Block-wise Training of Residual Networks via the Minimizing Movement Scheme

Skander Karkar, Ibrahim Ayed, Emmanuel de Bezenac, Patrick Gallinari

Block-wise Training of Residual Networks via the Minimizing Movement Scheme

Skander Karkar,[1 2] **Ibrahim Ayed,**[1 3] **Emmanuel de Bézenac,**[1] **Patrick Gallinari**[1 2]

[1]LIP6, Sorbonne Université, France
[2]Criteo AI Lab, Criteo, France
[3]Theresis lab, Thales, France
{skander.karkar, ibrahim.ayed, emmanuel.de-bezenac, patrick.gallinari}@lip6.fr

Abstract

End-to-end backpropagation has a few shortcomings: it requires loading the entire model during training, which can be impossible in constrained settings, and suffers from three locking problems (forward locking, update locking and backward locking), which prohibit training the layers in parallel. Solving layer-wise optimization problems can address these problems and has been used in on-device training of neural networks. We develop a layer-wise training method, particularly well-adapted to ResNets, inspired by the minimizing movement scheme for gradient flows in distribution space. The method amounts to a kinetic energy regularization of each block that makes the blocks optimal transport maps and endows them with regularity. It works by alleviating the stagnation problem observed in layer-wise training, whereby greedily-trained early layers overfit and deeper layers stop increasing test accuracy after a certain depth. We show on classification tasks that the test accuracy of block-wise trained ResNets is improved when using our method, whether the blocks are trained sequentially or in parallel.

Introduction

End-to-end backpropagation is the standard training method of neural nets. But there are reasons to look for alternatives. It is considered biologically unrealistic (Mostafa, Ramesh, and Cauwenberghs 2018) and it requires loading the whole model during training which can be impossible in constrained settings such as training on mobile devices (Teng et al. 2020; Tang et al. 2021). It also prohibits training layers in parallel as it suffers from three locking problems (forward locking: each layer must wait for the previous layers to process its input, update locking: each layer must wait for the end of the forward pass to be updated, and backward locking: each layer must wait for errors to backpropagate from the last layer to be updated) (Jaderberg et al. 2017). These locking problems force the training and deployment of networks to be sequential and synchronous, and breaking them would allow for more flexibility when using networks that are distributed between a central agent and clients and that operate at different rates (Jaderberg et al. 2017). Greedily solving layer-wise optimization problems, sequentially (i.e. one after the other) or in parallel (i.e. batch-wise), solves update locking (and so also backward locking). When combined with buffers, parallel layer-wise training solves all three problems (Belilovsky, Eickenberg, and Oyallon 2020) and allows distributed training of the layers. Layer-wise training is appealing in memory-constrained settings as it works without most gradients and activations needed in end-to-end training, and when done sequentially, only requires loading and training one layer at a time. Despite its simplicity, layer-wise training has been shown (Belilovsky, Eickenberg, and Oyallon 2019, 2020) to scale well. It outperforms more complicated ideas developed to address the locking problems such as synthetic (Jaderberg et al. 2017; Czarnecki et al. 2017) and delayed (Huo et al. 2018; Huo, Gu, and Huang 2018) gradients. We can also deduce theoretical results about a network of greedily-trained shallow sub-modules from the theoretical results about shallow networks (Belilovsky, Eickenberg, and Oyallon 2019, 2020). Module-wise training, where the network is split into modules that are trained greedily, may not offer the same computational gains as full layer-wise training, but it gets closer to the accuracy of end-to-end training and is explored often (Pyeon et al. 2021; Wang et al. 2021).

