Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 264 (2025) 270–279

Procedia Computer Science

www.elsevier.com/locate/procedia

International Neural Network Society Workshop on Deep Learning Innovations and Applications

One Period to Rule Them All: Identifying Critical Learning Periods in Deep Networks

Vinicius Yuiti Fukase[a,*], Heitor Gama[a,*], Barbara Bueno[a], Lucas Libanio[a], Anna Helena Reali Costa[a], Artur Jordao[a]

[a]*Escola Politécnica, Universidade de São Paulo, São Paulo, SP 05508-010, Brazil*

Abstract

Critical Learning Periods comprehend an important phenomenon involving deep learning, where early epochs play a decisive role in the success of many training recipes, such as data augmentation. Existing works confirm the existence of this phenomenon and provide useful insights. However, the literature lacks efforts to precisely identify when critical periods occur. In this work, we fill this gap by introducing a systematic approach for identifying critical periods during the training of deep neural networks, focusing on eliminating computationally intensive regularization techniques and effectively applying mechanisms for reducing computational costs, such as data pruning. Our method leverages generalization prediction mechanisms to pinpoint critical phases where training recipes yield maximum benefits to the predictive ability of models. By halting resource-intensive recipes beyond these periods, we significantly accelerate the learning phase and achieve reductions in training time, energy consumption, and CO_2 emissions. Experiments on standard architectures and benchmarks confirm the effectiveness of our method. Specifically, we achieve significant milestones by reducing the training time of popular architectures by up to 59.67%, leading to a 59.47% decrease in CO_2 emissions and a 60% reduction in financial costs, without compromising performance. Our work enhances understanding of training dynamics and paves the way for more sustainable and efficient deep learning practices, particularly in resource-constrained environments. In the era of the race for foundation models, we believe our method emerges as a valuable framework. Code available at https://github.com/baunilhamarga/critical-periods.

Keywords: Deep Learning; Neural Networks; Critical Periods; Regularization; Green AI

1. Introduction

Critical learning periods refer to a deep learning phenomenon where early epochs determine success in many training recipes, including regularization and the capacity of the model to combine information from diverse sources [1, 20]. Understanding critical learning periods in deep learning can significantly boost training efficiency and overall model performance. Despite their acknowledged significance, accurately identifying such periods during

* Equal contribution.
E-mail address: {vinicius.fukase, heitor_gama}@usp.br

1877-0509
10.1016/j.procs.2025.07.138

the training phase remains elusive. However, these periods significantly influence the dynamics and effectiveness of learning. Thus, systematically identifying them is essential to optimize training efficiency and model generalization.

Prior research confirms that critical periods manifest early in the training process, beyond which numerous training methods yield minimal to no additional advantage [1]. Kleinman et al. [20] emphasize the capacity of neural networks to combine data from varied origins, significantly hinging on their exposure to appropriately correlated stimuli during initial training stages. Intricate and unpredictable early transient dynamics illustrate the emergence of critical periods. These dynamics play a crucial role in determining final efficacy and the nature of representations acquired by the system upon completion of training. The seminal work by Kleinman et al. [21] demonstrates that learning ability does not increase monotonically during training and that a memorization phase occurs in early epochs, where models retain most discriminative information. Furthermore, the authors show that critical periods occur during the memorization phase and, after this, the model may start to weaken its ability to retain knowledge (forgetting phase).

Existing studies argue that the effectiveness of regularization during model training extends beyond merely preventing solution entrapment in local minima [1]. Specifically, they find that adjusting regularization practices after a critical period in training changes weight values and thereby the position of the model in the loss landscape. Consistent generalization confirms that this adjustment does not improve the predictive ability of models. Rather, its importance lies in guiding initial stages of training towards areas of the loss landscape that are rich in diversity, yet equally effective solutions with strong generalization characteristics. Although these works play an important role in critical learning periods, none of them offer a systematic way to identify critical periods. It is worth mentioning that Golatkar et al. [13] find that critical periods emerge early in training but not exactly when (i.e., the epoch). Therefore, a natural question that arises is:

How to identify critical periods during the course of training?

Given calls for more efficient training in the era of foundation models [7], answering this question yields two notable gains. Firstly, we could apply strong regularization only while the model is apt to absorb information (before critical periods). Secondly, we could reduce the number of training examples through data pruning only after the critical period; thus, preventing loss in predictive ability. Together, these strategies enable better allocation of computational resources, avoiding unnecessary demands, and significantly speeding up the training process.

Research Statement and Contributions. To sum up, our work has the following research statement. *Throughout the course of training, a simple generalization estimation enables successfully identifying when the critical period emerges. Adjusting training recipes at this point – such as stopping data augmentation or refining data pruning – preserves predictive ability, while significantly reducing training costs.* Among our contributions, we highlight the following. 1) We emphasize the importance of discovering the moment (epoch) at which the critical periods emerge. 2) We introduce a systematic approach for identifying such a moment during the training process. 3) Across various benchmarks and architectures, we show that our method significantly reduces computational costs by eliminating training recipes after critical periods. This efficiency comes with a minimal trade-off in accuracy. Overall, our contributions not only enhance understanding of the training dynamics, but also offer a practical tool for optimizing resource use in deep learning.

