DAREL: Training and Fine-Tuning Acceleration of Real and Hypercomplex Models

Alexander Demidovskij
Aleksei Trutnev
Artyom Tugaryov
Igor Salnikov
Stanislav Pavlov

Introduction

The industry is predisposed to the growth of neural network parameters and the increase in datasets size. It necessarily leads to an increase in the required computational resources as well as the training time. Furthermore, every hour that the GPU leads to an increase in carbon dioxide emissions into the atmosphere due to its required power production [Strubell et al., 2019].

This paper makes the following primary contributions:

1. Introduced a novel two-stage method that is designed to accelerate the pre-training of CNN and fine-tuning of Large Language Models (LLM) by applying the importance sampling strategies based on loss information. As a result, solid acceleration of ResNet18 pre-training (up to 2.03x) and GPT2-M fine-tuning (up to 1.43x) are demonstrated.
2. Presented the concept of a training budget for CV pre-training as a combination of maximum GPU memory utilization and maximum training time.
3. DAREL improves the state-of-the-art method for LLM fine-tuning (LoRA [Hu et al., 2021].) and allows for the increase of BLEU score for GPT2-M by 1.81 p.p. with 25% acceleration for E2E-NLG dataset.

Background study

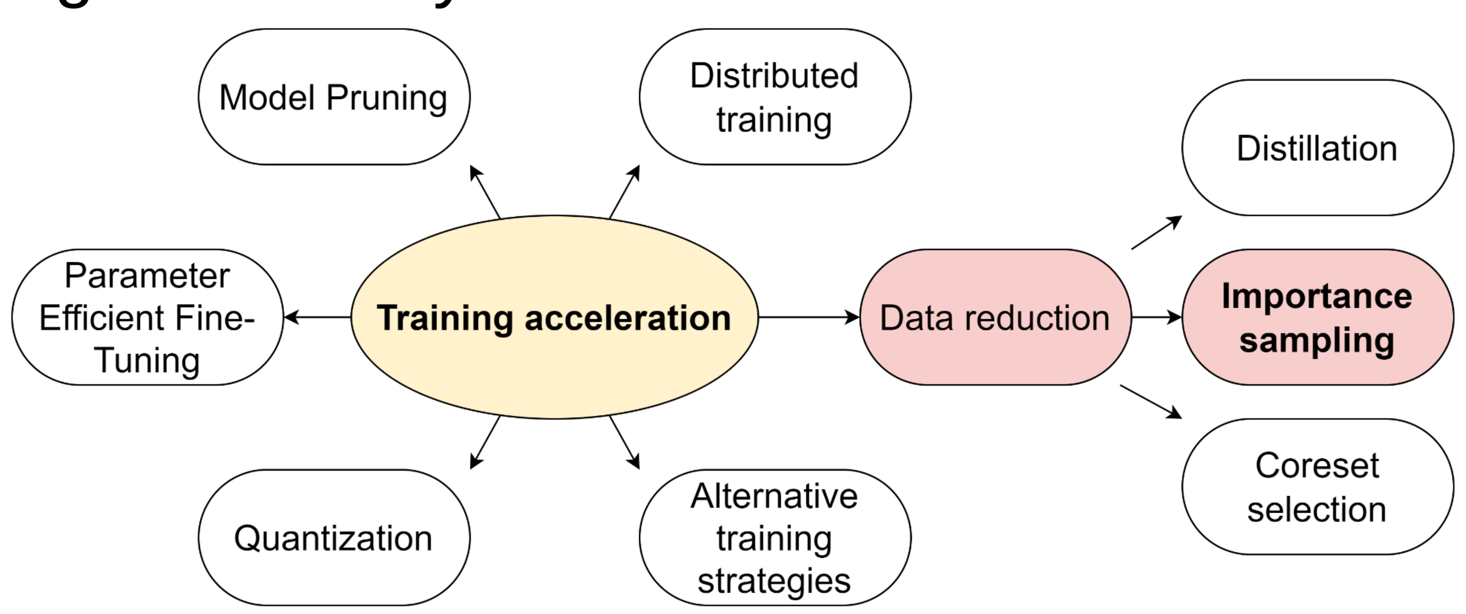
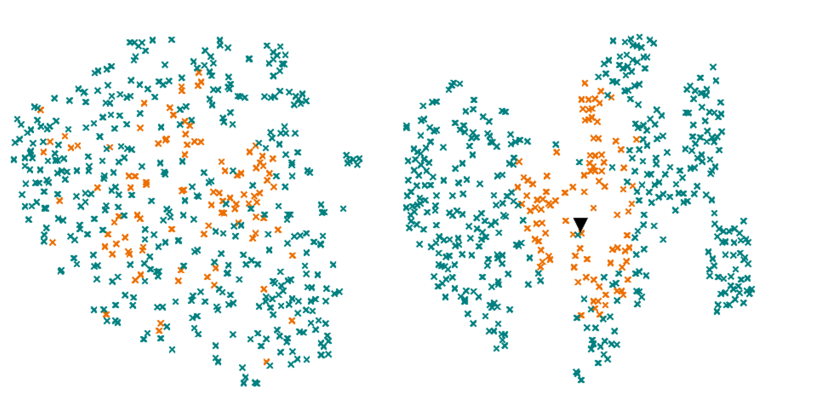


