Appendix
[KAKURENBO: Adaptively Hiding Samples in Deep Neural Network Training]

Anonymous Author(s)
Affiliation
Address
email

Appendix A. Proof of Lemma 1

Lemma 1. *Let $F(\mathbf{w}) = \mathbb{E}[f_i(\mathbf{w})]$ be non-convex. Set $\sigma^2 = \mathbb{E}[\|\nabla f_i(\mathbf{w_M})\|^2]$ with $\mathbf{w}^* := argminF(\mathbf{w})$. Suppose $\eta \leq \dfrac{1}{\sup_i L_i}$. Let $\Delta_t = \mathbf{w_t} - \mathbf{w}$. After T iterations, SGD satisfies:*

$$\mathbb{E}\left[\|\Delta_T\|^2\right] \leq (1 - 2\eta\hat{C})^T \|\Delta_0\|^2 + \eta R_\sigma$$

where $\hat{C} = \lambda(1 - \eta \sup_i L_i)$ and $R_\sigma = \dfrac{\sigma^2}{\hat{C}}$.

Proof. $\|\nabla f_i(\mathbf{w})\| = 0$ in the noiseless setting, and so $\sigma := 0$. For $\boldsymbol{x}_k$ being the input at i random index for iteration k, there exists a parameter $\lambda_{\mathbf{w_t}}$ for λ_{max} (Eq. 7), and $w = w_\lambda$, we have for step size γ

$$\begin{aligned}
\mathbb{E}\left[\|\Delta_T\|^2\right] &= \|\boldsymbol{x}_k - \boldsymbol{x}_\star - \gamma\nabla f_i(\boldsymbol{x}_k)\|^2 \\
&= \|(\boldsymbol{x}_k - \boldsymbol{x}_\star) - \gamma(\nabla f_i(\boldsymbol{x}_k) - \nabla f_i(\boldsymbol{x}_\star)) - \gamma\nabla f_i(\boldsymbol{x}_\star)\|^2 \\
&= \|\boldsymbol{x}_k - \boldsymbol{x}_\star\|^2 - 2\gamma \boldsymbol{x}_k - \boldsymbol{x}_\star * \nabla f_i(\boldsymbol{x}_k) + \gamma^2\|\nabla f_i(\boldsymbol{x}_k) - \nabla f_i(\boldsymbol{x}_\star) + \nabla f_i(\boldsymbol{x}_\star)\|^2 \\
&\leq \|\boldsymbol{x}_k - \boldsymbol{x}_\star\|^2 - 2\gamma \boldsymbol{x}_k - \boldsymbol{x}_\star * \nabla f_i(\boldsymbol{x}_k) + 2\gamma^2\|\nabla f_i(\boldsymbol{x}_k) - \nabla f_i(\boldsymbol{x}_\star)\|^2 + 2\gamma^2\|\nabla f_i(\boldsymbol{x}_\star)\|^2 \\
&\leq \|\boldsymbol{x}_k - \boldsymbol{x}_\star\|^2 - 2\gamma \boldsymbol{x}_k - \boldsymbol{x}_\star * \nabla f_i(\boldsymbol{x}_k) \\
&\quad + 2\gamma^2 L_i \boldsymbol{x}_k - \boldsymbol{x}_\star + \nabla f_i(\boldsymbol{x}_k) - \nabla f_i(\boldsymbol{x}_\star) + 2\gamma^2\|\nabla f_i(\boldsymbol{x}_\star)\|^2
\end{aligned}$$

where we employ Jensen's inequality in the first inequality for $\sigma^2 = \mathbb{E}[\|\nabla f_i(\mathbf{w_M})\|^2]$. Then $\nabla f_i(\boldsymbol{x}) = F(\boldsymbol{x})$, and we obtain

$$\begin{aligned}
\mathbb{E}\left[\|\Delta_T\|^2\right] &\leq \|\boldsymbol{x}_k - \boldsymbol{x}_\star\|^2 - 2\gamma \boldsymbol{x}_k - \boldsymbol{x}_\star * F(\boldsymbol{x}_k) + 2\gamma^2\left[L_i \boldsymbol{x}_k - \boldsymbol{x}_\star, \nabla f_i(\boldsymbol{x}_k) - \nabla f_i(\boldsymbol{x}_\star)\right] \\
&\quad + 2\gamma^2\|\nabla f_i(\boldsymbol{x}_\star)\|^2 \\
&\leq \|\boldsymbol{x}_k - \boldsymbol{x}_\star\|^2 - 2\gamma \boldsymbol{x}_k - \boldsymbol{x}_\star * F(\boldsymbol{x}_k) + 2\gamma^2 \sup_i L_i \boldsymbol{x}_k - \boldsymbol{x}_\star, \nabla f_i(\boldsymbol{x}_k) - \nabla f_i(\boldsymbol{x}_\star) \\
&\quad + 2\gamma^2\|\nabla f_i(\boldsymbol{x}_\star)\|^2 \\
&= \|\boldsymbol{x}_k - \boldsymbol{x}_\star\|^2 - 2\gamma \boldsymbol{x}_k - \boldsymbol{x}_\star * F(\boldsymbol{x}_k) + 2\gamma^2 \sup L \boldsymbol{x}_k - \boldsymbol{x}_\star, F(\boldsymbol{x}_k) - F(\boldsymbol{x}_\star) + 2\gamma^2\sigma^2
\end{aligned}$$

Table 1: Datasets and Models Used in Experiments (* Down-stream training using the pre-trained model).