Extensive experiments across a broad range of benchmarks (CIFAR-10/100 [22, 23], EuroSat [16], Tiny ImageNet [25] and ImageNet30 [17]) and architectures confirm our research statement and contributions. Specifically, we reduce training time by up to 59.67% with a negligible accuracy drop. In terms of Green AI [24, 11, 30], these results represent a significant advancement in minimizing carbon emissions associated with energy use during model deployment. In particular, we reduce CO_2 emissions by 59.47% and financial costs by 60%.

2. Related Work

Critical Periods. The training dynamics of neural networks reveal that early stages of learning play a decisive role in determining model performance [1, 13, 28]. These stages, named critical periods, represent phases where regularization techniques, such as weight decay and data augmentation, have the most significant impact on generalization [13, 1]. Once these periods pass, persistently applying regularization yields diminishing returns, adding computational overhead without comparable benefits [20].

Recent studies emphasize the importance of understanding and leveraging critical learning periods. For example, Golatkar et al. [13] and Achille et al. [1] suggest that reducing or even removing regularization after initial learning phases can lead to more effective training. Recent works explore the role of critical learning periods beyond traditional deep machine learning, particularly in federated learning. For instance, Yan et al. [37, 39, 38] investigate how leveraging critical periods can enhance robustness against adversarial attacks and optimize client selection strategies in federated settings. From the lens of combining sources of information, Kleinman et al. [20] observed that critical periods also impair the ability to synergistically merge data across multiple sources. Although these studies provide insights into critical periods phenomena, mainly from the theoretical perspective, none suggest a systematic method for identifying them. Importantly, identifying and acting upon critical periods offers a promising direction for reducing computational cost and improving training efficiency [12]. Our work fills this gap and introduces an effective strategy for pinpointing critical periods. In practical terms, this enables halting training recipes at optimal points, achieving comparable or improved accuracy while significantly reducing training time.

Generalization Estimation. Estimating how neural networks generalize to unseen data early in the training process is crucial for both practical applications and for advancing the theoretical understanding of deep models [2]. For example, Sankararaman et al. [34] estimate training convergence using the gradient confusion among batches of samples during the SGD updates. Similarly, Chen et al. [4] estimate training dynamics according to the stability of gradient directions across batches and parameter updates. From a different perspective, Carbonnelle and Vleeschouwer [3] introduce a simple yet effective indicator of generalization. Their method, named *Layer Rotation*, estimates generalization performance at a given training epoch taking into account the cosine distance between its weights and those from the random initialization.

As we shall see, our method leverages the metrics above to discover critical periods. Throughout our analysis, we observe that the methods by Sankararaman et al. [34] and Chen et al. [4] become unreliable for this purpose as they reveal inconsistent relationships with model accuracy, exhibit significant noise or have high computational cost.

Data Augmentation. The current paradigm for solving cognitive tasks using deep learning involves training models on large amounts of data [9]. Modern models, such as Llama 3, reinforce that the secret ingredient behind positive results lies in web-scale and high-quality data [9]. In this direction, data augmentation becomes one of the most important training recipes. It turns out that, due to the stochastic nature of state-of-the-art techniques, many of them allow the creation of multiple samples from a single one [40]. Thus, it is possible to increase both data quantity and diversity without a labor-intensive and costly labeling process. For example, popular augmentations such as PixMix [18], MixUp [41], and Cutout [42] apply transformations to the original sample with a given probability p. From this perspective, one could generate arbitrarily large training datasets just using data augmentations k times per input.

Regardless of whether the increase in data quantity stems from collection or data augmentation, handling more data incurs high computational, energy, and financial costs.

Data Pruning. While data augmentation focuses on enriching the dataset, data pruning aims to reduce computational costs by selecting a smaller, representative subset of data, without sacrificing predictive performance. Existing methods typically fall into two categories. Importance-based approaches [14, 35, 5] evaluate the relevance of individual data points. Optimization-based techniques aim to retain the core characteristics of the original dataset [27, 10, 36, 26]. However, both approaches often involve complex and computationally expensive procedures. Interestingly, Okanovic et al. [31] demonstrate that well-designed random pruning strategies can match or even outperform many sophisticated methods, underscoring the potential of simpler, more scalable solutions.

Despite positive advancements, most pruning techniques overlook the critical learning periods that occur early in training, when data selection may exert a disproportionate impact on model performance. To address this gap, we propose applying data pruning only after our method identifies the critical period. This approach ensures that sufficient data supports the early stages of learning, allowing even random pruning methods – like the one suggested by Okanovic et al. [31] – to achieve superior performance with reduced complexity.

According to our results, this strategy reduces the overall training time by up to 2.5× without degrading generalization. Additionally, unlike computationally intensive techniques, our proposed form of data pruning incurs no additional costs.

3. Preliminaries and Proposed Method

Preliminaries. Assume X and Y a set of training samples and their respective class labels (i.e., categories). Let $\mathcal{F}(\cdot, \theta)$ be a neural network parameterized by a set of weights θ. From a random initialization θ^0, an iterative process (e.g., SGD) updates θ^i towards a minimum of the loss function $\mathcal{L}$, where i indicates the i-th iteration of this process. Unless stated otherwise, we conduct the iterative update process of θ across N training epochs.

