VANDERBILT UNIVERSITY

Extra Credit Projects: Model and Data Compression

4262/5262 Foundations of Machine Learning

Chayne Thrash & Soheil Kolouri

1 Introduction

Recent advancements in Artificial Intelligence (AI) and Machine Learning (ML) have led to the development of increasingly large models and datasets, creating substantial challenges for academic and research institutions with limited computational resources. These obstacles have restricted much of the progress in AI and ML to a few well-resourced industrial entities. Efficient techniques for the storage, transmission, and processing of large-scale data and models are crucial to overcoming these barriers and broadening access to cutting-edge AI technology. Developing novel model and data compression methods offers a promising solution, potentially democratizing AI and enabling diverse institutions—from academic labs to tech startups—to contribute meaningfully to the field. Furthermore, advancements in extreme compression strategies not only expand access but also deepen our theoretical understanding of large foundational models. The following projects focus on model and data compression in machine learning and more particularly deep learning.

2 Project 1: Image Compression via INRs

Implicit Neural Representations (INRs) [3] offer a novel approach to image compression by encoding images not as pixel arrays but as the weights of a neural network optimized to map pixel locations to RGB values. In this framework, the image is effectively represented by a compact model that can be reconstructed by evaluating the neural network. This method reduces storage requirements and has shown potential in surpassing traditional image compression standards like JPEG at lower bit rates.

2.1 Problem Formulation:

Let I denote an image, where $I[x]$ gives the RGB values at pixel location $x = (x_1, x_2)$. The goal is to approximate I using a neural network $f_\theta : \mathbb{R}^2 \to \mathbb{R}^3$ with parameters θ, which maps each pixel location to its corresponding RGB values. This is achieved by minimizing the mean squared error (MSE) objective function:

$$\min_\theta \text{MSE}(\theta) = \sum_x \|f_\theta(x) - I[x]\|_2^2$$

where the sum is over all pixel locations. The optimized parameters θ then serve as the compressed representation of the image, and the image can be reconstructed by evaluating f_θ at each pixel location. COIN [1] showed that using sinusoidal activations [3] and quantizing the network weights, θ, to 16-bit and performing a hyperparameter search, INRs can outperform JPEG as a compression method.

In this project, you will push the limits of the COIN method for image compression. The project consists of three main tasks as discussed below.

2.2 Task 1. Establishing Baselines

The first task is to establish a reliable baseline for your method. To that end, you will need to replicate the results reported in [1] by training an INR and report PSNR over the Kodak dataset for varying bit-per-pixel (bpp). You should use the same architecture and experimental setup as in section A of the supplemental material of the COIN [1] paper. After training, you should quantize weights to 16-bits and then compute Peak signal-to-noise ratio (PSNR). The PSNR is a common metric used to evaluate the quality of image compression by comparing the compressed image to the original. For an image I and its compressed approximation $\hat{I}$, represented by the neural network f_θ, PSNR is defined as:

$$\text{PSNR} = 10 \cdot \log_{10}\left(\frac{\text{MAX}^2}{\text{MSE}(\theta^*)}\right),$$

where MAX is the maximum possible pixel value of the image (typically 255 for 8-bit images). Higher PSNR values indicate a closer match to the original image and, thus, better compression quality. Report your replicated results alongside the results reported in COIN and ensure alignment with the results in the COIN.

2.3 Task 2. Training-Time Quantization

COIN's post-hoc quantization is currently limited to higher precision (e.g., 16-bit quantization). In this task, you will explore the impact of training-time weight quantization on COIN's performance. Note that quantization is a non-differentiable operation; thus, to enable training with quantized weights, you'll need to apply a relaxation technique (analogous to relaxing a step function to a sigmoid in logistic regression). Specifically, you should follow the Quantization Networks framework [5] and replicate your baseline experiments from Task 1 using different quantization levels: 1-bit, 2-bit, 4-bit, 8-bit, and 16-bit. You will need to provide the PSNR values as a function of quantization levels. You should also investigate the effect of quantizing only weights and weights and biases.

