial \mathcal{L}}{\partial \text{sign}(x)} & |x| \leq 1 \\ 0 & |x| > 1 \end{cases}$$

The sign function extracts the sign of the original floating point weight, transforming the entire weight matrix into its binary representation. Since the sign function is non-differentiable, straight-through estimator (STE) is often used to approximate the derivative.

- **Salient Weight and Partial Binarization:** Since naively binarizing LLMs would lead to the model collapsing, PB-LLM keeps some of the important weights in its original floating point precision to compress model size without sacrificing much of the model performance. The salient weight is determined through the saliency metric:

$$v_i = \frac{w_{F,i}^2}{[H^{-1}]_{ii}}$$

where $w_{F,i}$ is the i th full-precision weight, and $[H^{-1}]_{ii}$ is the i th diagonal entry of the inverse Hessian matrix (computed with respect to the quantization error). This allows for average quantization of a model to be less than 2-bits, and the team achieved quantization on LLaMA-7B with minimal accuracy loss.

- **Post-Training Quantization:** The quantization method introduced by PB-LLM can be categorized as post-training quantization, which means a fully trained model is taken as an input through the quantization process. This method aligns with our project’s need, since this allows us to use the pre-trained DINO model’s weight directly for quantization.

2.2 Background

For information on relevant evaluation metrics or baselines, please refer to Section 3 and Section 4.

3 Methodology

3.1 Model Description

DINOv2 is a self-supervised vision model based on the Vision Transformer (ViT) architecture, designed to learn high-quality image representations without labeled data. It employs a self-distillation framework with a teacher-student paradigm and multi-crop training.

The model consists of:

- **Vision Transformer (ViT) Backbone:** Processes image patches and extracts hierarchical features using multi-head self-attention.
- **Teacher-Student Framework:** A momentum-updated teacher network guides the student network to learn meaningful representations.
- **Projection Head:** A Multi-Layer Perceptron (MLP) that maps feature embeddings into a high-dimensional space for contrastive learning.

DINOv2 utilizes a self-distillation strategy:

- **Multi-Crop Training:** Different global and local image views enhance feature robustness.
- **Contrastive Loss:** Encourages similarity between different views of the same image while pushing apart different images:

$$L = - \sum_i \sum_{v, v^+} w_i \log \frac{\exp(f(v) \cdot f(v^+)/\tau)}{\sum_{v'} \exp(f(v) \cdot f(v')/\tau)} \quad (1)$$

where $f(v)$ is the feature representation of view v , τ is a temperature parameter, and w_i is a weighting factor.

- **Momentum Teacher Update:**

$$\theta_t \leftarrow \lambda \theta_t + (1 - \lambda) \theta_s \quad (2)$$

where θ_t and θ_s are the teacher and student model parameters, respectively, and λ is a momentum coefficient.

3.2 Skin lesion classification dataset: HAM10000

We utilized the HAM10000 dataset to evaluate the performance of the our post-training low-bit quantized DINOv2 model.

3.2.1 Description

The HAM10000 dataset (Human Against Machine with 10,000 training images) is a publicly available dataset that contains 10,015 dermatoscopic images of pigmented skin lesions. These images were collected from various sources and annotated by expert dermatologists. The dataset includes a representative set of all major diagnostic categories in the field of pigmented skin lesions and is classified into seven types of skin disease:

- Actinic keratoses and intraepithelial carcinoma / Bowen’s disease(akiec)
- Basal cell carcinoma (bcc)
- Benign keratosis-like lesions (including solar lentigines, seborrheic keratoses, and lichen planus-like keratoses) (bkl)
- Dermatofibroma (df)
- Melanoma (mel)
- Melanocytic nevi (nv)
- Vascular lesions (including hemangioma, angiokeratoma, pyogenic granuloma, and hemorrhage) (vasc)

The HAM10000 dataset is widely used for training and benchmarking machine learning models for skin lesion classification.

