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.

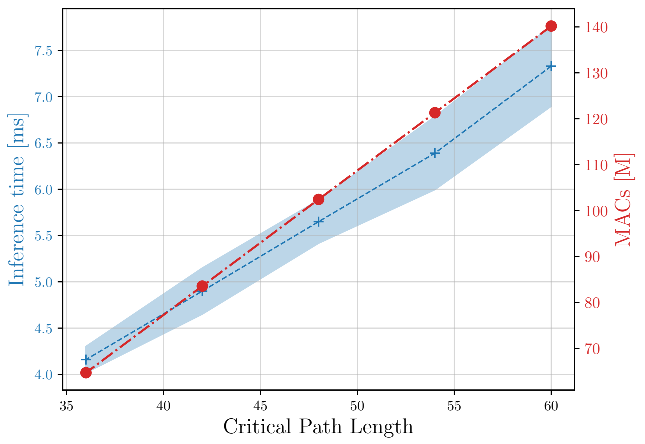


Figure 5. Relationship between Critical Path Length (CPL), inference time, and Multiply-Accumulate Operations (MACs) for a ResNet-18 model on the CIFAR-10 dataset. As the CPL decreases, both the inference time and the number of MACs decrease, indicating improved computational efficiency and faster inference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

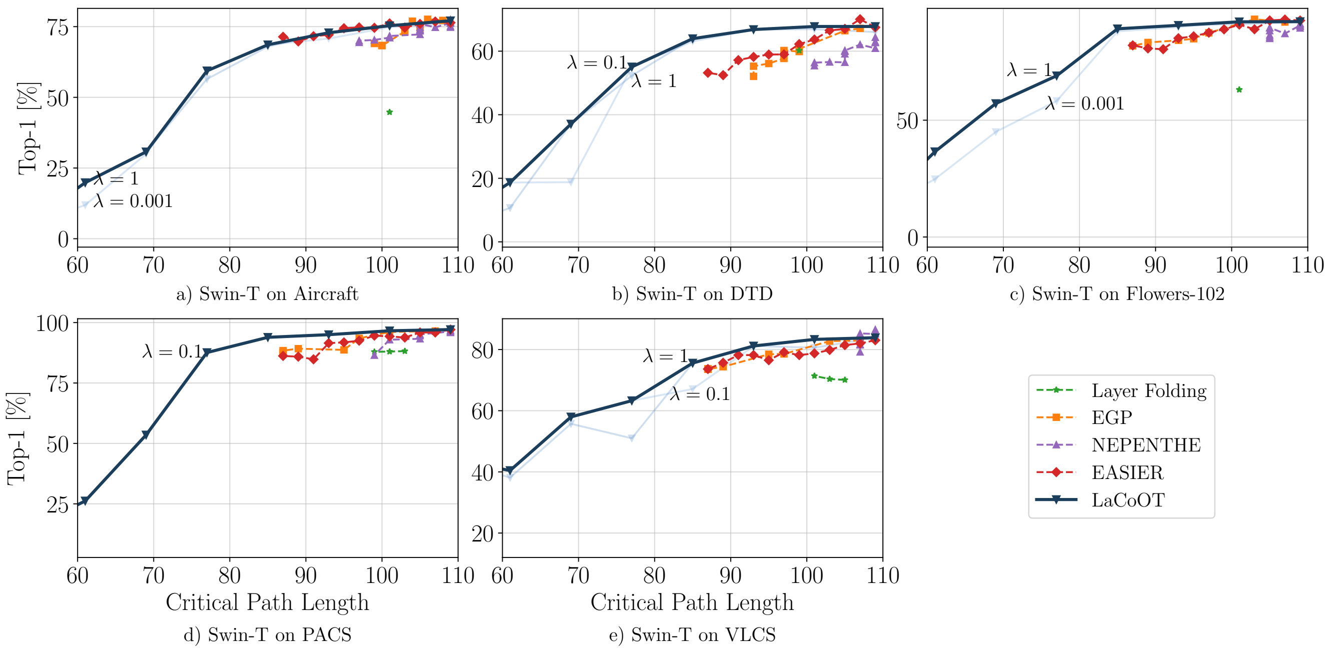


Figure 6. Test performance (Top-1 [%]) in function of the Critical Path Length for Swin-T trained on Aircraft (a), DTD (b), Flowers-102 (c), PACS (d) and VLCS (e). For each setup, we showcase the results achieved by LaCoOT for different values of λ, forming in **dark blue** the pareto frontier of our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