Model	**Dataset**	**#Samples**	**#Epoch**	**#GPUs**	**minibatch (per GPU)**	**Task**
Resnet50 He u. a. (2016)	ImageNet-1K Deng u. a. (2009)	1.2M	100	32	64	Image Classification
EfficientNet-b3 Tan und Le (2019)					32	
WideResNet-28-10 Zagoruyko und Komodakis (2016)	CIFAR-100 Krizhevsky und Hinton (2009)	50K	200	32	32	Image Classification
DeepCAM Kurth u. a. (2018)	DeepCAM Kurth u. a. (2018)	$\sim$ 122K	35	1024	1	Image Segmentation
DeiT-Tiny-224 Touvron u. a. (2021)	Fractal-3K Kataoka u. a. (2022)	3M	80	32	16	Image Classification
	(*) CIFAR-10 Krizhevsky und Hinton (2009)	50K	1000	8	96	
	(*) CIFAR-100 Krizhevsky und Hinton (2009)	50K	1000	8	96	

when $\gamma \leq \frac{1}{\sup L}$. Recursively applying this bound over the first k iterations yields the desired result

$$\begin{aligned}\mathbb{E}\left[\|\Delta_T\|^2\right] &\leq \Big(1-2\gamma\mu(1-\gamma)\Big)^k \|\boldsymbol{x}_0-\boldsymbol{x}_\star\|^2 + 2\sum_{j=0}^{k-1}\Big(1-2\gamma\mu(1-\gamma)\Big)^j \gamma^2\sigma^2 \\ &\leq \Big(1-2\gamma\mu(1-\gamma)\Big)^k \|\boldsymbol{x}_0-\boldsymbol{x}_\star\|^2 + \frac{\gamma\sigma^2}{\mu(1-\gamma)}.\end{aligned}$$

□

Appendix B. Experiments Details

B.1. System detail

We run our experiments on a supercomputer with 1000s of compute nodes, each equipped with 2 Intel Xeon Gold 6148 CPUs, 384 GiB of RAM, 4 NVidia V100 GPUs, and Infiniband EDR NICs (100Gbps×2). We run 4 MPI ranks per compute node so that each rank has a dedicated access to a GPU.

B.2. Model training method details and dataset information:

Table 1 summarizes the models and datasets used in this work. In details, we evaluate KAKURENBO using several models on various datasets as the following:

- **ImageNet-1K** Deng u. a. (2009): We use the subset of the ImageNet dataset containing 1000 classes each containing around 1300 images (1,282,048 images in total). We also test the trained model on the validation set of $50,000$ samples. We train ResNET-50 and EfficientNet-b3 provided by 'torchvision v0.12.0' on ImageNet-1K dataset.
- **CIFAR-10/CIFAR-100** Krizhevsky und Hinton (2009): The CIFAR-10/CIFAR-100 dataset dataset consists of 60,000 colour images. It has 100 categories each containing 600 images. The dataset provides 50,000 training images and 10,000 test images with a size of 32×32 pixels. CIFAR-100 dataset is available at https://www.cs.toronto.edu/ kriz/cifar.html.
- **DeepCAM** Kurth u. a. (2018): DeepCAM dataset for image segmentation, which consists of approximately 122K samples and requires 8.8TBs of storage. We use the settings in Kurth u. a. (2018) to train DeepCAM with the top learning rate of 0.0055.
- **Fractal-3K** Kataoka u. a. (2022) A rendered dataset from the Visual Atom method Kataoka u. a. (2022). Fractal-3K dataset comprise of 3 million images of visual atoms, where the number of classes is C = 3000 and the number of images per class is N = 1000. We train the DeiT-Tiny-224 model on Fractal-3K dataset and fine tune it with CIFAR-10 and CIFAR-100 datasets.

Table 2: Hyper-parameters used for different training in the paper and the baseline top-1 testing accuracy. We also considers different hyper-parameters for ResNet-50 model on ImageNet-1K dataset.

	ImageNet-1K				**CIFAR-100**	**Fractal-3K**	**CIFAR-10**	**CIFAR-100**
	ResNet-50	**ResNet-50 (A)**	**ResNet-50 (B)**	**EfficientNet-b3**	**WideResNet-28-10**	**DeiT-Tiny-224**		
Train Res	224	224	224	224	32	224	224	224
Test Res	232	224	232	224	32	-	32	32
Epochs	600	100	600	100	200	80	1000	1000
Number of workers	32	32	32	32	32	32	8	8
Batch size	2048	1024	1024	1024	1024	512	768	768
Optimizer	SGD	SGD	SGD	SRMSProp	SGD	adamw	SGD	SGD
Momentum	0.9	0.9	0.9	0.9	0.9	-	0.9	0.9
LR	0.11	0.0125	0.125	0.01	0.025	0.001	0.01	0.01
Weight decay	1e-5	5e-5	2e-5	5e-5	5e-4	0.05	1e-4	1e-4
LR decay	cosineLR	step	cosineAnnealing	step	step	Cosine_iter	Cosine_iter	Cosine_iter
Decay rate	-	0.1	-	0.9	0.2	-	-	-
Decay epochs	-	[30, 60, 80]	-	2	[60, 120, 160]	-	-	-
Warmup epochs	5	5	5	5	1	5	5	5
Warmup method	linear	linear	linear	linear	linear	linear	linear	linear
Label Smoothing	0.1	-	-	-	-	-	0.1	0.1
H.flip	YES	YES	YES	YES	YES	YES	YES	YES
Erasing prob.	0.1	-	0.1	-	-	0.5	0.5	0.5
Auto augument	ta_wide	-	ta_wide	-	-	rand-m9-mstd0.5-inc1		
Interpilation	bilinear	-	bilinear	-	-	bicubic	bicubic	bicubic
Train crop	176	-	176	-	-	224	224	224
Test crop	224	-	224	-	-	-	-	-
EMA	YES	-	-	-	-	-	-	-
EMA steps	32	-	-	-	-	-	-	-
EMA decay	0.99998	-	-	-	-	-	-	
Loss	Cross Entropy	Cross Entropy	Cross Entropy	Cross Entropy	Cross Entropy	Cross Entropy	Soft Target Cross Entropy	
Baseline acc.	74.89	73.68	76.58	76.63	77.49	-	95.03	79.69
Max fraction	0.3	0.3	0.3	0.3	0.3	0.3	-	-
Max fraction decay	[1, 0.8, 0.6]	[1, 0.8, 0.6]	[1, 0.8, 0.6]	[1, 0.8, 0.6]	[1, 0.8, 0.6]	[1, 0.8, 0.6]	-	-
Fraction decay epoch	[200, 400, 600]	[30, 60, 80]	[200, 400, 600]	[30, 60, 80]	[60, 120, 160]	[30, 60, 80]	-	-
KAKURENBO acc.	75.15	73.52	76.62	76.23	77.21	-	95.28	79.35

