LaCoOT: Layer Collapse through Optimal Transport

Victor Quétu[1] Zhu Liao[1] Nour Hezbri[2] Fabio Pizzati[3] Enzo Tartaglione[1]
[1]LTCI, Télécom Paris, Institut Polytechnique de Paris, France
[2]ENSAE, Institut Polytechnique de Paris, France
[3]MBZUAI, UAE
victor.quetu@telecom-paris.fr

Abstract

Although deep neural networks are well-known for their outstanding performance in tackling complex tasks, their hunger for computational resources remains a significant hurdle, posing energy-consumption issues and restricting their deployment on resource-constrained devices, preventing their widespread adoption.
In this paper, we present an optimal transport-based method to reduce the depth of over-parametrized deep neural networks, alleviating their computational burden. More specifically, we propose a new regularization strategy based on the Max-Sliced Wasserstein distance to minimize the distance between the intermediate feature distributions in the neural network. We show that minimizing this distance enables the complete removal of intermediate layers in the network, achieving better performance/depth trade-off compared to existing techniques. We assess the effectiveness of our method on traditional image classification setups and extend it to generative image models. Our code is available at https://github.com/VGCQ/LaCoOT.

1. Introduction

Over the last few years, the field of deep learning has undergone a significant transformation with the advent of foundation models. These are large-scale, pre-trained models capable of performing a wide range of tasks across different domains, including computer vision. As exemplars we can mention CLIP [52] and ALIGN [26] for image classification, DiT [47], Stable Diffusion [54] and DALL-E [53] for image generation, or SAM [29] for semantic segmentation. The effectiveness of these foundation models is primarily driven by empirical patterns observed through scaling laws [23]: the improvements achieved by these models correlate with the exponential increase in computational requirements due to the growth of both their size and the number of training data [10, 55].

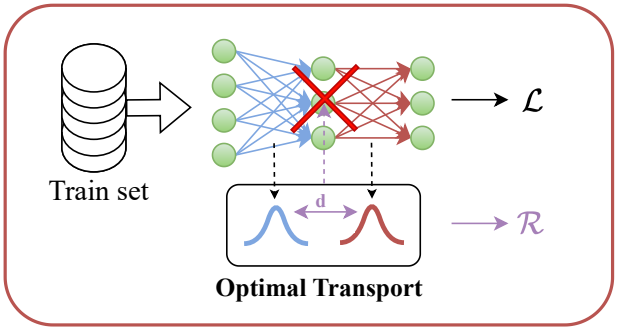

Figure 1. With LaCoOT, we finetune existing networks with an OT-inspired regularization $\mathcal{R}$ augmenting the loss $\mathcal{L}$, reducing intermediate feature distribution discrepancy. This enables the complete removal of layers.

However, the progress enabled by these new models (consisting of billions of parameters) comes at the price of higher computational costs, consuming more energy, thus contributing to carbon emissions [62]. For instance, training a generative model is comparable to driving a car for 10km, while generating 10k samples is estimated to be equivalent to driving 160 km [56]. Although training costs are expensive, the open-sourcing of these foundation models multiplies inference costs across multiple users and contributes to carbon emissions in a significant manner. The need to reduce the environmental impact of these models at inference by proposing computational reduction is therefore apparent. Consequently, the rise of complexity-reduction approaches such as pruning [21], quantization [18], and knowledge distillation [25] is motivated by the need for more efficient architectures to alleviate their resource demands. Reducing deep neural network (DNN) complexity is not an easy task: generalization and model complexity are inextricably related [23], but since pre-trained models are often employed for downstream tasks, they tend to be over-parameterized.

This paper has been accepted for publication at the IEEE/CVF International Conference on Computer Vision 2025 (ICCV25).

This gives us hope: in principle, it is possible to compress these models without any (or only little) performance degradation [59]. This observation is further supported by the *collapse* phenomenon in neural networks, which has been observed at both the neuron [65] and layer levels [19]. On the one hand, individual neurons stop learning, leading to constant or trivial outputs, while on the other hand, entire layers fail to learn and become redundant or inactive. Consequently, these layers could, in principle, be removed.

Nevertheless, few methods are capable of removing entire layers from a neural network. Some of them have been designed to mitigate the depth of DNNs while maintaining performance, exploiting the deletion of several layers [34] or ad-hoc architectural search [2]. However, these methods are computationally challenging since they either require retraining or rely on huge search spaces, leading to significant search costs. Moreover, the focus of previous works is often more on removing non-linearities, leaving the fusion of the remaining consecutive linear layers to further research, which shows that this is not straightforward in many common cases [49].

Driven by the motivation of reducing the depth of DNNs, we leverage optimal transport (OT) [48, 61] to develop a framework allowing post-training the complete removal of layers from the architecture. Compared to existing OT-based frameworks incorporating it into neural architecture search [27, 44, 63] or knowledge distillation pipelines [7, 38], our strategy does not involve training more than one network but rather operates inside the model. In our case, we use OT to minimize the distributional changes between layers inside the same model, allowing us to strategically and efficiently remove layers (as showcased in Fig. 1). Overall, our contributions can be summarized as follows.

- We propose a novel OT-based and block collapse inductive regularization (Sec. 4), seamlessly integrated into the main training pipeline of neural networks. Our approach consists of minimizing a block-wise OT discrepancy measure, specifically the Max-Sliced Wasserstein distance, between the input and output features' probability distributions of the blocks of the network (Sec. 4.2).
- We motivate our strategy (Sec. 4.1 and Sec. 4.3) by showing how it allows, post-training, the complete removal of several blocks from the architecture at once.
- Our proposed regularization strategy demonstrates its effectiveness in reducing the depth of over-parameterized DNNs with marginal performance loss with respect to competing state-of-the-art techniques (Sec. 5.2).

2. Related Works

Neural network pruning. In the last decades, neural network pruning has risen as the one privileged approach to compress deep neural networks: complimentary to other popular approaches like quantization, it leads to heavy parameter reduction through the proper cut of groups of parameters (or filters in convolutional architectures) that are less important for the specific downstream task under exam. Its effectiveness is empirically certified by several works [4, 8, 22] and justified by the known overparametrization of such models [36]. Among these, we historically distinguish between *unstructured* pruning approaches that eliminate parameters without considering the neural network's structure [21, 59] and *structured pruning*, where entire channels, neurons, or filters are removed [22, 58].

Unstructured pruning methods are grouped into two main categories based on the nature of the importance score used to prune weights: *gradient*-based methods rank the parameters according to the gradient magnitude [32, 59] (or higher-order derivatives), while *magnitude*-based ones [21, 39, 64] use the weights' magnitude as a significance score to prune them. In a famous study, [4] compared the effectiveness of these two approaches, concluding that magnitude-based techniques are often more accurate than gradient-based ones while offering a better trade-off between complexity and competitiveness. Following up on this work, [17] even showcased that simple magnitude pruning methods can achieve results that are comparable to more complex ones, establishing a solid comparison baseline. Although some studies suggest that unstructured pruning may actually harbor structured effects [35], in general, they provide few practical benefits when deploying the neural network on generic computing resources [6].

Unlike unstructured pruning, structured pruning brings immediate advantages for both memory and computation, despite resulting in lower overall sparsity [6]. Despite this, when employing recent computing resources, removing entire filters on recent computing resources has only a marginal effect on the improved latency, given the availability of resources in parallel. The real bottleneck in parallel computation resides in the computational *critical path*[1] [41], which can be mitigated by reducing the model's depth. Although some existing approaches, like knowledge distillation to shallow student models [25] already tackle this issue, maintaining the performance cannot be guaranteed, as the optimal architecture of the target model is not known a priori, which may lead to significant performance loss.

Neural network depth reduction. Several recent works have proposed approaches to reduce the depth of DNNs. The most common practice is to *remove non-linearities* between layers: this (in principle) enables two successive layers to be merged together. Among these approaches, Layer Folding [14] is one of the earlier attempts: it eval-

[1] We refer to the critical path as the longest path, in terms of time or computational cost, through the computational graph that must be executed sequentially during inference, thus determining the model's minimum achievable latency.

uates whether non-linear activations can be discarded, replacing ReLUs with PReLU (having a trainable slope for the negative part). More recently, Entropy-Guided Pruning (EGP) [34] proposes to reduce the depth of DNNs by prioritizing the pruning of connections in layers with little use of the non-linearity (estimated through an entropic measure). On the same trend, NEPENTHE [35] improved EGP's entropy estimator and introduced a budget for the number of parameters to prune. Taking a more global approach, EASIER [51] was designed to determine the effect of removing a non-linearity, considering the introduced error at the output of the model, estimated through a validation set.

Although in principle effective in reducing the critical path length of DNNs, these methods do not provide significant gains in all cases, due to some impossibilities in merging consecutive linear layers. One example is the problem associated with ResNet-type architectures: if we have padding in the second convolutional layer, then there is no analytical solution for merging the two consecutive layers [49]. Moreover, if an activation is removed where there is a residual connection, no fusion can be applied. A major architectural change is necessary to observe a real impact in practice.

Unlike those works, our method does not rely on linearizing activations and merging two consecutive layers. Instead, LaCoOT minimizes the Max-Sliced Wasserstein distance between the input and output features' distributions of the blocks of the network. Post-training, layers having the lowest Max-Sliced Wasserstein distance are removed. Developed to be model-agnostic, we compare our method with these works and show its effectiveness in Sec. 5.

3. Preliminaries

3.1. Background on Optimal Transport

In this subsection, we present a succinct overview of OT and the Wasserstein distance for discrete distributions [48].

Given two metric spaces $\mathcal{X}$ and $\mathcal{Y}$ and a cost function c defined over $\mathcal{X} \times \mathcal{Y}$, the goal of the OT problem is to determine the most efficient manner to transport mass from one distribution, defined over $\mathcal{X}$ to another supported over $\mathcal{Y}$, where the transportation cost is dictated by the chosen function c.

For $\mathcal{X} = \mathcal{Y} = \mathbb{R}^d$, we consider two discrete probability measures and we recall the *Monge Formulation of the OT problem*:

$$OT(\mu, \nu, c) = \min_{T} \sum_{i} \alpha_i c[\boldsymbol{x}_i, T(\boldsymbol{x}_i)], \tag{1}$$

where μ and ν defined as:

$$\mu = \sum_{i=1}^{N} \alpha_i \delta_{\boldsymbol{x}_i}, \qquad \nu = \sum_{i=1}^{M} \beta_i \delta_{\boldsymbol{y}_i}, \tag{2}$$

where δ_x refers to the Dirac (unit mass) distribution at point $\boldsymbol{x}$. The weights α and β reside in the probability simplex $\{a \in \mathbb{R} | \sum a_i = 1\}$, and T is defined as $T : \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\} \to \{\boldsymbol{y}_1, \dots, \boldsymbol{y}_M\}$ and verifying:

$$\beta_j = \sum_{i:T(\boldsymbol{x}_i)=\boldsymbol{y}_j} \alpha_i, \quad \forall j \in [\![M]\!], \tag{3}$$

or more compactly $T_\sharp \mu = \nu$.