3.2.2 Data Preprocessing & Augmentation

- **Preprocessing Pipeline** To optimize the performance of the DINOv2 model on the HAM10000 dataset, we designed a preprocessing and augmentation pipeline aimed at improving model generalization and robustness. For the training set, we applied a RandomResizedCrop operation with an output size of 224×224 pixels to introduce scale invariance, followed by RandomHorizontalFlip to simulate left-right asymmetries commonly observed in dermoscopic images. Subsequently, images were converted to tensors via ToTensor, and normalized using the standard ImageNet statistics (mean = [0.485, 0.456, 0.406]; standard deviation = [0.229, 0.224, 0.225]) to ensure consistent input distributions. For the validation set, images were first resized so that the shorter side measured 256 pixels using Resize(256), followed by CenterCrop(224) to extract the central region for evaluation, maintaining a standardized input size.
- **Augmentation Strategies** To further enhance model generalization, we employed geometric transformations like RandomResizedCrop and RandomHorizontalFlip, which add scale and orientation variations, as well as photometric transformations like Normalize, which standardizes pixel values. These augmentation techniques help simulate real-world variations, increase the diversity of the dataset, and improve model robustness. By applying these techniques, we expect improved classification accuracy, reduced overfitting, and enhanced adaptability to varying input conditions, leading to a more reliable and generalizable model for skin lesion classification.

3.3 Evaluation Metric

3.3.1 HAM10000 (Skin Lesion Classification)

Task: Multi-class classification (7 categories).

Metrics:

- **Accuracy:** Overall correctness.
- **Inference time:** 1 batch inference time in ms.
- **Model size:** Saved model size in MB.

3.4 Quantization Methods

3.4.1 TensorRT

TensorRT provides Python API for its quantization workflow. For baseline, we built the INT8 quantization using the workflow outlined as below:

- **Export to ONNX:** The pre-trained Pytorch model was exported to the ONNX (Open Neural Network Exchange) format to ensure framework-agnostic compatibility.
- **TensorRT Parsing:** The ONNX model was parsed using TensorRT’s ONNX parser, which initializes the network definition and identifies layer operations.

- **Weight Calibration:** After parsing the ONNX model, TensorRT provides calibration workflow for post-training quantization. The calibration workflow allows measuring the distribution of activations within each activation tensor as the network executes on representative input data and then uses that distribution to estimate scale values for each tensor. The calibration dataset is created with the HAM10000 dataset and IInt8EntropyCalibrator2 interface, which handles the representative data extraction.
- **Engine Configuration and Build:** The builder will first a fp32 engine that runs on the calibration set, and then apply different Applying the correct setting to allow for building INT8 quantized model engine from the original model.
 - `config.set_flag(trt.BuilderFlag.INT8) # Set the engine for INT8 Quantization`
 - `config.set_memory_pool_limit(trt.MemoryPoolType.WORKSPACE, 10 << 30) # Allocate 10GB of memory for quantization`

3.4.2 Pytorch

In midterm report, we employ post-training dynamic quantization to reduce the computational demands of the DINOv2_vits14 model with compromising its classification accuracy. We begin by defining a custom module, `QuantizedDINOv2Classifier`, which encapsulates the DINOv2 backbone and the classification layer. Within the module’s `forward` method, an input image is first passed through the quantized DINOv2 model to extract feature representations. These features are then normalized using the model’s internal normalization layer before being forwarded to the quantized classifier. Dynamic quantization is applied using PyTorch’s `torch.quantization.quantize_dynamic` function on the `nn.Linear` layers in both the backbone and the classifier, with the data type set to `torch.qint8`. The resulting quantized modules are moved to the CPU to further enhance inference efficiency. Finally, we evaluate the quantized model on a test dataset by iterating over test batches in a no-gradient context, using `tqdm` to monitor the test accuracy in real time.

3.4.3 TensorRT Model Optimizer

TensorRT Model Optimizer is a python library within the TensorRT ecosystem that’s comprised of model optimization techniques such as quantization, pruning, distillation, etc. For quantization, Model Optimizer supports quantizing through ONNX models or Pytorch models. Model Optimizer quantization also require an weight calibration dataset defined by the user. During quantization, Model Optimizer will automatically insert `TensorQuantizer` modules between layers that support quantization.

3.5 Loss Function

The loss function used for training the model is the **Cross-Entropy Loss**, which is widely used in multi-class classification tasks. It measures the difference between the predicted probability distribution and the true class labels.

$$\mathcal{L} = - \sum_{i=1}^C y_i \log(\hat{y}_i) \quad (3)$$

where:

- C is the number of classes.
- y_i is the ground-truth label, represented as a one-hot encoded vector.
- $\hat{y}_i$ is the predicted probability for class i , obtained using the softmax function:

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^C e^{z_j}} \quad (4)$$

where z_i represents the raw output (logits) of the model before applying softmax.

The Cross-Entropy Loss encourages the model to output high probabilities for the correct class and penalizes incorrect predictions. It is optimized using the AdamW optimizer during training.