To improve generalization of $\mathcal{F}$, previous works typically apply regularization and data augmentation mechanisms during the optimization iterative process [6, 18, 19, 33]. Particularly, in this work, we focus on regularization through data augmentation techniques, as we formalize below.

Let $T(\cdot)$ be a function that receives samples from X and modifies its content, producing a new set of same size (i.e., $|T(X)| = |X|$). Typically, modern data augmentation methods incorporate stochastic elements enabling the creation of arbitrarily large datasets by applying T multiple times [6, 18, 19]. Formally, we can augment the original dataset by applying T k times, denoted by $\{(T(X), Y)\}^k$. To simplify the notation, let $\mathcal{D} = (X, Y)$ represent the pair of samples and their respective labels. Thus, after performing data augmentation k times, it is possible to rewrite data augmentation as $\{\mathcal{D}\}^k$.

By applying data augmentation techniques, we can formalize the iterative process of updating θ as follows:

$$\theta^{i+1} = \theta^i - \eta \frac{1}{B} \sum_{b=1}^{B} \nabla \mathcal{L}(\{\mathcal{D}\}_b^k, \theta^i), \tag{1}$$

where B indicates the *batch size*, $\mathcal{D}_b^k$ is a *batch* of b samples from augmented data, $\nabla \mathcal{L}$[1] corresponds to the gradient of the loss function with respect to the parameters θ^i and η denotes the update magnitude (learning rate).

Building on the previous formalism, the end of a critical learning period is a training epoch $i^* \in \{0, \ldots, N\}$ from which point onward different training recipes provide little or no benefit to generalization. Therefore, our goal is to identify i^* in a way that effectively minimizes, or ideally eliminates, computationally intensive training recipes. Moreover, we can apply *data pruning* (i.e., $k < 1$) to further reduce the computational demands of the training phase. In this context, we adopt the method proposed by Okanovic et al. [31]. Their work emphasizes that repeated random sampling is a simple yet effective method that can outperform more complex techniques. Hence, this method becomes attractive as it incurs no additional costs while introducing variability and enhancing generalization. At each epoch, we apply dynamic random data pruning by randomly selecting a percentage of the training dataset. Therefore, the training data changes with every epoch, ensuring the model sees different data points. Formally, at each epoch i, we select a random subset $D_i \subset D$ of the training data, where D is the full training set and $D_i = k \cdot D$, where k represents the proportion of the dataset selected. It is important to note that our training phase employs this process only when we consider data pruning mechanisms.

In summary, by discovering i^*, we notably speed up the overall training process while preserving predictive ability. It is worth emphasizing that previous studies confirm the possibility of successfully reducing or eliminating training recipes after an iteration i [1, 13]. To the best of our knowledge, however, there are no efforts to determine *when* to reduce it.

Proposed Method. To grasp the intriguing generalization properties present in deep neural networks, it is crucial to identify numerical indicators of generalization performance that remain applicable across diverse training settings. In this context, Carbonnelle and Vleeschouwer [3] propose a groundbreaking approach to understanding and improving neural network generalization, named *Layer Rotation*. This approach focuses on tracking the evolution of the cosine distance between each weight vector of each layer and its initial state throughout the training process. Following Mason-Williams and Dahlqvist [29], instead of computing the cosine similarity between each weight of a given layer (as originally suggested by Carbonnelle and Vleeschouwer [3]), we concatenate all the weights composing the model $\mathcal{F}$ and linearize them to form a single vector. According to their work [29], this process enables a representation of the neural network parameters. For ease of exposition, we will keep indicating the single vector representing all weights of $\mathcal{F}$ as θ^0 (random initialization) and θ^i (with $i > 0$).

[1] It is possible to rewrite the gradient as follows: $\mathcal{L}(\{(Y_b, \mathcal{F}(T(X_b)))\}^k, \theta^i)$, where X_b and Y_b are data *batches* and respective labels.

Given the previous definition, the cosine distance between θ^0 and θ^i defines layer rotation at training epoch i. Thus, we estimate the layer rotation in terms of

$$CosineDistance = 1 - \frac{\theta^{\mathbf{0}} \cdot \theta^{\mathbf{i}}}{||\theta^{\mathbf{0}}||||\theta^{\mathbf{i}}||}. \tag{2}$$

Through a comprehensive suite of experiments encompassing a broad spectrum of datasets, network architectures, and training regimes, we uncover a consistent pattern: *larger layer rotations (i.e. as cosine distance between the final and initial weights increases) reliably predict enhanced generalization performance*.

In order to visualize layer rotation evolution during training, we track the cosine distance between the current weight vector of each layer and its initial state across various training steps. Upon analyzing these curves on a validation set, we notice a characteristic shape pattern emerging throughout the training process. To systematically identify critical periods within this evolution, we adopt an approach that performs linear regression over a window of 5 epochs (w). This choice proves itself effective in our experiments, emerging as the smallest window size that still allows capturing significant changes in layer behavior. A smaller window would make the analysis too vulnerable to minor fluctuations.