In the following, we will consider only the case of uniform weights and the same support size, taking $M = N$ and $\alpha_i = \beta_j = \frac{1}{N}$. We also take as the cost function $c(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|_p^p$ for $\boldsymbol{x} \in \mathcal{X}$, $\boldsymbol{y} \in \mathcal{Y}$, $p \in \mathbb{R}_{>0}$. In this case, OT establishes a measure of distance between the probability distributions. Such a distance, known as the **p-Wasserstein distance**, is in general defined as $\mathcal{W}_p = OT(\mu, \nu, c)^{\frac{1}{p}}$. When we have the dimension of the ground space being $d = 1$, the p-Wasserstein distance takes on a closed form, given by:

$$\mathcal{W}_p = \left(\frac{1}{N} \sum_{i=1}^{N} |\boldsymbol{x}_i - \boldsymbol{y}_i|^p \right)^{\frac{1}{p}}, \tag{4}$$

where we assume $x_1 < \dots < x_N$ and $y_1 < \dots < y_N$ such that $\boldsymbol{x}_i \mapsto \boldsymbol{y}_i, \forall i$.

Given the closed-form expression in one dimension, sliced variants of the p-Wasserstein distance have been introduced. These variants transform sample assignment and distance calculation by sorting the one-dimensional projection of the samples. This process yields a sufficient approximation of the high-dimensional p-Wasserstein distance, which is immune to the curse of dimensionality [57]. Specifically, our focus lies on the **p-Max-Sliced Wasserstein distance**, introduced in [12], and defined as follows:

$$\max \tilde{W}_p(\mu, \nu) = \max_{\theta \in \mathcal{U}(\mathbb{S}^{d-1})} \mathcal{W}_p(\theta_\sharp \mu, \theta_\sharp \nu), \tag{5}$$

where $\theta_\sharp$ stands for the pushforwards of the projection $X : \mathbb{R}^d \mapsto \langle \theta, X \rangle$, $\langle \cdot, \cdot \rangle$ for the dot product operator and $\mathcal{U}(\mathbb{S}^{d-1})$ for the uniform distribution on the unit hypersphere of dimension $d - 1$.

Essentially, the Max-Sliced Wasserstein distance represents a version of the sliced Wasserstein distance where we select the optimal direction to project the probability measures, *i.e.*, the direction along which the projected distance is maximized, also possessing valid metric properties [5, 42, 43]. In our work, we will consider the Max-Sliced Wasserstein distance, for its previously discussed convenience, specifically computed for $p = 2$, to quantify the distance between intermediate probability distributions between blocks inside a neural network model, as it will be presented in the next section.

3.2. Learning Framework

In this subsection, we introduce our learning framework. Let us define $\mathcal{T} = T_K \circ \cdots \circ T_1$ as the DNN we wish to train on the dataset $\mathcal{D}$, where each T_k is an elementary module (which can be defined as single or multiple layers). Given a loss function $\mathcal{L}$ we aim at minimizing, the objective entails minimizing the problem:

$$(\mathcal{T}, F) \in \arg\min_{\mathcal{T}, F} \sum_{i=1}^{|\mathcal{D}|} \mathcal{L}\left[(F \circ \mathcal{T})(\boldsymbol{x}_{1,i}), y_i\right], \qquad (6)$$

where F is a classifier layer, $\boldsymbol{x}_{1,i}$ the i-th input sample for the DNN, and y_i the associated ground-truth label and $|\mathcal{D}|$ the number of samples in the dataset. For each module T_k, we define the *input probability distribution* μ_k as:

$$\mu_k := \frac{1}{|\mathcal{D}|}\delta_{\boldsymbol{y}_{k-1,\mathcal{D}}} = \frac{1}{|\mathcal{D}|}\delta_{\boldsymbol{x}_{k,\mathcal{D}}} = \frac{1}{|\mathcal{D}|}\sum_{i=1}^{|\mathcal{D}|}\delta_{\boldsymbol{x}_{k,i}},$$

which is also the output of the preceding module T_{k-1}. Similarly, the *output probability distribution* ν_k is defined as:

$$\nu_k := \frac{1}{|\mathcal{D}|}\delta_{\boldsymbol{y}_{k,\mathcal{D}}} = \frac{1}{|\mathcal{D}|}\sum_{i=1}^{|\mathcal{D}|}\delta_{\boldsymbol{y}_{k,i}}.$$

According to our notation, $\nu_k \equiv \mu_{k+1}$, given that $\boldsymbol{y}_{k,i} = \boldsymbol{x}_{k+1,i} \forall i, k$. Furthermore, we assume that $\boldsymbol{x}_{k,i}$ and $\boldsymbol{y}_{k,i}$ possess identical dimensions, and consequently μ_k and ν_k to live in the same dimensional space, allowing for the computation of the distance between them. In the following, our goal would be to control these distances during training to allow for the isolation and removal of certain blocks post-training, with almost no loss of performance.

4. LaCoOT

In this section, we detail our method, LaCoOT, for reducing neural network depth using optimal transport (Sec. 4.1): we propose a regularization strategy based on the Max-Sliced Wasserstein distance to minimize the distance between intermediate feature distributions in the neural network. Fig. 1 provides a general overview of our method. We also present some insights into this strategy and how it enables the removal of intermediate layers in the network after training, with, in principle, minimal performance degradation (Sec. 4.3).

4.1. Proposed Regularization

Our objective is to reduce the depth of the neural network. To achieve such a goal, we will incorporate a penalization of the distance between two consecutive blocks during training. This is to assist the DNN in learning a target input-output function $\mathcal{T}$ while following the shortest path. This yields, after training, to identifying blocks that can be removed from the architecture without impacting the performance. Specifically, these blocks introduce marginal statistical modification on their corresponding input features. We include this constraint in the learning translates into minimizing, besides the loss $\mathcal{L}$, in the form of a regularizer:

$$\mathcal{R} = \frac{1}{K}\sum_{k=1}^{K} \max \tilde{W}_2(\hat{\mu}_k, \hat{\nu}_k), \qquad (7)$$

where the probability distributions $\hat{\mu}_k$ and $\hat{\nu}_k$ are the empirical counterparts of the previously defined distributions μ_k and ν_k. They are constructed over uniformly-weighted samples of a N-sized minibatch, and so defined by $\hat{\mu}_k = \frac{1}{N}\sum_{i=1}^{N}\delta_{\boldsymbol{x}_{k,i}}$, where $\boldsymbol{x}_{k,i}$ is taken as the flattened input vector of the block corresponding to the i-th element of the minibatch.

Post-training, if the distance $\widehat{\mathcal{R}}_k := \max \tilde{W}_2(\hat{\mu}_k, \hat{\nu}_k)$ falls below a fixed threshold ε, the corresponding block T_k can be pruned from the architecture. Namely, this threshold is related to a tolerated performance drop budget δ.

Algorithm 1 Our proposed method LaCoOT.

1: **function** LaCoOT($\boldsymbol{w}^{\text{INIT}}$, $\mathcal{D}$, λ, δ)
2: $\quad\boldsymbol{w} \leftarrow$Train($\boldsymbol{w}^{\text{INIT}}$, $\mathcal{D}_{\text{train}}$, λ)
3: $\quad$dense_acc $\leftarrow$Evaluate($\boldsymbol{w}$, $\mathcal{D}_{\text{val}}$)
4: $\quad$current_acc $\leftarrow$ dense_acc
5: $\quad$**while** (dense_acc - current_acc) $> \delta$ **do**
6: $\quad\quad\widehat{\mathcal{R}} = [\widehat{\mathcal{R}}_1, \widehat{\mathcal{R}}_2, ..., \widehat{\mathcal{R}}_K]$
7: $\quad\quad l \leftarrow$ argmin($\widehat{\mathcal{R}}$)
8: $\quad\quad T_l$ = Identity()
9: $\quad\quad$current_acc $\leftarrow$ Evaluate($\boldsymbol{w}$, $\mathcal{D}_{\text{val}}$)
10: $\quad$**end while**
11: $\quad$**return** $\boldsymbol{w}$
12: **end function**

4.2. Overview on the Procedure

Depicted in Alg. 1, we present here LaCoOT to remove the layers having the lowest Max-Sliced Wasserstein distances. Indeed, the layer having the lowest Max-Sliced Wasserstein distance is likely to have a function close to the identity function. Therefore, this layer can be linearized, as keeping it is unnecessary, as illustrated in Tab. 2 in Supp. Mat. Aiming at this, we first train the neural network, represented by its weights at initialization $\boldsymbol{w}^{\text{INIT}}$ on the training set $\mathcal{D}_{\text{train}}$ with our regularization set by λ (line 2) and evaluate it on the validation set $\mathcal{D}_{\text{val}}$ (line 3). We then calculate the Max-Sliced Wasserstein distance $\widehat{\mathcal{R}}_k$ for each considered layer k for all the K considered layers, collected in the vector $\mathcal{R}$ (line 6), following Eq. 7. We then find the layer having the lowest Max-Sliced Wasserstein distance, represented by its index l (line 7) and replace it with the Identity (line 8).

In the following steps, this layer is, obviously, no longer taken into consideration. The performance of the model is re-evaluated on the validation set $\mathcal{D}_{\text{val}}$ (line 9). Once the performance on the validation set drops below the threshold δ, the final model is obtained.

4.3. Properties of the proposed regularization

The regularization acts like a soft 1-Lipschitz constraint. By looking closer into μ_k and ν_k (the input and output distributions of the k-th block), the central limit theorem suggests that μ_k can be regarded asymptotically as a Gaussian distribution with mean $\boldsymbol{m}_k$ and covariance Σ_k. Then, by employing the delta method, ν_k can be approximated asymptotically as a Gaussian distribution with mean $T_k(\boldsymbol{m}_k)$ and covariance $J_k^T \Sigma_k J_k$, where J_k represents the Jacobian matrix of the block transformation T_k.

The constraint $\mu_k = \nu_k$ can be interpreted as an orthogonality constraint on the Jacobian of the block transformation, indicating that our regularization imposes a similar effect as enforcing orthogonality on the Jacobian. This implies a block-wise soft Lipschitz constraint on the neural network by preserving gradient norms that drive the network to be 1-Lipschitz. This type of constraint was investigated in the literature [1, 3, 33], and it has been particularly shown in [3] that a 1-Lipschitz constraint does not limit the expressiveness, *i.e*. the capacity and learning flexibility of a neural network, for classification tasks. Instead, this regularization should offer a different stance on the trade-off between generalization and accuracy. This also coincides with the results in [28] that highlight the generalization-enhancing effect of such a regularization. During training, provided that a proper weight on the regularization (through some hyperparameter λ) is tuned, the neural network's expressive power should, in principle, remain intact, while adhering to the least action principle, thereby preventing arbitrary amplification of small differences and big distributional changes.