B.3. Hyper-parameters

It is worth noting that we follow the hyper-parameters reported in Vryniotis for training ResNet-50, Zagoruyko und Komodakis (2016) for training WideResNet-28-10, Tan und Le (2019) for training EfficientNet-b3, and Kurth u. a. (2018) for DeepCAM. We also use the setting in Kataoka u. a. (2022) for both pretrain and finetune tasks in Fractal-3K. Table 2 shows the detail of our hyper-parameters. Specifically, We follow the guideline of 'TorchVision' to train the ResNet-50 that uses the CosineLR learning rate scheduler [1], auto augments, and random erasing, etc Vryniotis. We also set the weight decay to $1e-05$ and crop the input image to 176×176 pixels and train for a long number of epochs, i.e., 600 (The ResNet-50 setting). We train the WideResNet-28-4 on the CIFAR-100 dataset in 200 epochs following the setting in Zagoruyko und Komodakis (2016). Specifically, we use the base learning rate of $0.025 \times k$, momentum 0.9, and weight decay 0.0005. For EfficientNet-b3, we use RMSProp optimizer with momentum 0.9; batch norm momentum 0.99 weight decay $1e-5$ (following Tan und Le (2019)). We use an initial learning rate of 0.016 that decays by 0.9 every 2 epochs. We set the minibatch size per worker (GPU) to b, e.g., the global batch size of $b \times p$ in the case of p GPUs. The minibatch size per GPU and the number of GPUs in each experiments are shown in Table 1.

To show the robustness of KAKURENBO, we also train ResNet-50 with different settings, e.g., marked as (A) and (B) in the Table 2 and discuss the result in Appendix C.3. For example, in ResNet-50 (A) setting, we follow the hyper-parameters reported in Goyal u. a. (2017). Specifically, we use using the Stochastic Gradient Descent (SGD) optimizer with a Nesterov momentum of 0.9 and weight decay of 0.00005. We trained all the models for 100 epochs and apply the linear scaling rule with the base learning rate of $0.0125 \times k$ where k is the number of workers. We reduce the learning rate by 0.1 at the 30th, 60th, and 80th epoch. We gradual warmup which starts with 0 and is

[1]implemented by timm https://github.com/huggingface/pytorch-image-models/tree/main/timm

linearly increased to the base learning rate over 5 epochs. We also use scale and aspect ratio data augmentation.The input image is a 224×224 pixel random crop from an augmented image or its horizontal flip.

B.4. Implementation

It is worth noting that KAKURENBO merely hides samples before the input pipeline. As a result, KAKURENBO can be easily implemented with simple extensions to PyTorch and TensorFlow implementations[2]. Using KAKURENBO with new models and datasets can be added to any training code by indicating so in the model launch parameters.

Appendix C. Ablation Studies

C.1. Analysis of the Factors Affecting KAKURENBO's Performance

In this section, we present an analysis of the factors affecting KAKURENBO's performance, e.g., the lagging loss and the prediction confidence.

Figure 1 shows the histogram of the loss as the number of epochs increases when training ResNet-50 (A) on the ImageNet-1K dataset. At the first few epochs, the histogram of the loss follows a Gaussian distribution. As the number of epochs increases, the number of samples with small loss increases significantly. For example, starting from epoch 30, more than 50% of the samples have a loss which is lower than 5% of the highest loss. As a result, there is an increase in the number of samples that provide about the same absolute contribution to the update, e.g., in the latter epochs. Hiding a fraction of (fixed) F samples during training in this case may lead to a relatively higher negative impact on the accuracy than that at some early epochs. Thus, we reduce the maximum hidden fraction at the epoch number increases (as mentioned in Section 3.4 in the main manuscript).

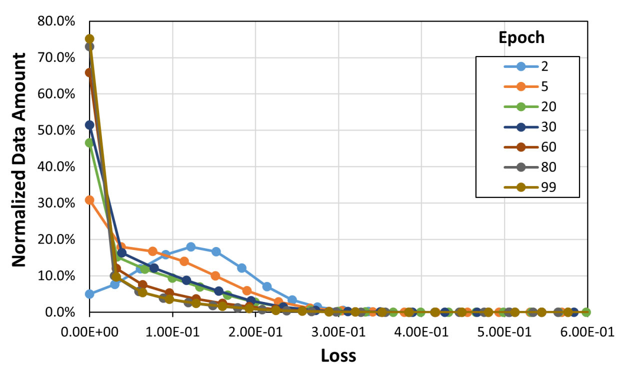


Figure 1: Histogram of the lagging-loss as the number of epoch increases during training (ResNet-50 w/ ImageNet-1K).

In addition, as the number of samples with the same absolute loss increases, there is a high probability that samples classified as important are in fact unimportant. To this end, we propose moving back samples from the hidden set based on their prediction confidence score (as per Section 3.2). Unlike the ahead-of-time method proposed by the authors in Toneva u. a. (2019), instead of computing the loss of all the samples before training and selecting samples to be removed from the training process, we compute the loss of the samples on the fly. With this method, at each epoch, a dynamic hiding fraction F^* is applied. Figure 2 shows the number of hidden samples of each class in KAKURENBO (ResNet-50, ImageNet-1K). The figure shows the result of the first 50 classes. The number on top of each column shows the rank over 1000 classes (a lower rank indicates a higher number of hidden samples). The result shows that our method could dynamically hide the samples at each epoch. For example, fewer samples in the class 25 are hidden while more and more samples in class 13 are selected to hide as epochs increase.

