Partitioned Gradient Matching based Data Subset Selection for Compute-Efficient & Robust ASR Training

Ashish Mittal[1,2,†], Durga Sivasubramanian[2,†] , Rishabh Iyer[3],
Preethi Jyothi[2], Ganesh Ramakrishnan[2]
[1] IBM Research, India
[2] Indian Institute of Technology Bombay, Mumbai, India
[3] The University of Texas at Dallas, Dallas, USA

Abstract

Training state-of-the-art ASR systems such as RNN-T often have a high associated financial and environmental cost. Training with a subset of training data could mitigate this problem if the subset selected could achieve performance on-par with training with the entire dataset. Although there are many data subset selection (DSS) algorithms, direct application to the RNN-T is difficult, especially the DSS algorithms that are adaptive and use learning dynamics such as gradients, since RNN-T tends to have gradients with a significantly larger memory footprint. In this paper we propose **P**artitioned **G**radient **M**atching (PGM) a novel distributable DSS algorithm, suitable for massive datasets like those used to train RNN-T. Through extensive experiments on Librispeech 100H and Librispeech 960H, we show that PGM achieves between $3\times$ to $6\times$ speedup with only a very small accuracy degradation (under 1% absolute WER difference). In addition, we demonstrate similar results for PGM even in settings where the training data is corrupted with noise.

1 Introduction

Owing to their simplicity in directly mapping an acoustic input sequence to a output sequence of characters, or words, or even word-pieces, neural end-to-end methods (Graves et al., 2006, 2013; Chan et al., 2016; Vaswani et al., 2017; He et al., 2019) have become ubiquitous. The most common end-to-end architectures include (i) Connectionist Temporal Classification (CTC) models (Graves et al., 2006; Gulati et al., 2020), (ii) Attention-based Encoder-Decoder models (AED) (Chan et al., 2016; Watanabe et al., 2017) and (iii) Sequence Transduction models (Graves, 2012) such as RNN-Ts (Graves et al., 2013). Due to their streaming and low-latency properties, sequence transduction

architectures such as RNN-T (Graves et al., 2013; Sainath et al., 2020; Saon et al., 2021) are becoming state-of-the-art for modeling the ASR problem.

These successes in the ASR have come at a cost, as most of the practical RNN-T models are trained on thousands of hours of labeled datasets (Rao et al., 2017; Zhao et al., 2021). Model training on these massive datasets leads to significantly increased training time, energy requirements, and consequently the carbon footprint (Sharir et al., 2020; Strubell et al., 2019; Schwartz et al., 2020; Parcollet and Ravanelli, 2021). As per Parcollet *et al.* (Parcollet and Ravanelli, 2021), training an RNN-T model on Librispeech 960H (Panayotov et al., 2015) emits more than 10kg CO_2 if trained in France, which becomes much worse for developing countries. This is exacerbated due to the many more training runs required for hyper-parameter tuning. This warrants a need for greener training strategies that rely on significantly lower resources while still achieving state-of-the-art results.

One way to make ASR training more efficient is to train on a subset of the training data, which ensures minimum performance loss (Killamsetty et al., 2021a; Wei et al., 2014; Kaushal et al., 2019; Coleman et al., 2020; Har-Peled and Mazumdar, 2004; Clarkson, 2010; Mirzasoleiman et al., 2020; Killamsetty et al., 2021b; Liu et al., 2017). Since training on a subset reduces end-to-end time, the hyperparameter tuning time is also reduced. While greedy subset selection algorithms employ various criteria to identify the appropriate subset of training points, the process of forming the subsets remains sequential. However, for a large scale speech corpus such as Librispeech (Panayotov et al., 2015) this requirement may be difficult to meet. In this work, we propose a Partitioned Gradient Matching (PGM) approach, which scales well with huge datasets used in ASR and takes advantage of distributed setups. To the best of our knowledge, this is the first such study performed for ASR systems.

---
†Equal contribution. Correspondence to: arakeshk@in.ibm.com, durgas@cse.iitb.ac.in

1.1 Contributions of this work

The PGM Algorithm: We present PGM a data subset selection algorithm which constructs partial subsets from data partitions of the original dataset. This circumvents the need to load the entire dataset at a time into the memory, which is otherwise prohibitively expensive for ASR systems such as RNN-T(see Section 3).

PGM is a distributable Algorithm: Training with a subset of the training data is beneficial only when the cost of selecting a subset is also less. Therefore, for subset selection algorithms to scale to larger datasets used in speech recognition, they must work across multiple GPUs, since training for ASR systems can then be distributed. In Section 4, we present PGM which is more suitable for ASR systems, more specifically for RNN-T.

Trade-off between efficiency and accuracy: A subset selection algorithm has to counter the contrasting goals of efficiency and accuracy. We perform extensive experiments to demonstrate the trade-off between efficiency and accuracy for PGM and provide a general recipe for a user to control the trade-off.

Effectiveness of PGM in a Noisy ASR setting: A subset selection algorithm should work well when the training data is corrupted with noise. In this work, we show the efficacy of PGM, even when a fraction of the labeled dataset is augmented with noise across varying signal-to-noise ratios.

2 Background: RNN Transducer

The RNN-T model (Graves et al., 2013; Graves, 2012) maps an input acoustic signal $(x_1, x_2, \ldots, x_T)$ to an output sequence $(y_1, y_2, \ldots, y_U)$, where each output symbol $y_i \in \mathcal{M}$, and $\mathcal{M}$ is the vocabulary. An RNN-T model consists of three components - (i) Transcription Network - which maps an acoustic signal $(x_1, x_2, \ldots, x_T)$ to an encoded representation $(h_1, h_2, \ldots, h_T)$, T being the length of the acoustic signal and x_i being a W dimensional feature representation, (ii) Prediction Network - which is a language model that maps the previously emitted non-blank tokens $\mathbf{y}_{<U} = y_1, y_2, \ldots, y_{u-1}$ to an output space g_U for the next output token. (iii) Joint Network - that combines the Transcription Network representation h_t and Prediction Network representation g_u to produce $z_{t,u}$ using a feed-forward network J and $\oplus$ as a combination operator (typically a sum).

$$h_t = TranscriptionNetwork(x, t) \quad (1)$$

$$g_u = PredictionNetwork(y, u) \quad (2)$$

During the training, the output probability $P_{\text{rnnt}}(y_{t,u})$ over the output sequence $\mathbf{y}$ is marginalized over all possible alignments using an efficient forward-backward algorithm to compute the log-likelihood. The training objective is to minimize the Negative Log Likelihood of the target sequence.

$$P_{\text{rnnt}}(y_{t,u}|\mathbf{y}_{<u}, x_t) = \text{softmax}(J(h_t \oplus g_u)) \quad (3)$$

$$\mathcal{L} = -lnPr(y|x) \quad (4)$$

For inference, the decoding algorithms (Graves, 2012; Saon et al., 2020) attempt to find the best (t, u) and their corresponding output sequence $\mathbf{y}$ using a beam search. In this work, we use the gradients of the joint network layer (J) for PGM, since the linear layer helps in fusing the audio(h_t) and the text(g_u) representations.