The typical setting of (sequential) module-wise training for minimizing a loss L, is, given a dataset $\tilde{\rho}_0$, to solve one after the other, for $0 \leq k \leq K$, the problems

$$(T_k, F_k) \in \arg\min_{T,F} \sum_{x \in \tilde{\rho}_0} L(F, T(G_{k-1}(x))) \qquad (1)$$

where $G_k = T_k \circ ... \circ T_0$ for $0 \leq k \leq K$ and $G_{-1} = \mathtt{id}$. Here, T_k is the module (a single layer or a group of layers) and F_k is an auxiliary network (a classifier if the task is classification) that processes the outputs of T_k so that the loss can be computed. In a generation task, auxiliary networks might not be needed. In all cases, module T_{k+1} receives the output of module T_k. The final network trained this way is $F_K \circ G_K$, but we can stop at any previous depth k and use $F_k \circ G_k$ if it performs better. In fact, and especially when modules are shallow, module-wise training suffers from a stagnation problem, whereby greedily-trained early modules overfit and deeper modules stop improving the test accuracy after a certain depth, or even degrade it (Marquez, Hare, and Niranjan 2018; Wang et al. 2021). We observe this experimentally (Figure 1) and propose a regularization for module-wise training that addresses this problem by increasing training stability. The regularization is particularly well-adapted to ResNets

(He et al. 2016a,b) and similar models such as ResNeXts (Xie et al. 2017) and Wide ResNets (Zagoruyko and Komodakis 2016), but is easily usable on other models, and ResNets themselves remain competitive (Wightman, Touvron, and Jégou 2021). The method leverages the analogy between ResNets and the Euler scheme for ODEs (Weinan 2017) to penalize the kinetic energy of the network. Intuitively, if the kinetic energy is penalized enough, the current module will barely move the points, thus at least preserving the accuracy of the previous module and avoiding its collapse.

After a discussion of related work in Section 2, we present the method in Section 3 and show that it amounts to a transport regularization of each module, which we prove forces the solution module to be an optimal transport map and makes it regular. This also suggests a simple principled extension of the method to non-residual networks. In Section 3, we link the method with gradient flows and the minimizing movement scheme in the Wasserstein space, which allows to invoke convergence results to a minimizer under additional hypotheses on the loss. Section 4 discusses different practical implementations and introduces a new variant of layer-wise training we call *multi-lap sequential* training. It is a slight variation on sequential layer-wise training that has the same advantages and offers a non-negligible improvement in many cases over sequential training for the same computational and memory costs. Experiments using different architectures and on different classification datasets in Section 5 show that our method consistently improves the test accuracy of block-wise and module-wise trained residual networks, particularly in small data regimes, whether the block-wise training is carried out sequentially or in parallel.

Related work

Layer-wise training of neural networks has been considered as a pre-training and initialization method (Bengio et al. 2006; Marquez, Hare, and Niranjan 2018) and was shown recently to perform competitively with end-to-end training (Belilovsky, Eickenberg, and Oyallon 2019; Nøkland and Eidnes 2019). This has led to it being considered in practical settings with limited resources such as embedded training (Teng et al. 2020; Tang et al. 2021). For layer-wise training, many papers consider using a different auxiliary loss, instead of or in addition to the classification loss: kernel similarity (Mandar Kulkarni 2016), information-theory-inspired losses (Sindy Löwe 2019; Nguyen and Choi 2019; Ma, Lewis, and Kleijn 2020; Wang et al. 2021) and biologically plausible losses (Sindy Löwe 2019; Nøkland and Eidnes 2019; Gupta 2020; Bernd Illing 2020; Yuwen Xiong 2020). Paper (Belilovsky, Eickenberg, and Oyallon 2019) reports the best experimental results when solving the layer-wise problems sequentially. Methods PredSim (Nøkland and Eidnes 2019), DGL (Belilovsky, Eickenberg, and Oyallon 2020), Sedona (Pyeon et al. 2021) and InfoPro (Wang et al. 2021) report the best results when solving the layer-wise problems in parallel, albeit each in a somewhat different setting. (Belilovsky, Eickenberg, and Oyallon 2019, 2020) do it simply through architectural considerations mostly regarding the auxiliary networks. However, (Belilovsky, Eickenberg, and Oyallon 2019) do not consider ResNets and PredSim state that their method does not perform well on ResNets, specifically because of the skip connections. DGL only considers a ResNet architecture by splitting it in the middle and training the two halves without backpropagating between them. All three focus on VGG architectures and networks that are not deep. Sedona applies architecture search to decide on where to split the network into 2 or 4 modules and what auxiliary classifier to use before module-wise training. Only BoostResNet (Huang et al. 2018) also proposes a block-wise training idea geared for ResNets. However, their results only show better early performance on limited experiments and end-to-end fine-tuning is required to be competitive. A method called ResIST (Dun et al. 2021) that is similar to block-wise training of ResNets randomly assigns residual blocks to one of up to 8 sub-networks that are trained independently and reassembled before another random partition. But only blocks in the third section of the ResNet are partitioned, the same block can appear in many sub-networks and the sub-networks are not necessarily made up of successive blocks. Considered as a distributed training method, it is only compared with local SGD (Stich 2019). These methods can all be combined with our regularization, and we use the auxiliary network architecture from (Belilovsky, Eickenberg, and Oyallon 2019, 2020). We also show the benefits of our method both with full layer-wise training and when the network is split into a few modules.