[2] Our PyTorch implementation is available at https://anonymous.4open.science/r/kakurenbo-8F10/

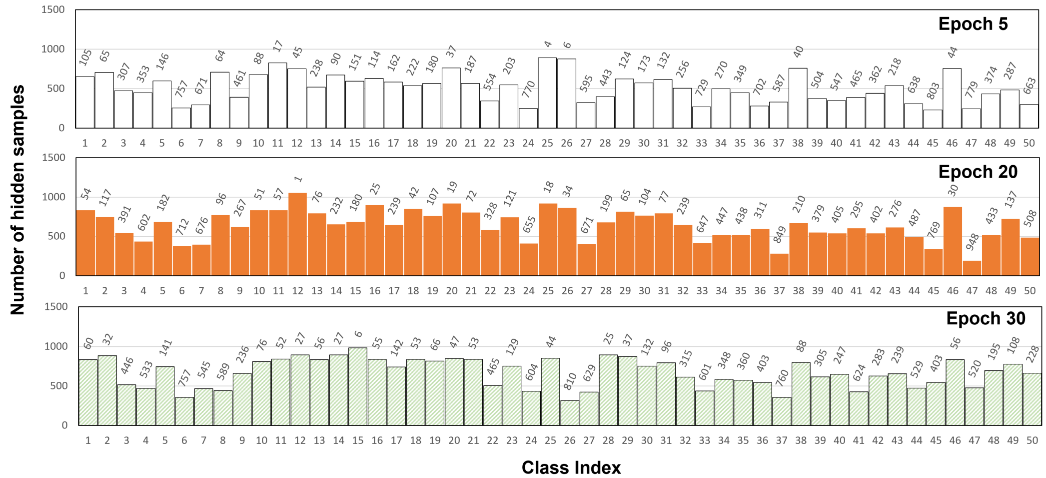


Figure 2: Number of hidden samples of each class in KAKURENBO (ResNet-50, ImageNet-1K). The figure shows the result of the first 50 classes. The number on top of each column shows the rank over 1000 classes (a lower rank indicates a higher number of hidden samples).

C.2. Evolution of the Hiding Fraction

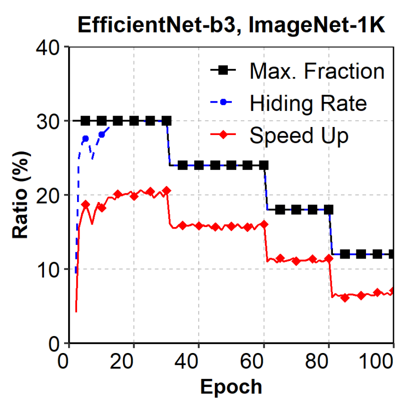


Figure 3: Reduction of hiding fraction, per epoch, and the resulting speedup.

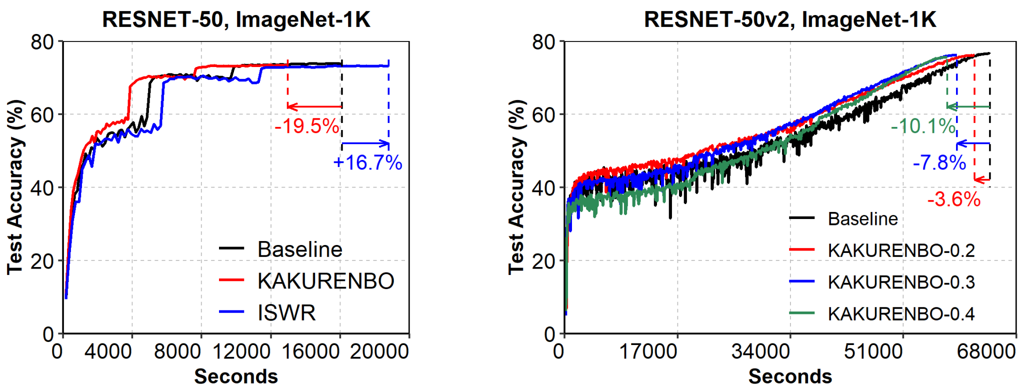
Figure 3 shows how KAKURENBO adapts the size of the hidden set during the training of EfficientNet-b3. At the beginning of the training, the maximum hiding fraction is set to 30 %. This fraction is progressively reduced after a few epochs followed by our fraction adjustment rule. The figure also reports the effective proportion of samples that are hidden at each epoch (*Hiding rate* in the Figure). As described in Section 3 in the main manuscript, KAKURENBO first cuts a part of the dataset before moving back samples that are mispredicted or correctly predicted but with low confidence. Figure 3 shows that the moving back strategy mostly impacts the beginning of the training when the model is still inaccurate.

Figure 3 also reports the measured speedup per epoch as compared to the baseline epoch duration. The speedup follows the same trend as the hiding rate. This is because reducing the number of samples in the training set impacts the speed of the training. The measured speedup does not reach the maximum hiding rate because of additional hidden sample selection and due to the need for computing the forward pass on samples in the hidden list.

C.3. Robustness of Our Method

Table 3: Test accuracy (Top-1) in percentage and total training time in seconds of KAKURENBO in the comparison with those of the baseline.

Setting	ResNet-50(A) + ImageNet-1K		ResNet-50(B) + ImageNet-1K	
	Accuracy	**Time** (sec)	**Accuracy**	**Time** (sec)
Baseline	73.68	16118	76.58	64060
KAKURENBO-0.2	-	-	76.11	61723
KAKURENBO-0.3	73.52	12984	76.17	59063
KAKURENBO-0.4	-	-	75.62	57582



(a) ResNet-50 (A)

(b) ResNet-50 (B)

Figure 4: Convergence and speedup of KAKURENBO with different setting of ResNet-50.

In this section, we demonstrate the robustness of KAKURENBO with different settings during training, e.g. (1) when using different techniques to improve accuracy and (2) the batch size is changed.

We first measure the robustness of KAKURENBO when using SoTA techniques in training, the ResNet-50 (A) and (B) described in Table 2. The result in Figure 4 and Table 3 show that our proposed method is also stable with different learning techniques. For example, KAKURENBO could reduce the total training time to 19.5% (7.8%) with only 0.2% (0.41) percent of accuracy reduction when the maximum hidden fraction is set to 30% for RESNET-50 (A) and (B), respectively.