Stationary point analysis. The whole optimization problem can be expressed as:

$$\mathcal{J} = \mathcal{L} + \lambda \mathcal{R} \Rightarrow \frac{\partial \mathcal{J}}{\partial w_{k_0}} = \frac{\partial \mathcal{L}}{\partial w_{k_0}} + \lambda \frac{\partial \mathcal{R}}{\partial w_{k_0}}, \quad k_0 \in [\![K]\!], \tag{8}$$

where λ is a positive hyperparameter. We then characterize the stationary point as:

$$\frac{\partial \mathcal{L}}{\partial w_{k_0}} + \lambda \frac{\partial \mathcal{R}}{\partial w_{k_0}} = 0 \Rightarrow \lambda = -\frac{\partial \mathcal{L}}{\partial w_{k_0}} \cdot \frac{1}{\frac{\partial \mathcal{R}}{\partial w_{k_0}}}. \tag{9}$$

From this equation, since $\lambda \geq 0$, we clearly observe that the loss and the regularizer are antagonists. Hence, while browsing the parameter space to minimize the loss, unrestricted DNNs are rather biased towards increasing intermediate distributional changes in the path they take. These changes, which can be evaluated when taking the inter-block distances in the vanilla setting, might be irrelevant: the DNN can converge to another local minimum in the loss landscape, with similar performance but without undergoing too many distributional changes.

In the learning process, we recall that the primary objective is to traverse the gap between the input distribution and the target output distribution. A crucial threshold is thus reached when the network's output converges to the ground truth. Namely, this inherent distance between the input and ground truth distribution defines a tight lower bound for the regularization value, corresponding to the minimal distributional changing capacity that still has to be maintained in the network to have a good performance and not to underfit. This is guaranteed by applying the Triangle inequality:

$$\max \tilde{W}_2(\mu_1, \nu_{GT}) \leq \sum_{k=1}^{K} \max \tilde{W}_2(\mu_k, \nu_k) + \max \tilde{W}_2(\nu_K, \nu_{GT}), \tag{10}$$

where ν_{GT} represents the ground truth label distribution.

5. Experiments

In this section, we empirically evaluate the effectiveness of our proposed approach across multiple architectures and datasets for traditional image classification setups and extend it to image generation.

5.1. Experimental setup

Networks and Datasets. On image classification setups, we test LaCoOT on three widely used models, ResNet-18, MobileNet-V2, and Swin-T trained on seven different datasets: CIFAR-10 [30], Tiny-ImageNet [31], PACS and VLCS from DomainBed [20], as well as Flowers-102 [45], DTD [9], and Aircraft [40]. To showcase its applicability to larger models on diverse tasks, we employ DiT-XL/2 [47], finetuned on ImageNet [11] for image generation. For all our experiments, we use the implementation of the Max Sliced Wasserstein distance available in the POT toolbox [16]. LaCoOT is only applied in subsequent blocks having the same dimensionality: this results in a subset of 4, 12, 12, and 28 blocks considered removable for ResNet-18, Swin-T, MobileNetV2, and DiT-XL/2, respectively.

Baselines. We compare our method with leading depth-reducing methods: Layer Folding [14], EGP [34], NEPENTHE [35], and EASIER [51]. The hyperparameters, augmentation techniques, and learning policies are presented in Supp. Mat., mainly following [50] and [51].

5.2. Results

5.2.1. Image Classification

Fig. 2 displays the test performance (top-1) as a function of the critical path length for CIFAR-10 and Tiny-

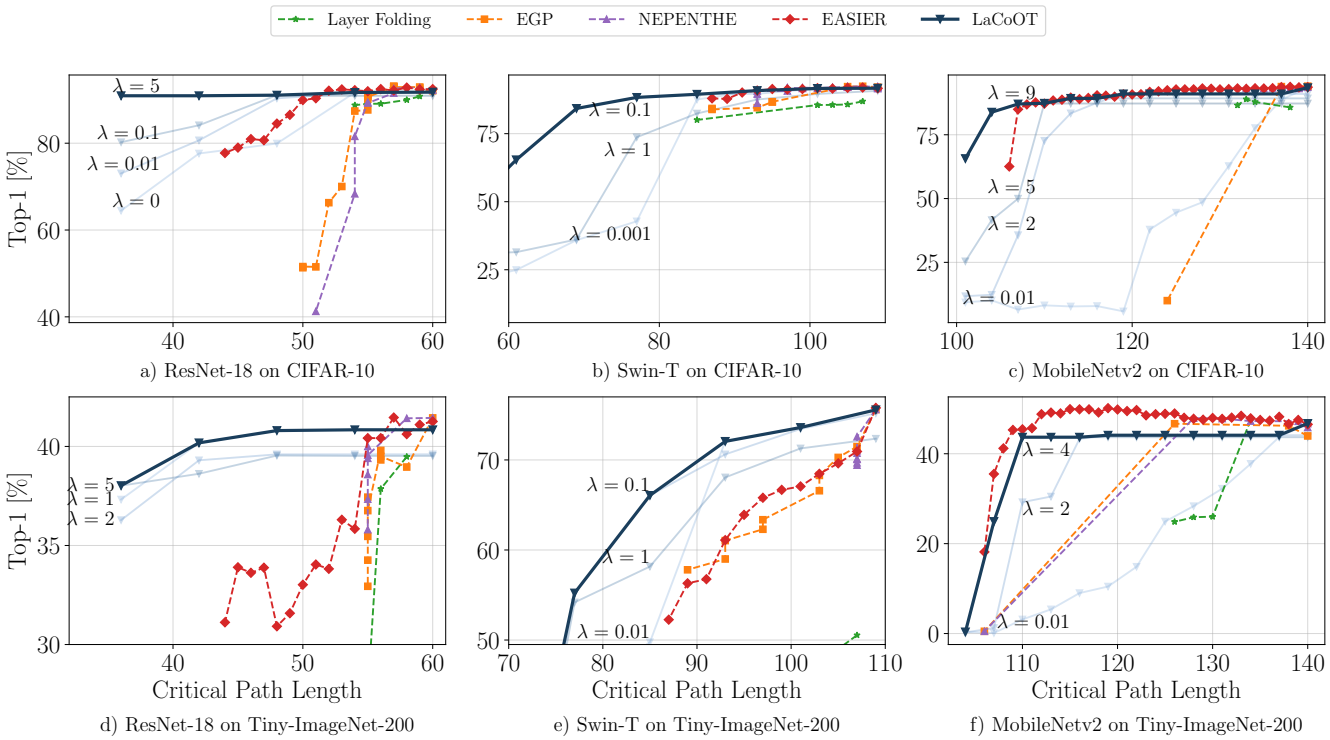

Figure 2. Test performance (Top-1 [%]) in function of the Critical Path Length for ResNet-18 (a,d), Swin-T (b,e) and MobileNetv2 (c,f) trained on CIFAR-10 (a,b,c) and Tiny-ImageNet-200 (d,e,f). For each dataset/architecture, we showcase the results achieved by LaCoOT for different values of λ, forming in **dark blue** the pareto frontier of our technique. *Top left corner is the best.*

ImageNet-200. The results achieved on other setups can be found in Sec. F in the Supp. Mat. Moreover, Fig. 5 in Supp. Mat. illustrates the relationship between Critical Path Length (CPL) and practical resource consumption: as the CPL decreases, both the inference time and the number of MACs (multiply-accumulate operation) decrease, indicating improved computational efficiency and faster inference speeds.

Critical Path Length. First, we can observe in Fig. 2 that the baseline methods showcase longer critical path lengths with respect to our method. Indeed, as discussed in Sec. 2, most methods have been focusing on removing non-linearities from the networks, leaving the fusion of subsequent layers as future work. However, while for Swin-T the fusion of two linear layers in the MLP block is straightforward, it is not the case when ResNet is employed: we recall that there is no analytical solution for merging two consecutive convolutional layers when padding is employed in the second one [49]. Hence, even if multiple non-linearities are removed from the network with these baselines, the critical path length only slowly decreases. Unlike these methods, LaCoOT focuses directly on full blocks divided by skip connections, hence quickly lowering the critical path length.

LaCoOT effectiveness. First, in most setups, we can observe LaCoOT's effectiveness. Indeed, for short critical path length, LaCoOT is the method performing overall the

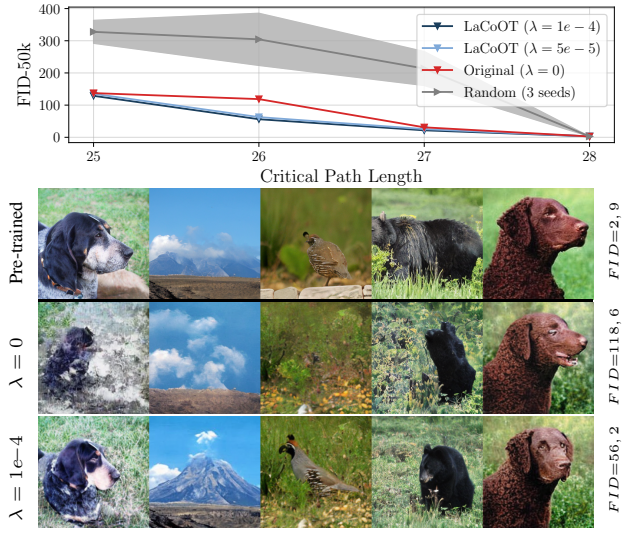

Figure 3. FID-50k as a function of the critical path length achieved by a DiT-XL/2 finetuned on ImageNet. It consistently achieves lower FID when finetuned with LaCoOT, even halving the FID when two DiT blocks are removed. The generated content is also better preserved (images for critical path length 26).

best, achieving a new Pareto Frontier when λ is increasing. For instance, for ResNet-18 trained on CIFAR-10 (Fig. 2a), our method reduces the critical path length even further, whereas previous methods could not. Besides, for Swin-T on Tiny-ImageNet-200 (Fig. 2e), LaCoOT sometimes outperforms current methods by 10% for the same critical path length. Additionally, looking at longer critical path lengths, we can observe that when λ is decreasing, LaCoOT achieves comparable results to other baselines.

We also report an issue faced with EGP. Indeed, by forcing a layer to have zero entropy, this method could prune it entirely, hence preventing the signal from passing through this layer, and thus causing the algorithm to completely fail. This is what is observed with the MobileNetv2 architecture on CIFAR-10, Flowers-102, DTD, or on Aircraft: from the first iteration, EGP prunes the last single layer before the classifier head entirely, leading to its complete removal, which completely cuts the information flow in the network, since there is no skip or residual connection at this stage.

Furthermore, on parameter-efficient architectures (such as MobileNetv2), we can observe that EASIER performs the best (Fig. 2f), while our method achieves comparable results (Fig. 2c). However, the advised reader will be able to put these results into perspective with the aim of EASIER, which focuses solely on removing non-linearities (as mentioned in Sec. 2). Moreover, the iterative nature of EASIER results in very few benefits in practice, as shown in Sec. 5.3. For instance, to achieve a path length of 105 on MobileNetv2, EASIER needs to carry out 34 trainings, whereas our method requires only one.

