Erasure Coded Neural Network Inference via Fisher Averaging

Divyansh Jhunjhunwala*, Neharika Jali*, and Gauri Joshi
Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA, USA
Email: {djhunjhu, njali, gaurij}@andrew.cmu.edu

Shiqiang Wang
IBM Research
Yorktown Heights, NY, USA
Email: wangshiq@us.ibm.com

***Abstract*—Erasure-coded computing has been successfully used in cloud systems to reduce tail latency caused by factors such as straggling servers and heterogeneous traffic variations. A majority of cloud computing traffic now consists of inference on neural networks on shared resources where the response time of inference queries is also adversely affected by the same factors. However, current erasure coding techniques are largely focused on linear computations such as matrix-vector and matrix-matrix multiplications and hence do not work for the highly non-linear neural network functions. In this paper, we seek to design a method to code over neural networks, that is, given two or more neural network models, how to construct a coded model whose output is a linear combination of the outputs of the given neural networks. We formulate the problem as a KL barycenter problem and propose a practical algorithm `COIN` that leverages the diagonal Fisher information to create a coded model that approximately outputs the desired linear combination of outputs. We conduct experiments to perform erasure coding over neural networks trained on real-world vision datasets and show that the accuracy of the decoded outputs using `COIN` is significantly higher than other baselines while being extremely compute-efficient.**

I. Introduction

Modern machine learning (ML) jobs are deployed on large-scale cloud-based computing infrastructure. With training being a one-time event, an overwhelming majority of cloud computing traffic now constitutes ML inference jobs, in particular, inference on neural network models. Inference queries are highly time-sensitive because delays and time-outs can directly impact the quality of service to users. However, the ML models in question are often foundation models trained on diverse large-scale datasets and fine-tuned for different specific downstream tasks [1]. Due to the size and computational complexity of these models, there can be significant variations in the time taken to process inference queries. Guaranteeing low inference latency is all the more challenging because applications now host multiple neural network models on the same shared infrastructure. Issues such as resource contention in multi-tenant clusters [2], network constraints [3] or hardware unreliability [4] can result in straggling servers. The straggler problem is even worse in ensemble inference scenarios [5], where the desired inference output is a combination of the outputs of an ensemble of models, because delays in the output of any one model in the ensemble can bottleneck the entire query. Heterogeneous traffic for different models hosted on the same infrastructure can also be a major issue affecting latency. In many applications, inference queries are routed to one of the several expert models based on the features of the query [6], [7]. For example, an object detection application may use specialized models for indoor and outdoor images or day and nighttime images. In such scenarios, the query traffic for each model can vary unpredictably over time [8] and it can be negatively correlated across models, i.e., if the volume of queries to one model is high, it is low for another model.

Techniques to handle stragglers and unpredictable traffic variations include launching redundant queries [9], [10], also referred to as speculative execution and replication of models to meet the highest possible traffic demand. These lead to inefficient execution of queries and idling of resources respectively. Erasure coding, which is a generalization of replication, is an effective solution for straggler mitigation in matrix computations [11]–[13] and handling heterogeneous traffic [14]–[16]. As an illustrative example, we look at a linear inference task of matrix computation consisting of models $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_N$. Consider an ensemble inference query, represented by vector $\boldsymbol{x}$, that requires outputs of all the N models and can be bottlenecked by straggling of any one of the outputs. If we create coded linear combinations of the N models, they can allow us to recover the N outputs even when one of the uncoded models slows down. To see the benefit of coding in handling heterogeneous traffic, consider an application scenario that routes the query $\boldsymbol{x}$ to one of the models $\mathbf{A}_1, \mathbf{A}_2, \ldots \mathbf{A}_N$, and the output required to be computed is $\mathbf{A}_i\boldsymbol{x}$. If the server storing $\mathbf{A}_i$ is congested or fails, the query routed to two servers one with $\mathbf{A}_j$ and one with coded $\mathbf{A}_i + \mathbf{A}_j$ and the outputs can be combined to recover the desired output $\mathbf{A}_i\boldsymbol{x}$. The coded server thus acts as a flexible model that can effectively be used to serve queries of both types or enable retrieval if one of the servers is slow.

Main Contributions.

A key missing element in previous works is that erasure coding is inherently linear and does not work for non-linear functions. Thus, it cannot be directly used for neural network inference. In this work, we consider the question of how to erasure code neural networks. In Section III, we define the

*Denotes equal contribution

coding objective that seeks to construct a coded network whose output is a linear combination of two or more neural networks. In Section IV we reduce our coding objective to the equivalent KL barycenter problem and propose a practical solution COIN that approximately produces the desired linear combination of outputs. Finally in Section V, we measure the accuracy of the decoded outputs using COIN on neural networks trained on real-world vision datasets and highlight its improvement over several competing baselines. To the best of our knowledge, COIN is one of the first works to do erasure coding for non-linear neural network functions in a compute-efficient manner.

II. Related Work

Erasure codes have been extensively studied and applied to distributed storage and computing. Effective solutions for both straggler mitigation and latency reduction use principles of replication and erasure codes [11]–[13]. Other works including [17]–[20] use erasure codes for straggler-resistant computation for convex optimization and gradient descent. In [14]–[16], the authors proposed the idea of using coded servers to handle variations in skewed traffic for efficient reduction of the response time of queries. However, the codes in the above works are for linear function computations and do not work in general for non-linear functions like neural networks.

There is very little work on employing erasure codes for non-linear functions and neural networks, some of which include the following relevant work. [21] decomposes non-linear functions into inexpensive linear functions and proposes rateless sum-recovery codes to alleviate the problem of stragglers in distributed non-linear computations. For inference on an ensemble of neural networks, [22] proposes learning a 'parity' network that is trained to transform erasure-coded queries into a form that enables a decoder to reconstruct slow or failed predictions. A major drawback of this, however, is that it requires training the parity model from scratch, which is expensive in both compute and data requirements. Another orthogonal line of work aims to learn the encoder and decoder neural networks that enable erasure coding in communication over noisy channels [23], [24]. The goal of this line of work is different and complementary to ours – they construct codes using deep neural networks while we are using codes to improve the reliability and latency performance of neural network inference.