4 Baseline and Extensions

4.1 Baseline Selection & Evaluation

We selected the full-precision DINOv2_vits14 linear probing model as the baseline for medical image classification on the HAM10000 dataset. This model was chosen due to its strong performance in extracting robust visual features and its established track record in self-supervised learning frameworks, which aligns well with our overall objective of balancing high accuracy with computational efficiency.

The baseline was implemented by training the original DINOv2_vits14 model with an attached classification head. The detailed testing on the full precision model yielded a test precision of 80.77%, an inference time of 20.40 ms for 1 batch on the L4 GPU and model size of 346 MB. These baseline metrics serve as critical reference points to assess the impact of our subsequent quantization and fine-tuning techniques.

4.2 Implemented Experiments

4.2.1 Partial Fine-Tuning

Fine-tuning is performed on the **DINOv2 ViT-B/14** backbone to enhance its ability to classify skin lesion images. The process involves two training approaches:

- **Linear Probing:** The pre-trained model is used without modifying its feature extraction layers. Only the newly added classifier head is trained while the backbone remains frozen.
- **Partial Fine-Tune:** The last two transformer blocks (`blocks.10` and `blocks.11`), along with the normalization layer (`norm`), are unfrozen and updated during training. This allows the model to adapt to the specific dataset while leveraging pre-trained knowledge.

During training, different learning rates are assigned:

- The classifier head is trained with a higher learning rate (1×10^{-3}) since it is randomly initialized.
- The unfrozen backbone layers are fine-tuned with a lower learning rate (1×10^{-5}) to prevent catastrophic forgetting.

Gradient updates are performed using the AdamW optimizer, and automatic mixed precision (AMP) is employed to optimize memory usage.

5 Comparison of Results

Our baseline linear probing achieved an accuracy of 80.77%. When we partially fine-tuned the last two layers, accuracy increased to 82.42%, showing that limited fine-tuning improves performance while keeping training efficient. Table 1 presents the comparison:

Method	Accuracy (%)	Inference Time (ms)	Model Size (MB)
Linear Probing	80.77	20.40	346
Partial Fine-Tune	82.42	20.35	346
Fine-Tune + PTQ	79.79	6.78	87.5
Fine-Tune + QAT	82.03	6.84	87.5

Table 1: Comparison of Fine-Tuned and Quantized Models on HAM10000

For quantization, we observed the following:

- **Post-Training Quantization (PTQ)** slightly reduced accuracy to **79.97%**.
- **Quantization-Aware Training (QAT)** preserved accuracy at **82.03%**, nearly matching the fine-tuned model.

Despite the small accuracy drop, quantization brought major efficiency gains:

- Inference time decreased from **20.4ms** to **6.84ms**.
- Model size dropped from **346MB** to **87.5MB**.

These results demonstrate that:

- **Partial fine-tuning** is an effective way to improve performance without overfitting.
- **QAT** is essential for retaining accuracy under quantization.
- Our approach **balances performance and deployment efficiency**, making it practical for real-world applications.

We further evaluated the effectiveness of two int-4 quantization format at retaining the model accuracy represented in Table 2.

Quantization Format	Test Accuracy (%)	PTQ Accuracy (%)	QAT Accuracy (%)
Original Model	83.47	-	-
INT8_SMOOTHQUANT	-	82.97	83.34
INT4_AWQ	-	76.71	81.74
W4A8_AWQ	-	73.08	81.18

Table 2: Accuracy Comparison of Models Quantized in Different Formats

The result of this table further reinforces the observation that QAT has instrumental impact in increasing the model accuracy from that of the PTQ phase. It also shows that for 4-bit integer quantization (INT4_AWQ, W4A8_AWQ), there’s a greater importance for applying QAT, since QAT provides the greatest accuracy increase for int-4 quantization. Int-4 quantization also shows noticeable accuracy drop compared to int-8 quantization (from around 83% to 81%). We were unable to benchmark the performance increase with the int-4 quantization, since our benchmarking environment onnxruntime only support precision as low as int-8. The trade-off between selecting int-8 quantization or int-4 quantization should be further investigated with deployment framework that supports int-4 quantization.