Comparison with the original model. Although in certain setups, such as ResNet-18 on CIFAR-10, LaCoOT effectively reduces model size while maintaining the original model's performance, it often results in some performance degradation compared to the original model. This is likely because the model is not re-trained after layer removal. In contrast, traditional compression schemes typically involve re-training the model after dropping some parameters to recover performance. We show in Tab. 8 in Supp. Mat. that the model can recover performance with a healing phase.

5.2.2. Image Generation

Unlike other baselines, which cannot scale due to their iterative nature, we show the possible extension of our method to foundation models by finetuning a pre-trained DiT-XL/2 on ImageNet with our method LaCoOT for 5k training steps. For 50k generated samples of size 256×256 with a classifier-free guidance scale of 1.5, Fig. 3 displays the FID-50k score depending on the critical path length, as well as examples of generated samples from the pre-trained model, and from models with two DiT blocks removed.

While no difference is observed when looking at the FID-50k score for the original model (critical path length of 28), the effect of our technique is visible when DiT blocks are removed: LaCoOT consistently achieves a lower FID-50k compared to the pre-trained model. For instance, when two DiT blocks are removed, we observe that the FID-50k score is twice as low. This is reflected in the quality of the generated images: the dog, the volcano, the bird, and the bear are not really visible, while the last dog contains artifacts. Indeed, the removal of blocks completely destroys generated images in the absence of the regularization, while the generated content is better preserved with its use. Therefore, as it required only a few fine-tuning steps, our approach LaCoOT can be suitable for foundation models.

Approach	top-1 [%]	MACs [M]	Inference time [ms]	Time
Original	91.77	140.19	7.90 ± 0.43	30'
Layer Folding	88.76	147.53	9.89 ± 1.11	160'
EGP	90.64	140.19	7.62 ± 0.20	376'
NEPENTHE	89.26	140.19	7.71 ± 0.40	288'
EASIER	90.35	140.19	7.07 ± 0.18	533'
LaCoOT	**90.99**	**64.69**	$\mathbf{4.78 \pm 0.34}$	**40'**

Table 1. Test performance (top-1), MACs, inference time on a NVIDIA A4500 and training time for ResNet-18 trained on CIFAR-10. Original refers to the trained model without layer deletion. The best results between Layer Folding, EGP, NEPENTHE, EASIER, and LaCoOT are in **bold**.

5.3. Practical Benefits

In this subsection, we showcase the practical benefits of our approach in terms of inference time as well as the efficiency of LaCoOT.

Tab. 1 shows the test performance, MACs, and inference time on an NVIDIA A4500, as well as the training time for a ResNet-18 trained on CIFAR-10 for all the considered approaches. The same analysis for Swin-T and MobileNetv2 is conducted in Sec. D in the Supp. Mat. As anticipated in Sec. 2, we observe that baseline methods do not reduce MACs in practice, as they simply rely on non-linearities removal without providing insights on how to merge consecutive layers. Unlike its competitors, LaCoOT produces a model whose inference has been reduced by 40%. Moreover, looking at the training time, LaCoOT is the most efficient. In Tab. 4 in the Supp. Mat., the same analysis is conducted for MobileNetv2 on CIFAR-10, corresponding to Fig. 2c. While Layer Folding, EGP, and NEPENTHE showcase performance drop at high critical path length, we achieve comparable performance as EASIER for the same critical path length in 20× less time, with real practical benefits since the inference time is reduced.

5.4. Ablation Study

In this subsection, we study the impact of λ, which balances the strength of our regularizer. Moreover, to validate the effectiveness of our proposed importance metric for layer removal, we compare it with two alternative methods for

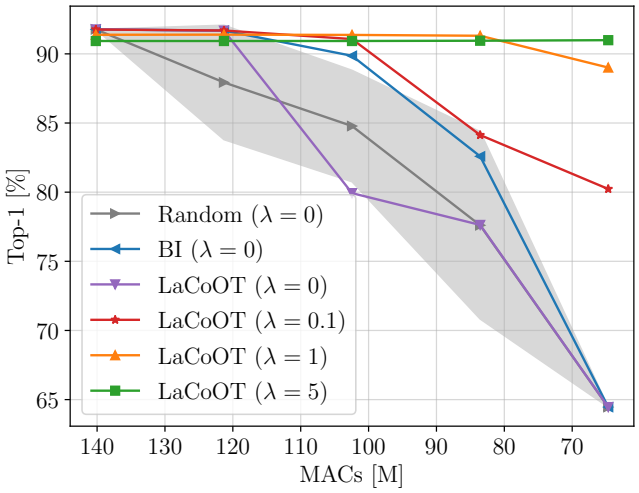

Figure 4. Comparison of LaCoOT, BI (theoretical best), and Random for ResNet-18 on CIFAR-10. LaCoOT ($\lambda = 5$) halves the MACs with minimal performance loss. Higher λ values further reduce MACs while maintaining performance.

selecting which layers to remove. Fig. 4 shows this comparison on a ResNet-18 on CIFAR-10.

Impact of λ. First, in the absence of regularization during training (*i.e.*, with $\lambda = 0$), we can observe that the Max-Sliced Wasserstein distance is not a faithful indicator of block importance, since it can be surpassed by random block removal. Considering our previous theoretical analysis in Sec. 4.3, this observation is largely expected, since without our regularization, the blocks operate changes on the intermediate features' distribution, which is unnecessary. Indeed, when our regularization is incorporated into the training process, the unnecessary distributional changes are minimized, and our metric becomes a reliable basis for ranking the importance of the model blocks. Looking at Fig. 4, the higher λ, the more blocks can be removed without harming performance. Indeed, we can observe that LaCoOT ($\lambda = 5$) halves the number of MACs with almost no performance drop to the dense model. From Eq. 8, selecting λ depends on the user's goal, since there is a trade-off between performance and complexity. In general, the higher the lambda, the more layers can be removed, but *potentially* the lower the performance.

Choice of the importance score. LaCoOT is here compared with two alternative methods for selecting which layers to remove. In the first scenario that we refer to as "block influence" (BI), we remove layers depending on their impact on the performance : we remove one layer at a time by selecting the one impacting the performance the least. In a second scenario, referred to as "Random", we remove one layer at a time by selecting it randomly. Based on randomness, error bars have been calculated on 10 seeds. From Fig. 4, we can observe that LaCoOT achieves a better performance/compression trade-off compared to the two other approaches Block Influence and Random, which validates the choice of our importance score.

5.5. Limitations and Future Work

While effective in alleviating the computational burden of DNNs, LaCoOT also has some limitations, highlighting opportunities for future improvements and research, as discussed below.

Performance degradation. Compressing existing parameter-efficient architectures is particularly challenging, a common issue in the field of model compression. Indeed, LaCoOT struggles to reduce the depth of an already underfitted architecture without compromising performance, as seen with MobileNetv2 on Tiny-ImageNet-200. This highlights the challenges of further compressing already efficient architectures and the need to carefully manage the trade-off between model depth and performance. However, LaCoOT effectively reduces the depth of over-fitted DNNs, especially given that only one training is required to achieve compression.

Extension of LaCoOT with the Gromov-Wasserstein distance. LaCoOT was primarily designed to operate on layers where the Max-Sliced Wasserstein distance can be computed directly. While this distance requires matching dimensions between distributions, our experiments in Sec. E in Supp. Mat. show that LaCoOT remains effective even for layers with mismatched dimensions, such as convolutional layers that modify the number of filters or feature sizes. However, we believe that a more principled treatment of such cases—potentially leveraging the Gromov-Wasserstein distance [60], which allows for comparing distributions whose supports do not necessarily lie in the same metric space—remains an avenue for future research.

6. Conclusion

In this work, we have proposed LaCoOT, a new optimal transport-based regularization strategy. Specifically, we use the Max-Sliced Wasserstein distance to minimize the distances between the intermediate feature distributions in the neural network. This regularization enables, post-training, the complete removal of layers from the architecture with a minor impact on performance. Experiments conducted on three widely used architectures across seven image classification datasets have demonstrated LaCoOT's capability and effectiveness in reducing the number of layers in the neural network. Unlike other approaches that rely on an iterative scheme, we have shown that extending LaCoOT to foundation models is possible since our approach requires only a few fine-tuning steps. Concerned about the increasing environmental impact of AI, we hope this work will inspire future optimization techniques and new approaches for network compression.

Acknowledgements

This work was supported by the French National Research Agency (ANR) in the framework of the JCJC project "BANERA" under Grant ANR-24-CE23-4369, from the European Union's Horizon Europe Research and Innovation Programme under grant agreement No. 101120237 (ELIAS), and by the Hi! PARIS Center on Data Analytics and Artificial Intelligence. Zhu Liao acknowledges financial support from the China Scholarship Council (CSC).

References

[1] Cem Anil, James Lucas, and Roger Baker Grosse. Sorting out lipschitz function approximation. In *ICML*, 2018. 5

[2] Dilyara Baymurzina, Eugene Golikov, and Mikhail Burtsev. A review of neural architecture search. *Neurocomputing*, 474:82–93, 2022. 2

[3] Louis Béthune, Thibaut Boissin, Mathieu Serrurier, Franck Mamalet, Corentin Friedrich, and Alberto Gonzalez Sanz. Pay attention to your loss : understanding misconceptions about lipschitz neural networks. In *NeurIPS*, 2022. 5

[4] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning? *MLSys*, 2020. 2

[5] Nicolas Bonneel, Julien Rabin, Gabriel Peyré, and Hanspeter Pfister. Sliced and Radon Wasserstein Barycenters of Measures. *Journal of Mathematical Imaging and Vision*, 2015. 3

[6] Andrea Bragagnolo, Enzo Tartaglione, Attilio Fiandrotti, and Marco Grangetto. On the role of structured pruning for neural network compression. In *ICIP*, 2021. 2

[7] Liqun Chen, Dong Wang, Zhe Gan, Jingjing Liu, Ricardo Henao, and Lawrence Carin. Wasserstein contrastive representation distillation. In *CVPR*, 2021. 2

[8] Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. *IEEE PAMI*, 2024. 2

[9] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. Describing textures in the wild. In *CVPR*, 2014. 5

[10] Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Peter Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, et al. Scaling vision transformers to 22 billion parameters. In *ICML*, 2023. 1

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009. 5

[12] Ishani Deshpande, Yuan-Ting Hu, Ruoyu Sun, Ayis Pyrros, Nasir Siddiqui, Oluwasanmi Koyejo, Zhizhen Zhao, David Alexander Forsyth, and Alexander G. Schwing. Max-sliced wasserstein distance and its use for gans. *CVPR*, 2019. 3, 4