3 Limitations of existing subset selection algorithms

An approach to the selection of a subset of points from the entire dataset is to rank points based on their suitability. This ranking can be done either via a some static metric such as diversity or representation among features (Wei et al., 2014; Kaushal et al., 2019) or via a dynamic metric using instance-wise loss gradients[1] to construct the subset greedily (Mirzasoleiman et al., 2020; Killamsetty et al., 2021b,a). In the latter case, ranking and re-ranking happens using instance-wise loss gradients. Specifically, during the selection process, loss gradients of the entire set of instances have to be available in the memory in order to perform greedy selection, since otherwise, subset selection time would be prohibitively large owing to disk reads, *etc*.

As keeping all the loss gradients in the memory would be resource intensive, we employ the following approximations, which have been also previously employed by (Mirzasoleiman et al., 2020; Killamsetty et al., 2021b,a), *viz.*, (i) only last layer gradients are used and (ii) subsets are constructed for each class. The latter technique is not relevant in ASR systems since ASR requires sequential decoding into a large size vocabulary. Similar to the

---

[1]gradient associated with an instance (x, y) as opposed to mean mini-batch loss gradient used in training the model
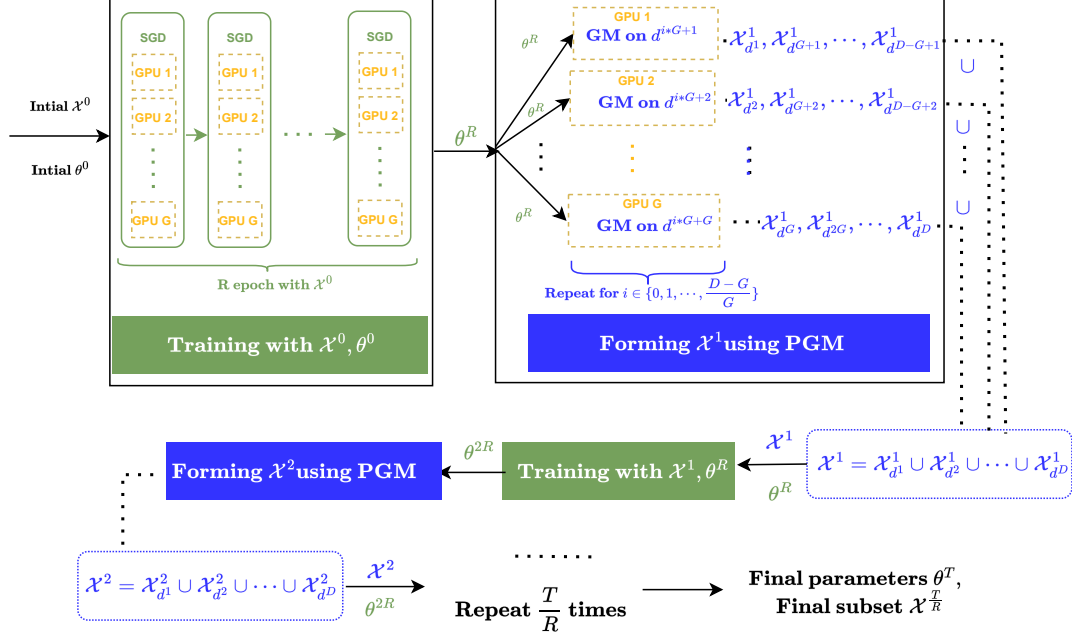
Figure 1: As PGM is a adaptive DSS algorithm, PGM is invoked after for every R epochs training RNN-T using stochastic gradient descent. At every time step, using the latest set of parameters, PGM forms partial subsets via Gradient Matching (GM) across GPUs. These partial subsets are combined and used for the next R epochs of RNN-T training. This is repeated until the final set of parameters is obtained.

last layer approximation, for the RNN-T model, we use the gradients of the joint network layer (J) which performs the important task of fusing speech (h_t) and text (g_u) features for sequence transduction. In Table 1 we present the memory footprint of the last layer gradient obtained while training ResNet18 (He et al., 2016) using CIFAR10 (Krizhevsky, 2009) and gradients of the joint network layer of RNN-T using Librispeech 100H. We compare against training ResNet18 using CIFAR10, since most of these subset selection algorithms are applied to image classification settings. In the first column of the table 1, we present the memory footprint of single instance's loss gradient. Clearly, the loss gradients used to train RNN-T have a much higher footprint than the ones used in image classification setting. The CIFAR10 dataset has 50,000 instances and Librispeech 100H has 20539. In the second column, we present the total memory required to store all the instance-wise loss gradients. The memory requirement for RNN-T's loss gradients prohibitively huge. Thus, storing all the instance-wise loss gradients at once is not feasible for RNN-T systems.

Killamsetty et al. (2021a) propose another technique, *viz.*, the *PerBatch* version, wherein one se-

Dataset	Single Gradient size (MB)	Total size (GB)	Per Batch size (GB)
CIFAR10	0.0215	1.049	0.0082
Librispeech 100H	4.096	111	28

Table 1: Memory footprint of last layer gradient obtained while training ResNet18 using CIFAR10 and gradients of the joint network layer of RNN-T using Librispeech 100H. We use a batch size of 128 for CIFAR10 and 4 for Librispeech 100H.

lects mini-batches (like used in SGD) instead of individual instances. Reduction in memory by using this technique is also not much for ASR systems such as RNN-T, since batch size used here is often small. For example, the batch size employed for the CIFAR10 dataset is typically of 128, as proposed by (He et al., 2016) whereas the batch size is 4 for Librispeech 100H as used in the Speech-Brain (Ravanelli et al., 2021) Librispeech RNN-T recipe. We present the memory required to store all the batch-wise loss gradients in the third column of Table 1. Although this requirement may seem satisfiable with some high end computing resource, however shown are the memory requirements to

store the instance-wise loss gradients only. If we add other memory needs such as space to store RNN-T model and space to process features and gradient computations, effectively one needs much larger GPU memory that the figures presented in Table 1. These memory issues become even more pronounced while performing subset selection with Librispeech 960H.