6 Results & Analysis

Accuracy and Inference Time Comparison

A bar-chart visualization of the results in Table 1 makes clear the trade-offs between accuracy and inference speed for different model versions. Linear probing (the frozen backbone) achieves 80.77% accuracy, while partial fine-tuning of the last layers boosts accuracy to 82.42%. Applying post-training quantization (PTQ) to the fully fine-tuned model cuts the model size to 87.5 MB and inference time to ~ 6.8 ms, but reduces accuracy to $\sim 79.8\%$. In contrast, quantization-aware training (QAT) of the same quantized model retains accuracy ($\sim 82.03\%$) almost as high as the full-precision fine-tuned model, while keeping the inference time (~ 6.84 ms) and model size (87.5 MB) nearly unchanged from PTQ. Overall, partial fine-tuning plus QAT offers nearly the best accuracy (82.0% vs. 82.4% partial-fine-tuned) with the dramatic efficiency gains (size cut by 4 $\times$, time cut by $\sim 3\times$).

Training vs. Validation Metrics

The table below contrasts each model/version and phase. Since the training-phase accuracies were not explicitly reported, the “Training” rows are marked as “-” (not reported). Validation (test) accuracy, inference time, and model size are taken from Tables 1–2:

Model / Method	Phase	Accuracy (%)	Inference Time (ms)	Model Size (MB)
Linear Probing	Training	–	–	346
Linear Probing	Validation	80.77	20.40	346
Partial Fine-Tune	Training	–	–	346
Partial Fine-Tune	Validation	82.42	20.35	346
Fine-Tune + PTQ	Training	–	–	87.5
Fine-Tune + PTQ	Validation	79.79	6.78	87.5
Fine-Tune + QAT	Training	–	–	87.5
Fine-Tune + QAT	Validation	82.03	6.84	87.5

Table 3: Side-by-side comparison of model versions, phases, and key metrics.

Discussion and Analysis

- Efficiency vs. Accuracy Trade-off:** The results show that careful fine-tuning plus quantization yields a very favorable trade-off. Partial fine-tuning improved accuracy with little extra cost. Meanwhile, aggressive quantization (4× size reduction) led to a major speedup (inference time from ~20.4 ms to ~6.8 ms) with only a *minor* accuracy penalty when QAT is used. Indeed, QAT retained 82.03% accuracy versus 82.42% (before quantization), a drop of ~0.4%. These figures underscore that quantization dramatically improves deployment efficiency with modest accuracy loss. The abstract also noted a < 1% drop in accuracy after quantization.
- Key Observations:** Partial fine-tuning increased accuracy (80.77% → 82.42%), showing that even unfrozen backbone layers boost performance. PTQ alone reduced accuracy to ~79.97%, whereas adding QAT brought accuracy back up to ~82.03%. In other words, *QAT is essential to preserve accuracy under quantization*. The inference speed-up (~3× faster) and size reduction (4× smaller) obtained by quantization are well supported by the data. Overall, the approach successfully balances high accuracy with lightweight deployment.
- Limitations and Risks:** A key risk is accuracy degradation under *aggressive* quantization. Table 2 shows that 4-bit PTQ (INT4) sharply drops accuracy (to ~73–77%), a large degradation. Only with QAT does 4-bit quantization recover performance (~81–82%). Thus, cutting precision from 8-bit to 4-bit yields ~5–10% worse accuracy if QAT is not used. Another limitation is deployment support: as noted, the team could not evaluate 4-bit inference speed because their ONNX framework only supports up to INT8. In practice, INT4 models may not run on current hardware or software, hindering use of the smallest model. These points show that while quantization offers efficiency, *over-quantizing* without proper methods (QAT) risks unacceptable accuracy loss, and framework support for lower precision (INT4) may be lacking.
- Sensitivity to Input/Quantization Changes:** The accuracy is sensitive to how quantization is applied. For example, the same int8 quantized model has ~82.97% accuracy with PTQ and 83.34% with QAT, nearly unchanged from the full-precision baseline (83.47%). In contrast, int4 quantization without QAT plummets accuracy by ~6–10%. This indicates that smaller bit-widths make the model more sensitive to quantization noise. In summary, the results suggest that small changes in the quantization method (e.g. adding QAT) have a significant impact on accuracy. Robust deployment should therefore include QAT or mixed-precision schemes, especially if the input data vary or come from different sensors, to guard against steep accuracy drops.

7 Future Directions

7.1 Hardware Deployment

- Explore INT4-Compatible Inference Frameworks:** Since ONNX Runtime currently lacks support for INT4 quantization, future work should evaluate deployment frameworks and hardware that natively support 4-bit inference. For example, NVIDIA TensorRT on Ampere/Hopper GPUs offers INT4 weight-only quantization capabilities. Custom FPGA implementations (e.g., using Xilinx or Intel FPGAs) can be configured for 4-bit operations.