We scale the number of epochs and cosine distance from 0 to their respective maximum values. This way, we can graphically calculate the learning variation during training by the angle α that indicates the rate of change within this window. Then, we determine its value by the linear regression of the normalized data and the arctangent calculation of the regression coefficient m in degrees. To accomplish this, we define a set of data points $\{(u_1, v_1), (u_2, v_2), \ldots, (u_w, v_w)\}$, where u_i represents the epoch (independent variable) and v_i the cosine distance between θ^i and θ^0 (dependent variable) – the Layer Rotation. The slope m of the regression line that best fits these data points (in the least squares sense) is:

$$m = \frac{\sum_{i=1}^{w}(u_i - \bar{u})(v_i - \bar{v})}{\sum_{i=1}^{w}(u_i - \bar{u})^2}. \tag{3}$$

We therefore present the formula for calculating the angle as follows:

$$\alpha = \frac{180}{\pi} \arctan(m). \tag{4}$$

In our initial experiments, we observe that an angle of 45° marks a pivotal moment in the training of neural networks, where the shift from rapid learning to careful refinement and optimization occurs. Therefore, we use this value throughout our work.

4. Experiments

Experimental Setup. In our experiments, we use training cycles of 200 epochs and SGD optimizer [13, 21] with a learning rate that starts at 0.01, and is divided by 10 at epochs 100 and 150. Furthermore, every sample is transformed randomly before each epoch by a combination of horizontal flips, crops, rotations, and translations. We apply this basic transformation even after we reduce the augmentation factor k to 1 or less mid-training. Unless stated otherwise, our data augmentation consists of repeating each sample k times, with $k = 3$ (formalism given in Section 3). By doing this repetition step before the random transformations, we ensure each copy is slightly different from the original. This is important because we want to simulate the effect of having more samples, not just simple copies.

Regarding the models and datasets, we use the popular Residual networks [15] at different depths, and the CIFAR-10/100 benchmarks. Overall, we apply these experimental settings (model × datasets) for most experiments because they are common practices in the context of critical period and data pruning [32, 13, 20]. However, to confirm the effectiveness of our method, we also evaluate it on large-scale datasets, including EuroSat [16], Tiny ImageNet [25], and ImageNet30 [17].

Throughout experiments and discussions, the term baseline refers to the model without removing training recipes. In other words, it means the model training on standard practices without any knowledge and intervention of critical periods.

Metrics and Evaluation. To evaluate the effectiveness of our method in terms of improvements in computational cost and generalization, we introduce two key metrics: Normalized Training Cost: This metric quantifies the computational demand of training relative to the baseline (training with initial $k = 3$ – see Equation 1 – during all epochs). It normalizes the cost by combining the number of epochs and the number of data points used per SGD update. Overall, this metric accounts for dynamic changes in dataset size during the training process, particularly during critical periods. Accuracy Delta: This metric measures the variation in model accuracy compared to the baseline model. It enables assessing the trade-off between computational efficiency and performance when removing training recipes.

Enhancing Generalization Through Repeated Augmentation. We start our analysis by illustrating the advantages of generating arbitrarily large datasets by applying the same data augmentation k times per sample. As we mentioned before, this is possible due to the stochastic nature of modern augmentation strategies. Specifically, because they employ transformations (i.e., rotation or crop) with a given probability p. To this end, we increase the number of training samples by three times ($k = 3$ in Equation 1) and compare the accuracy when using the dataset with data augmentation without changing its size (i.e., $k = 1$). For a fair comparison, we consider the same initialization.

On the ResNet32 architecture, we observe an improvement of roughly 1 percentage point (pp) while using the increased dataset ($k = 3$) and a similar gain with a deeper, high-capacity architecture (ResNet86[2]). From these findings, we confirm that an effective training recipe for improving generalization is simply to expand the dataset size by repeating the same data augmentation k times. Additionally, we highlight the following key observations. First, although these improvements may seem small, modern and complex data augmentation methods achieve similar gains [41, 42]. Second, despite improving generalization, the training time increases proportionally; for example, moving from $k = 1$ to $k = 3$ increases the training time by roughly three times. Most importantly, this setting becomes a potential candidate for exploring the practical benefits of our method in discovering critical periods. In particular, we can begin training a model on the expanded version of a dataset ($k = 3$) and then reduce the dataset to its original size ($k = 1$), or even use smaller versions ($k < 1$ – data pruning), after identifying the critical period. Therefore, adapting the training size through k enables leveraging the best of both worlds: *higher generalization and lower training time.*

Revisiting Critical Periods. In this experiment, we re-examine the existence of critical periods. However, in contrast to previous works [21, 20, 1], we analyze it from the lens of potential values of i^*, taking into account the compromise between accuracy and computational cost. Specifically, our main objective is to identify the end of the critical period, i.e., an early epoch i^* where we reduce training recipes until the training is complete and final accuracy is sufficiently close to the baseline accuracy. To achieve this, we create oracle models that reveal every possible accuracy outcome after reducing the augmentation factor k from 3 to 1 at epoch i. It is worth mentioning that, after eliminating data augmentation at epoch i, the training continues until completing 200 epochs. Therefore, each oracle model uses $k = 3$ for the initial i epochs and $k = 1$ for the remaining $200 - i$ epochs.