Table 4: Test accuracy (Top-1) in percentage of KAKURENBO in comparison with those of the baseline when the batch size changes.

Setting	ResNet-50 (A) + ImageNet-1K			
#GPUs	32	64	128	256
Batch size	1024	2048	4096	8192
Baseline	73.68	73.98	73.59	73.81
KAKURENBO-0.4	73.60	73.21	73.03	72.84

We now fix the mini-batch size per worker to 32 and then increase the number of workers (GPUs), i.e., we increase the global batch size in the case of ResNet-50 (A). Table 4 shows the top-1 testing accuracy of ResNet-50 (A) on the ImageNet-1K dataset when the batch size changes from 1024 to 8192. The result shows that KAKURENBO can maintain the accuracy (or with a trivial reduction of accuracy) even with large batch sizes. KAKURENBO could help with large-scale training which has become common when training DL models on a large supercomputer or cluster.

Table 5: The impact of different components of KAKURENBO on testing accuracy (Resnet-50 (A), ImageNet-1K, $F = 0.4$) including **HE**: Hiding $F\%$ lowest-loss examples, **MB**: Moving Back, **RF**: Reducing the Fraction by epoch, **LR**: Adjusting Learning Rate. Numbers inside the (.) indicate the gap in percentage compared to the full version of KAKURENBO.

	Component				Accuracy
	HE	MB	RF	LR	
Baseline	×	×	×	×	73.68
v1000	✓	×	×	×	72.25 (-1.8%)
v1001	✓	×	×	✓	73.08 (-0.7%)
v1010	✓	×	✓	×	72.81 (-1.1%)
v1011	✓	×	✓	✓	73.27 (-0.4%)
v1100	✓	✓	×	×	72.37 (-1.7%)
v1101	✓	✓	×	✓	73.09 (-0.7%)
v1110	✓	✓	✓	×	72.96 (-0.9%)
KAKUR. (v1111)	✓	✓	✓	✓	73.6

C.4. Impact of different components of KAKURENBO

We evaluate how KAKURENBO's individual internal strategies, and their combination, affect the testing accuracy of a neural network. Table 5 reports the results we obtained when training ResNet-50 on ImageNet-1K with a maximum hiding fraction of 40% . The results show that when only HE (Hiding Examples) of the 40% lowest loss samples is performed, accuracy slightly degrades. Combining HE with other strategies, namely MB (Move-Back), RF (Reducing Fraction), and LR (Learning Rate adjustment) gradually improves testing accuracy. In particular, all combinations with RF achieve higher accuracy than the ones without it. For example, the accuracy of v1110 is higher than that of v1100 by about 0.59%. We also observe that using LR helps to improve the training accuracy by a significant amount, i.e., from 0.46% to 0.83%. The MB strategy also improves accuracy. For example, the accuracy of v1010 is 72.81%, compared to v1110 which is 72.96%. This small impact of MB on the accuracy is due to moving back samples at the beginning of the training, as seen in Appendix C.3. By using all the strategies, KAKURENBO achieves the best accuracy of 73.6%, which is very close to the baseline of 73.68%.

Appendix D. Discussion on DeepCAM

We have shown how KAKURENBO's internal strategies, and their combination, affect the testing accuracy of a neural network in the case of ResNet-50 and the ImageNet-1K dataset. Figure 5 presents the same result on the DeepCAM dataset. In this experiment, we evaluate two combinations: **v1000** and **v1001**. For **v1000** we hide $F\%$ lowest-loss samples only (Hiding Example or HE for short). For **v1001** we combine HE and learning rate adjustment techniques. It is worth noting that our proposed method, **KAKURENBO**, is the combination of HE, LR, MB (Moving Back sample), and FR (Reducing the Fraction by epoch). The result with different maximum hidden fractions, e.g. F from 0.2 to 0.4, shows that using LR helps to improve the training accuracy by a significant amount, and KAKURENBO achieves the best accuracy which is very close to the baseline. This result is similar to what we observed with ResNet-50 and the ImageNet-1K dataset.

For DeepCAM, we also observed that the loss of the samples with the highest loss does not decrease significantly during the last few epochs of training and remain substantially above the rest of other samples. Those samples may be hard to learn or represent noise in the data. Figure 6 demonstrates this phenomena showing the loss distributions of the full, bottom 98% and top 2% of the dataset according to the loss values, respectively. As seen, the top 2%'s loss distribution remains high until the very last epoch.

This observation motivated us to consider a version in which we cut 2% of the highest-loss samples at each epoch (DropTop). Interestingly, it helps to improve the testing accuracy of DeepCAM, e.g., from 77.16% in KAKURENBO to 77.37% with a maximum fraction of 0.3. For version v1001, Droptop increases the accuracy by 0.82%.

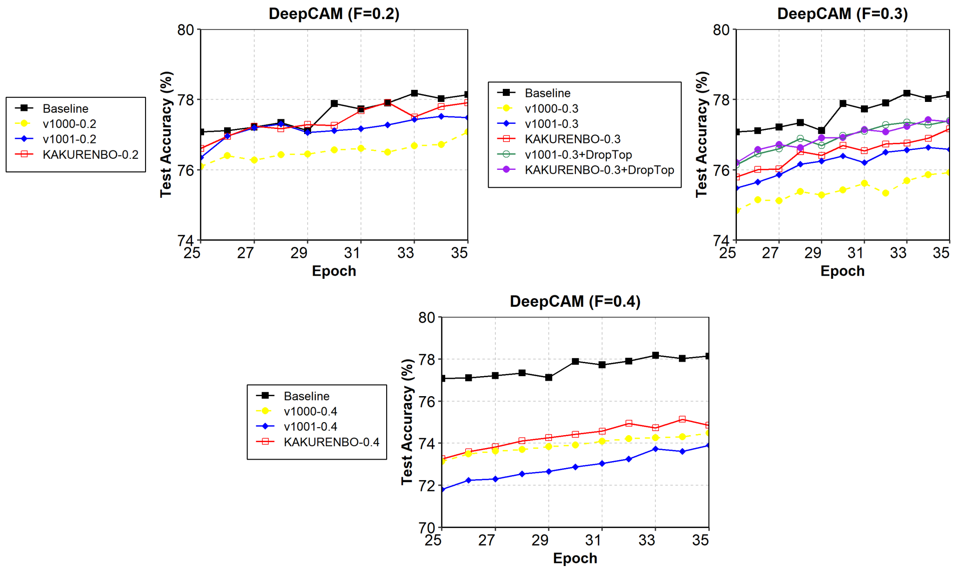


Figure 5: The impact of different components of KAKURENBO on testing accuracy (DeepCAM). **v1000**: Hiding $F\%$ lowest-loss samples only (HE). **v1001**: HE + LR (Adjusting Learning Rate). **KAKURENBO**: our proposed method with HE + LR + MB (Moving Back) + FR (Reducing the Fraction by epoch). We also consider the version in which we cut 2% of the highest-loss samples at each epoch (DropTop).

