Knowledge Distilation for Smaller Models in Computer Vision

Wonjun Park*, **Abhinay Kotla***
Computer Science and Engineering
University of Texas at Arlington
Arlington, TX, USA
{ *wxp7177, axk5827* }*@mavs.uta.edu*

***Abstract*—Knowledge distilation is a technique that allows deep learning models to transfer knowledge from a larger, and more complex model (the teacher) to a smaller, and more efficient model (the student). The technique is able to be adopted in various tasks including image classification and image generation. Among the tasks in computer vision, the paper specifically focuses on the image classification task to demonstrate the effectiveness of knowledge distilation, further outlining the framework to apply the technique to other tasks. The results from the image classification task that addressed in this paper shows that the knowledge distilation is able to successfully decrease the size of the model while preserving the performance of the model. Notably, the distilled model achieves slightly better performance than the teacher model. We expect that the framework can be applied to other tasks in computer vision without performance degradation.**

***Index Terms*—deep learning, computer vision, knowledge distillation, teacher-student learning, and quantization.**

I. Introduction

Deep neural networks have achieved remarkable success across a wide spectrum of computer vision problems, from a simple image classification task, ImageNet [1] for example, to high-fidelity image generation with diffusion [2]–[4] and Generative Adversarial Network (GAN) models [5], [6]. Much of this progress, however, has been powered by ever-larger model capacities; billions of parameters, expansive training datasets, and considerable computational budgets, even though such computationally expensive models deliver state-of-the-art performance. Their computational-intensive nature poses serious obstacles for deploying them on resource-constrained devices and real-time and latency-sensitive backend services such as smartphones or even low-end GPUs. Closing this gap between accuracy and efficiency therefore becomes one of central research questions in deep learning.

Image classification is a basic and fundamental task in computer vision. It is the task of predicting the class of an image from a set of classes. Many other tasks like object detection, image segmentation, and image generation are successfully built on top of image classification since AlexNet [7] overwhelmed the traditional methods in computer vision with its deep learning-based approach.

* Equal Contribution.

This paper is a part of the final project of the course for Spring 2025, CSE-5368-001: Neural Networks at the University of Texas at Arlington under Dr. Francklin Rivas.

In this project, the initial plan was to implement a knowledge distillation framework for an image inpainting task with diffusion models which restores images from masked regions. Unfortunately, however, while training scripts for student models were successfully implemented and run with the guidance of the teacher model from DiffIR [4], not only the problem of the memory but also the problem of the computing operations led the training time to be extremely long. More than 200 hours were required per epoch for training on the Places dataset [8] with our student model from DiffIR, even if it operated on 8-bit floating point. Therefore, we decided to change the task from image inpainting to image classification to demonstrate the effectiveness of knowledge distillation and to further the foundation of the framework to apply the technique in the future.

Based on our code implemented in Assignment 2 of the course, CSE 5368, we built a knowledge distillation framework for the image classification task. In the subsequent sections, the paper describes our knowledge distillation method, presents experimental results demonstrating its efficiency, and discusses potential directions for future work.

II. Related Works

In this section, the paper reviews the performance optimization in the image classification task via three approaches: *II-A lightweight backbones*, *II-B knowledge distillation*, and *II-C quantization*. The first approach focuses on the design of efficient architectures, the second approach emphasizes the transfer of knowledge from a larger model to a smaller one, and the third approach discusses the quantization of models, which is a common technique used to reduce the size of models and improve their performance on low-compute devices.

A. Lightweight Backbones

Early breakthroughs in large-scale image classification were driven by deeper and wider Convolutional Neural Network (CNN) such as AlexNet, VGG, Inception and ResNet [7], [9]–[11]. While these architectures steadily improved top-1 accuracy on ImageNet [1], their memory footprint and compute cost grew proportionally, prompting a parallel line of research on efficiency-focused approaches. Lightweight backbones such as MobileNet [12], [13], EfficientNet [14], and Data-efficient

Vision Transformer (DeiT) [15], demonstrated that accuracy can be retained with careful depth-width scaling or attention re-use. Nevertheless, these models still require their heavy weights when trained from scratch.

B. Knowledge distillation

Knowledge distillation is a technique used to transfer knowledge from a large complex model to a smaller, more efficient model, enabling deployment on resource-constrained devices without significant loss of performance. Hinton et al. [16] introduced logit-matching between a cumbersome teacher and a compact student. Subsequent works enriched the transfer signal by aligning intermediate feature maps [17] or the spatial attention of convolutional layers [18]. The latter approach, self-distillation [19], was shown to be effective in improving the performance of a single model by training it with the same architecture.