The aforementioned process produces 200 oracle models and Figure 1 shows the accuracy of each one at the end of the training phase. Figure 1 reinforces the existence of critical periods early in training and supports that, after pinpointing i^* (i.e. the end of the critical period), reducing augmentation is optimal in terms of accuracy and computational cost. Following this reasoning, epoch 24 marks the potential end of the critical period. It is important to note that better-performing models only emerge by epoch 60, but the accuracy gain is not significant enough to justify the additional computational cost.

Despite using checkpoints to avoid retraining initial epochs, this experiment is very resource-intensive. For this reason, we choose CIFAR-10 and ResNet32, as both are computationally light while remaining sufficiently challenging and capable of delivering competitive performances for this task [15, 8].

From Figure 1, the optimal epoch i^* is explicit, however, this is the epoch we wish to identify without completing all these multiple models, allowing the decision to stop applying training recipes at that point in a single run (i.e., on-the-fly during a single training phase). This is what our method aims to achieve. Therefore, we now turn our attention towards automatically and systematically identifying i^*.

Effectiveness of Generalization Estimators. Due to the nature of critical periods — epochs where training recipes promote significant impact on generalization [13, 1] — we argue that generalization estimation metrics can successfully identify i^*. Specifically, these metrics enable estimating the epoch where generalization begins to decline by

[2] Here, we avoid using the popular ResNet56 and ResNet110 to prevent any bias in our subsequent analysis.

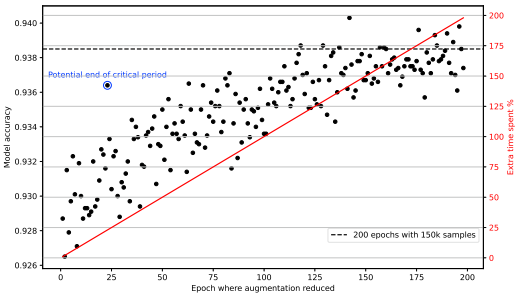

Fig. 1. Critical period in ResNet32 on CIFAR-10. Each point indicates a model with a complete training cycle of 200 epochs. The figure shows the accuracy trends when data augmentation reduces from 150k to 50k samples per epoch at different epochs. The red line indicates the extra time cost.

analyzing their behavior among a range of epochs during training. In informal terms, we believe these metrics can identify the circled point in Figure 1.

Knowing the optimal i^* beforehand from the previous experiment enables validating our argument to use generalization estimation metrics. Following previous studies, we explore the following metrics: Gradient Confusion [34], Gradient Predictiveness [4] and Layer Rotation [3].

Regarding the first two metrics, we observe notable drawbacks. For example, Gradient Confusion is capable of accurately identifying critical periods (i.e., identify the epoch 24 in Figure 1), but its computational cost is as intensive as the cost saved by reducing training recipes, making it impractical for real-time applications where reducing training time is a key objective. In contrast, Gradient Predictiveness identifies the critical period far from the optimal point. Particularly, this metric shows no correlation between its values and model accuracy on a validation set.

Regarding Layer Rotation, we observe that as epochs progress, the cosine distance between the weights of each epoch and the initial weights increases. However, we notice that, at a certain point, the growth of this distance begins to diminish. Carbonnelle and Vleeschouwer [3] state that Layer Rotation achieves a network-independent optimum when the cosine distance of all layers reaches 1. However, our practical observations contradict this, revealing that distance values significantly fluctuate with the architecture. Consequently, the pure application of this metric turns out to be architecture-dependent, making the Layer Rotation value arbitrary and unique to each architecture. Therefore, it becomes necessary to implement the learning variation during a window of epochs we introduce in Equations 3 and 4. These changes allow analyzing the slope of the line generated by the linear regression of a 5-epoch window, evidencing the reduction in generalization over epochs (i.e., temporally). From these experiments, we identify epoch 24 as the turning point, at which the angle α becomes less than 45º, indicating the transition of the curve to a smoother phase. The comparison with the oracle in Figure 1 highlights this point as having significant potential for the critical period. Thereby, we confirm the potential of Layer Rotation in discovering the end of critical periods.

Besides accurately identifying the critical period, Layer Rotation is a quite efficient metric that demands negligible computational resources, as its calculations are simple and always based on comparing the current weights of the epoch with the initial ones (see Equation 2). Therefore, we employ Layer Rotation as the main metric implemented in our systematic method.

Comparison with State-of-the-Art Data Pruning. Existing data pruning methods provide positive results in reducing computational costs during the training of deep models [32, 5]. However, to the best of our knowledge, no method currently takes into account the critical period phenomenon.