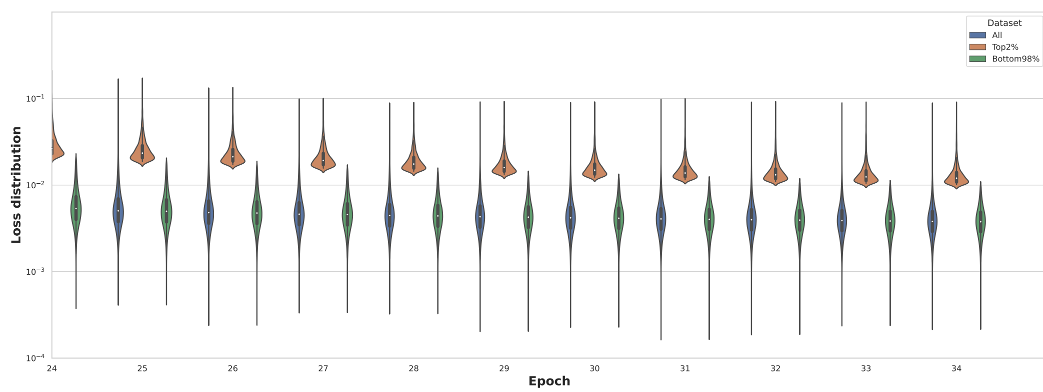


Figure 6: Loss distributions of DeepCAM training samples (full dataset, bottom 98% and top 2%) in the last 10 epoch of training.

Appendix E. Related work

As the size of training datasets and the complexity of deep-learning models increase, the cost of training neural networks becomes prohibitive. Several approaches have been proposed to reduce this training cost without degrading accuracy significantly.

Biased with-Replacement Sampling has been proposed as a method to improve the convergence rate in SGD training Katharopoulos und Fleuret (2018); Mindermann u. a. (2022). Importance sampling is based on the observation that not all samples are of equal *importance* when it comes to training, and accordingly replaces the regular uniform sampling used to draw samples from datasets with a biased sampling function that assigns a likelihood to a sample being drawn proportional to its importance; the more important the sample is, the higher the likelihood it would be selected. The with-replacement strategy of importance sampling maintains the total number of samples the network trains on.

Several improvements over importance sampling have been proposed. Reducible Holdout Loss Selection (RHO-LOSS) Mindermann u. a. (2022) is a selection function that quantifies by how much each sample would reduce the loss on unseen data had it been trained on. Mercury uses an importance-aware data sharding technique in order to speed up distributed training Zeng u. a. (2021). It distributes important samples across workers between iterations. This allows important samples to be uniformly distributed between workers, and it reduces the number of samples to communicate for each epoch since non-important samples are kept local.

The importance of a sample can be estimated with several methods. In Wu u. a. (2017), authors use distance weighted sampling to determine the importance of samples. Zhao und Zhang (2015) uses stochastic optimization to reduce the stochastic variance. Allen-Zhu u. a. (2016) selects each coordinate with a probability proportional to the square root of its smoothness parameter (applied to accelerated coordinate descent). RAIS Johnson und Guestrin (2018) proposes approximating the ideal sampling distribution, which introduces little computational overhead.

Overall, biased with-replacement sampling aims at increasing the convergence speed of SGD by focusing on samples that induce a measurable change in the model parameters, which would allow a reduction in the number of epochs. While these techniques promise to converge in fewer epochs on the whole dataset, each epoch requires computing the importance of samples which is time-consuming; and the actual speedup in terms of time-to-solution remains unclear. Moreover, existing studies Katharopoulos und Fleuret (2018); Mindermann u. a. (2022); Zeng u. a. (2021) only evaluate small datasets. Our experiments show that the biased with-replacement, importance sampling Katharopoulos und Fleuret (2018), the algorithm does not speedup the training when applied to large-scale datasets (demonstrated in the evaluation section in the paper).

Data Pruning techniques are used to reduce the size of the dataset by removing less important samples. Pruning the dataset requires training on the full dataset and adds significant overheads for quantifying individual differences between data points Sorscher u. a. (2022). However, the assumption is that the advantage would be a reduced dataset that replaces the original datasets when used by others to train. Several studies investigate the selection of the samples to discard from a dataset. In Toneva u. a. (2019), authors detect unforgettable samples that are correctly classified during the course of training. EL2N Paul u. a. (2021) uses the loss gradient norm of samples to identify the important ones and prune the unimportant samples from the dataset after a few epochs. While this work does not require fully training the model before pruning, it remains unclear if EL2N reduces the total training time. Another work uses memorization to identify outliers or mislabeled samples in a given dataset Feldman und Zhang (2020). Removing these atypical samples accelerates the training without altering the trained model accuracy. Ensemble Active Learning Chitta u. a. (2021) trains an ensemble of networks and uses ensemble uncertainty to identify which samples are hard to learn. They manage to reduce the ImageNet dataset by 20% without degrading the accuracy of the trained model, but again, their method is prohibitive for models and datasets that require excessive resources for training.