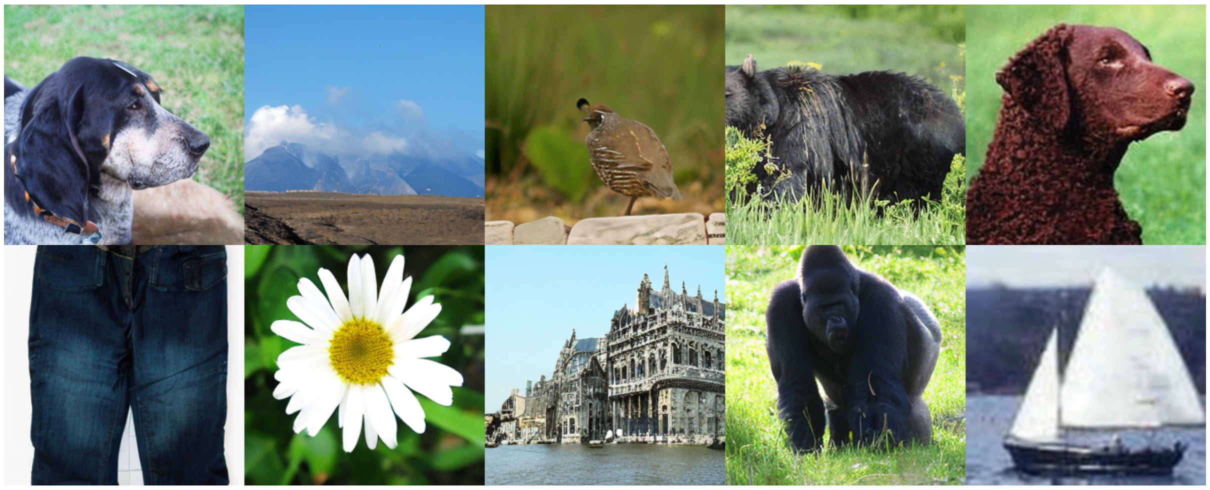
From the POT library [16], two sliced OT distances can be used in our proposed regularization strategy. We propose here to explore the impact of using the Max-Sliced Wasserstein Distance (MSWD), or the the Sliced Wasserstein Distance (SWD) as a regularization in our method.

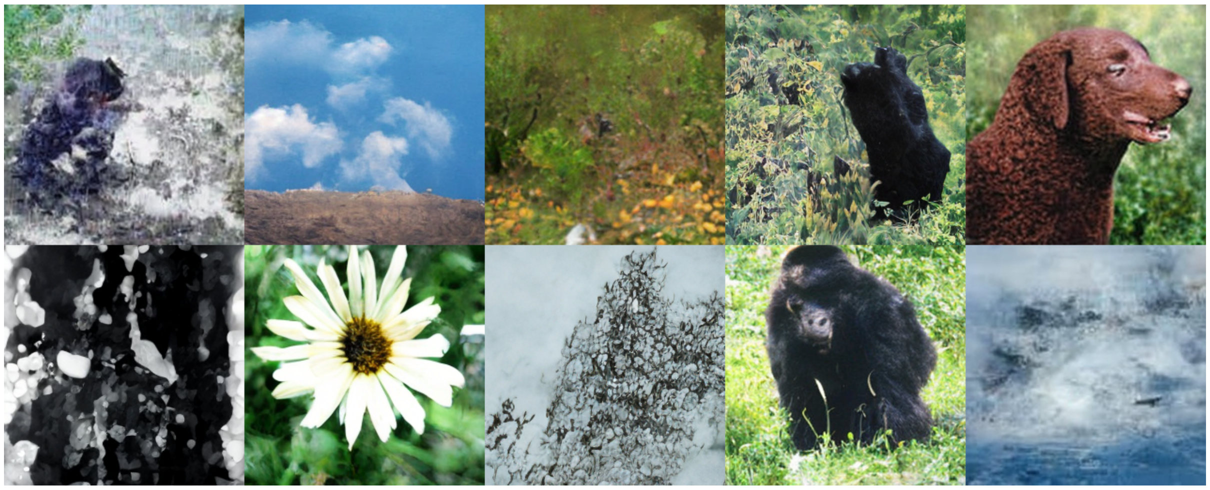
From a theoretical perspective, it is preferable to use the MSWD as it guarantees convergence [12]. Indeed, the MSWD minimizes the worst-case difference in distribution between the two measures over all possible projections. Since the MSWD is a global measure over all slices, it enforces convergence 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.