A recent relevant line of work in ML has investigated the problem of 'model fusion', i.e., combining the weights of two or more independent neural networks into a single network that broadly speaking inherits the properties of the fused networks [25]–[28]. In this aspect, model merging is closer to multi-task learning [29] where the goal is to learn a single model that can perform well on all tasks. Our goal on the other hand is strictly to produce a model whose output is a linear combination of the given neural networks; we do not care about the performance of the coded model on individual tasks. The closest work to ours is [28] which also proposes to use the diagonal Fisher when merging models. Nonetheless, we believe our motivation for using the Fisher information for erasure coding is novel as discussed in Section III along with our experiments in Section V which show that our proposed approach significantly improves decoding accuracy compared to approaches which are adopted from model merging literature. We discuss other related works on model merging in Section V.

III. Problem Formulation and Preliminaries

In the rest of the paper, we use lowercase bold letters, e.g. $\boldsymbol{x}$, to denote vectors and use x_i to denote the i-th element of the vector $\boldsymbol{x}$. We use $\|\cdot\|_2$ to denote the L_2 norm, $\mathbb{R}$ to denote the set of real numbers and $[N]$ to denote the set of numbers $\{1, 2, \ldots, N\}$. Probability density functions are represented by $p(\boldsymbol{x})$. We use $\mathrm{KL}(p(\boldsymbol{x})||q(\boldsymbol{x}))$ to denote the KL divergence between two densities $p(\boldsymbol{x})$ and $q(\boldsymbol{x})$.

Multi-Model Inference Setup. We consider a multi-model inference setup with N neural networks denoted by $f_{\boldsymbol{\theta}_1}(\boldsymbol{x}), f_{\boldsymbol{\theta}_2}(\boldsymbol{x}), \ldots, f_{\boldsymbol{\theta}_N}(\boldsymbol{x})$ where $\boldsymbol{\theta}_i \in \mathbb{R}^d$ parameterizes the weights of the i-th neural network. Each neural network takes an input $\boldsymbol{x} \in \mathbb{R}^s$ (e.g., an image) and produces an output $\boldsymbol{y} \in \mathbb{R}^K$ (e.g., image label). To simplify our discussion and also allow comparison with other baseline methods in Section V, we assume that these neural networks have the same architecture.

Erasure Coding Objective.

We consider a class of erasure codes called systematic maximum distance separable (MDS) codes [30], [31] that take N source symbols (N neural networks in our case) and produce N_c coded symbols such that using the N_c coded symbols and any subset of $(N - N_c)$ source symbols, we can recover the other N_c source symbols. In this paper, we focus on the $N_c = 1$ case and leave extensions to general k as future work. That is, given N neural networks $f_{\boldsymbol{\theta}_1}(\boldsymbol{x}), f_{\boldsymbol{\theta}_2}(\boldsymbol{x}), \ldots, f_{\boldsymbol{\theta}_N}(\boldsymbol{x})$, our goal is to produce a coded neural network $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ that can be used to recover the output of $f_{\boldsymbol{\theta}_i}(\boldsymbol{x})$ for any i using the output of the remaining $(N-1)$ neural networks. To do so, we want to express $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ as a convex combination of $f_{\boldsymbol{\theta}_i}(\boldsymbol{x})$'s:

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) \approx \textstyle\sum_{i=1}^{N} \beta_i f_{\boldsymbol{\theta}_i}(\boldsymbol{x}) \tag{1}$$

where $\beta_i > 0, \sum_{i=1}^{N} \beta_i = 1$ are the coding weights.

Given such a coded neural network $f_{\boldsymbol{\theta}}(\boldsymbol{x})$, it is easy to approximately recover $f_{\boldsymbol{\theta}_i}(\boldsymbol{x})$ using the other $N-1$ networks $\{f_{\boldsymbol{\theta}_j}(\boldsymbol{x})\}_{j=1, j\neq i}^{N}$ as follows,

$$\hat{f}_{\boldsymbol{\theta},i}(\boldsymbol{x}) = \frac{1}{\beta_i}\left(f_{\boldsymbol{\theta}}(\boldsymbol{x}) - \sum_{j=1, j\neq i}^{N} \beta_j f_{\boldsymbol{\theta}_j}(\boldsymbol{x})\right) \approx f_{\boldsymbol{\theta}_i}(\boldsymbol{x}). \tag{2}$$

The quality of the decoded approximation can be measured by the mismatch between $f_{\boldsymbol{\theta}_i}(\boldsymbol{x})$ and $\hat{f}_{\boldsymbol{\theta},i}(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathbb{R}^s$ and $i \in [N]$, which we define using the squared loss function:

$$L(\boldsymbol{\theta}) = \frac{1}{2N}\mathbb{E}_{\boldsymbol{x}}\left[\sum_{i=1}^{N}\left\|f_{\boldsymbol{\theta}_i}(\boldsymbol{x}) - \hat{f}_{\boldsymbol{\theta},i}(\boldsymbol{x})\right\|_2^2\right] \tag{3}$$

$$= \frac{\bar{\beta}}{2}\mathbb{E}_{\boldsymbol{x}}\left[\left\|f_{\boldsymbol{\theta}}(\boldsymbol{x}) - \sum_{i=1}^{N}\beta_i f_{\boldsymbol{\theta}_i}(\boldsymbol{x})\right\|_2^2\right], \tag{4}$$

where the last equality follows from substituting Equation (2) in Equation (3) and defining $\bar{\beta} = (\sum_{i=1}^{N} 1/\beta_i^2)/N$. Since the distribution $q(\boldsymbol{x})$ over $\boldsymbol{x}$ is typically unknown, we only assume access to P samples $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_P$ drawn from $q(\boldsymbol{x})$.