Pruning the dataset does reduce the training time without significantly degrading the accuracy Toneva u. a. (2019); Feldman und Zhang (2020). However, these techniques require fully training the model on the whole dataset to identify the samples to be removed, which is compute intensive. While most of the proposed solutions perform well on small datasets such as CIFAR, many fail to maintain accuracy on larger datasets like ImageNet Sorscher u. a. (2022).

Selective-Backprop Jiang u. a. (2019) combines importance sampling and online data pruning. It reduces the number of samples to train on by using the output of each sample's forward pass to estimate the sample's importance and cuts a fixed fraction of the dataset at each epoch. While this method shows notable speedups, it has been evaluated only on tiny datasets without providing any measurements on how accuracy is impacted. In addition, the authors allow up to 10% reduction in test error in their experiments. EIF Wu u. a. (2020) is similar to Selective-Backprop: it reduces the computation cost of training by filtering out the samples with the lowest loss. E^2-Train Wang u. a. (2019) shows that the combination of randomly dropping samples during training with selective layer update in CNNs can significantly reduce the training time, while slightly degrading the accuracy. However, E^2-Train targets edge environments and is evaluated only on very small datasets.

GRAD-MATCH Killamsetty u. a. (2021) is an online method that selects a subset of the samples that would minimize the gradient matching error, where the error of the gradients of a matched subset samples (and their weights) becomes minimum. To avoid the impractical storing and computation of the optimization of the gradients of all instances, the authors approximate the gradients by only using the gradients of the last layer, use a per-class approximation, and run data selection every R epochs, in which case, the same subsets and weights will be used between epochs. The infrequent selection, however, means the model is limited in its capacity to learn in intermediate epochs - where selection occurs - since it trains on the same limited subset of samples. This often leads to a longer numbers of epochs needed to converge to the same validation accuracy that can be achieved by the baseline or the baseline reaching much higher accuracy Pooladzandi u. a. (2022). Another important point worth mentioning is that GRAD-MATCH is impractical in distributed training, which is a de facto requirement in large dataset and models (e.g., the DeepCAM model/dataset). That is since the approximation of the classes would require very expensive high-volume collective communication operations to gather the gradients scattered across different samples belonging to the same class. The communication cost would be O($N.R.G$) where N is the number of samples, R is the frequency of selection, and G is the gradients (of the last layer, if gradient approximation is to be used). Distributed GRAD-MATCH would require a scatter communication to collect the class approximations and a collective all-reduce of the gradients to then do the matching optimization. This is practically a very high cost for communication per epoch that could even exceed the average time per epoch. Finally, the mini-batch variant of GRAD-MATCH can only be effective for small mini-batches. However, since in distributed training the mini-batch grows with the scale (i.e., the mini-batch aggregates the local mini-batch of all workers), the cost of communication amplifies by B (where B is mini-batch size).

References

[Allen-Zhu u. a. 2016] ALLEN-ZHU, Zeyuan ; QU, Zheng ; RICHTÁRIK, Peter ; YUAN, Yang: Even faster accelerated coordinate descent using non-uniform sampling. In: *International Conference on Machine Learning* PMLR (Veranst.), 2016, S. 1110–1119 9

[Chitta u. a. 2021] CHITTA, Kashyap ; ÁLVAREZ, José M ; HAUSSMANN, Elmar ; FARABET, Clément: Training data subset search with ensemble active learning. In: *IEEE Transactions on Intelligent Transportation Systems* (2021) 9

[Deng u. a. 2009] DENG, Jia ; DONG, Wei ; SOCHER, Richard ; LI, Li-Jia ; LI, Kai ; FEI-FEI, Li: Imagenet: A large-scale hierarchical image database. In: *Conf. on comp. vision and pattern recogn.* IEEE (Veranst.), 2009, S. 248–255 2

[Feldman und Zhang 2020] FELDMAN, Vitaly ; ZHANG, Chiyuan: What Neural Networks Memorize and Why: Discovering the Long Tail via Influence Estimation. In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020 (NIPS'20). – ISBN 9781713829546 9

[Goyal u. a. 2017] GOYAL, Priya ; DOLLÁR, Piotr ; GIRSHICK, Ross ; NOORDHUIS, Pieter ; WESOLOWSKI, Lukasz ; KYROLA, Aapo ; TULLOCH, Andrew ; JIA, Yangqing ; HE, Kaiming: Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour. In: *CoRR* abs/1706.02677 (2017) 3

[He u. a. 2016] HE, Kaiming ; ZHANG, Xiangyu ; REN, Shaoqing ; SUN, Jian: Deep residual learning for image recognition. In: *CVPR*, 2016, S. 770–778 2

[Jiang u. a. 2019] JIANG, Angela H. ; WONG, Daniel L. K. ; ZHOU, Giulio ; ANDERSEN, David G. ; DEAN, Jeffrey ; GANGER, Gregory R. ; JOSHI, Gauri ; KAMINKSY, Michael ; KOZUCH, Michael ; LIPTON, Zachary C. ; PILLAI, Padmanabhan: *Accelerating Deep Learning by Focusing on the Biggest Losers*. 2019. – URL `https://arxiv.org/abs/1910.00762` 9

[Johnson und Guestrin 2018] JOHNSON, Tyler B. ; GUESTRIN, Carlos: Training deep models faster with robust, approximate importance sampling. In: *Advances in Neural Information Processing Systems* 31 (2018) 9

[Kataoka u. a. 2022] KATAOKA, Hirokatsu ; HAYAMIZU, Ryo ; YAMADA, Ryosuke ; NAKASHIMA, Kodai ; TAKASHIMA, Sora ; ZHANG, Xinyu ; MARTINEZ-NORIEGA, Edgar J. ; INOUE, Nakamasa ; YOKOTA, Rio: Replacing Labeled Real-Image Datasets With Auto-Generated Contours. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, S. 21232–21241 2, 3

[Katharopoulos und Fleuret 2018] Katharopoulos, Angelos ; Fleuret, François: Not all samples are created equal: Deep learning with importance sampling. In: *International conference on machine learning* PMLR (Veranst.), 2018, S. 2525–2534 8, 9