Another problem with the existing subset selection algorithms is that they are sequential in nature. This doesn't allow the selection algorithm to enjoy the speedup achieved using state of the art techniques such as parallelizing across multiple GPUs etc. This may cause the subset selection algorithm to be a bottleneck while training RNN-T with datasets of the scale of Librispeech. Therefore, there is a need to design an data subset algorithm that doesn't need all the loss gradient to form a subset and could be distributed across GPUs.

4 Partitioned Gradient Matching Algorithm

Let $\mathcal{U} = \{(x_i, y_i)\}_{i=1}^N$ denote the set of training examples, and $\mathcal{V} = \{(x_j, y_j)\}_{j=1}^M$, the validation set. Let θ denote the ASR system's parameters with θ^t as the ASR system's parameters at the tth epoch. The training loss associated with the ith instance is denoted by $L_T^i(\theta) = L_T(x_i, y_i, \theta) = -\ln Pr(y_i|x_i)$. We denote the validation loss by $L_V = -\sum_{i \in \mathcal{V}} ln Pr(y_i|x_i)$. Let the training data be divided into D partitions, i.e., $\mathcal{U} = d^1 \cup d^2 \cup \cdots \cup d^D$ where each partition comprises of $\frac{N}{D}$ instances. Let B be the batch size, $b_n = N/B$ be the total number of mini-batches and $b_k = k/B$ the number of batches to be selected.

Let L_T^{dp} be the training loss associated with a data partition d^p and $\nabla_\theta L_T^{dp} = \{\nabla_\theta L_T^{dp B_1}(\theta^t), \cdots, \nabla_\theta L_T^{dp B_l}(\theta^t)\}$ denote the set of mini-batch gradients associated with the data partition d^p, where $l = \frac{b_n}{D}$. Let $L_T^{b_n}$ denote the set of mini-batch gradients. For each data partition d^p, we wish to perform gradient matching (GM), by optimising the following problem,

$$\underset{\mathcal{X}_{dp}^t \subseteq d^p, |\mathcal{X}_{dp}^t| \le \frac{b_k}{D}}{\operatorname{argmin}} \min_{\mathbf{w}_{dp}^t} \mathrm{E}_\lambda(\mathbf{w}_{dp}^t, \mathcal{X}_{dp}^t, L_T^{dp}, \nabla_\theta L_T^{dp}, \theta^t)$$

where,

$$\mathrm{E}_\lambda(\mathbf{w}_{dp}^t, \mathcal{X}_{dp}^t, L_T^{dp}, \nabla_\theta L_T^{dp}, \theta^t) = \lambda \|\mathbf{w}_{dp}^t\|^2 +$$
$$\| \sum_{i \in \mathcal{X}_{dp}^t} \mathbf{w}_{idp}^t \nabla_\theta L_T^{dp B_i} - \nabla_\theta L_T^{dp}(\theta^t) \| \quad (5)$$

This selects a subset of batches $\mathcal{X}_{dp}^t$ and associated weights $\mathbf{w}_{dp}^t$, such that the weighted sum of loss gradients associated with each instance in the subset are the best approximation of the loss gradient of the entire data partition d^p while honoring the budget constraints. We perform gradient matching on mini-batch wise loss gradients only as it helps in reducing the memory needs. Similarly, we can define gradient matching problem with loss associated with the validation set as,

$$\underset{\mathcal{X}_{dp}^t \subseteq d^p, |\mathcal{X}_{dp}^t| \le \frac{b_k}{D}}{\operatorname{argmin}} \min_{\mathbf{w}_{dp}^t} \mathrm{E}_\lambda(\mathbf{w}_{dp}^t, \mathcal{X}_{dp}^t, L_V, \nabla_\theta L_T^{dp}, \theta^t)$$

where,

$$\mathrm{E}_\lambda(\mathbf{w}_{dp}^t, \mathcal{X}_{dp}^t, L_V, \nabla_\theta L_T^{dp}, \theta^t) = \lambda \|\mathbf{w}_{dp}^t\|^2 +$$
$$\| \sum_{i \in \mathcal{X}_{dp}^t} \mathbf{w}_{idp}^t \nabla_\theta L_T^{dp B_i} - \nabla_\theta L_V(\theta^t) \| \quad (6)$$

The optimization problem given in Eq.(5) is weakly submodular (Killamsetty et al., 2021a; Natarajan, 1995). Hence, we can effectively solve it using a greedy algorithm with approximation guarantees – we use orthogonal matching pursuit (OMP) algorithm (Elenberg et al., 2018) to find the subset and their associated weights. We also add to Eq.(5) an l_2 regularization component to discourage large weight assignments to any of the instances selected in the subset, thereby preventing the model from overfitting on some samples.

The complete algorithm is presented in Algorithm 1. In the algorithm, 'Val' is a boolean flag that indicates whether to match the subset loss gradient with validation set loss gradient like in noisy settings ('Val=True') or with training set loss gradient ('Val=False'). Depending on the choice of the loss gradient, we perform gradient matching with L_T^{dp}, current model parameters θ_t, budget $\frac{b_k}{D}$, and a stopping criterion ϵ. We describe gradient matching in details in Algorithm 2. Once the appropriate batch for selection is determined, we form $\mathcal{X}_f$ adding all the samples constituting the selected mini-batch. The model is then trained using the mini-batch SGD. We randomly shuffle elements in the subset $\mathcal{X}^t$, divide them up into mini-batches of size B, and run mini-batch SGD with instance weights.