Figure 1. Acceleration methods. The highlighted area was investigated in this work.

The state-of-the-art in data reduction is achieved with *Intellectual Data Selection* (IDS) and *Adaptive Online Importance Sampling* (ADONIS) methods [Demidovskij et al., 2023]. The IDS methodology filters the training datasets by selecting diverse samples from each class in a labeled dataset by clustering samples' embeddings with K-Means. The ADONIS aims to reduce the number of backward passes by choosing samples from the available training data and constructing new training batches containing only the important samples. IDS and ADONIS are greedy in terms of required resources, hyperparameters have to be handcrafted and no support for LLM fine-tuning.



(a) DAREL offline (b) IDS

Figure 2. Selected samples from CIFAR100 class #58 (pear) via DAREL offline (a) and IDS (b). In (a) and (b) cyan color denotes selected samples, and the orange color signifies rejected samples. In (b) black triangle stands for the class prototype

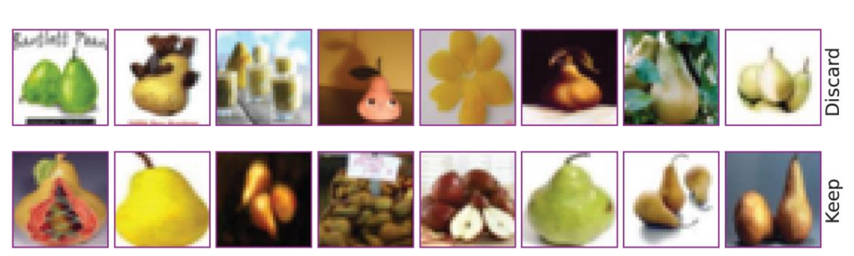


Figure 3. Samples were selected via DAREL offline from CIFAR100 from class #58 (pear). The first row contains samples removed from the training set and the second – is kept.

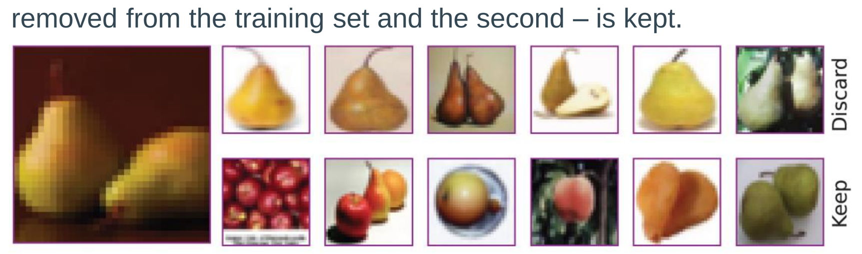


Figure 4. Samples selected via IDS from CIFAR100 from class #58 (pear). The first row contains samples removed from training set, the second – kept. The leftmost image – class prototype.

Proposed Method

DAta REduction with Losses (DAREL) is a two-stage training and fine-tuning acceleration method that is designed to be budget-aware and based on the idea that reducing the number of samples due to a certain rule decreases the number of training steps, thus reducing the overall training time for a CNN and fine-tuning time for LLM. The two stages of the algorithm are called offline (happens before training) and online (happens during training). The hyperparameters of both stages are automatically adapted for the given budget.