Given these P samples and the neural networks $f_{\boldsymbol{\theta}_1}(\boldsymbol{x}), f_{\boldsymbol{\theta}_2}(\boldsymbol{x}), \cdots, f_{\boldsymbol{\theta}_n}(\boldsymbol{x})$, we can define the following empirical coding loss

$$\hat{L}(\boldsymbol{\theta}) = \frac{\bar{\beta}}{2P} \sum_{l=1}^{P} \left\| f_{\boldsymbol{\theta}}(\boldsymbol{x}_l) - \sum_{i=1}^{N} \beta_i f_{\boldsymbol{\theta}_i}(\boldsymbol{x}_l) \right\|_2^2. \tag{5}$$

We now discuss a baseline approach to minimize the empirical objective, followed by our proposed approach in Section IV.

Ensemble Distillation Baseline.

From Equation (5), we see that we want the output of our coded neural network $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ to match the output of the 'ensemble' of neural networks given by $\sum_{i=1}^{N} \beta_i f_{\boldsymbol{\theta}_i}(\boldsymbol{x}_l)$. This idea has been well studied in the context of ensemble distillation [32]–[34] where the goal is to distill the knowledge from an ensemble of models or 'teachers' into a single model or 'student'.

Treating the output $\sum_{i=1}^{N} \beta_i f_{\boldsymbol{\theta}_i}(\boldsymbol{x})$ as a pseudo-label $\hat{\boldsymbol{y}}_l$ for every $l \in [P]$, we see that our objective becomes exactly the same as squared loss regression and can be optimized with standard gradient-based techniques. However, there are some drawbacks to this approach. Firstly performing such a gradient based optimization step imposes a significant computation cost. Secondly, it is not easy to modify the coded network to account for changes in the coding weights β_i's or add a new neural network $f_{\boldsymbol{\theta}_{N+1}}(\boldsymbol{x})$ to our coding setup. We would need to re-train the coded network in such cases. Lastly, in the case where the number of samples P is small, there is a serious risk of overfitting. We demonstrate this in our experiments where we show that the coded network obtained via ensemble distillation generalizes poorly for samples outside of the training set. Standard regularization techniques such as early stopping and weight decay are also unable to help with the overfitting as we show in the Appendix.

IV. COIN: CODED INFERENCE OF NEURAL NETWORKS

In this section we show how the problem of minimizing the objective in Equation (3) can be reformulated to get an equivalent problem known as the KL barycenter problem [35]. Next we discuss how to get an approximate solution to the KL barycenter problem in our setup and how we can practically implement this solution.

Neural Network as a Statistical Model. To motivate our proposed solution, we use the idea of a neural network as a parameterized statistical model that defines a probability density function $p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y})$ over all input-label pairs $(\boldsymbol{x}, \boldsymbol{y})$ in $\mathbb{R}^{s \times K}$. In particular, we define $p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y}) = q(\boldsymbol{x}) p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}) = q(\boldsymbol{x}) \exp\left(-\|\boldsymbol{y} - f_{\boldsymbol{\theta}}(\boldsymbol{x})\|_2^2\right)/\sqrt{2\pi}$ where $\boldsymbol{x} \sim q(\boldsymbol{x})$ is the input distribution over $\mathbb{R}^s$, which is independent of parameters $\boldsymbol{\theta}$. This is a standard idea in statistical learning that draws an equivalence between minimizing the squared loss and maximizing the log likelihood of the observed data under a Gaussian model since $-\log p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y}) = \|f_{\boldsymbol{\theta}}(\boldsymbol{x}) - \boldsymbol{y}\|_2^2 + c$, where c is some constant which does not depend on $\boldsymbol{\theta}$.

Reduction to KL Barycenter Problem. Expanding the norm in Equation (4) and since $\sum_{i=1}^{N} \beta_i = 1$, we get

$$L(\boldsymbol{\theta}) = \underbrace{\frac{\bar{\beta}}{2} \sum_{i=1}^{N} \bar{\beta}_i \mathbb{E}_{\boldsymbol{x}} \left[\|f_{\boldsymbol{\theta}_i}(\boldsymbol{x}) - f_{\boldsymbol{\theta}}(\boldsymbol{x})\|_2^2 \right]}_{L_1(\boldsymbol{\theta})} - \frac{\bar{\beta}}{2} \sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} \beta_i \beta_j \mathbb{E}_{\boldsymbol{x}} \left[\left\| f_{\boldsymbol{\theta}_i}(\boldsymbol{x}) - f_{\boldsymbol{\theta}_j}(\boldsymbol{x}) \right\|_2^2 \right] \tag{6}$$

where $\bar{\beta}_i = \beta_i \sum_{j=1}^{N} \beta_j$. Since the second term in Equation (6) does not depend on the coded model's parameters $\boldsymbol{\theta}$, we focus on minimizing just the first term $L_1(\boldsymbol{\theta})$:

$$L_1(\boldsymbol{\theta}) = \frac{1}{2} \sum_{i=1}^{N} \bar{\beta}_i \mathbb{E}_{\boldsymbol{x}} \left[\|f_{\boldsymbol{\theta}_i}(\boldsymbol{x}) - f_{\boldsymbol{\theta}}(\boldsymbol{x})\|_2^2 \right] \tag{7}$$

$$= \sum_{i=1}^{N} \bar{\beta}_i \mathbb{E}_{\boldsymbol{x}} \left[\mathrm{KL}(p_{\boldsymbol{\theta}_i}(\boldsymbol{y}|\boldsymbol{x}) || p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})) \right] \tag{8}$$

$$= \sum_{i=1}^{N} \bar{\beta}_i \mathrm{KL}(p_{\boldsymbol{\theta}_i}(\boldsymbol{x}, \boldsymbol{y}) || p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y})) \tag{9}$$

where Equation (8) follows from our definition of $p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})$ above and uses the fact that $\mathrm{KL}(\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}) || \mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma})) = \|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\|_2^2/2$. Thus we see that minimizing $L_1(\boldsymbol{\theta})$ is equivalent to finding the density function $p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y})$ that is a *weighted average* (in the KL divergence sense) of the density functions $p_{\boldsymbol{\theta}_i}(\boldsymbol{x}, \boldsymbol{y})$ with weights proportional to $\bar{\beta}_i$. This is known as the KL barycenter problem and has been studied in previous work in the context of clustering [36] and model-fusion [35].