The complete block diagram of PGM is presented in Figure 1. As the subset selection process is dependant on the model parameters, we repeat the subset selection every R epochs. For each data

Algorithm 1 PGM: **P**artitioned **G**radient **M**atching

Require: Train set: $\mathcal{U} = d^1 \cup d^2 \cup \cdots \cup d^D$ consisting of D partitions; validation set: $\mathcal{V}$; initial subset: $\mathcal{X}^0$; subset size: b_k; TOL: ϵ; initial params: θ^0; learning rate: α; total epochs: T, selection interval: R, Validation Flag: Val, Batchsize: B
  for epochs t in $1, \cdots, T$ **do**
    if $(t \bmod R == 0)$ **then**
      $\mathcal{X}^t = \phi, \mathbf{w}^t = []$
      for data partition p in $d^1, \cdots, d^D$ **do**
        if Val **then**
          $\mathcal{X}_d^t, \mathbf{w}_d^t = \mathrm{GM}(L_V, \nabla_\theta L_T^{dp}, \theta^t, \frac{b_k}{D}, \epsilon)$
        else
          $\mathcal{X}_d^t, \mathbf{w}_d^t = \mathrm{GM}(L_T^{dp}, \nabla_\theta L_T^{dp}, \theta^t, \frac{b_k}{D}, \epsilon)$
        end if
        $\mathcal{X}^t = \mathcal{X}^t \cup \mathcal{X}_d^t$
        Extend $\mathbf{w}^t$ with $\mathbf{w}_d^t$
      end for
    else
      $\mathcal{X}^t = \mathcal{X}^{t-1}$
    end if
    $\theta_{t+1} = \mathrm{BatchSGD}(\mathcal{X}^t, \mathbf{w}^t, \alpha, B)$
  end for
  Output final model parameters θ^T

Algorithm 2 Gradient Matching (GM)

Require: Loss of the entire dataset(train or validation) : L, set of mini-batch gradients $\nabla_\theta L_T^B$, current parameters θ^t, budget k, TOL: ϵ;
  $\mathcal{X} = \phi, \mathcal{X}_f = \phi, r = \nabla_\theta L$
  for $|\mathcal{X}| \le k$ or $\mathrm{E}_\lambda(w, \mathcal{X}, L, \nabla_\theta L_T^B, \theta^t) > \epsilon$ **do**
    Pick a element j in $\nabla_\theta L_T^B$ which a maximum alignment with r
    $\mathcal{X} = \mathcal{X} \cup j$
    $\mathcal{X}_f = \mathcal{X} \cup \{\text{set of instances in the batch } j\}$
    Update $w = \min_w \mathrm{E}_\lambda(w, \mathcal{X}, L, \nabla_\theta L_T^B, \theta^t)$
    Update $r = r - \mathrm{E}_\lambda(w, \mathcal{X}, L, \nabla_\theta L_T^B, \theta^t)$
  end for
  Return $\mathcal{X}_f, w$

$$\mathrm{E}_\lambda(\mathbf{w}^t, \mathcal{X}^t, L, L_T^{b_n}, \theta^t) = \lambda \|\mathbf{w}^t\|^2 + \\ \|\sum_{i \in \mathcal{X}^t} \mathbf{w}_i^t \nabla_\theta L_T^{B_i}(\theta^t) - \nabla_\theta L(\theta^t)\|$$

$L_T^{b_n}$ denotes the set of all mini-batch gradients, defined as $L_T^{b_n} = \nabla_\theta L_T^{d1} \cup \nabla_\theta L_T^{d2} \cup \cdots \cup \nabla_\theta L_T^{dD}$ and L is either the training loss of the entire dataset L_T defined as $L_T = \mathbb{E}(L_T^{dp})$ or L_V depending on what sort matching we seek for. The problem tries to find subset and it associated weights so that the gradients of the mini-batches best approximate the either gradient associated with the full dataset or the validation set. We show that GRAD-MATCHPB is lower bound to PGM, that is

$$\mathbb{E}(\mathrm{E}_\lambda(\mathbf{w}_{dp}^t, \mathcal{X}_{dp}^t, L_T^{dp}, \nabla_\theta L_T^{dp}, \theta^t)) \\ \ge \mathrm{E}_\lambda(\mathbf{w}^t, \mathcal{X}^t, L_T, L_T^{b_n}, \theta^t))$$

and
$$\mathbb{E}(\mathrm{E}_\lambda(\mathbf{w}_{dp}^t, \mathcal{X}_{dp}^t, L_V, \nabla_\theta L_T^{dp}, \theta^t)) \\ \ge \mathrm{E}_\lambda(\mathbf{w}^t, \mathcal{X}^t, L_V, L_T^{b_n}, \theta^t))$$

For the proof, we refer the reader to Appendix A.

---

partition d^p, we perform gradient matching (GM) individually and obtain partial subsets $\mathcal{X}_{dp}^t$, sequentially, one after another. However in the presence of multi-GPU settings, since the gradient matching within a a data partition can be performed independently from gradient matching in other data partitions, the gradient matchings could be executed in parallel. This allows one to take advantage of multi-GPU settings which is critical to efficiently process large datasets typically used to train RNN-T. In Figure 1, we illustrate parallelization of PGM on the system with G GPUs. Here, every G partial subsets are obtained in parallel and this process is repeated $\frac{D}{G}$ times.

4.1 Connection to existing work

In this section we discuss the connection of PGM with GRAD-MATCHPB (Killamsetty et al., 2021a) where subset is selected via solving the following problem,

$$\operatorname*{argmin}_{\mathcal{X}^t \subseteq \mathcal{U}, |\mathcal{X}^t| \le b_k} \min_{\mathbf{w}^t} \mathrm{E}_\lambda(\mathbf{w}^t, \mathcal{X}^t, L, L_T^{b_n}, \theta^t)$$

where

5 Experiments

Datasets We perform all our experiments on the Librispeech dataset (Panayotov et al., 2015). We present results on the medium-scale Librispeech 100H as well as on the large-scale Librispeech 960H datasets.

Along with the standard Librispeech benchmark, we also perform experiments on noisy Librispeech, where the speech is augmented with noise across varying signal-to-noise ratios (up to 15db) on a fraction of the training data. We refer to this dataset
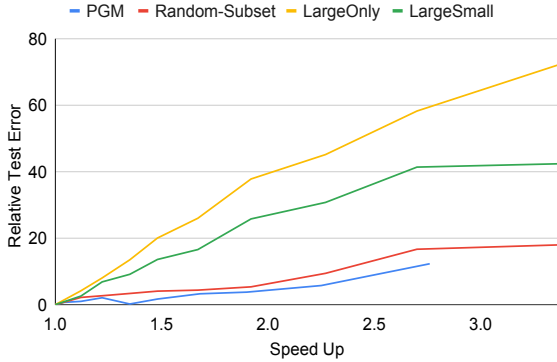
Figure 2: Relative Test Error(↓) *vs.* Speed Up(↑) for subset selection methods on Librispeech 100H TEST-CLEAN test set.