In this experiment, we investigate the behavior of applying data pruning *only after* our method identifies critical periods and compare the results with state-of-the-art methods. For this purpose, we build upon the experimental framework established by Qin et al. [32], using their evaluation of static and dynamic data pruning methods[3] as a benchmark. To this baseline, we integrate results from our approach, introducing data pruning after identifying the critical periods during training. For a fair comparison, we use ResNet18 as the backbone model, training it on CIFAR-

[3] Static data pruning removes a predetermined subset of data, with no reintroduction during training, whereas dynamic pruning allows for the subset to change as training progresses, adapting to the requirements of the model [32]. Our method, as specified in Section 3, uses the dynamic approach.

Table 1. Comparison of accuracy delta (ΔAcc) with state-of-the-art data pruning methods. Bold, underline, ↑ and ↓ mean the best results, second-best results, accuracy improvement and accuracy decrease, respectively.

	Dataset	CIFAR-10			CIFAR-100		
	Pruning Ratio %	30	50	70	30	50	70
Static	Random	↓1.0	↓2.3	↓5.4	↓4.4	↓6.1	↓8.5
	CD	↓0.6	↓1.3	↓4.8	↓4.0	↓5.9	↓7.9
	K-Center	↓0.9	↓1.7	↓4.7	↓4.1	↓6.0	↓8.0
	Least Confidence	↓0.6	↓1.1	↓5.3	↓4.0	↓5.9	↓8.4
	Margin	↓0.7	↓1.3	↓4.7	↓4.2	↓6.0	↓8.0
	Forgetting	↓0.9	↓1.5	↓3.9	↓2.9	↓5.1	↓8.3
	GraNd-4	↓0.3	↓1.0	↓4.4	↓3.6	↓6.8	↓9.4
	DeepFool	↓0.5	↓1.5	↓5.6	↓4.0	↓5.0	↓6.4
	Craig	↓0.8	↓3.3	↓7.2	↓3.8	↓6.3	↓8.5
	Glister	↓0.4	↓1.6	↓4.7	↓3.6	↓5.0	↓7.8
	EL2N-20	↓0.3	↓0.5	↓3.7	↓1.0	↓6.1	-
	DP	↓0.7	↓1.8	↓4.8	↓1.0	↓5.1	-
Dynamic	Random	↓0.8	↓1.1	↓2.6	↓0.9	↓2.9	-
	ϵ-greedy	↓0.4	↓0.7	↓1.5	↓1.8	↓3.4	-
	UCB	↓0.3	↓0.9	↓1.7	↓0.9	↓2.9	-
	InfoBatch	**↑0.0**	↓0.5	**↓0.9**	↑0.0	**↓0.1**	**↓1.7**
	Ours	**↑0.0**	**↓0.3**	↓1.6	**↑0.2**	↓1.4	↓1.8

10/100 datasets with pruning ratios of 30%, 50% and 70%. It is worth mentioning our method diverges from traditional data pruning techniques by initially training the model on the full dataset (i.e $k = 1$) for the first few epochs and then applying the pruning ratio from the critical periods onward. We compute the accuracy delta relative to a model trained for 200 epochs without data pruning, ensuring a fair comparison. Table 1 shows the results.

According to Table 1, our method outperforms all static pruning methods across both datasets and all pruning ratios, consistently achieving lower accuracy degradation. Compared to dynamic methods, our approach delivers competitive performance, rivaling the state-of-the-art InfoBatch method [32]. For example, on CIFAR-10 with a 50% pruning ratio, our method results in an accuracy drop of only 0.3. Additionally, on CIFAR-100 with a 30% pruning ratio, our method leads to a surprising increase in accuracy by 0.2, showcasing its robustness.

Overall, the previous results emphasize the importance of timing in data pruning, demonstrating that applying these mechanisms after critical periods allows the model to retain more data during the most effective learning phases; thereby optimizing performance while reducing trade-offs.

Effectiveness on Large Datasets and Optimizers. To validate the effectiveness of our method, we analyze its performance across different datasets and optimizers. Our objective is to demonstrate that the method is agnostic to both dataset and optimizer choices, further reinforcing its general applicability. To this end, we consider the following datasets: CIFAR-10/100 [22, 23], EuroSAT [16], TinyImageNet [25] and ImageNet30 [17]. On these datasets, the popular architecture is ResNet50; therefore, we employ it in these experiments.

Table 2 introduces the accuracy delta and time saved when applying our method on large datasets. The results confirm that our approach maintains competitive performance while greatly reducing computational costs, independent of the dataset. Specifically, we save up to 65.79% training time while increasing accuracy on EuroSAT by 1.02 pp.

Similarly, Table 3 examines the impact of different optimizers on the effectiveness of our method in identifying critical periods. This experiment is important to show that our method is neither confined nor biased to the iterative

Table 2. Accuracy delta (ΔAcc) in percentage points and time saved of our method across different datasets using ResNet50 with augmentation factor $k = 3$. ↑ and ↓ mean accuracy improvement and decrease, respectively.

	CIFAR-10	**CIFAR-100**	**EuroSAT**	**TinyImageNet**	**ImageNet30**
ΔAcc	↓0.32	↓2.36	↑1.02	↓0.20	↑0.34
Saved (%)	49.35	49.47	65.79	60.31	61.70

Table 3. Accuracy delta (ΔAcc) in percentage points and time saved of our method across different optimizers using ResNet50 on CIFAR-10 with augmentation factor $k = 2$. ↑ and ↓ mean accuracy improvement and decrease, respectively.