Solving the KL Barycenter Problem. In the case where $p_{\boldsymbol{\theta}_i}(\boldsymbol{x}, \boldsymbol{y})$ belongs to the exponential family of distributions with natural parameter $\boldsymbol{\theta}$, it is known that there exists an analytical expression for the parameters $\boldsymbol{\theta}$ of the distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y})$ that minimizes Equation (9) [35]. However, this is not the case in our setup because $p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})$ in Equation (8) is Gaussian with respect to $f_{\boldsymbol{\theta}}(\cdot)$ and not $\boldsymbol{\theta}$ itself. Thus, we need to resort to some approximations to get a analytical solution. We use the following approximation for the KL divergence between $p_{\boldsymbol{\theta}_i}(\boldsymbol{x}, \boldsymbol{y})$ and $p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y})$ [37],

$$\mathrm{KL}(p_{\boldsymbol{\theta}_i}(\boldsymbol{x}, \boldsymbol{y}) || p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y})) \approx (\boldsymbol{\theta} - \boldsymbol{\theta}_i)^\top F_{\boldsymbol{\theta}_i} (\boldsymbol{\theta} - \boldsymbol{\theta}_i) \tag{10}$$

where $F_{\boldsymbol{\theta}_i}$ is the Fisher information matrix of $\boldsymbol{\theta}_i$ defined as follows

$$F_{\boldsymbol{\theta}_i} = \mathbb{E}_{\boldsymbol{x}} \left[\mathbb{E}_{\boldsymbol{y} \sim p_{\boldsymbol{\theta}}(\cdot|\boldsymbol{x})} \left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y}) \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y})^\top \right] \right]_{\boldsymbol{\theta}=\boldsymbol{\theta}_i} \tag{11}$$

$$= \mathbb{E}_{\boldsymbol{x}} \left[\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\boldsymbol{x}) \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\boldsymbol{x})^\top \right]_{\boldsymbol{\theta}=\boldsymbol{\theta}_i} \tag{12}$$

This approximation comes from treating $\mathrm{KL}(p_{\boldsymbol{\theta}_i}(\boldsymbol{x}, \boldsymbol{y}) || p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y}))$ as a function of $\boldsymbol{\theta}$ and taking a second order Taylor expansion around $\boldsymbol{\theta}_i$ (zeroth and first order terms are zero). As is the case with Taylor expansions, the quality of the approximation degrades as the distance

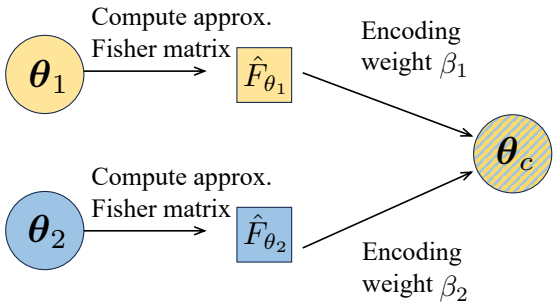

Fig. 1: Illustration of how our proposed method COIN (see Algorithm 1) computes the coded model's parameters $\boldsymbol{\theta}_c$ such that its output $f_{\theta_c}(\boldsymbol{x}) \approx \beta_1 f_{\theta_1}(\boldsymbol{x}) + \beta_2 f_{\theta_2}(\boldsymbol{x})$, a linear combination of the outputs of $f_{\theta_1}(\boldsymbol{x})$ and $f_{\theta_2}(\boldsymbol{x})$. Unlike ensemble distillation, the parameters $\boldsymbol{\theta}_c$ are computed without requiring training the model from scratch.

Algorithm 1 COIN

Input: neural networks $f_{\boldsymbol{\theta}_1}(\boldsymbol{x}), f_{\boldsymbol{\theta}_2}(\boldsymbol{x}), \ldots, f_{\boldsymbol{\theta}_N}(\boldsymbol{x})$ to be coded, coding weights $\beta_1, \beta_2, \ldots, \beta_N$, input data samples $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_P$, penalty parameter λ

For $i \in [N]$ **do:**

  Compute $\bar{\beta}_i = \beta_i \sum_{j=1}^{N} \beta_j$

  Compute $\hat{F}_{\boldsymbol{\theta}_i} = \text{diag}\left(\frac{1}{P}\sum_{l=1}^{P}[\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\boldsymbol{x}_l)\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\boldsymbol{x}_l)^\top]_{\boldsymbol{\theta}=\boldsymbol{\theta}_i}\right)$

Return $\boldsymbol{\theta}_c = (\sum_{i=1}^{N} \bar{\beta}_i(\hat{F}_{\boldsymbol{\theta}_i} + \lambda I))^{-1} \sum_{i=1}^{N} \bar{\beta}_i(\hat{F}_{\boldsymbol{\theta}_i} + \lambda I)\boldsymbol{\theta}_i$.