Figure 3: Word Error Rate (WER) on the TEST-CLEAN test set of Librispeech 100H for all the methods.

as **Librispeech-noise**, where up to 30% examples in the original dataset are augmented with noise across varying signal-to-noise ratios.

Architecture. We perform all our experiments on the Speechbrain's(Ravanelli et al., 2021) Librispeech transducer recipe. The transcription network of the RNN-T consists of a CRDNN encoder which has 2 CNN blocks followed by 4 layers of bi-LSTMs and subsequently followed by 2 DNN layers. The prediction network consists of an embedding layer followed by a single layer GRU unit. A joint network is a single linear layer that projects 1024 dimensional representations to output a vocabulary of 1000 BPE. The decoding is done through a time-synchronous decoding algorithm (Graves, 2012; Hannun et al., 2019) with a beam size of 4. The decoding involves an external transformer language model trained on the Librispeech corpus (Kannan et al., 2018; Hrinchuk et al., 2020; Wolf et al., 2019).

Training Details. For the training, we employ a learning rate of 2.0 with an annealing factor of 0.8 for the relative improvement of 0.0025 on validation loss (sometimes referred to as newbob scheduler). The training on Librispeech 100H is performed on two A100 40GB GPUs with the effective batch size of 8, whereas for Librispeech 960H, we employ two A100 80GB GPUs with an effective batch size of 24. All the training is done for 30 epochs. In all our experiments, the PGM algorithm is invoked after every 5th epoch ($R = 5$) after performing warm-start (training on full data) for 7 and 2 epochs on Librispeech 100H and Librispeech 960H datasets respectively. The results for each setting are averaged over 3 runs with different random seeds.
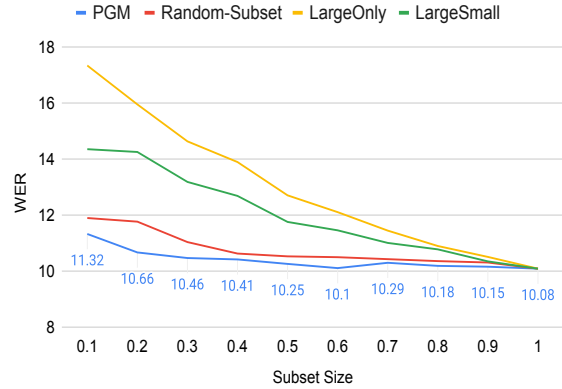
PGM Details. For doing the subset selection with PGM, we use the gradients of the Joint Network parameters, which we believe would have the maximum information concentrated for the sequence. We freeze the rest of the network while we compute the gradient of the Joint Network of the RNN-T. We use $D = 7$ and $D = 50$ (data partitions) to obtain subsets using the PGM algorithm over gradients of training data for Librispeech 100H and 960H datasets respectively. Subset selection is performed using ***training set loss gradients*** in experiments performed using Librispeech 100H (Figures 2,3) and Librispeech 960H (table 2). For experiments with **Librispeech-noise** (Table 3) we employ the ***validation gradients*** for performing the subset selection, *since we are also concerned with robustness in the presence of noise.*

Baselines. We compare the results obtained using the PGM method against three intuitive baselines - (i) ***Random-Subset*** baseline, in which the subset of the dataset is obtained by choosing points with uniform probability. (ii) ***LargeOnly*** - For each subset, we employ only the largest utterances based on duration. (iii) ***LargeSmall*** - For each subset size, half of the subset is filled with smallest utterances and the other half with the largest utterances based on duration, to remove the length bias of the ***LargeOnly*** baseline.

5.1 Results

To compare the efficacy of the PGM, we compare the word error rate (WER), relative test error, and speed-up compared to training with the entire dataset. We compute these metrics for both the Librispeech 100H and Librispeech 960H benchmarks. Additionally, we also, present energy ratios *vs.* relative test error rate tradeoff on Librispeech 100H.

Subset	Method	WER (Rel. Test Error) (↓)		Speed Up (↑)
		TEST-CLEAN	TEST-OTHER	
100%	-	4.21 (0.0)	11.59 (0.0)	-
10%	*Random-Subset*	5.87 (39.43%)	15.39 (32.79%)	6.25
	PGM	5.71 (35.63%)	14.66 (26.49%)	4.43
20%	*Random-Subset*	5.08 (20.67%)	13.89 (19.84%)	3.95
	PGM	5.01 (19%)	12.78 (10.27%)	3.30
30%	*Random-Subset*	4.62 (12.65%)	12.65 (9.74%)	2.89
	PGM	4.58 (8.79%)	12.45 (7.42)	2.64

Table 2: Results showing WER (Relative Test Error) and Speed Up on TEST-CLEAN and TEST-OTHER test splits of Librispeech 960H.

	Subset	Noise = 10%		Noise = 20%		Noise = 30%	
		Random-Subset	PGM	*Random-Subset*	PGM	*Random-Subset*	PGM
100H	100%	10.59		11.16		11.39	
	10%	11.79	11.86	11.64	11.67	11.96	11.82
	20%	11.53	10.8	11.27	**11.12**	11.39	11.25
	30%	11.33	**10.7**	11.74	11.42	12.05	**11.17**
960H	100%	4.52		4.65		4.68	
	10%	6.5	6.28	6.44	6.54	6.58	6.43
	20%	5.61	5.58	5.44	5.65	5.84	5.5
	30%	4.99	**4.97**	5.16	**5.02**	5.62	**5.17**

Table 3: Results showing WER on TEST-CLEAN test set of Librispeech 100H trained using noisy Librispeech dataset using PGM and *Random-Subset*.

In Figure 2, we present the comparison of WER for PGM against various baselines for various subset sizes of the full dataset. With just 20% of the subset size, the PGM method yields a WER of 10.66 as opposed to 10.08 obtained by training on the full dataset. For Librispeech 100H, PGM consistently outperforms all the baseline, thus illustrating the effect of selecting subsets using the gradient matching algorithm. Also note, *Random-Subset* baseline is consistently better than other heuristic based baselines such LargeOnly and LargeSmall. In Figure 3, we plot the speed up against the Relative Test Error for Librispeech 100H. While *Random-Subset* baseline is observed to attain higher speed up in comparison to the PGM because of the simple selection strategy, *Random-Subset* baseline also incurs higher relative test error in comparison to the PGM.