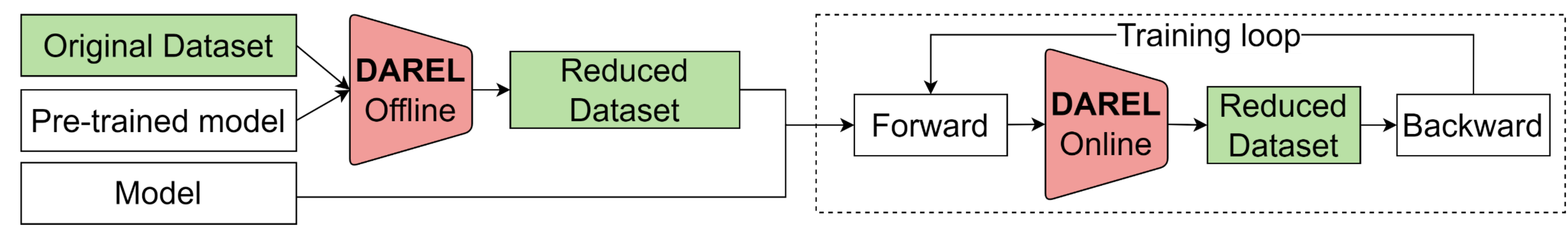


Figure 5. Scheme of Data Reduction with Losses algorithm. For CV pre-trained ResNet18 is utilized as a pre-trained model, for LLM fine-tuning pre-trained model and model are the same models.

Algorithm 1 *DAREL* (online stage)

```
H ← deque(γ): history of losses as double-ended queue of size γ
batch_train ← {}: batch for training
η, η ∈ ℕ, η ≥ 1: number of epochs when loss information is considered actual
losses ← {}: loss history
E: number of epochs to train
ω: number of epoch to start DAREL online
last_update ← 0: number of last epoch when loss history was updated
k ← 1: training step
while epoch i < E do
    for batch batch_train from D do
        if i ≤ ω then
            f_w(k) ← trainBatch(b_train, f_w(k−1))
            k ← k + 1
            batch_train ← {}
            continue
        end if
        if i − last_update < η then
            losses ← f_w(k)(j)
        end if
        for example e from batch_train, loss l from losses do
            H ← H ∪ l
            Φ ← buildHistogram(H): losses histogram representing losses
            p ← PMF(Φ, l): give preference to the most frequent losses
            is_chosen ← choose(p)
            if is_chosen = 1 then
                batch_train ← batch_train ∪ e
            end if
            if |batch_train| = b then
                f_w(k) ← trainBatch(b_train, f_w(k−1))
                batch_train ← {}
            end if
        end for
    end for
end while
```

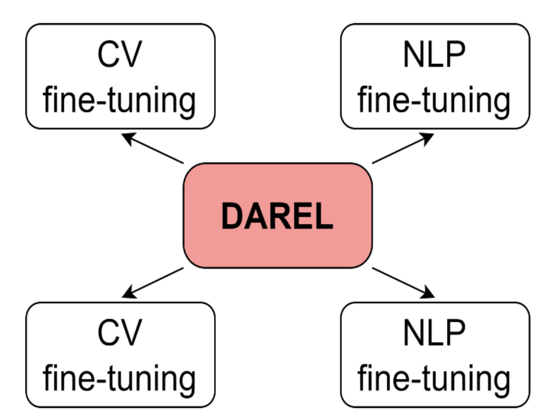


Figure 6. DAREL application areas

Algorithm 2 *DAREL* (offline stage)

```
Mode: mode of selection {EASY, HARD}
α: selection ratio
μ: pre-trained model for a classification task, for example ResNet18
Q = {}: final dataset
for each class D^i in D do
    L^i = |D^i|: number of samples in class
    Θ^i = {inf}: losses for each sample in class D^i
    for each sample j in D^i do
        θ^i_j ← μ(j): calculate loss for each sample
    end for
    O^i ← sort(Θ, Mode): sort samples w.r.t by loss values in ascending order if mode is EASY, descending otherwise
    Q ← Q ∪ {O^i_j : j = 1, 2, ..., ⌈αL^i⌉}: add selected samples to final dataset
end for
```