$\|\boldsymbol{\theta} - \boldsymbol{\theta}_i\|_2^2$ increases. To capture this we also propose to add a penalty term $\lambda \|\boldsymbol{\theta} - \boldsymbol{\theta}_i\|_2^2, \lambda > 0$ to this approximation. With this, our new objective $G(\boldsymbol{\theta})$ is given by,

$$L_1(\boldsymbol{\theta}) \approx G(\boldsymbol{\theta}) \tag{13}$$

$$= \sum_{i=1}^{N} \bar{\beta}_i(\boldsymbol{\theta} - \boldsymbol{\theta}_i)^\top F_{\boldsymbol{\theta}_i}(\boldsymbol{\theta} - \boldsymbol{\theta}_i) + \lambda \sum_{i=1}^{N} \bar{\beta}_i \|\boldsymbol{\theta} - \boldsymbol{\theta}_i\|_2^2 \tag{14}$$

We see that $G(\boldsymbol{\theta})$ is a strongly convex function whose global minimizer is given by,

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^d} G(\boldsymbol{\theta})$$

$$= \left[\sum_{i=1}^{N} \bar{\beta}_i(F_{\boldsymbol{\theta}_i} + \lambda I)\right]^{-1} \sum_{i=1}^{N} \bar{\beta}_i(F_{\boldsymbol{\theta}_i} + \lambda I)\boldsymbol{\theta}_i. \tag{15}$$

Thus given the parameters of our uncoded networks $\boldsymbol{\theta}_i$, Equation (15) outlines how in theory, we can compute $\boldsymbol{\theta}^*$ such that $f_{\boldsymbol{\theta}^*}(\boldsymbol{x}) \approx \sum_{i=1}^{N} \beta_i f_{\boldsymbol{\theta}_i}(\boldsymbol{x})$.

Practical Solver. In practice, computing the exact Fisher $F_{\boldsymbol{\theta}_i}$ for all $i \in [N]$ in Equation (15) is challenging since it involves $\mathcal{O}(d^2)$ operations, with d being in the order of millions for neural networks. Also recall that we only have access to P samples $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_P$ from the distribution $q(\boldsymbol{x})$. Thus in order to get a fast and tractable solution, we propose to approximate the true Fisher $F_{\boldsymbol{\theta}_i}$ with the diagonal of the empirical Fisher as done in several other works dealing with computing the Fisher [38], [39]. In other words we have,

$$F_{\boldsymbol{\theta}_i} \approx \hat{F}_{\boldsymbol{\theta}_i} = \text{diag}\left(\frac{1}{P}\sum_{l=1}^{P}[\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\boldsymbol{x}_l)\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\boldsymbol{x}_l)^\top]_{\boldsymbol{\theta}=\boldsymbol{\theta}_i}\right). \tag{16}$$

With this approximation, the parameters of our coded model $\boldsymbol{\theta}_c$ are given by,

$$\boldsymbol{\theta}_c = (\sum_{i=1}^{N} \bar{\beta}_i(\hat{F}_{\boldsymbol{\theta}_i} + \lambda I))^{-1} \sum_{i=1}^{N} \bar{\beta}_i(\hat{F}_{\boldsymbol{\theta}_i} + \lambda I)\boldsymbol{\theta}_i. \tag{17}$$

We find that using the diagonal Fisher is sufficient to provide a consistent improvement over other coding baselines including ensemble distillation as shown in Section V. Furthermore it also overcomes other limitations of the ensemble distillation baseline - it is simple and cheap to compute, effectively taking only $O(d)$ operations and can be modified easily to incorporate changes in the coding weights β_i or $\boldsymbol{\theta}_i$ without needing an expensive retraining step. It has also been shown that the empirical approximation of the Fisher is sample-efficient [40]; we only choose P to be about 200 to get a good approximation to the true Fisher in our experiments.

V. Experiments

In this section we demonstrate the effectiveness of COIN for erasure coding on neural networks trained on real-world vision datasets. To do so, we first introduce the metric that we use in our experiments.

Normalized Decoding Accuracy. Recall in Section III we use $f_{\boldsymbol{\theta}_i}(\boldsymbol{x})$ to denote the outputs of the i-th neural network and $\hat{f}_{\boldsymbol{\theta},i}(\boldsymbol{x})$ to denote the decoded outputs for network i for some given coded model $\boldsymbol{\theta}$ (see Equation (2)). Let $\mathcal{S}_i^{\text{test}}$ be the test data associated with neural network i. We define the *Normalized Decoding Accuracy* (NDA) for the i-th network as follows:

$$100 \times \frac{\sum_{(\boldsymbol{x},\boldsymbol{y}) \in \mathcal{S}_i^{\text{test}}} \mathbb{I}\left\{\arg\max\left(\hat{f}_{\boldsymbol{\theta},i}(\boldsymbol{x})\right) = y\right\}}{\sum_{(\boldsymbol{x},\boldsymbol{y}) \in \mathcal{S}_i^{\text{test}}} \mathbb{I}\left\{\arg\max\left(f_{\boldsymbol{\theta}_i}(\boldsymbol{x})\right) = y\right\}} \tag{18}$$

where $\mathbb{I}\{\cdot\}$ is the indicator function. We see that the numerator of Equation (18) measures the accuracy of the decoded outputs while the denominator measures the accuracy of the original network $f_{\boldsymbol{\theta}_i}(\cdot)$. For linear models since $\hat{f}_{\boldsymbol{\theta},i}(\boldsymbol{x}) = f_{\boldsymbol{\theta}_i}(\boldsymbol{x})$, this ratio would always be 100; however for non-linear networks, due to the approximations introduced in the coding step, i.e., $\hat{f}_{\boldsymbol{\theta},i}(\boldsymbol{x}) \approx f_{\boldsymbol{\theta}_i}(\boldsymbol{x})$, this ratio is usually less than 100. Thus, $(100 - \text{NDA})$ gives us a measure of the error introduced by the approximate erasure coding over non-linear models.

Baselines. We compare COIN with 4 other baselines including 3 which are adopted from the model merging literature. Vanilla Averaging [25] is the first and most common baseline in model merging literature where the merged/coded model is constructed by a simple weighted average of the parameter vectors of the individual models, i.e., $\boldsymbol{\theta} = \sum_{i=1}^{n} \beta_i \boldsymbol{\theta}_i$. Next, we compared with Task Arithmetic [26], where the coded

TABLE I: Normalized Decoding Accuracy results when coding over experts trained on different partitions of the same dataset. COIN shows a significant improvement in performance compared to baselines while being compute-efficient.