In Figure 4, we present the plot of relative test error *w.r.t* energy efficiency for the full training setting. We use pyJoules[2] for measuring the energy consumed by GPU cores. We show that with PGM, the training time is halved and energy efficiency is doubled while incurring the relative test error of
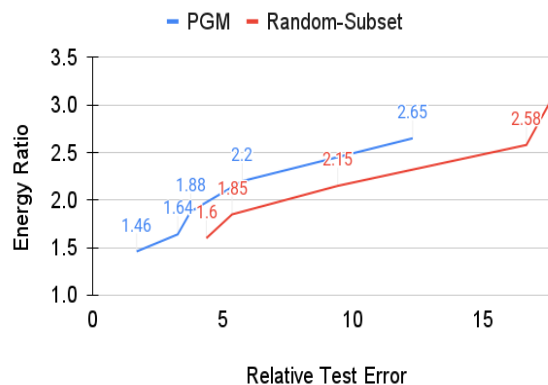

Figure 4: Energy Ratio(↑) *vs.* Relative Test Error(↓) for PGM and *Random-Subset* on Librispeech 100H.

less than 5% as compared to the training on the entire dataset. For higher speedups, where there is a degradation in the WER, the loss is relatively better for PGM as compared to the baseline. We do not show the energy efficiency for LargeOnly and LargeSmall baselines as their relative test error is consistently poor as compared to the Random Subset baseline as shown in Figure 3.

For the ASR task we recommend using at least 30% of the dataset for training the model or using

---

	Random-Subset	PGM
Overlap Index	20.2%	6.37%
Noise Overlap Index	0.82%	0.83%

Table 4: Overlap Indices - measures the overlap between consecutive subsets for PGM and *Random-Subset* methods.

Subset Size	nGPU = 1 LR = 1.0	nGPU=2 LR = 1.0	nGPU=2 LR = 2.0
0.1	11.26	13.99	11.32
0.2	10.6	12.58	10.66
0.3	10.4	11.58	10.46

Table 6: Effect of Learning Rate on WER for PGM on TEST-CLEAN test set of Librispeech 100H.

more warm-start epochs as described in Section 5.2. In Table 2, we present comparison of the PGM method with the baseline for the Librispeech 960H dataset on both the TEST-CLEAN and TEST-OTHER test sets. As shown in the Table, with just 30% of the training data, PGM is within 10% of the relative test error (1% of absolute error difference) when compared against training on the full data, thus yielding a speedup of 2.64. Similar results hold on the challenging TEST-OTHER test set of the Librispeech which shows the better generalization of PGM in comparison to the *Random-Subset* baseline.

Results on Librispeech-noise: We augment randomly selected signals from the dataset with noise across varying signal-to-noise ratios to mimic a more practical setting where subset selection algorithms need to address the noise while selecting useful subsets. We show the results on the Librispeech-noise 100H and 960H datasets for different subsets in Table 3. PGM consistently outperforms the *Random-Subset* baseline for different subsets with lower relative test error when compared against the full training and still yields significant speed up to reduce training time and maintain robustness.

5.2 Ablation Study

Next, we do an ablation study to understand the effect of learning rate on PGM for Librispeech 100H dataset. Since, the goal of subset selection algorithms is to reduce the training data for training, the older recipes (especially learning rate) on full training data do not work as-is for the PGM because of the distributed nature of the training.

Subset Size	WS = 2 epochs	WS= 3 epochs
10%	5.71	5.3
20%	5.01	4.82
30%	4.58	4.54

Table 5: Effect of warm-start (WS) on WER for PGM on TEST-CLEAN test set for Librispeech 960H

In Table 6, we show the effect of learning rate on multi-gpu training of the PGM method. The recipe for single GPU borrowed as-is for the multi-gpu training setting, performed poorly because the number of gradient updates in the distributed setting halved. To overcome this barrier, we doubled the learning rate to take larger steps and reach convergence within the same number of epochs.

We perform some ablation studies to understand why subsets selected by PGM tend to outperform a relatively simple *Random-Subset* baseline. We compute the following two metrics:

Overlap Index (OI): This is the fraction of common points selected in the last two subset selection rounds with the subset size. This metric computes the diversity of the points being selected by the methods in the subsequent subset selection rounds.

Noise Overlap Index (NOI): This is the fraction of noise points selected by the subset selection methods divided by the total number of noisy points. Both the metrics are computed by averaging the index for all the runs with the same subset selection method.

As shown in Table 4, PGM selects more diverse points across different subset selection rounds which explains the better generalization of the TEST-OTHER test set. At the same time, both the methods select a similar amount of noisy points during the subset selection indicating that PGM selects more diverse points from the non-noisy points.

Finally, we study the effect of warm-start on the performance of the PGM algorithm. Since it is an adaptive data selection algorithm, PGM needs a good starting point for computing reasonable estimates of the gradients for subset selection. Table 5 shows the effect of warm-start epoch ablation on the TEST-CLEAN test set for Librispeech 960H. As we increase the warm start, the performance of the PGM algorithm improves at the cost of speed up.

5.3 Comparing PGM and GRAD-MATCHPB

Running GRAD-MATCHPB for Librispeech is prohibitively expensive since the amount of memory

Subset-Size	*Random-Subset*	*LargeSmall*	*LargeOnly*	GRAD-MATCHPB	PGM
0.1	16.64	17.98	17.27	16.14	16.23
0.2	16.43	17.23	16.35	15.89	16.03
0.3	16.28	16.35	16.22	15.79	15.95

Table 7: Comparison of WER obtained with *Random-Subset*, *LargeSmall*, *LargeOnly*, GRAD-MATCHPB and PGM on TIMIT Phone recognition dataset.

required to store all the gradients would exceed the memory size of available commercial GPUs as described in Section 3. To address this, we compare Phone Error (PER) on the TIMIT Phone recognition dataset (Garofolo, 1993) (containing 3680 utterances with 630 speakers) for all the methods.