[13] Enmao Diao, Ganghua Wang, Jiawei Zhang, Yuhong Yang, Jie Ding, and Vahid Tarokh. Pruning deep neural networks from a sparsity perspective. In *ICLR*, 2023. 3

[14] Amir Ben Dror, Niv Zehngut, Avraham Raviv, Evgeny Artyomov, Ran Vitek, and Roy Jevnisek. Layer folding: Neural network depth reduction using activation linearization. *BMVC*, 2022. 2, 5

[15] Gongfan Fang, Xinyin Ma, Mingli Song, Michael Bi Mi, and Xinchao Wang. Depgraph: Towards any structural pruning. In *CVPR*, 2023. 3

[16] Rémi Flamary, Nicolas Courty, Alexandre Gramfort, Mokhtar Z. Alaya, Aurélie Boisbunon, Stanislas Chambon, Laetitia Chapel, Adrien Corenflos, Kilian Fatras, Nemo Fournier, Léo Gautheron, Nathalie T.H. Gayraud, Hicham Janati, Alain Rakotomamonjy, Ievgen Redko, Antoine Rolet, Antony Schutz, Vivien Seguy, Danica J. Sutherland, Romain Tavenard, Alexander Tong, and Titouan Vayer. Pot: Python optimal transport. *Journal of Machine Learning Research (JMLR)*, 2021. 5, 1, 4

[17] Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019. 2

[18] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. In *Low-Power Computer Vision*, pages 291–326. Chapman and Hall/CRC, 2022. 1

[19] Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A Roberts. The unreasonable ineffectiveness of the deeper layers. *ICLR*, 2025. 2

[20] Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. In *ICLR*, 2020. 5

[21] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *NeurIPS*, 2015. 1, 2

[22] Yang He and Lingao Xiao. Structured pruning for deep convolutional neural networks: A survey. *IEEE PAMI*, 2023. 2

[23] Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017. 1

[24] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *NeurIPS*, 2017. 8

[25] Geoffrey Hinton. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 1, 2

[26] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *ICML*, 2021. 1

[27] Kirthevasan Kandasamy, Willie Neiswanger, Jeff Schneider, Barnabas Poczos, and Eric P Xing. Neural architecture search with bayesian optimisation and optimal transport. In *NeurIPS*, 2018. 2

[28] Skander Karkar, Ibrahim Ayed, Emmanuel de Bezenac, and Patrick Gallinari. A Principle of Least Action for the Training of Neural Networks. In *ECML PKDD*, 2020. 5

[29] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *ICCV*, 2023. 1

[30] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images, 2009. 5

[31] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015. 5

[32] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip Torr. Snip: Single-shot network pruning based on connection sensitivity. In *ICLR*, 2019. 2

[33] Qiyang Li, Saminul Haque, Cem Anil, James Lucas, Roger Baker Grosse, and Joern-Henrik Jacobsen. Preventing gradient attenuation in lipschitz constrained convolutional networks. *NeurIPS*, 2019. 5

[34] Zhu Liao, Victor Quétu, Van-Tam Nguyen, and Enzo Tartaglione. Can unstructured pruning reduce the depth in deep neural networks? In *ICCV*, 2023. 2, 3, 5, 8

[35] Zhu Liao, Victor Quétu, Van-Tam Nguyen, and Enzo Tartaglione. Nepenthe: Entropy-based pruning as a neural network depth's reducer. *arXiv preprint arXiv:2404.16890*, 2024. 2, 3, 5

[36] Lucas Liebenwein, Cenk Baykal, Brandon Carter, David Gifford, and Daniela Rus. Lost in pruning: The effects of pruning neural networks beyond test accuracy. *Proceedings of Machine Learning and Systems*, 3:93–138, 2021. 2

[37] Jing Liu, Bohan Zhuang, Zhuangwei Zhuang, Yong Guo, Junzhou Huang, Jinhui Zhu, and Mingkui Tan. Discrimination-aware network pruning for deep model compression. *TPAMI*, 2021. 3

[38] Suhas Lohit and Michael Jones. Model compression using optimal transport. In *WACV*, 2022. 2

[39] Christos Louizos, Max Welling, and Diederik P. Kingma. Learning sparse neural networks through l_0 regularization. In *ICLR*, 2018. 2

[40] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft, 2013. 5

[41] Christian HX Ali Mehmeti-Göpel and Jan Disselhoff. Nonlinear advantage: trained networks might not be as complex as you think. In *ICML*, 2023. 2

[42] Kimia Nadjahi, Alain Durmus, Lénaïc Chizat, Soheil Kolouri, Shahin Shahrampour, and Umut Simsekli. Statistical and topological properties of sliced probability divergences. *NeurIPS*, 2020. 3

[43] Kimia Nadjahi, Alain Durmus, Pierre Jacob, Roland Badeau, and Umut Simsekli. Fast approximation of the sliced-wasserstein distance using concentration of random projections. In *NeurIPS*, 2021. 3

[44] Vu Nguyen, Tam Le, Makoto Yamada, and Michael A Osborne. Optimal transport kernels for sequential and parallel neural architecture search. In *ICML*, 2021. 2

[45] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*, 2008. 5

[46] Gaurav Parmar, Richard Zhang, and Jun-Yan Zhu. On aliased resizing and surprising subtleties in gan evaluation. In *CVPR*, 2022. 8

[47] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, 2023. 1, 5, 8

[48] Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019. 2, 3

[49] Giommaria Pilo, Nour Hezbri, André Pereira e Ferreira, Victor Quétu, and Enzo Tartaglione. Layerfold: A python library to reduce the depth of neural networks. *SoftwareX*, 2025. 2, 3, 6

[50] Victor Quétu and Enzo Tartaglione. Dsd^2: Can we dodge sparse double descent and compress the neural network worry-free? In *AAAI*, 2024. 5, 8

[51] Victor Quétu, Zhu Liao, and Enzo Tartaglione. The simpler the better: An entropy-based importance metric to reduce neural networks' depth. In *ECML PKDD*, 2024. 3, 5

[52] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 1

[53] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1 (2):3, 2022. 1

[54] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 1

[55] Jonathan S Rosenfeld. Scaling laws for deep learning. *arXiv preprint arXiv:2108.07686*, 2021. 1

[56] Marvin Seyfarth, Salman Ul Hassan Dar, and Sandy Engelhardt. Latent pollution model: The hidden carbon footprint in 3d image synthesis. In *International Workshop on Simulation and Synthesis in Medical Imaging*, 2024. 1

[57] Nian Si, Jose Blanchet, Soumyadip Ghosh, and Mark Squillante. Quantifying the empirical wasserstein distance to a set of measures: Beating the curse of dimensionality. In *NeurIPS*, 2020. 3

[58] Enzo Tartaglione, Andrea Bragagnolo, Francesco Odierna, Attilio Fiandrotti, and Marco Grangetto. Serene: Sensitivity-based regularization of neurons for structured sparsity in neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 33(12):7237–7250, 2021. 2

[59] Enzo Tartaglione, Andrea Bragagnolo, Attilio Fiandrotti, and Marco Grangetto. Loss-based sensitivity regularization: towards deep sparse neural networks. *Neural Networks*, 2022. 2

[60] Vayer Titouan, Rémi Flamary, Nicolas Courty, Romain Tavenard, and Laetitia Chapel. Sliced gromov-wasserstein. In *NeurIPS*, 2019. 8

[61] Cédric Villani et al. *Optimal transport: old and new*. Springer, 2009. 2

[62] Carole-Jean Wu, Ramya Raghavendra, Udit Gupta, Bilge Acun, Newsha Ardalani, Kiwan Maeng, Gloria Chang, Fiona Aga, Jinshi Huang, Charles Bai, Michael Gschwind, Anurag Gupta, Myle Ott, Anastasia Melnikov, Salvatore Candido, David Brooks, Geeta Chauhan, Benjamin Lee, Hsien-Hsin

Lee, Bugra Akyildiz, Maximilian Balandat, Joe Spisak, Ravi Jain, Mike Rabbat, and Kim Hazelwood. Sustainable ai: Environmental implications, challenges and opportunities. In *Proceedings of Machine Learning and Systems*, pages 795–813, 2022. 1

[63] Jiechao Yang, Yong Liu, and Hongteng Xu. Hotnas: Hierarchical optimal transport for neural architecture search. In *CVPR*, 2023. 2

[64] Michael H. Zhu and Suyog Gupta. To prune, or not to prune: Exploring the efficacy of pruning for model compression. In *ICLR*, 2018. 2

[65] Zhihui Zhu, Tianyu DING, Jinxin Zhou, Xiao Li, Chong You, Jeremias Sulam, and Qing Qu. A geometric analysis of neural collapse with unconstrained features. In *NeurIPS*, 2021. 2

LaCoOT: Layer Collapse through Optimal Transport

Supplementary Material

A. Gradients of the regularizer

Herein, we provide further details on the regularizer, namely, by deriving its gradients.

Namely, $\forall k_0 \in [\![K]\!]$,

$$\mathcal{R}_{k_0} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\theta^T \boldsymbol{x}_{k_0,i} - \theta^T T_{k_0}(\boldsymbol{x}_{k_0,i}; w_{k_0}))^2} \quad (11)$$

In our setting, we are also assuming that $T_{k_0}(\boldsymbol{x}_{k_0,i}; w_{k_0}) = \boldsymbol{x}_{k_0,i} + f_{k_0}(\boldsymbol{x}_{k_0,i}; w_{k_0})$. Hence, we make use of this expression to derive the following analytical expression for $\frac{\partial \mathcal{R}_{k_0}}{\partial w_{k_0}}$.

$$\frac{\partial \mathcal{R}_{k_0}}{\partial w_{k_0}} = \frac{1}{N\mathcal{R}_{k_0}}\sum_{i=1}^{N}(\theta^T f_{k_0}(\boldsymbol{x}_{k_0,i}; w_{k_0}))\frac{\partial \theta^T f_{k_0}(\boldsymbol{x}_{k_0,i}; w_{k_0})}{\partial w_{k_0}} \quad (12)$$

$$\Rightarrow \frac{\partial \mathcal{R}}{\partial w_{k_0}} = \frac{1}{K}\sum_{k=k_0}^{K}\frac{\partial \mathcal{R}_k}{\partial w_{k_0}} \quad (13)$$

Computing the Wasserstein distance requires a sorting oracle, even in one dimension. In practice, backpropagation works through subgradients of the sorting operation. While argsort is non-differentiable, by leveraging automatic differentiation with PyTorch, in the POT library [16] gradients flow through the *take_along_axis* operation, treating sorting indices as constants during backpropagation. This is sufficient because the Wasserstein distance remains well-defined even at points where the sorting order changes.