C. Quantization

Quantization techniques have been highly regarded recently for their ability to allow Large Language Model (LLM) to run on low-compute devices. Especially, Microsoft [1] has pioneered the quantization of LLM with their LLM Phi-3 [20] and Phi-4 [21] models, as well as the 1-bit quantized model, BitNet [22]. Those floating-point quantization techniques can also be applied to the image classification task, enabling the deployment of large models on low-compute devices.

III. Methodology

A. Intel Image Dataset

The Intel Image Classification Dataset [23] was curated as part of a data hackathon organized by Intel [2] on an online platform Kaggle [3], with the aim of fostering engagement and innovation within data science communities. The dataset comprises 25,000 RGB color images, each with a resolution of 150 $\times$ 150 pixels, depicting a variety of natural and urban scenes from around the world. The images, originally captured by Jan Böttinger on Unsplash [4], are labeled into six distinct classes: buildings, forest, glacier, mountain, sea, and street.

The dataset is partitioned into three subsets: approximately 14,000 images for training, 3,000 images for testing, and 7,000 images for prediction tasks. Each subset is provided as a separate zipped file through Kaggle.

The primary object of this dataset is to support participants in developing and evaluating image classification models. By providing a diverse and well-labeled collection of scenes, the dataset encourages the advancement of robust classification algorithms capable to distinguish objects in various environments with reasonable accuracy.

[1]https://www.microsoft.com/en-us/
[2]https://www.intel.com/content/www/us/en/homepage.html
[3]https://www.kaggle.com/
[4]https://unsplash.com/

B. Model Architecture

In this section, the architecture of the teacher model is described. We implemented a custom CNN model, including the residual block [11], which hierarchically extracts and aggregates spatial features from the input images, before mapping them to one of six scene categories.

The model consists of 12 residual blocks, progressively increasing the channel capacity while reducing spatial resolution, and a fully connected layer at the end as a classifier which regularly decreases the number from 1024, 512, 256, 64, and 6. In total, 26,534,358 parameters are in the model.

All operations in the model are performed using floating point 32, which is the default precision in PyTorch [24] in most GPUs.

C. Weight Reduction

From the teacher model, the number of residual blocks and the layers of the classifier are reduced. In the perspective of weight reduction, four reduction models are created, excluding the teacher model, which are 10, 8, 6, and 4 residual blocks and (512, 256, 64, 16), (256, 64, 16), (128, 64, 16), and (64, 32) layers of the classifiers. The number of parameters in the models is 6,601,686, 1,648,470, 408,854, and 98,294, respectively.

D. Precision Quantization

The quantization is utilized in both model weights and floating point operations. Two different levels of precision, floating point 16 and 8, are used. To implement the quantization, `torch.set_default_dtype()` and the `bnb` [25] Python framework are used.

IV. Experiments

A. Training Details

Several student models are configured based on the teacher model in III-B, applying the reduction and quantization techniques introduced in III-C and III-D. All training processes are conducted on a single NVIDIA GeForce RTX 4070 Laptop GPU. The PyTorch [24] framework forms the basis of the implementation, leveraging its extensive libraries and tools for deep learning. Adam optimizer [26] is employed with a learning rate of 0.001. A mini-batch size of 224 is used as well as the KL Divergence is utilized as the loss function, which effectively measures the differences between the distributions predicted by the teacher and the outputs of the student models. Early stopping is applied to prevent overfitting with a patience of 3 epochs.

B. Evaluation

The performance of the teacher model is on the table I. The table II shows the performance of the student models along with reducing the number of parameters in the models. Furthermore, the table III presents the performance along with the quantization of the model weights and floating-point operations.

TABLE I
TEACHER MODEL PERFORMANCE

	Teacher model
the number of parameters	26,534,358
accuracy	85.77%
training time per epoch (secs)	72.32

TABLE II
WEIGHT REDUCTION COMPARISON

	10 blocks, (512, 256, 64, 16)
the number of parameters	6,601,686
accuracy	83.23%
training time per epoch (secs)	68.6

	8 blocks, (256, 64, 16)
the number of parameters	1,648,470
accuracy	**86.43%**
training time per epoch (secs)	67.0

	6 blocks, (128, 64, 16)
the number of parameters	408,854
accuracy	85.77%
training time per epoch (secs)	66.7

	4 blocks, (64, 32)
the number of parameters	98,294
accuracy	84.57%
training time per epoch (secs)	**64.7**

Corresponding to the common sense, the training time per epoch is reduced as the number of parameters in the model is reduced. The smallest number of the student model shows the fastest training time per epoch 64.7 seconds. Also, another that the performance is not proportional to the number of parameters in the model when it comes to the fixed dataset and the architecture of the model is observed, highlighted by the 8-block student model, which shows the best performance.