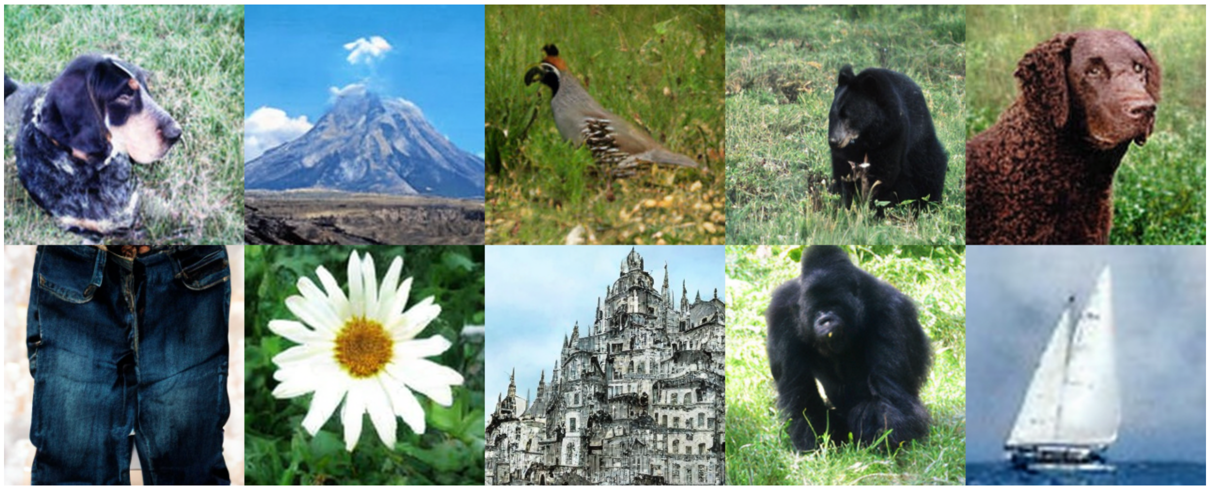

(c) Samples generated from a DiT-XL/2 finetuned with LaCoOT ($\lambda = 1e{-}4$) with two DiT blocks removed.

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.

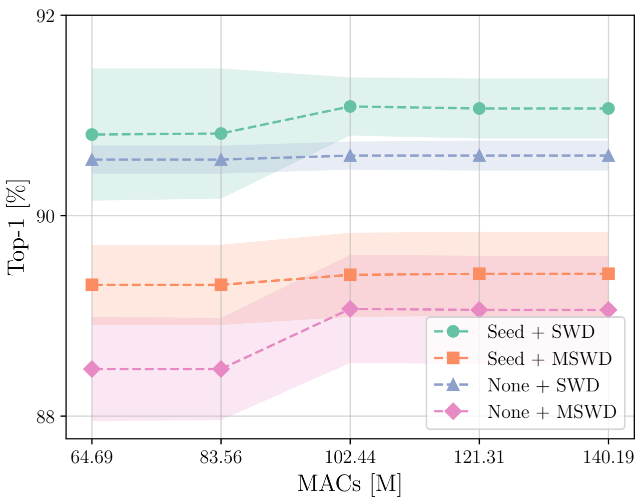


Figure 8. MSWD vs. SWD and impact of seeding the generator for the projections for a ResNet-18 trained on CIFAR-10 with LaCoOT ($\lambda = 5$). SWD yields better results than MSWD.