Table 7 shows WER obtained with PGM, GRAD-MATCHPB *Random-Subset* and other subset selection baselines such as *LargeSmall*, *LargeOnly*. For PGM we use data partitioning $D = 2$. We see that the WER of PGM is slightly higher than that of GRAD-MATCHPB, as the error term that PGM minimises is a upper bound of error term minimised by GRAD-MATCHPB as discussed in section 4.1. However, PGM's WER is very close to that of GRAD-MATCHPB, indicating that the partitioning doesn't deteriorate the bounds while allowing to scale for larger datasets and utilize multiple GPUs which allows PGM to enjoy better speedups over GRAD-MATCHPB.

Statistical Significance: WER reductions using PGM compared to the *Random-Subset* baseline are statistically significant at $p < 0.001$ using a matched pairs test.[3]

6 Conclusion

We propose PGM, a distributable data subset selection algorithm which avoids the need to load the entire dataset at a time, by constructing partial subsets from smaller data partitions. PGM is an adaptive subset selection algorithm that improves the training time of the ASR models while maintaining low relative test error as compared to the ASR model trained with the entire dataset. This speed-up improves the efficiency of the training process and subsequently reduces the carbon footprint of training such models. Our approach performs consistently better than *Random-Subset* baseline whilst providing good speed up, and robustness in the presence of noise. Although we test the method on the RNN-T model, we believe

that similar results could be obtained for other ASR models and we leave that as future work.

Limitations

In this paper we investigate the usefulness of subset selection algorithms for the ASR task for the first time on a popular RNN-Transducer ASR architecture which typically consumes vast volume (~40000 hours of labelled audio and more) of training data. At such industrial scale, the overhead of PGM for gradient matching over the entire training set would limit the utility of the algorithm. These practical considerations warrant more careful design of the subset selection algorithms so as to scale well with such huge workloads. We also limit our results to the RNN-T architecture and believe that the results also hold for other less popular architecture by taking gradients of the last few layers. While we show the efficient training for the ASR task, we believe a similar study should be carried out for the self-supervised pre-training approaches.

Ethics Statement

In this work we present a gradient matching based data subset selection algorithm for compute efficient and robust ASR model training. Since we do not modify any existing speech architecture or propose new benchmarks, but provide a mechanism for faster training of such models, we see no new ethical concerns arising from our work.

7 Acknowledgements

---

[3]https://github.com/talhanai/wer-sigtest

References

William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. 2016. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4960–4964. IEEE.

Kenneth L Clarkson. 2010. Coresets, sparse greedy approximation, and the frank-wolfe algorithm. *ACM Transactions on Algorithms (TALG)*, 6(4):1–30.

Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. 2020. Selection via proxy: Efficient data selection for deep learning.

Ethan R Elenberg, Rajiv Khanna, Alexandros G Dimakis, Sahand Negahban, et al. 2018. Restricted strong convexity implies weak submodularity. *The Annals of Statistics*, 46(6B):3539–3568.

John S Garofolo. 1993. Timit acoustic phonetic continuous speech corpus. *Linguistic Data Consortium, 1993*.

Alex Graves. 2012. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. Ieee.

Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al. 2020. Conformer: Convolution-augmented transformer for speech recognition. *arXiv preprint arXiv:2005.08100*.

Awni Hannun, Ann Lee, Qiantong Xu, and Ronan Collobert. 2019. Sequence-to-sequence speech recognition with time-depth separable convolutions. *arXiv preprint arXiv:1904.02619*.

Sariel Har-Peled and Soham Mazumdar. 2004. On coresets for k-means and k-median clustering. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 291–300.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Yanzhang He, Tara N Sainath, Rohit Prabhavalkar, Ian McGraw, Raziel Alvarez, Ding Zhao, David Rybach, Anjuli Kannan, Yonghui Wu, Ruoming Pang, et al. 2019. Streaming end-to-end speech recognition for mobile devices. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6381–6385. IEEE.

Oleksii Hrinchuk, Mariya Popova, and Boris Ginsburg. 2020. Correction of automatic speech recognition with transformer sequence-to-sequence model. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7074–7078. IEEE.

Anjuli Kannan, Yonghui Wu, Patrick Nguyen, Tara N Sainath, Zhijeng Chen, and Rohit Prabhavalkar. 2018. An analysis of incorporating an external language model into a sequence-to-sequence model. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5828. IEEE.

Vishal Kaushal, Rishabh Iyer, Suraj Kothawade, Rohan Mahadev, Khoshrav Doctor, and Ganesh Ramakrishnan. 2019. Learning from less data: A unified data subset selection and active learning framework for computer vision. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1289–1299. IEEE.

Krishnateja Killamsetty, Durga S, Ganesh Ramakrishnan, Abir De, and Rishabh Iyer. 2021a. Grad-match: Gradient matching based data subset selection for efficient deep model training. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5464–5474. PMLR.

Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, and Rishabh Iyer. 2021b. Glister: Generalization based data subset selection for efficient and robust learning. *In AAAI*.

Alex Krizhevsky. 2009. Learning multiple layers of features from tiny images. Technical report.

Yuzong Liu, Rishabh Iyer, Katrin Kirchhoff, and Jeff Bilmes. 2017. Svitchboard-ii and fisver-i: Crafting high quality and low complexity conversational english speech corpora using submodular function optimization. *Computer Speech & Language*, 42:122–142.

Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. 2020. Coresets for data-efficient training of machine learning models.

Balas Kausik Natarajan. 1995. Sparse approximate solutions to linear systems. *SIAM journal on computing*, 24(2):227–234.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE international conference on acoustics, speech*

and signal processing (ICASSP), pages 5206–5210. IEEE.

Titouan Parcollet and Mirco Ravanelli. 2021. The energy and carbon footprint of training end-to-end speech recognizers. *Interspeech*.

Kanishka Rao, Haşim Sak, and Rohit Prabhavalkar. 2017. Exploring architectures, data and units for streaming end-to-end speech recognition with rnn-transducer. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 193–199. IEEE.

Mirco Ravanelli, Titouan Parcollet, Peter Plantinga, Aku Rouhe, Samuele Cornell, Loren Lugosch, Cem Subakan, Nauman Dawalatabad, Abdelwahab Heba, Jianyuan Zhong, et al. 2021. Speechbrain: A general-purpose speech toolkit. *arXiv preprint arXiv:2106.04624*.

Tara N Sainath, Yanzhang He, Bo Li, Arun Narayanan, Ruoming Pang, Antoine Bruguier, Shuo-yiin Chang, Wei Li, Raziel Alvarez, Zhifeng Chen, et al. 2020. A streaming on-device end-to-end model surpassing server-side conventional model quality and latency. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6059–6063. IEEE.