B. Correlation between Wasserstein distance and performance degradation

In Tab. 2, we analyze the correlation between the Wasserstein distance for each considered block (B1, B2, B3, B4), and performance degradation for a ResNet-18 on CIFAR-10. The higher λ, the lower the distances, thus the more likely the block has a function close to the identity, and therefore the higher the performance since removing a block close to identity does not lead to any changes. Indeed, if the Wasserstein distance is zero, by definition the output matches the input and therefore the whole block encodes the identity function. The higher the distance is, the larger the perturbation introduced will be, increasing the risk of performance loss. Consequently, as λ increases, performance improves since removing a block functioning close to identity has minimal impact on the model's behavior.

λ	B1	B2	B3	B4	Mean	top-1
0.01	0.210	0.126	0.104	0.044	0.121	73.04
0.1	0.068	0.074	0.036	0.016	0.048	80.23
1	4.63e-4	0.0246	5.80e-3	2.59e-4	7.78e-3	89.01
5	4.41e-4	7.93e-3	4.35e-4	1.34e-4	2.23e-3	90.99

Table 2. Correlation between the Wasserstein distance of each block (B1, B2, B3, B4) and the performance for a ResNet-18 on CIFAR-10.

C. Relationship between critical path length and practical resource consumption

In this section, we show that optimizing the critical path length can lead to significant improvements in both inference speed and computational efficiency. Indeed, Fig. 5 demonstrates the relationship between critical path length, inference time (measured on a NVIDIA A4500) and computational complexity (MACs) for a ResNet-18 on CIFAR-10.

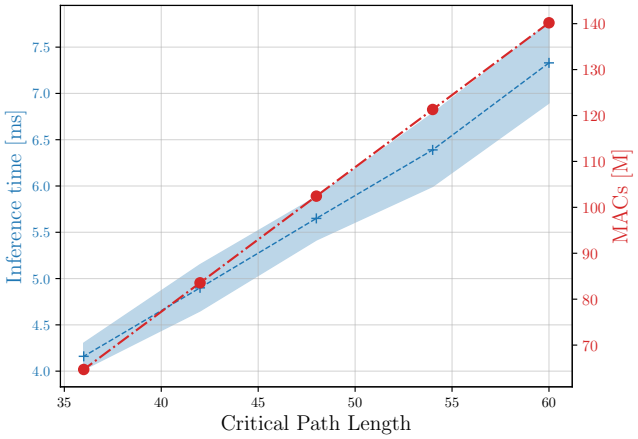

Figure 5. Relationship between Critical Path Length (CPL), inference time, and Multiply-Accumulate Operations (MACs) for a ResNet-18 model on the CIFAR-10 dataset. As the CPL decreases, both the inference time and the number of MACs decrease, indicating improved computational efficiency and faster inference speeds.

This analysis highlights a trade-off between computational complexity and inference speed. Reducing the critical path length leads to both faster inference times and fewer computational operations: shorter critical paths are more efficient in terms of both time and computational resources.

D. Practical Benefits

Following the analysis conducted in Sec. 5.3, in this subsection, we showcase the practical benefits of our approach in terms of efficiency as well as inference time.

To clarify the role of block eligibility in our method, Tab. 3 reports, for each tested backbone: the total number of blocks (# Blocks), the number of blocks satisfying the equal-shape criterion and thus eligible for MSW scoring (# MSW), the number of blocks ultimately pruned (# Pruned), along with the resulting percentage reductions in latency and MACs, and the resulting Top-1 accuracy on CIFAR-10. Tab. 3 provides a detailed quantification of block eligibility and its impact on both efficiency and performance. For the cases where this assumption does not hold, we describe a workaround in Sec. E.

Backbone	# Blocks	# MSW	# Pruned	Latency	MACs	Top-1
ResNet-18	8	4	4	-39.50 %	-53.86 %	90.99
Swin-T	12	12	3	-38.48 %	-61.55 %	89.47
MobileNetv2	17	12	10	-23.66 %	-3.81 %	87.25

Table 3. Block eligibility and impact of pruning on CIFAR-10.

Tab.4 shows the test performance, MACs, and inference time on an NVIDIA A4500, as well as the training time for a MobileNetv2 trained on CIFAR-10 for all the considered approaches.

Approach	top-1 [%]	MACs [M]	Inference time [ms]	Time
Original	93.50	87.98	13.57 ± 0.82	112’
Layer Folding	86.56	87.98	20.29 ± 0.19	529’
EGP	9.70	87.98	13.38 ± 0.18	732’
NEPENTHE	86.75	87.98	13.29 ± 0.24	3165’
EASIER	87.19	87.98	13.22 ± 0.54	3514’
LaCoOT	**87.25**	**84.63**	$\mathbf{10.36 \pm 0.60}$	**132’**

Table 4. Test performance (top-1), MACs, inference time on a NVIDIA A4500 and training time for MobileNetv2 trained on CIFAR-10. Original refers to the trained model without layer deletion. The best results between Layer Folding, EGP, NEPENTHE, EASIER and LaCoOT are in **bold**.

In this setup, while Layer Folding and EGP showcase performance drop at high critical path length, we achieve comparable performance as EASIER or NEPENTHE for the same critical path length in 20× less time, with real practical benefits since the inference time and the MACs are reduced.

Tab. 5 shows the test performance, MACs, and inference time on an NVIDIA A4500, as well as the training time for a Swin-T trained on CIFAR-10 for all the considered approaches. In this setup, since the fusion of two linear layers is straightforward, we merge the layers for the baseline methods when the non-linearity in between has been removed.

Approach	top-1 [%]	MACs [M]	Inference time [ms]	Time
Original	91.67	518.94	13.54 ± 0.32	113’
Layer Folding	85.73	510.80	14.89 ± 0.11	383’
EGP	92.01	514.95	13.51 ± 0.17	228’
NEPENTHE	**92.29**	510.82	13.24 ± 0.26	688’
EASIER	91.25	494.28	11.04 ± 0.15	803’
LaCoOT	89.47	**199.54**	$\mathbf{8.33 \pm 0.02}$	**135’**

Table 5. Test performance (top-1), MACs, inference time on a NVIDIA A4500 and training time for Swin-T trained on CIFAR-10. Original refers to the trained model without layer deletion. The best results between Layer Folding, EGP, NEPENTHE, EASIER and LaCoOT are in **bold**.

In this setup, we can observe that the other methods reduce the number of MACs. However, there is little (if any) benefit in practice: the inference time is not reduced. Indeed, these methods focus solely on removing non-linearities, unlike our method, which removes complete blocks. On the other hand, although a slight loss of performance is noticeable, our method LaCoOT considerably reduces the number of MACs and decreases the inference time by more than 35%, while being far more efficient at training time than its competitors.

E. Extension of LaCoOT to layers with mismatched dimensionalities

LaCoOT was primilary designed to operate on layers where the Max-Sliced Wasserstein distance can be computed directly. This distance requires matching dimensions between distributions, which prevents a direct calculation of this distance for layers with different input and output dimensions. Nevertheless, we propose in this section to address this issue by studying the case of a 3x3 convolutional layer inside a ResNet-18.

Our goal here is to remove this specific layer in the network. However, directly removing the layer would result in a mismatch in both spatial resolution and channel dimensionality, disrupting the flow of activations through the network. To mitigate this, we introduce an alternative transformation that preserves the overall network structure while ensuring compatibility with subsequent layers. Specifically, we replace the 3×3 convolutional layer with a combination of a spatial downsampling operation and a 1×1 convolution. The downsampling is achieved using an average pooling layer (AvgPool2d) with a 2×2 kernel and a stride of 2, which reduces the spatial resolution. The 1×1 convolution then adjusts the number of output channels to match the expected input dimensions of the following layers. To ensure both configurations could be trained simultaneously, we implemented a dual-path approach within the modified block, where both the original and new transformations coexisted. During training, the introduced path with the 1x1

convolution is only trained using the Max-Sliced Wasserstein distance, computed between the output distribution of the 1x1 convolution and the output distribution of the original 3x3 convolution. Post-training, the original 3x3 convolution is discarded, and replaced by the average pooling and the newly trained 1x1 convolution.

Following the same training policy detailed in Sec. L, a ResNet-18 with the introduced transformation is trained on CIFAR-10. On the one hand, with $\lambda = 0$, the resulting network with the proposed transformation loses 0.98% performance compared to its full version. On the other hand, with $\lambda = 0.1$, the resulting network achieves comparable performance with only a 0.18% performance loss compared to its full version, highlighting the effectiveness of our method in this case.

To conclude, by incorporating this modified structure into the ResNet-18 architecture, we enable a seamless integration of LaCoOT which addresses the case of layers with mismatched dimensions.

F. Additional Results on Image Classification Setups

To complete the comparisons carried out in Sec. 5, we compare in this section the effectiveness of LaCoOT with respect to other baselines methods on Swin-T trained on PACS, VLCS, Aircraft, Flowers-102 and DTD in Fig 6. Indeed, as demonstrated in the previous section (Sec. D), Swin-T is the only architecture where competing methods can lead to practical benefits, since the fusion of two consecutive linear layers is straightforward.

In most setups, we can observe the effectiveness of LaCoOT. Indeed, for short critical path length, LaCoOT is the method performing overall the best, achieving a new Pareto Frontier when λ is increasing. In some cases like DTD, LaCoOT outperforms current methods by 10% for the same critical path length. Additionally, looking at longer critical path lengths, we can observe that when λ is decreasing, LaCoOT achieves comparable results to other baseline methods. Overall, since our method focuses on removing blocks, shorter critical path lengths can be achieved even though the performance drops dramatically. For very short critical path length (around 60), applying a healing policy or finetuning to the model could help recovering performance. However, we leave this aspect to future work.

G. Ranking with the Lipschitz constant

While Fig. 2 and 3 emphasize critical path length, we also report results using standard metrics (MACs, inference time, and training time) in Tab. 1, 4 and 5, which confirm the advantage of LaCoOT across multiple resource and performance dimensions. Furthermore, we show here in Tab. 6 that the ranking of the methods is preserved by displaying the Lipschitz constants across all blocks (B1–B4) of ResNet-18 on CIFAR-10. The global Lipschitz constant is upper bounded by the product of each block's Lipschitz constant.

Approach	B1	B2	B3	B4	Global
Original	3.61	3.42	2.12	1.47	38.48
Layer Folding	**1.01**	7.52	5.19	1.01	39.81
EGP	3.83	3.70	**1.01**	**1.01**	14.17
NEPENTHE	4.53	3.03	**1.01**	**1.01**	13.73
EASIER	4.72	1.44	1.35	2.69	24.68
LaCoOT ($\lambda = 5$)	**1.01**	**1.04**	1.10	1.18	**1.36**

Table 6. Lipschitz constants for ResNet-18 on CIFAR-10.

H. Comparison with structured pruning

To complete the comparisons carried out in Sec. 5, we compare in this section the effectiveness of LaCoOT with respect to a traditional model pruning method : Depgraph [15] in Tab. 7. LaCoOT outperforms DepGraph and achieves superior latency reduction. DepGraph's low performance is due to the absence of retraining after pruning (to fairly compare to us). Furthermore, since [13, 15, 37] perform channel pruning while LaCoOT removes entire layers, these methods operate at different levels of granularity. Rather than addressing the exact same problem, they are complementary: applying DepGraph on top of LaCoOT (as a refinement at finer granularity) yields even greater latency gains.