	SGD	**AdamW**	**RMSprop**	**AdaGrad**
ΔAcc	↓1.04	↓0.18	↓0.59	↓0.32
Saved (%)	33.75	33.68	32.99	33.57

learning process of SGD. Table 3 confirms the consistency of our method across different optimizers, demonstrating its optimizer-agnostic nature and ensuring applicability in diverse training settings. The results indicate that changing the optimizer has minimal impact on our method. The accuracy deltas remain within a narrow range, and the time saved is consistently around 33% across all optimizers, reinforcing the robustness of our approach.

These results, combined with our previous experiments on different architectures, reinforce that our method is not only model-agnostic, as previously demonstrated, but also dataset-agnostic and optimizer-agnostic, further supporting its wide-ranging applicability.

Computational Benefits and GreenAI. Existing works show that modern models emit high levels of carbon dioxide (CO_2) due to their substantial processing capacity and energy requirements during training and implementation [24, 12, 30]. However, our approach drastically lowers the carbon footprint through a direct increase in computational efficiency. Specifically, applying our method to ResNet56 results in a 58.89% reduction in CO_2 emissions. Identifying critical periods also reduces financial costs, achieving a 58.33% reduction for this model. For other architectures, we achieve results nearing a 60% reduction in both CO_2 emissions and financial costs. We estimate these values using the Machine Learning Impact Calculator [24].

To summarize, our efforts yield significant advancements in GreenAI by effectively lowering carbon footprint and enhancing the financial accessibility of deep learning models.

5. Conclusions

Existing works suggest that early epochs, known as critical periods, play a decisive role in the success of many training recipes. In this work, we propose a systematic method to identify critical learning periods in neural network training. Our method leverages a simple yet effective prediction estimator, named Layer Rotation, to analyze the generalization behavior during training and then identify when critical periods emerge.

Extensive experiments confirm a consistent pattern in our idea: larger layer rotations – i.e., as cosine distance between the final and initial weights increases – reliably predict enhanced generalization performance and hence indicate the emergence of critical periods. Importantly, our method fills the gap in existing studies on critical learning periods that fail to offer ways to identify them. From a practical perspective, our approach significantly improves training time by restricting resource-intensive training recipes (such as data augmentation) to the critical learning periods. As a concrete example, this results in up to 2.5× reduction in training cost with minimal accuracy trade-offs across diverse benchmarks. Our findings underscore the untapped potential of early-phase analysis for refining training recipes, offering practical insights for sustainable machine learning and resource allocation. We hope this work inspires further exploration into adaptive training methods and efficient deep learning practices.

Acknowledgments

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001. Artur Jordao Lima Correia would like to thank Edital Programa de Apoio a Novos Docentes 2023. Processo USP nº: 22.1.09345.01.2. Anna H. Reali Costa would like to thank grant #312360/2023-1 CNPq.

References

[1] Achille, A., Rovere, M., Soatto, S., 2019. Critical learning periods in deep networks, in: ICLR.