- "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.

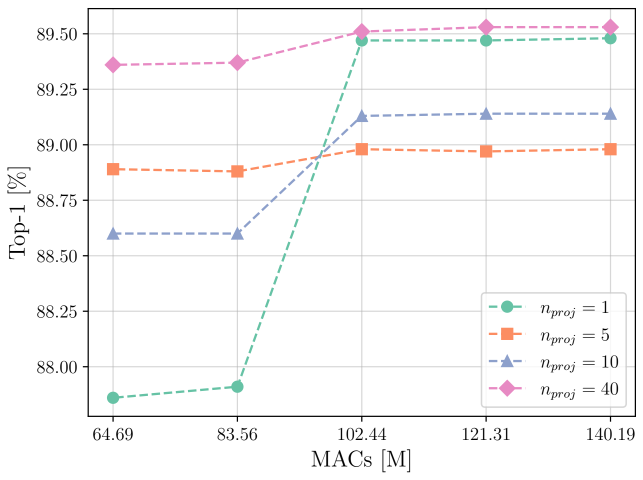


Figure 9. Ablation on the number of projection n_{proj} for a ResNet-18 trained on CIFAR-10 with LaCoOT ($\lambda = 5$).

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.

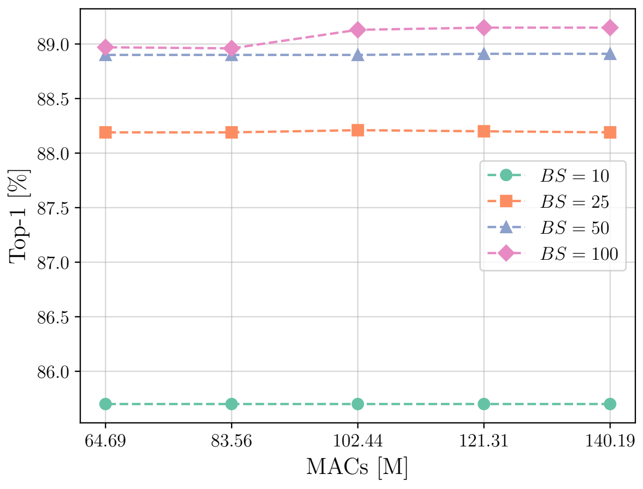


Figure 10. Ablation on the batch size BS for a ResNet-18 trained on CIFAR-10 with LaCoOT ($\lambda = 5$).

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 reduced, 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.

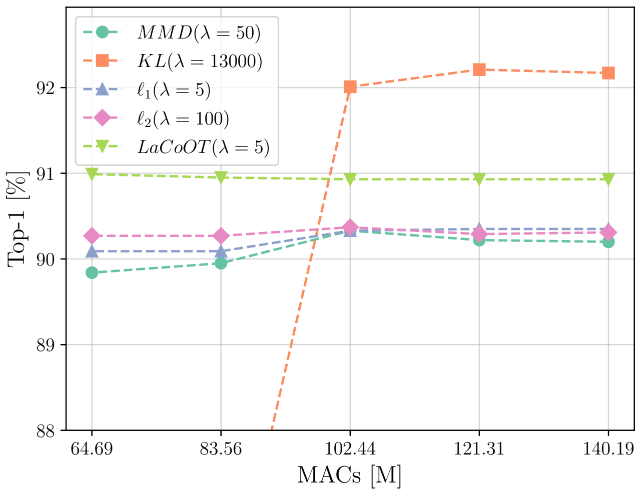


Figure 11. Ablation on LaCoOT replacing the proposed regularization with ℓ_1, ℓ_2, MMD or KL divergence, for a ResNet-18 trained on CIFAR-10.

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.