2.4 Task 3. Low-Dimensional Reparameterization

In this Task, you will perform a neural reparameterization of an INR using Manifold Constrained Network Compression [4]. In short, assume that f_θ is a multi-layer perceptron with $\theta \in \mathbb{R}^d$. Consider optimizing this model, where $\theta = \theta_0 + \Delta\theta$, with θ_0 fixed (e.g., all zeros, random initialization, or pretrained weights). Using polar decomposition, we can express $\Delta\theta = \beta u$, where $\beta \in \mathbb{R}$ is the amplitude and $u = \frac{\Delta\theta}{\|\Delta\theta\|_2} \in \mathbb{S}^{d-1}$ represents the direction on the hypersphere $\mathbb{S}^{d-1}$. Now, consider a segment of the real line $[-L, L]$, representing a string of length $2L$, wrapped around $\mathbb{S}^{d-1}$ via a nonlinear mapping $\phi : \mathbb{R} \to \mathbb{S}^{d-1}$, parameterizing a one-dimensional manifold with $\alpha \in [-L, L]$. Instead of optimizing in d-dimensional space, we can optimize over the amplitude β and the manifold parameter α. In general one can wrap a k-dimensional space $[-L, L]^k$ around $\mathbb{S}^{d-1}$ to increase coverage. This reduces the parameter space from d to $k+1$, reparameterizing θ by α and β. Formally, let $\mathcal{L}(\cdot) : \mathbb{R}^d \to \mathbb{R}_+$ represent the loss function for a parametric learning problem. MCNC then constrain the training of the parameter $\theta \in \mathbb{R}^d$ as:

$$\theta^* = \theta_0 + \beta^* \phi(\alpha^*), \qquad (\alpha^*, \beta^*) =_{\alpha,\beta} \mathcal{L}(\theta_0 + \beta\phi(\alpha)). \tag{1}$$

To manage the reparameterization of deep models with possibly billions of parameters, we partition the parameters into smaller chunks of size d. Each chunk is reparameterized using $k+1$ parameters through the generator $\phi(.)$. This will result in a $d/(k+1)$ compression of the network.

In this task, you need to investigate the compression capability of MCNC for image compression. For ϕ, you will use a randomly initialized MLP with sinusoidal activations, such that $\phi : \mathbb{R}^k \to \mathbb{S}^{d-1}$. You will need to perform rigorous experiments on the image compression potential of the compressed model. You will need to report PSNR vs. bpp for the compressed images.

3 Project 2: Model Compression

In this project, your goal is to find compressed neural network models trained for classification tasks. We will use a MLP as the target model and solve the classification problem for MNIST. The project is organized in three phases.

3.1 Task 1. Establishing Baselines

For this task, you will train models of varying depths and widths on the MNIST dataset to evaluate the trade-off between the network's parameter count and its test accuracy. For a fixed depth (e.g., depth=3), vary the width (e.g., width=32,64,128,256,512) and train a model on MNIST. If possible, repeat the experiment 3-5 times and report the median accuracy on MNIST as a function of width. Repeat the experiment for different depths (e.g., depth=2,3,4), and report the accuracy vs. depth for each one. This will serve as our baseline for the following tasks.

3.2 Task 2. Low-Dimensional Reparameterization

Using the MCNC framework [4], reviewed in Section 2.4, you will compress a target MLP model to various compression rates. Repeat the experiments from Task 1 with MCNC, applying different compression rates. Add the MCNC results to the plots generated in Task 1, showing the impact of different compression rates (i.e., variations in d and k).

3.3 Task 3. Knowledge-Distillation + Low-Dimensional Reparameterization

Knowledge distillation, introduced by Hinton et al. [2], is a technique for transferring knowledge from a larger model (the "teacher") to a smaller model (the "student"). By applying model compression techniques, we can reduce a network's size while retaining the advantages of a larger architecture. In this project, we will investigate whether knowledge distillation can improve the performance of a compressed model. For simplicity, you should use the distillation method described by Hinton et al. [2].

Select the best-performing model from Task 1 (i.e., the model with the highest test accuracy) as the teacher model. Then, treat the MCNC models from Task 2 as the student models, and repeat the experiments from Task 2, incorporating the knowledge distillation loss. Provide a head-to-head comparison of results from Task 2 and Task 3 to evaluate the impact of knowledge distillation on model compression.

References

[1] Dupont, E., Golinski, A., Alizadeh, M., Teh, Y. W., and Doucet, A. Coin: Compression with implicit neural representations. In *Neural Compression: From Information Theory to Applications–Workshop@ ICLR 2021*.

[2] Hinton, G. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[3] Sitzmann, V., Martel, J., Bergman, A., Lindell, D., and Wetzstein, G. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems 33* (2020), 7462–7473.

[4] Thrash, C., Abbasi, A., Nooralinejad, P., Koohpayegani, S. A., Andreas, R., Pirsiavash, H., and Kolouri, S. Mcnc: Manifold constrained network compression. *arXiv preprint arXiv:2406.19301* (2024).

[5] Yang, J., Shen, X., Xing, J., Tian, X., Li, H., Deng, B., Huang, J., and Hua, X.-s. Quantization networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2019), pp. 7308–7316.