Approach	top-1 [%]	MACs [M]	Latency
Original	91.77	140.19	100%
LaCoOT ($\lambda = 5$)	90.99	64.69	-38%
Depgraph (0.3)	59.40	70.96	-11%
LaCoOT ($\lambda = 5$) + Depgraph (0.2)	90.96	57.22	-45%
LaCoOT ($\lambda = 5$) + Depgraph (0.3)	89.27	49.63	-49%

Table 7. Depgraph vs. LaCoOT for ResNet-18 on CIFAR-10.

I. A closer look at generated samples

We display in Fig. 7 some generated samples from the pre-trained DiT-XL/2 (Fig. 7a), a DiT-XL/2 with two DiT blocks removed without the use of LaCoOT (Fig. 7b), and from a DiT-XL/2 finetuned with LaCoOT($\lambda = 1e{-}4$) with two DiT blocks removed (Fig. 7c).

While the removal of blocks is completely destroying generated images in absence of the regularization, the generated content is better preserved when the DiT-XL/2 is

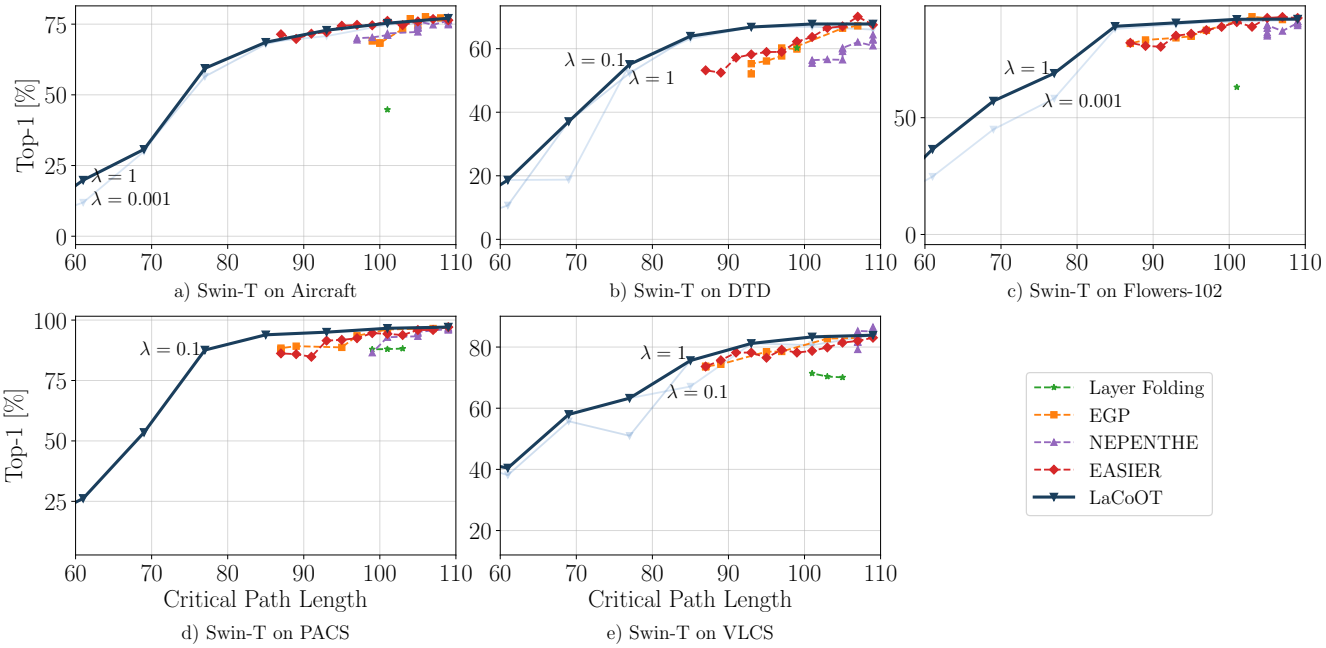

Figure 6. Test performance (Top-1 [%]) in function of the Critical Path Length for Swin-T trained on Aircraft (a), DTD (b), Flowers-102 (c), PACS (d) and VLCS (e). For each setup, we showcase the results achieved by LaCoOT for different values of λ, forming in **dark blue** the pareto frontier of our technique. *Top left corner is the best.*

fine-tuned with our method on 5k training steps. Indeed, the resulting images are much better than the the generated images produced without LaCoOT. However, we can observe a little loss in visual fidelity with respect to the pre-trained DiT-XL/2. For instance, although we can still perceive the buildings in the third image of the second column in Fig. 7c, we can no longer discern the lake in the foreground compared to the pre-trained model image in Fig. 7a. Nevertheless as it required only a few finetuning steps and given the quality of the generated samples with our method compared to without, we believe that our approach LaCoOT can be applied and suitable for foundation models.

J. Ablation Study

In this section, we conduct multiple ablation studies. First, we explore in Sec. J.1 the impact of using the Max-Sliced Wasserstein Distance, or the the Sliced Wasserstein Distance as a regularization in our method. Second, we evaluate the impact of the number of projections in Sec. J.2 and the batch size in Sec. J.3 toward LaCoOT success. Finally, in Sec. J.4, we compare our method LaCoOT replacing the Max-Sliced Wasserstein Distance with other existing metrics to quantify differences between distributions.

J.1. Theoretical guarantees versus practical benefits

From the POT library [16], two sliced OT distances can be used in our proposed regularization strategy. We propose here to explore the impact of using the Max-Sliced Wasserstein Distance (MSWD), or the the Sliced Wasserstein Distance (SWD) as a regularization in our method.

From a theoretical perspective, it is preferable to use the MSWD as it guarantees convergence [12]. Indeed, the MSWD minimizes the worst-case difference in distribution between the two measures over all possible projections. Since the MSWD is a global measure over all slices, it enforces convergence in the full measure space. This makes it a robust and convergent method for comparing distributions, ensuring that all possible distances are minimized when the maximum distance is minimized.

Moreover, when performing projections to calculate the Wasserstein Distance, we evaluate the impact of seeding the generator. Specifically, we investigate whether initializing the random seed for the generator during the projection process affects the stability or performance of our model.

This leads to four distinct configurations:

- "Seed + SWD" refers to the case where the SWD is used as a regularizer in our framework, and the generator is seeded during projections;
- "Seed + MSWD" refers to the case where the MSWD is used as a regularizer in our framework, and the generator is seeded during projections;
- "None + SWD" refers to the case where the SWD is used as a regularizer in our framework, and the generator is not seeded during projections, allowing for randomness to influence the projection directions;

(a) Samples generated from a pre-trained DiT-XL/2.

(b) Samples generated from a DiT-XL/2 with two DiT blocks removed, without LaCoOT. The generated content tends to be indiscernible.

(c) Samples generated from a DiT-XL/2 finetuned with LaCoOT ($\lambda = 1e{-}4$) with two DiT blocks removed.

Figure 7. Generated samples from different configurations of a DiT-XL/2. When finetuned with LaCoOT, when two DiT blocks are removed, the generated content is better preserved. Indeed, the removal of blocks is completely destroying generated images in absence of the regularization, while the generated content is better preserved with its use.

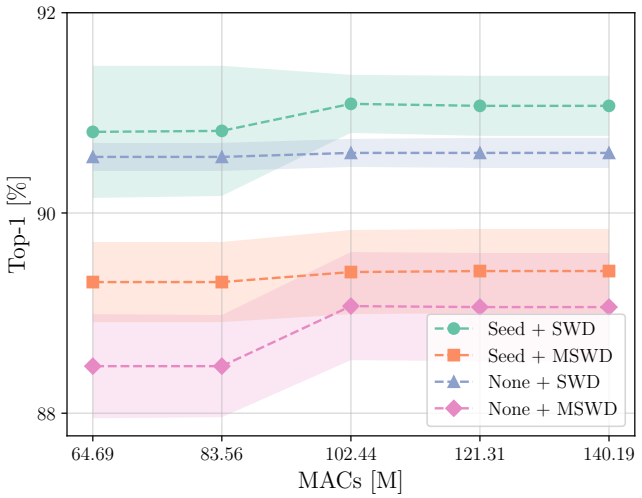

Figure 8. MSWD vs. SWD and impact of seeding the generator for the projections for a ResNet-18 trained on CIFAR-10 with LaCoOT ($\lambda = 5$). SWD yields better results than MSWD.

- "None + MSWD" refers to the case where the MSWD is used as a regularizer in our framework, and the generator is not seeded during projections, allowing for randomness to influence the projection directions.

Since the unseeded approach may expose the model to more generalization across different slices, and that MSWD provides theoretical convergence guarantees, we use the "None + MSWD" configuration in all our experiments except where otherwise stated.

Fig. 8 displays the results of the ablation over the 4 configurations on a ResNet-18 trained on CIFAR-10 with our method LaCoOT with $\lambda = 5$. Since variability between runs can occur, we report standard deviations over 5 runs.

Interestingly, despite offering theoretical convergence guarantees, the use of MSWD as a regularizer yields worse results compared to the use of the SWD. Moreover, looking at the standard deviations, we can observe that the "None + SWD" configuration display the lowest, which shows its stability. Thus, we draw the reader's attention to the fact that better results can be obtained in practice if one allow himself to dispense with the theoretical convergence guarantees.

J.2. Ablation on the number of projections

In this subsection, we evaluate the impact of the number of projections n_{proj} toward LaCoOT success. Indeed, Fig. 9 shows the results achieved for LaCoOT($\lambda = 5$) when lowering the number of projections used to calculate the MSWD.

While for the extreme case ($n_{proj} = 1$), a drop in performance is observed when blocks are removed and MACs reduced, we can already obtain decent results with $n_{proj} = 5$. Indeed, it appears that the number of projections plays a role in the trade-off between the model's original performance (at 140.19 MACs) and the possibility of removing layers without performance loss. In fact, $n_{proj} = 40$ showcases the best results with the best performance at lower MACs, and a very slight loss of performance with respect to its original counterpart at 140,19 MACs.

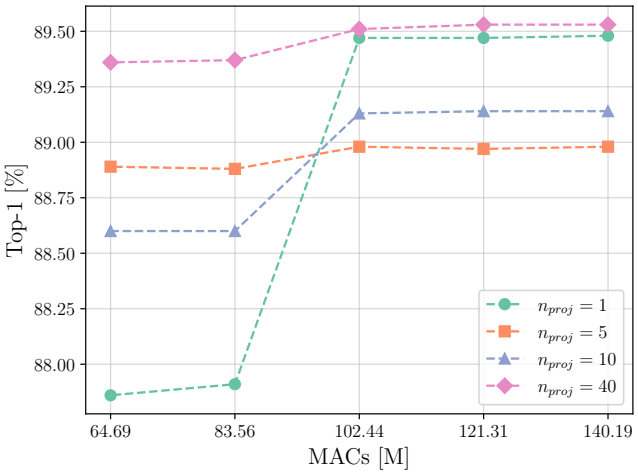

Figure 9. Ablation on the number of projection n_{proj} for a ResNet-18 trained on CIFAR-10 with LaCoOT ($\lambda = 5$).