Budget-aware configuration

The training budget $\mathbb{B}(t,m)$ is defined relatively with t as a ratio of full training time T, so that maximum training time is $t \cdot T$, and also m as a ratio of memory required for full training M, so that maximum memory required is $m \cdot M$. Let D denote a training dataset, E – number of epochs.

$$T_{new} = t_b \cdot N_b \cdot E \qquad (1)$$

$$A = \begin{cases} \frac{T \cdot t}{T_{new} \cdot \alpha}, & \text{if } T \cdot t \geq T_{new} \cdot \alpha \cdot 0.1 \\ 0.1, & \text{otherwise} \end{cases} \qquad (2)$$

The t_b is the time required to train epoch with the size of batch equal to b. T_{new} is an approximation of full training time within a memory budget. The α is the selection ratio of the DAREL offline stage, A is the selection ratio of the online stage, η is the scheduler of the online stage.

$$\alpha = \begin{cases} 0.8, & \text{by default} \\ \frac{T \cdot t}{T_{new} \cdot A}, & \text{if } T \cdot t < T_{new} \cdot 0.8 \cdot A \end{cases} \qquad (3)$$

$$\eta = \begin{cases} 1, & \text{by default} \\ 2, & \text{if } 0.5 < A < 0.8 \\ 3, & \text{if } A \leq 0.5 \end{cases} \qquad (4)$$

Before training starts the A is clipped to range $[0.1, 0.5]$ to guarantee the acceleration.

Evaluation

CV Training

The optimal budget for CV training is defined as $\mathbb{B}(t = 0.8\, m = 0.8)$. Within this budget, DAREL accelerates the ResNet18 training on CIFAR-100 by 25% with a 4.57 p.p. accuracy drop and Hypercomplex ResNet18 by 67% with a 2.91 p.p. drop.

Table 1: Budget-aware training of ResNet18 and Hypercomplex ResNet18 (ResNet18-HC) on CIFAR-100 with DAREL (CPU 3.0GHz 32 cores, 1xGPU 16GB)

Model	Method	Batch	α	A	η	Epoch	Boost, x	Acc. drop, p.p.	Memory cut, x	CO_2e cut, x
ResNet18	$\mathbb{B}(t=1, m=1)$	128	0.8	0.28	3	200	**1.46**	**3.01**	**1.66**	**3.12**
	$\mathbb{B}(t=0.80, m=0.80)$	64	0.8	0.43	3	121	1.25	**4.57**	1.93	2.05
	$\mathbb{B}(t=0.80, m=0.70)$	64	0.8	0.5	2	121	**1.26**	4.97	1.93	1.46
	$\mathbb{B}(t=0.80, m=0.55)$	32	0.8	0.29	3	88	1.25	15.04	**2.03**	**2.71**
	$\mathbb{B}(t=0.70, m=0.80)$	64	0.8	0.43	3	109	1.43	**10.58**	1.93	2.3
	$\mathbb{B}(t=0.70, m=0.70)$	64	0.8	0.5	3	81	1.43	11.3	1.93	1.37
	$\mathbb{B}(t=0.70, m=0.55)$	32	0.8	0.1	3	61	**1.44**	21.53	**2.03**	**5.46**
	$\mathbb{B}(t=0.50, m=0.80)$	64	0.8	0.11	3	78	1.99	11.43	1.93	3.0
	$\mathbb{B}(t=0.50, m=0.70)$	64	0.8	0.11	3	75	2.0	**11.4**	1.95	3.2
	$\mathbb{B}(t=0.50, m=0.55)$	32	0.6	0.1	3	49	**2.03**	32.18	**2.03**	**7.83**
ResNet18-HC	$\mathbb{B}(t=1, m=1)$	128	0.8	0.5	1	200	2.09	3.03	1.00	1.99
	$\mathbb{B}(t=0.80, m=0.80)$	64	0.8	0.5	1	200	1.67	2.91	1.48	1.73
	$\mathbb{B}(t=0.70, m=0.80)$	64	0.8	0.5	2	195	1.60	2.65	1.54	1.72