[Killamsetty u. a. 2021] Killamsetty, Krishnateja ; Durga, S ; Ramakrishnan, Ganesh ; De, Abir ; Iyer, Rishabh: Grad-match: Gradient matching based data subset selection for efficient deep model training. In: *International Conference on Machine Learning* PMLR (Veranst.), 2021, S. 5464–5474 9

[Krizhevsky und Hinton 2009] Krizhevsky, A. ; Hinton, G.: Learning multiple layers of features from tiny images. In: *Master's thesis, Dep. of Comp. Sci. Univ. of Toronto* (2009) 2

[Kurth u. a. 2018] Kurth, Thorsten ; Treichler, Sean ; Romero, Joshua ; Mudigonda, Mayur ; Luehr, Nathan ; Phillips, Everett ; Mahesh, Ankur ; Matheson, Michael ; Deslippe, Jack ; Fatica, Massimiliano u. a.: Exascale deep learning for climate analytics. In: *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis* IEEE (Veranst.), 2018, S. 649–660 2, 3

[Mindermann u. a. 2022] Mindermann, Sören ; Brauner, Jan M. ; Razzak, Muhammed T. ; Sharma, Mrinank ; Kirsch, Andreas ; Xu, Winnie ; Höltgen, Benedikt ; Gomez, Aidan N. ; Morisot, Adrien ; Farquhar, Sebastian ; Gal, Yarin: Prioritized Training on Points that are Learnable, Worth Learning, and not yet Learnt. In: *Proceedings of the 39th International Conference on Machine Learning*, PMLR, Jun 2022, S. 15630–15649. – ISSN 2640-3498 8, 9

[Paul u. a. 2021] Paul, Mansheej ; Ganguli, Surya ; Dziugaite, Gintare K.: Deep Learning on a Data Diet: Finding Important Examples Early in Training. In: Ranzato, M. (Hrsg.) ; Beygelzimer, A. (Hrsg.) ; Dauphin, Y. (Hrsg.) ; Liang, P.S. (Hrsg.) ; Vaughan, J. W. (Hrsg.): *Advances in Neural Information Processing Systems* Bd. 34, URL https://proceedings.neurips.cc/paper/2021/file/ac56f8fe9eea3e4a365f29f0f1957c55-Paper.pdf, 2021, S. 20596–20607 9

[Pooladzandi u. a. 2022] Pooladzandi, Omead ; Davini, David ; Mirzasoleiman, Baharan: Adaptive second order coresets for data-efficient machine learning. In: *International Conference on Machine Learning* PMLR (Veranst.), 2022, S. 17848–17869 10

[Sorscher u. a. 2022] Sorscher, Ben ; Geirhos, Robert ; Shekhar, Shashank ; Ganguli, Surya ; Morcos, Ari S.: *Beyond neural scaling laws: beating power law scaling via data pruning*. 2022. – URL https://arxiv.org/abs/2206.14486 9

[Tan und Le 2019] Tan, Mingxing ; Le, Quoc: EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In: Chaudhuri, Kamalika (Hrsg.) ; Salakhutdinov, Ruslan (Hrsg.): *Proceedings of the 36th International Conference on Machine Learning* Bd. 97, PMLR, 09–15 Jun 2019, S. 6105–6114. – URL https://proceedings.mlr.press/v97/tan19a.html 2, 3

[Toneva u. a. 2019] Toneva, Mariya ; Sordoni, Alessandro ; Combes, Remi T. des ; Trischler, Adam ; Bengio, Yoshua ; Gordon, Geoffrey J.: An Empirical Study of Example Forgetting during Deep Neural Network Learning. In: *7th International Conference on Learning Representations (ICLR)*, URL https://openreview.net/forum?id=BJlxm30cKm, 2019 4, 9

[Touvron u. a. 2021] Touvron, Hugo ; Cord, Matthieu ; Douze, Matthijs ; Massa, Francisco ; Sablayrolles, Alexandre ; Jegou, Herve: Training data-efficient image transformers & distillation through attention. In: Meila, Marina (Hrsg.) ; Zhang, Tong (Hrsg.): *Proceedings of the 38th International Conference on Machine Learning* Bd. 139, PMLR, 18–24 Jul 2021, S. 10347–10357. – URL https://proceedings.mlr.press/v139/touvron21a.html 2

[Vryniotis] Vryniotis, Vasilis: *How to Train State-Of-The-Art Models Using TorchVision's Latest Primitives*. – URL https://pytorch.org/blog\/how-to-train-state-of-the-art-models\-using-torchvision-latest-primitives/ 3

[Wang u. a. 2019] WANG, Yue ; JIANG, Ziyu ; CHEN, Xiaohan ; XU, Pengfei ; ZHAO, Yang ; LIN, Yingyan ; WANG, Zhangyang: E2-train: Training state-of-the-art cnns with over 80% energy savings. In: *Advances in Neural Information Processing Systems* 32 (2019) 9

[Wu u. a. 2017] WU, Chao-Yuan ; MANMATHA, R ; SMOLA, Alexander J. ; KRAHENBUHL, Philipp: Sampling matters in deep embedding learning. In: *Proceedings of the IEEE International Conference on Computer Vision*, 2017, S. 2840–2848 9

[Wu u. a. 2020] WU, Yawen ; WANG, Zhepeng ; SHI, Yiyu ; HU, Jingtong: Enabling on-device cnn training by self-supervised instance filtering and error map pruning. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39 (2020), Nr. 11, S. 3445–3457 9

[Zagoruyko und Komodakis 2016] ZAGORUYKO, Sergey ; KOMODAKIS, Nikos: Wide Residual Networks. In: *BMVC*, September 2016, S. 87.1–87.12 2, 3

[Zeng u. a. 2021] ZENG, Xiao ; YAN, Ming ; ZHANG, Mi: Mercury: Efficient On-Device Distributed DNN Training via Stochastic Importance Sampling. In: *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, 2021, S. 29–41 9

[Zhao und Zhang 2015] ZHAO, Peilin ; ZHANG, Tong: Stochastic optimization with importance sampling for regularized loss minimization. In: *international conference on machine learning* PMLR (Veranst.), 2015, S. 1–9 9