J.3. Ablation on the batch size

In this subsection, we evaluate the impact of the batch size BS on LaCoOT results. Indeed, Fig. 10 shows the results achieved for LaCoOT($\lambda = 5$) when lowering the batch size used to calculate the MSWD.

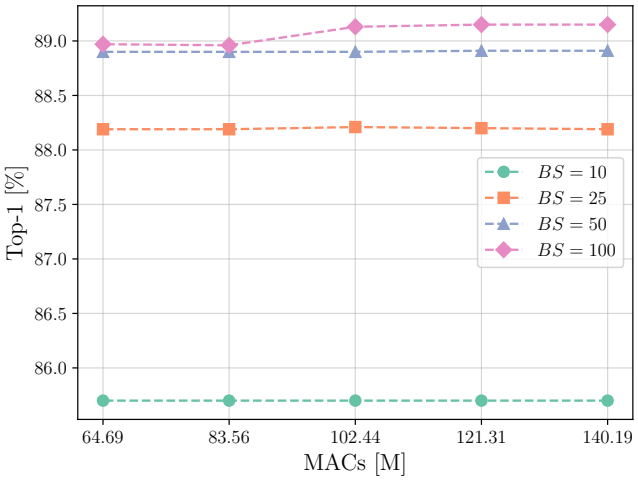

Figure 10. Ablation on the batch size BS for a ResNet-18 trained on CIFAR-10 with LaCoOT ($\lambda = 5$).

Looking at the results, it appears evident that reducing the batch size produces worse results. Although performance remains constant (or presents a slight decrease for $BS = 100$) when layers are removed and MACs are re-

duced, it appears evident that the smaller the batch size, the lower the performance of the original model (at 140.19 MACs). Hence, whenever possible, LaCoOT should always be applied with a sufficiently large batch size that can fit in the memory of the used computing resources.

J.4. Comparison with other metrics

In this subsection, we compare our method LaCoOT using other existing metrics to quantify differences between distributions. Indeed, the MSWD regularization can be replaced by the ℓ_1 distance, the ℓ_2 distance, the Maximum Mean Discrepancy (MMD) or the Kullback-Leibler (KL) Divergence. For each comparison, we show the best configuration of λ yielding the best results for the trade-off between top-1 performance and MACs. Indeed, a grid search on λ is carried out to find the best trade-offs. Fig. 11 displays the results for a ResNet-18 trained on CIFAR-10.

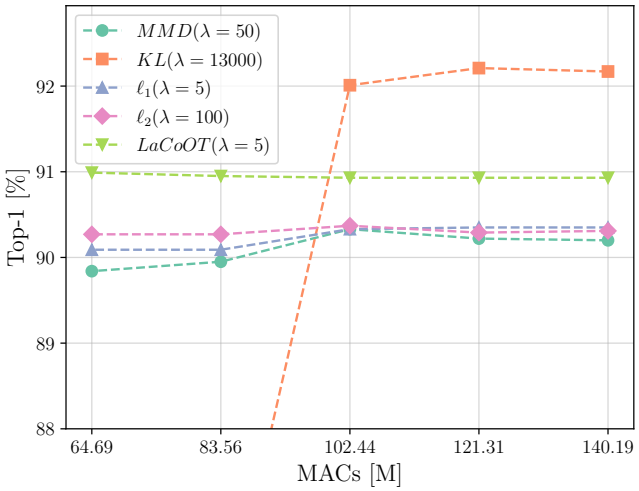

Figure 11. Ablation on LaCoOT replacing the proposed regularization with ℓ_1, ℓ_2, MMD or KL divergence, for a ResNet-18 trained on CIFAR-10.

On the one hand, while for high MACs, the KL divergence shows its competitiveness, the performance drops dramatically as blocks are removed and MACs reduced. On the other hand, we can observe that ℓ_1, ℓ_2 and MMD obtain similar performance/complexity trade-off with relatively few performance drops. Our method LaCoOT with the MSWD outperforms the other compared metrics for lower MACs.

While the choice of metric seems to have very little impact on the performance obtained, we draw the reader's attention to the fact that this is due to the idea we are proposing. Indeed, minimizing the distance between feature distributions of successive layers appears to be more important to reduce the depth of DNNs than the choice of the metric itself. Thus, the other metrics presented here can also be used as regularizations in our method, but these may produce worse results.

K. Training from scratch with lower initial depth

While having an oracle baseline on each setup is computationally expensive as brute-force research for all the combinations (+full retraining) is required, we present in Tab. 8 below the performance of 16 residual networks trained from scratch on CIFAR-10 following the set of Tab. 9. Indeed, we remove at initialization (a combination of) layers inside a ResNet-18 and train the network from scratch. We call this the "brute-force approach".

Combination	# B. Rem.	Top-1 [%]	MACs [M]
2222 (Original)	0	91.77	140.19
2221	1	91.86	121.31
2212	1	91.29	121.31
1222	1	90.82	121.31
2122	1	90.59	121.31
2211	2	91.96	102.44
1221	2	91.58	102.44
2112	2	91.45	102.44
2121	2	91.27	102.44
1212	2	91.25	102.44
1122	2	91.02	102.44
2111	3	91.78	83.56
1211	3	91.64	83.56
1121	3	90.94	83.56
1112	3	90.88	83.56
1111	4	91.45	64.69
LaCoOT($\lambda = 5$)	4	90.99	64.69
LaCoOT($\lambda = 5$) + retraining	4	91.42	64.69

Table 8. ResNet-18 trained from scratch on CIFAR-10 with layers removed initially. For a given combination, we associate the number of blocks removed (# B. Rem.), the top-1 performance and associated MACs at inference.

Our approach LaCoOT is achieving comparable performance compared to these models. However, while the "brute-force approach" has to perform 16 separate trainings, LaCoOT can produce the same 16 subnetworks in one training only, hence being very efficient. By further retraining the pruned architecture, we recover performance, comparable to the original model, as shown in the last line.

L. Details on the learning strategies employed

Image Classification. The training hyperparameters used in the experiments are presented in Table 9. Our code is available at https://github.com/VGCQ/LaCoOT.

CIFAR-10 is augmented with per-channel normalization, random horizontal flipping, and random shifting by up to four pixels in any direction. For the datasets of DomainBed, the images are augmented with per-channel normalization,

Model	Dataset	Epochs	Batch	Opt.	Mom.	LR	Milestones	Drop Factor	Weight Decay
ResNet-18	CIFAR-10	160	128	SGD	0.9	0.1	[80, 120]	0.1	1e-4
Swin-T	CIFAR-10	160	128	SGD	0.9	0.001	[80, 120]	0.1	1e-4
MobileNetv2	CIFAR-10	160	128	SGD	0.9	0.1	[80, 120]	0.1	1e-4
ResNet-18	Tiny-ImageNet-200	160	128	SGD	0.9	0.1	[80, 120]	0.1	1e-4
Swin-T	Tiny-ImageNet-200	160	128	SGD	0.9	0.001	[80, 120]	0.1	1e-4
MobileNetv2	Tiny-ImageNet-200	160	128	SGD	0.9	0.1	[80, 120]	0.1	1e-4
ResNet-18	PACS	30	16	SGD	0.9	0.001	[24]	0.1	5e-4
Swin-T	PACS	30	16	SGD	0.9	0.001	[24]	0.1	5e-4
MobileNetv2	PACS	30	16	SGD	0.9	0.001	[24]	0.1	5e-4
ResNet-18	VLCS	30	16	SGD	0.9	0.001	[24]	0.1	5e-4
Swin-T	VLCS	30	16	SGD	0.9	0.001	[24]	0.1	5e-4
MobileNetv2	VLCS	30	16	SGD	0.9	0.001	[24]	0.1	5e-4
ResNet-18	Flowers-102	50	16	Adam		1e-4			0
Swin-T	Flowers-102	50	16	Adam		1e-4			0
MobileNetv2	Flowers-102	50	16	Adam		1e-4			0
ResNet-18	DTD	50	16	Adam		1e-4			0
Swin-T	DTD	50	16	Adam		1e-4			0
MobileNetv2	DTD	50	16	Adam		1e-4			0
ResNet-18	Aircraft	50	16	Adam		1e-4			0
Swin-T	Aircraft	50	16	Adam		1e-4			0
MobileNetv2	Aircraft	50	16	Adam		1e-4			0

Table 9. The different employed learning strategies.

random horizontal flipping, random cropping, and resizing to 224. The brightness, contrast, saturation, and hue are also randomly affected with a factor fixed to 0.4. Tiny-ImageNet-200 is augmented with per-channel normalization and random horizontal flipping. Moreover, the images of Flowers-102 are augmented with per-channel normalization, random horizontal and vertical flipping combined with a random rotation, and cropped to 224. DTD and Aircraft are augmented with random horizontal and vertical flipping, and with per-channel normalization.

Following [34] and [50], on CIFAR-10 and Tiny-ImageNet-200, all the models are trained for 160 epochs, optimized with SGD, having momentum 0.9, batch size 128, and weight decay 1e-4. The learning rate is decayed by a factor of 0.1 at milestones 80 and 120. The initial learning rate ranges from 0.1 for ResNet-18 and MobileNetv2, to 1e-3 for Swin-T. Moreover, on PACS and VLCS, all the models are trained for 30 epochs, optimized with SGD, having momentum 0.9, a learning rate of 1e-3 decayed by a factor 0.1 at milestone 24, batch size 16, and weight decay 5e-4. Furthermore, on Aircraft, DTD, and Flowers-102, all the models are trained following a transfer learning strategy. Indeed, each model is initialized with its pre-trained weights on ImageNet, trained for 50 epochs, optimized with Adam, having a learning rate 1e-4 and batch size 16.

The experiments were mostly performed using an NVIDIA RTX 3090.

Image Generation. DiT-XL/2 at 256×256 image resolution is fine-tuned on ImageNet for 5k training steps on 3 NVIDIA L40S using AdamW, no weight decay, with a global batch size of 60 and a learning rate 1e-4. Only horizontal flips were used to augment the training set. Following common practice in the generative modeling literature, we maintain an exponential moving average (EMA) of DiT weights over training with a decay of 0,9999. All results reported use the EMA model. The pre-trained model is taken from the original paper [47] and the diffusion was done using the same details as in the original paper. We evaluate the quality of the generated samples with Fréchet Inception Distance (FID) [24], the standard metric for evaluating generative models of images. Following convention, we report FID-50k using 250 sampling steps with clean-fid [46] and classifier-free guidance scale of 1,5.