[2] Ballester, R., Clemente, X.A., Casacuberta, C., Madadi, M., Corneanu, C.A., Escalera, S., 2024. Predicting the generalization gap in neural networks using topological data analysis. Neurocomputing .
[3] Carbonnelle, S., Vleeschouwer, C.D., 2019. Layer rotation: a surprisingly simple indicator of generalization in deep networks?, in: ICML.
[4] Chen, Y., Yuille, A., Zhou, Z., 2023. Which layer is learning faster? a systematic exploration of layer-wise convergence rate for deep neural networks, in: ICLR.
[5] Choi, H., Ki, N., Chung, H.W., 2024. BWS: best window selection based on sample scores for data pruning across broad ranges, in: ICML.
[6] Cubuk, E.D., Zoph, B., Shlens, J., Le, Q., 2020. Randaugment: Practical automated data augmentation with a reduced search space, in: NeurIPS.
[7] DeepSeek-AI, Liu, A., et al., B.F., 2024. Deepseek-v3 technical report. ArXiv .
[8] Deng, W., Feng, Q., Gao, L., Liang, F., Lin, G., 2020. Non-convex learning via replica exchange stochastic gradient MCMC, in: ICML.
[9] Dubey, A., et al., 2024. The llama 3 herd of models. ArXiv .
[10] Engstrom, L., Feldmann, A., Madry, A., 2024. Dsdm: Model-aware dataset selection with datamodels, in: ICML.
[11] Faiz, A., Kaneda, S., Wang, R., Osi, R.C., Sharma, P., Chen, F., Jiang, L., 2024a. Llmcarbon: Modeling the end-to-end carbon footprint of large language models, in: ICLR.
[12] Faiz, A., Kaneda, S., Wang, R., Osi, R.C., Sharma, P., Chen, F., Jiang, L., 2024b. Llmcarbon: Modeling the end-to-end carbon footprint of large language models, in: ICLR.
[13] Golatkar, A., Achille, A., Soatto, S., 2019. Time matters in regularizing deep networks: Weight decay and data augmentation affect early learning dynamics, matter little near convergence, in: NeurIPS.
[14] Han, X., Simig, D., Mihaylov, T., Tsvetkov, Y., Celikyilmaz, A., Wang, T., 2023. Understanding in-context learning via supportive pretraining data, in: ACL.
[15] He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition, in: CVPR.
[16] Helber, P., Bischke, B., Dengel, A., Borth, D., 2018. Introducing eurosat: A novel dataset and deep learning benchmark for land use and land cover classification, in: IGARSS.
[17] Hendrycks, D., Mazeika, M., Kadavath, S., Song, D., 2019. Using self-supervised learning can improve model robustness and uncertainty, in: NeurIPS.
[18] Hendrycks, D., Zou, A., Mazeika, M., Tang, L., Li, B., Song, D., Steinhardt, J., 2022. Pixmix: Dreamlike pictures comprehensively improve safety measures, in: CVPR.
[19] Kim, I., Lee, H., Lee, H.E., Shin, J., 2025. Controllable blur data augmentation using 3d-aware motion estimation, in: ICLR.
[20] Kleinman, M., Achille, A., Soatto, S., 2023. Critical learning periods for multisensory integration in deep networks, in: CVPR.
[21] Kleinman, M., Achille, A., Soatto, S., 2024. Critical learning periods emerge even in deep linear networks, in: ICLR.
[22] Krizhevsky, A., Nair, V., Hinton, G., 2009a. Cifar-10 (canadian institute for advanced research) .
[23] Krizhevsky, A., Nair, V., Hinton, G., 2009b. Cifar-100 (canadian institute for advanced research) .
[24] Lacoste, A., Luccioni, A., Schmidt, V., Dandres, T., 2019. Quantifying the carbon emissions of machine learning, in: NeurIPS.
[25] Le, Y., Yang, X.S., 2015. Tiny imagenet visual recognition challenge.
[26] Li, Z., Wu, T., Tan, J., Zhang, M., Wang, J., Lin, D., 2025. IDIV: Intrinsic decomposition for arbitrary number of input views and illuminations, in: ICLR.
[27] Mahabadi, S., Trajanovski, S., 2023. Core-sets for fair and diverse data summarization, in: NeurIPS.
[28] Maini, P., Mozer, M.C., Sedghi, H., Lipton, Z.C., Kolter, J.Z., Zhang, C., 2023. Can neural network memorization be localized?, in: ICML.
[29] Mason-Williams, G., Dahlqvist, F., 2024. What makes a good prune? maximal unstructured pruning for maximal cosine similarity, in: ICLR.
[30] Morrison, J., Na, C., Fernandez, J., Dettmers, T., Strubell, E., Dodge, J., 2025. Holistically evaluating the environmental impact of creating language models, in: ICLR.
[31] Okanovic, P., Waleffe, R., Mageirakos, V., Nikolakakis, K.E., Karbasi, A., Kalogerias, D.S., Gürel, N.M., Rekatsinas, T., 2024. Repeated random sampling for minimizing the time-to-accuracy of learning, in: ICLR.
[32] Qin, Z., Wang, K., Zheng, Z., Gu, J., Peng, X., Xu, Z., Zhou, D., Shang, L., Sun, B., Xie, X., You, Y., 2024. Infobatch: Lossless training speed up by unbiased dynamic data pruning, in: ICLR.
[33] Robine, J., Höftmann, M., Harmeling, S., 2025. Simple, good, fast: Self-supervised world models free of baggage, in: ICLR.
[34] Sankararaman, K.A., De, S., Xu, Z., Huang, W.R., Goldstein, T., 2020. The impact of neural network overparameterization on gradient confusion and stochastic gradient descent, in: ICML.
[35] Xia, M., Malladi, S., Gururangan, S., Arora, S., Chen, D., 2024. LESS: selecting influential data for targeted instruction tuning, in: ICML.
[36] Xiao, G., Tang, J., Zuo, J., junxian guo, Yang, S., Tang, H., Fu, Y., Han, S., 2025. Duoattention: Efficient long-context LLM inference with retrieval and streaming heads, in: ICLR.
[37] Yan, G., Wang, H., Li, J., 2022. Seizing critical learning periods in federated learning, in: AAAI.
[38] Yan, G., Wang, H., Yuan, X., Li, J., 2023a. Criticalfl: A critical learning periods augmented client selection framework for efficient federated learning, in: KDD.
[39] Yan, G., Wang, H., Yuan, X., Li, J., 2023b. Defl: Defending against model poisoning attacks in federated learning via critical learning periods awareness, in: AAAI.
[40] Yun, S., Han, D., Chun, S., Oh, S.J., Yoo, Y., Choe, J., 2019. Cutmix: Regularization strategy to train strong classifiers with localizable features, in: ICCV.
[41] Zhang, H., Cissé, M., Dauphin, Y.N., Lopez-Paz, D., 2018. mixup: Beyond empirical risk minimization, in: ICLR.
[42] Zhong, Z., Zheng, L., Kang, G., Li, S., Yang, Y., 2020. Random erasing data augmentation, in: AAAI.