	MNIST			FashionMNIST			CIFAR10		
Algorithm	Split 1	Split 2	Avg.	Split 1	Split 2	Avg.	Split 1	Split 2	Avg.
Vanilla Averaging [25]	95.61	83.52	89.56	98.75	92.21	95.48	94.28	86.09	90.19
Task Arithmetic [26]	96.82	83.56	90.19	98.73	92.21	95.47	95.19	86.70	90.94
RegMean [27]	95.36	83.91	89.63	96.50	89.37	92.93	91.05	85.05	88.05
Ensemble Distillation	97.19	97.06	97.12	92.98	84.63	88.80	85.33	90.86	88.09
COIN(ours)	98.72	97.56	**98.14**	97.68	97.39	**97.54**	97.63	98.30	**97.96**

TABLE II: Normalized Decoding Accuracy results when coding over experts trained on different datasets. COIN shows a significant improvement in performance compared to baselines while being compute-efficient.

	MNIST + FashionMNIST			CIFAR10 + FashionMNIST			CIFAR10 + MNIST		
Algorithm	MNIST	FashionMNIST	Avg.	CIFAR10	FashionMNIST	Avg.	CIFAR10	MNIST	Avg.
Vanilla Averaging [25]	42.33	73.27	57.80	66.65	80.12	73.39	86.08	68.84	77.46
Task Arithmetic [26]	52.96	80.99	66.98	76.83	84.25	80.54	86.08	68.84	77.46
RegMean [25]	73.18	78.83	76.00	87.01	86.62	86.81	83.20	69.40	76.30
Ensemble Distillation	82.63	75.27	78.95	62.44	70.38	66.41	65.56	87.29	76.42
COIN(ours)	80.36	83.99	**82.17**	89.12	85.86	**87.49**	92.89	84.34	**88.62**

model is constructed as $\boldsymbol{\theta} = \boldsymbol{\theta}_0 + \alpha \sum_{i=1}^{n} (\boldsymbol{\theta}_i - \boldsymbol{\theta}_0)$ with $\boldsymbol{\theta}_0$ being our base foundation model and α being a hyperparameter which is tuned using validation data. RegMean [27] is a recently proposed state-of-the-art model fusion method which uses the Gram matrices of the data for model fusion. Lastly, we also compare with the Ensemble Distillation baseline, as outlined in Section III.

Experimental Setup. We use a ResNet50 pretrained on ImageNet as our foundation model. The datasets we consider are MNIST, FashionMNIST and CIFAR10, all of which consist of 10 classes, i.e., $K = 10$. In all experiments, we set the number of coded models $n = 2$ and coding coefficients to be $\beta_1 = 0.5$ and $\beta_2 = 0.5$ for simplicity. Now to simulate $n = 2$ experts, each of which specializes in a particular type of query, we consider the following two settings. In the first case we consider experts that are trained on different partitions of the same dataset. We split the given dataset into two partitions $\mathcal{S}_1$ and $\mathcal{S}_2$ where $\mathcal{S}_1$ consists of all the data corresponding to labels $\{1, 2, \ldots, 5\}$ and $\mathcal{S}_2$ consists of the data corresponding to labels $\{6, 7 \ldots, 10\}$ and fine-tune a neural network on each partition. In the second case, we consider experts that are fine-tuned on different datasets itself, for e.g., where $\mathcal{S}_1$ is the CIFAR-10 dataset and $\mathcal{S}_2$ is the MNIST dataset. For algorithms which require access to data to create the coded model (RegMean, Ensemble Distillation, COIN), we sample $P' = 100$ datapoints from both $\mathcal{S}_1$ and $\mathcal{S}_2$ giving us $P = 200$ datapoints in total, which is less than 1% of the total data in $\mathcal{S}_1$ and $\mathcal{S}_2$. Additional details and an ablation study evaluating the effect of P on the normalized decoding accuracy can be found in the Appendix.

Discussion. Table I shows the normalized decoding accuracy results when coding over experts trained on different partitions of the same dataset while Table II shows the results of coding over experts trained on different datasets for different combinations of datasets. In all cases we see that COIN achieves the highest average normalized decoding accuracy while avoiding any expensive computational procedures such as distillation (Ensemble Distillation) or computing the Gram matrix of data (RegMean). Specifically for Table I we see that COIN is the only algorithm which consistently achieves greater than 97.5% average normalized decoding accuracy which implies that there is a less than 2.5% loss in accuracy compared to the individual models. In Table II we see that there is a larger drop in accuracy when coding over models trained on different datasets which can be attributed to the greater data heterogeneity used to fine-tune the respective models. Nonetheless, COIN continues to outperform baselines with almost 10% in some cases like CIFAR10+MNIST.

VI. Concluding Remarks

In this paper, we propose COIN, an algorithm that leverages erasure coding for multi-model neural network inference using an equivalence with the KL barycenter problem in its design. Our solution is both efficient in resource utilization (needs less than 1% of training data) and avoids any expensive computational procedures such as ensemble distillation. We demonstrate via experiments over that our method significantly improves decoding accuracy compared to baselines when coding over neural networks trained on real-world vision datasets in various settings. Directions for future work include characterizing the performance on a wider range of model architectures such as transformers and coding over a larger set of models.

Acknowledgment

This work was supported in part by NSF CCF 2045694, CNS-2112471, CPS-2111751, and ONR N00014-23-1-2149, and the CMU Benjamin Garver Lamme/Westinghouse Fellowship. The authors would also like to thank Tuhinangshu Choudhury for insightful discussions.

REFERENCES

[1] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill *et al.*, "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021.

[2] Y. Xu, Z. Musgrave, B. Noble, and M. Bailey, "Bobtail: Avoiding long tails in the cloud," in *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*. Lombard, IL: USENIX Association, Apr. 2013, pp. 329–341. [Online]. Available: https://www.usenix.org/conference/nsdi13/technical-sessions/presentation/xu_yunjing

[3] D. Crankshaw, X. Wang, G. Zhou, M. J. Franklin, J. E. Gonzalez, and I. Stoica, "Clipper: A Low-Latency online prediction serving system," in *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*. Boston, MA: USENIX Association, Mar. 2017, pp. 613–627. [Online]. Available: https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/crankshaw

[4] G. Ananthanarayanan, S. Kandula, A. Greenberg, I. Stoica, Y. Lu, B. Saha, and E. Harris, "Reining in the outliers in map-reduce clusters using mantri," in *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'10. USA: USENIX Association, 2010, p. 265–278.