George Saon, Zoltán Tüske, and Kartik Audhkhasi. 2020. Alignment-length synchronous decoding for rnn transducer. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7804–7808. IEEE.

George Saon, Zoltán Tüske, Daniel Bolanos, and Brian Kingsbury. 2021. Advancing rnn transducer technology for speech recognition. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5654–5658. IEEE.

Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. 2020. Green ai. *Communications of the ACM*, 63(12):54–63.

Or Sharir, Barak Peleg, and Yoav Shoham. 2020. The cost of training nlp models: A concise overview. *arXiv preprint arXiv:2004.08900*.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Shinji Watanabe, Takaaki Hori, Suyoun Kim, John R Hershey, and Tomoki Hayashi. 2017. Hybrid ctc/attention architecture for end-to-end speech recognition. *IEEE Journal of Selected Topics in Signal Processing*, 11(8):1240–1253.

Kai Wei, Rishabh Iyer, and Jeff Bilmes. 2014. Fast multi-stage submodular maximization. In *International conference on machine learning*, pages 1494–1502. PMLR.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Rui Zhao, Jian Xue, Jinyu Li, Wenning Wei, Lei He, and Yifan Gong. 2021. On addressing practical challenges for rnn-transducer. *arXiv preprint arXiv:2105.00858*.

A Connections between PGM and GRAD-MATCHPB

Lemma 1 *(triangle inequality). Let $v_1, ..., v_\tau$ be τ vectors in $\mathbb{R}^d$. Then the following is true:*

$$\|\sum_{i=1}^{\tau} v_i\| \geq \sum_{i=1}^{\tau} \|v_i\| \tag{7}$$

Corollary 1 *Following inequality holds between the objectives of* PGM *and* GRAD-MATCHPB

$$\mathbb{E}(E_\lambda(\mathbf{w}_{d^p}^t, \mathcal{X}_{d^p}^t, L_T^{d^p}, \nabla_\theta L_T^{d^p}, \theta^t))$$
$$\geq E_\lambda(\mathbf{w}^t, \mathcal{X}^t, L_T, L_T^{b_n}, \theta^t))$$

and

$$\mathbb{E}(E_\lambda(\mathbf{w}_{d^p}^t, \mathcal{X}_{d^p}^t, L_V, \nabla_\theta L_T^{d^p}, \theta^t))$$
$$\geq E_\lambda(\mathbf{w}^t, \mathcal{X}^t, L_V, L_T^{b_n}, \theta^t))$$

Proof.

Using the triangle inequality,

$$\sum_{i=p}^{D}(\|\sum_{i\in\mathcal{X}_{d^p}^t} \mathbf{w}_{id^p}^t \nabla_\theta L_T^{d^p B_i} - \nabla_\theta L_T^{d^p}(\theta^t)\|$$
$$+ \lambda\|\mathbf{w}_{d^p}^t\|^2)$$
$$\geq \|\sum_{i=p}^{D}(\sum_{i\in\mathcal{X}_{d^p}^t} \mathbf{w}_{id^p}^t \nabla_\theta L_T^{d^p B_i} - \nabla_\theta L_T^{d^p}(\theta^t))\|$$
$$+ \lambda\|\sum_{i=p}^{D} \mathbf{w}_{d^p}^t\|^2$$

We divide both sides by D,

$$\frac{1}{D}\sum_{i=p}^{D}(\|\sum_{i\in\mathcal{X}_{d^p}^t} \mathbf{w}_{id^p}^t \nabla_\theta L_T^{d^p B_i} - \nabla_\theta L_T^{d^p}(\theta^t)\|$$
$$+ \lambda\|\mathbf{w}_{d^p}^t\|^2)$$
$$\geq \|\sum_{i=p}^{D}(\sum_{i\in\mathcal{X}_{d^p}^t} \frac{\mathbf{w}_{id^p}^t}{D} \nabla_\theta L_T^{d^p B_i})$$
$$- \frac{\sum_{i=p}^{D}(\nabla_\theta L_T^{d^p}(\theta^t))}{D}\| + \lambda\|\frac{\sum_{i=p}^{D} \mathbf{w}_{d^p}^t}{D}\|^2$$

$$\mathbb{E}(E_\lambda(\mathbf{w}_{d^p}^t, \mathcal{X}_{d^p}^t, L_T^{d^p}, \nabla_\theta L_T^{d^p}, \theta^t))$$
$$\geq \|\sum_{i=p}^{D}(\sum_{i\in\mathcal{X}_{d^p}^t} \frac{\mathbf{w}_{id^p}^t}{D} \nabla_\theta L_T^{d^p B_i})$$
$$- \frac{\sum_{i=p}^{D}(\nabla_\theta L_T^{d^p}(\theta^t))}{D}\| + \lambda\|\frac{\sum_{i=p}^{D} \mathbf{w}_{d^p}^t}{D}\|^2$$

Since $L_T = \mathbb{E}(L_T^{d^p})$ and therefore $\mathbb{E}(\sum_{i=p}^{D}(\sum_{i\in\mathcal{X}_{d^p}^t} \frac{\mathbf{w}_{id^p}^t}{D} \nabla_\theta L_T^{d^p B_i})) = \sum_{i\in\mathcal{X}^t} \mathbf{w}_i^t \nabla_\theta L_T^{B_i}(\theta^t)$ as they are obtained via gradient matching,

$$\mathbb{E}(E_\lambda(\mathbf{w}_{d^p}^t, \mathcal{X}_{d^p}^t, L_T^{d^p}, \nabla_\theta L_T^{d^p}, \theta^t))$$
$$\geq \|\sum_{i\in\mathcal{X}^t} \mathbf{w}_i^t \nabla_\theta L_T^{B_i}(\theta^t) - \nabla_\theta L_T(\theta^t)\|$$
$$+ \lambda\|\mathbf{w}^t\|^2$$

$$\mathbb{E}(E_\lambda(\mathbf{w}_{d^p}^t, \mathcal{X}_{d^p}^t, L_T^{d^p}, \nabla_\theta L_T^{d^p}, \theta^t))$$
$$\geq E_\lambda(\mathbf{w}^t, \mathcal{X}^t, L_T, L_T^{b_n}, \theta^t))$$

6010