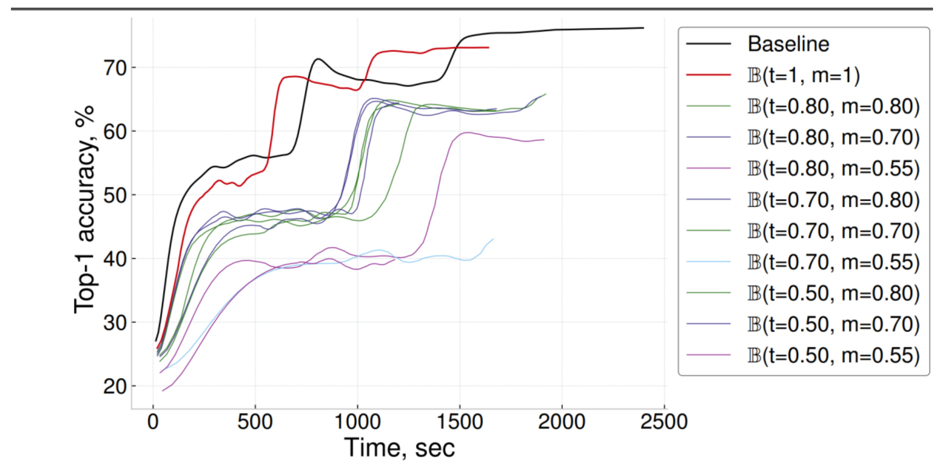


Figure 7. The chart of the accuracy of ResNet18 CIFAR-100 budgeted training with DAREL.

LLM Fine-tuning

DAREL allows acceleration of the LoRA methodology and achieves 1.43x acceleration for GPT2-M fine-tuning with a corresponding increase of BLEU by 1.81 p.p. with a selection parameter equal to 0.7.

Table 2: Fine-tuning experiments of GPT2 family on E2E-NLG with DAREL (CPU 3.0GHz 32 cores, 2xGPU 16GB)

Models	Method	Boost	BLEU↑	TER↓	METEOR↑	NIST↑
GPT2-S	LoRA	-	67.3	66.43	**75.82**	**6.39**
	LoRA + *DAREL* ($\alpha = 0.9$)	1.12	68.22	65.59	73.79	6.06
	LoRA + *DAREL* ($\alpha = 0.8$)	1.26	69.52	**64.92**	74.79	6.09
	LoRA + *DAREL* ($\alpha = 0.7$)	**1.43**	**69.53**	65.62	73.59	5.93
GPT2-M	LoRA	-	65.9	69.36	**79.48**	**6.97**
	LoRA + *DAREL* ($\alpha = 0.9$)	1.11	67.65	68.07	79.37	6.96
	LoRA + *DAREL* ($\alpha = 0.8$)	1.25	**67.71**	**67.54**	78.46	6.93
	LoRA + *DAREL* ($\alpha = 0.7$)	**1.44**	66.03	68.24	77.91	6.81
GPT2-L	LoRA	-	69.93	67.45	**81.73**	7.32
	LoRA + *DAREL* ($\alpha = 0.9$)	1.11	**70.02**	**67.38**	81.68	**7.33**
	LoRA + *DAREL* ($\alpha = 0.8$)	1.24	68.36	68.02	81.02	7.2
	LoRA + *DAREL* ($\alpha = 0.7$)	**1.43**	68.07	68.48	81.01	7.15

Conclusion

Training acceleration for ResNet18 is up to 2.03x and accelerates Hypercomplex ResNet18 by up to 2.09x while for fine-tuning DAREL allows to achieve 1.43x acceleration for GPT2-M fine-tuning with corresponding increase of BLEU by 1.81 p.p. Further directions: NLU support and interoperability with other LLM PEFT methods such as IA^3 and Prompt-Tuning.

Acknowledgements

Authors express special thanks to reviewers for valuable comments and to Samuel Ragland Francis Nadine Suzanne for helpful proofreading.

References

Alexander Demidovskij et al. (2023) "Lightweight and elegant data reduction strategies for training acceleration of convolutional neural networks"
Emma Strubell et al. (2019) "Energy and policy considerations for deep learning in nlp"
Edward J Hu et al. (2021) "Lora: Low-rank adaptation of large language models"