Specialized edge AI accelerators (such as the Google Coral Edge TPU) should also be investigated. These platforms may better exploit the reduced model size and provide faster on-device inference for skin lesion analysis.

- **Benchmark on Specialized Hardware:** Perform detailed profiling on target devices to quantify runtime and resource usage. For instance, measure inference latency, throughput, and memory footprint on NVIDIA Jetson Orin (using TensorRT INT4) and compare against CPU/GPU baselines. Evaluate the quantized model on a Xilinx or Intel FPGA development board to assess real-time performance and power consumption. These benchmarks will reveal computational and memory bottlenecks and guide optimizations, which is critical for deployment in resource-constrained clinical or embedded settings.

7.2 Advanced Quantization Techniques

- **Hybrid Bit-Width Quantization:** Investigate schemes that assign different bit precisions to different layers of the model. For example, keep sensitive components (such as the initial patch embedding or final classifier) at INT8 or FP16, while quantizing the remaining layers to INT4. This hybrid approach can preserve accuracy in critical parts of the network while still significantly reducing overall model size.
- **Adaptive and Layer-wise Quantization:** Explore adaptive quantization strategies that tailor precision on a per-layer or per-channel basis. For instance, perform layer sensitivity analysis or use quantization-aware training to identify layers most affected by quantization noise. Preserve higher precision for layers like normalization or attention if needed, and use learnable scale/zero-point parameters to optimize quantization for each channel. Such adaptive methods aim to retain important feature fidelity under low-bit quantization.

7.3 Fine-Tuning and Training Improvements

- **Expanded Layer Unfreezing:** Broaden the partial fine-tuning strategy by gradually unfreezing additional transformer blocks of the DINOv2 model. Instead of only training the projection head, subsequent layers (e.g., the last few encoder blocks, followed by earlier ones) can be fine-tuned with lower learning rates. This layered unfreezing enables the model to adapt deeper feature representations to the dermoscopic domain while still leveraging pretrained weights, potentially improving classification accuracy on HAM10000.
- **Advanced Optimization Schedules:** Apply advanced learning rate schedules and regularization to stabilize fine-tuning. For example, use cyclical learning rates or cosine annealing to better explore the loss landscape. Experiment with optimizers like AdamW with carefully tuned weight decay. These strategies can help the model converge more reliably on limited medical data and reduce overfitting when more layers are trainable.

8 Administrative Details

8.1 Team Contributions

Team Member	Contributions
Zeqi Jia	Methodology: Model Description & Dataset & Evaluation Metric
Tianyi Ren	Overview and context; Code debugging Development, execution, & evaluation of QAT program.
Furi Xiang	Objective, ModelOpt Quantization methodology & results analysis
Di Wen	Related work & background for DINO&DINOv2, results & analysis

Table 4: Team Contributions

8.2 GitHub Repository

All project code and related materials are available on our GitHub repository:
<https://github.com/TianyiRnj/dino-classification-extent>.

References

- [1] Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., & Joulin, A. (2021). Emerging Properties in Self-Supervised Vision Transformers. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 9630-9640.
- [2] Papa, L., Russo, P., Amerini, I., & Zhou, L. (2023). A Survey on Efficient Vision Transformers: Algorithms, Techniques, and Performance Benchmarking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46, 7682-7700.
- [3] Oquab, M., Darcet, T., Moutakanni, T., Vo, H.Q., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., Assran, M., Ballas, N., Galuba, W., Howes, R., Huang, P., Li, S., Misra, I., Rabbat, M.G., Sharma, V., Synnaeve, G., Xu, H., Jégou, H., Mairal, J., Labatut, P., Joulin, A., & Bojanowski, P. (2023). DINOv2: Learning Robust Visual Features without Supervision. *ArXiv, abs/2304.07193*.
- [4] Shang, Y., Yuan, Z., Wu, Q., & Dong, Z. (2023). PB-LLM: Partially Binarized Large Language Models. *ArXiv, abs/2310.00034*.
- [5] Yoo, S.W., Kim, E., Jung, D., Lee, J., & Yoon, S. (2023). Improving Visual Prompt Tuning for Self-supervised Vision Transformers. *ArXiv, abs/2306.05067*.
- [6] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jegou, Julien Mairal, Piotr Bojanowski, Armand Joulin. Emerging Properties in Self-Supervised Vision Transformers. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, Montreal, QC, Canada, 2021, pp. 9630-9640.