With the best performance model, an experiment that applies the precision quantization is conducted. The result in Table IV demonstrates that the 8-block student model benefits from fp16 precision. With an accuracy of 87.00% and the fastest training time per epoch of 60.8 seconds, fp16 quantization outperforms fp8, which achieves 86.07% accuracy with a slower training time of 69.0 seconds. This highlights that choosing the appropriate precision can significantly enhance both model performance and computational efficiency.

TABLE III
PRECISION QUANTIZATION ON TEACHER MODEL

	fp16	*fp8*
accuracy	85.54%	**86.38%**
training time per epoch (secs)	**61.3**	70.5

TABLE IV
PRECISION QUANTIZATION ON 8 BLOCKS

	fp16	*fp8*
accuracy	**87.00%**	86.07%
training time per epoch (secs)	**60.8**	69.0

V. CONCLUSION

In this paper, we have presented a comprehensive study on the parameter reduction and quantization of image classification models using knowledge distillation. We have demonstrated that knowledge distillation is an effective technique for transferring knowledge from a larger, more complex model (the teacher) to a smaller, more efficient model (the student) in the context of image classification tasks. Based on this result, we believe that our knowledge distilation framework can also be applied to other tasks in computer vision including image inpainting without significant performance degradation. We expect that this framework will pave the way for the development of other efficient models in various computer vision tasks.

REFERENCES

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[2] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[3] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 684–10 695.

[4] B. Xia, Y. Zhang, S. Wang, Y. Wang, X. Wu, Y. Tian, W. Yang, and L. Van Gool, "Diffir: Efficient diffusion model for image restoration," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 13 095–13 105.

[5] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[6] K. Nazeri, E. Ng, T. Joseph, F. Z. Qureshi, and M. Ebrahimi, "Edgeconnect: Generative image inpainting with adversarial edge learning," *arXiv preprint arXiv:1901.00212*, 2019.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[8] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 6, pp. 1452–1464, 2017.

[9] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[10] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[12] A. G. Howard, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[13] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[14] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*. PMLR, 2019, pp. 6105–6114.

[15] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *International conference on machine learning*. PMLR, 2021, pp. 10 347–10 357.

[16] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[17] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," *arXiv preprint arXiv:1412.6550*, 2014.

[18] S. Zagoruyko and N. Komodakis, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," *arXiv preprint arXiv:1612.03928*, 2016.

[19] L. Zhang, J. Song, A. Gao, J. Chen, C. Bao, and K. Ma, "Be your own teacher: Improve the performance of convolutional neural networks via self distillation," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 3713–3722.

[20] M. Abdin, J. Aneja, H. Awadalla, A. Awadallah, A. A. Awan, N. Bach, A. Bahree, A. Bakhtiari, J. Bao, H. Behl *et al.*, "Phi-3 technical report: A highly capable language model locally on your phone," *arXiv preprint arXiv:2404.14219*, 2024.

[21] M. Abdin, J. Aneja, H. Behl, S. Bubeck, R. Eldan, S. Gunasekar, M. Harrison, R. J. Hewett, M. Javaheripi, P. Kauffmann *et al.*, "Phi-4 technical report," *arXiv preprint arXiv:2412.08905*, 2024.

[22] J. Wang, H. Zhou, T. Song, S. Mao, S. Ma, H. Wang, Y. Xia, and F. Wei, "1-bit ai infra: Part 1.1, fast and lossless bitnet b1. 58 inference on cpus," *arXiv preprint arXiv:2410.16144*, 2024.

[23] P. Singh, "Intel image classification," https://www.kaggle.com/datasets/puneet6060/intel-image-classification, 2019, accessed: 2025-04-21.

[24] A. Paszke, "Pytorch: An imperative style, high-performance deep learning library," *arXiv preprint arXiv:1912.01703*, 2019.

[25] bitsandbytes Foundation, "bitsandbytes: Accessible large language models via k-bit quantization for pytorch," https://github.com/bitsandbytes-foundation/bitsandbytes. [Online]. Available: https://github.com/bitsandbytes-foundation/bitsandbytes

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.