[5] O. Sagi and L. Rokach, "Ensemble learning: A survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1249, 2018.

[6] M. I. Jordan and R. A. Jacobs, "Hierarchical mixtures of experts and the em algorithm," *Neural Computation*, vol. 6, no. 2, p. 181–214, mar 1994.

[7] W. Fedus, J. Dean, and B. Zoph, "A review of sparse expert models in deep learning," *arXiv*, 2022. [Online]. Available: https://arxiv.org/abs/2209.01667

[8] Google, "What are we searching for? https://trends.google.com/trends/,."

[9] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *ACM Commun. Mag.*, vol. 51, no. 1, pp. 107–113, Jan. 2008.

[10] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: cluster computing with working sets," in *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, vol. 10, 2010, p. 10.

[11] G. Ananthanarayanan, A. Ghodsi, S. Shenker, and I. Stoica, "Effective straggler mitigation: Attack of the clones," in *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, Apr. 2013, pp. 185–198.

[12] G. Joshi, "Synergy via redundancy: Boosting service capacity with adaptive replication," *SIGMETRICS Performance Evaluation Review*, vol. 45, no. 3, pp. 21–28, Mar. 2018. [Online]. Available: http://doi.acm.org/10.1145/3199524.3199530

[13] A. Mallick, U. Sheth, G. Palanikumar, M. Chaudhari, and G. Joshi, "Rateless Codes for Near-Perfect Load Balancing in Distributed Matrix-Vector Multiplication," in *Proceedings of ACM Sigmetrics 2020*, May 2020. [Online]. Available: "https://arxiv.org/abs/1804.10331"

[14] M. Aktas, S. E. Anderson, A. Johnston, G. Joshi, S. Kadhe, G. L. Matthews, C. Mayer, and E. Soljanin, "On the service capacity of accessing erasure coded content," in *Proc. Allerton Conf. Commun., Control and Computing*, Oct. 2017.

[15] S. E. Anderson, A. Johnston, G. Joshi, G. L. Matthews, C. Mayer, and E. Soljanin, "Service rate region of content access from erasure coded storage," in *Proc. Allerton Conf. Commun., Control and Computing*, Oct. 2017.

[16] T. Choudhury, W. Wang, and G. Joshi, "Tackling heterogeneous traffic in multi-access systems via erasure coded servers," ser. MobiHoc '22. Association for Computing Machinery, 2022, p. 171–180.

[17] C. Karakus, Y. Sun, and S. Diggavi, "Encoded distributed optimization," in *IEEE International Symposium on Information Theory (ISIT)*, June 2017, pp. 2890–2894.

[18] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 70, 2017, pp. 3368–3376.

[19] W. Halbawi, N. Azizan, F. Salehi, and B. Hassibi, "Improving distributed gradient descent using reed-solomon codes," in *2018 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2018, pp. 2027–2031.

[20] S. Dutta, Z. Bai, H. Jeong, T. M. Low, and P. Grover, "A unified coded deep neural network training strategy based on generalized polydot codes," in *2018 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2018, pp. 1585–1589.

[21] A. Mallick and G. Joshi, "Rateless sum-recovery codes for distributed non-linear computations," in *2022 IEEE Information Theory Workshop (ITW)*. IEEE, 2022, pp. 160–165.

[22] J. Kosaian, K. V. Rashmi, and S. Venkataraman, "Parity models: A general framework for coding-based resilience in ML inference," *CoRR*, vol. abs/1905.00863, 2019. [Online]. Available: http://arxiv.org/abs/1905.00863

[23] H. Kim, Y. Jiang, R. B. Rana, S. Kannan, S. Oh, and P. Viswanath, "Communication algorithms via deep learning," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=ryazCMbR-

[24] H. Kim, Y. Jiang, S. Kannan, S. Oh, and P. Viswanath, "Deepcode: Feedback codes via deep learning," *Advances in neural information processing systems*, vol. 31, 2018.

[25] M. Wortsman, G. Ilharco, S. Y. Gadre, R. Roelofs, R. Gontijo-Lopes, A. S. Morcos, H. Namkoong, A. Farhadi, Y. Carmon, S. Kornblith *et al.*, "Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time," in *International Conference on Machine Learning*. PMLR, 2022, pp. 23 965–23 998.

[26] G. Ilharco, M. T. Ribeiro, M. Wortsman, L. Schmidt, H. Hajishirzi, and A. Farhadi, "Editing models with task arithmetic," in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. [Online]. Available: https://openreview.net/pdf?id=6t0Kwf8-jrj

[27] X. Jin, X. Ren, D. Preotiuc-Pietro, and P. Cheng, "Dataless knowledge fusion by merging weights of language models," in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. [Online]. Available: https://openreview.net/pdf?id=FCnohuR6AnM

[28] M. S. Matena and C. A. Raffel, "Merging models with fisher-weighted averaging," *Advances in Neural Information Processing Systems*, vol. 35, pp. 17 703–17 716, 2022.

[29] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, pp. 41–75, 1997.

[30] E. Berkelamp, *Algebraic coding theory*. New York, USA: McGraw-Hill, 1968.

[31] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. USA: Wiley Publishers, 2006.

[32] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *ArXiv*, Mar. 2015. [Online]. Available: https://arxiv.org/abs/1503.02531

[33] T. Lin, L. Kong, S. U. Stich, and M. Jaggi, "Ensemble distillation for robust model fusion in federated learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 2351–2363, 2020.

[34] M. Freitag, Y. Al-Onaizan, and B. Sankaran, "Ensemble distillation for neural machine translation," *arXiv preprint arXiv:1702.01802*, 2017.

[35] S. Claici, M. Yurochkin, S. Ghosh, and J. Solomon, "Model fusion with kullback-leibler divergence," in *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, ser. Proceedings of Machine Learning Research, vol. 119. PMLR, 2020, pp. 2038–2047. [Online]. Available: http://proceedings.mlr.press/v119/claici20a.html

[36] A. Banerjee, I. S. Dhillon, J. Ghosh, S. Sra, and G. Ridgeway, "Clustering on the unit hypersphere using von mises-fisher distributions." *Journal of Machine Learning Research*, vol. 6, no. 9, 2005.

[37] J. Martens, "New insights and perspectives on the natural gradient method," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 5776–5851, 2020.

[38] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[39] Y. LeCun, J. Denker, and S. Solla, "Optimal brain damage," *Advances in neural information processing systems*, vol. 2, 1989.

[40] S. P. Singh and D. Alistarh, "Woodfisher: Efficient second-order approximation for neural network compression," *Advances in Neural Information Processing Systems*, vol. 33, pp. 18 098–18 109, 2020.

Appendix

VII. Additional Experimental Details

We use PyTorch to run all our experiments. For fine-tuning we use the AdamW optimizer with a learning rate of 10^{-5}, batch size of 128 and weight decay 0.1. For MNIST dataset we fine-tune for 1 epoch, for FashionMNIST we fine-tune for 5 epochs and for CIFAR10 we fine-tune for 3 epochs. During fine-tuning, we freeze the BatchNorm parameters of the model. We find that while this does not affect the fine-tuning accuracy it significantly improves the normalized decoding accuracy for all algorithms. A more extensive evaluation on the effect of using BatchNorm for erasure coding is left as future work. For Task Arithmetic we use the available P samples as the validation data and tune α in the range $[0.05, 0.1, 0.15, \ldots, 1.0]$ to find the α which achieves the highest normalized decoding accuracy on the validation data. For `COIN`, we similarly tune λ in the range $[10^{-5}, 10^{-4}, \ldots, 1]$ using the P samples as validation data. To implement RegMean we use the code publicly available on the official repository on Github. For Ensemble Distillation we again use the AdamW optimizer with a learning rate of 10^{-5}, batch size of 8, weight decay 0.1 and run the optimization for 20 epochs.

VIII. Additional Experiments and Results

We conduct additional experiments in the setting where we are coding over neural networks fine-tuned on CIFAR-10 and MNIST respectively to showcase the overfitting behavior of the Ensemble Distillation baseline and the effect of the number of datapoints P on the decoding accuracy. Figure 2(a) shows the average normalized decoding accuracies computed on the train set and test set for the Ensemble Distillation baseline as we train the coded model. We see that while the decoding accuracy for the train set quickly reaches close to 100, the accuracy for the test set saturates close to 75, implying that the coded model is clearly overfitting the training set. Note that we are using a weight decay of 0.1 in the optimization procedure which is a standard technique to prevent overfitting. Figure 2(b) shows the average normalized decoding accuracy for `COIN`, RegMean and Ensemble Distillation as a function of the number of datapoints P. We see that as P increases, the performance of Ensemble Distillation improves significantly, which is expected since the coded model is less likely to overfit as the size of the training data increases. Nonetheless, we note that the cost of computing the coded model using Ensemble Distillation also grows significantly as P increases. On the other hand there is only a slight improvement in the accuracy for `COIN` which reinforces the data efficiency and implicit computational ease of our proposed method. RegMean also sees an improvement as we increase the number of samples P which can be attributed to a more accurate estimation of the Gram matrices of the data.

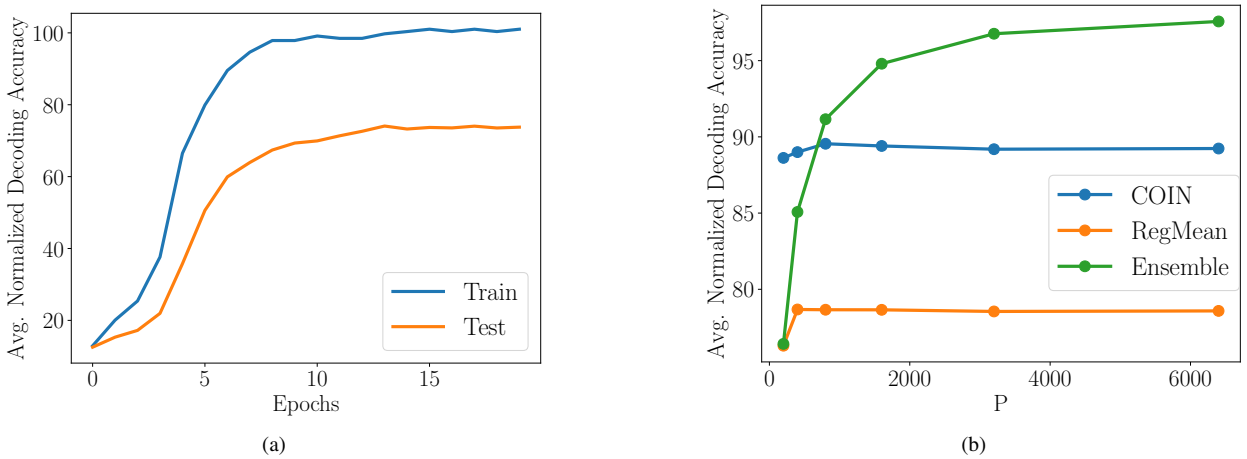

Fig. 2: (a) shows the average normalized decoding accuracies computed on the train set and test set for the Ensemble Distillation baseline as a function of the number of optimization epochs when coding over networks trained on CIFAR-10 and MNIST. The accuracy on the train set reaches close to 100 but accuracy on test set saturates close to 75, implying overfitting. (b) shows the average normalized decoding accuracy for `COIN`, RegMean and Ensemble Distillation in the same setting as a function of the number of datapoints P. We see only a slight increase in the accuracy of `COIN` as we increase P, which demonstrates the data-efficiency of our approach.