Besides layer-wise training, methods such as DNI (Jaderberg et al. 2017; Czarnecki et al. 2017), DDG (Huo et al. 2018) and Features Replay (Huo, Gu, and Huang 2018), solve the update locking problem and the backward locking problem with an eye towards parallelization by using delayed or synthetic predicted gradients, or even predicted inputs to address forward locking. But they only fully apply this to quite shallow networks and only split deeper ones into a small number of sub-modules (less than five) that don't backpropagate to each other and observe training issues with more splits (Huo, Gu, and Huang 2018). This makes them compare unfavorably to layer-wise training (Belilovsky, Eickenberg, and Oyallon 2020). The high dimension of the predicted gradient which scales with the size of the network renders (Jaderberg et al. 2017; Czarnecki et al. 2017) challenging in practice. Therefore, despite its simplicity, greedy layer-wise training is more appealing when working in a constrained setting.

Viewing residual networks as dynamical transport systems (de Bézenac, Ayed, and Gallinari 2019; Karkar et al. 2020) followed from their view as a discretization of differential equations (Weinan 2017; Lu et al. 2018). Transport regularization was also used in (Finlay et al. 2020) to accelerate the training of the NeuralODE model (Chen et al. 2018). Transport regularization of ResNets in particular is motivated by the observation that they are naturally biased towards minimally modifying their input (Jastrzebski et al. 2018; Hauser 2019; Karkar et al. 2020). We further link this transport viewpoint with gradient flows in the Wasserstein space to apply it in a principled way to module-wise training. Gradient flows in the Wasserstein space operating on the data space appeared recently in deep learning. In (Alvarez-Melis and Fusi 2021), the focus is on functionals of measures whose first variations are known in closed form and used, through their gradients,

in the algorithm. This limits the scope of their applications to transfer learning and similar tasks. Likewise, (Gao et al. 2019; Liutkus et al. 2019; Arbel et al. 2019; Ansari, Ang, and Soh 2021) use the explicit gradient flow of f-divergences and other distances between measures for generation and generator refinement. In contrast, we use the minimizing movement scheme which does not require computation of the first variation and allows to consider classification.

Regularized block-wise training of ResNets

In this section we state the module-wise problems we solve and show that the modules they induce are regular as they approximate optimal transport maps. We show that solving these problems sequentially means following a minimizing movement that approximates the Wasserstein gradient flow that minimizes the loss, which offers hints as to why it works well in practice outside the theoretical hypotheses.

Method statement

In a ResNet, a data point x_0 is transported by applying $x_{m+1} = x_m + g_m(x_m)$ for M ResBlocks (a ResBlock is a function $\mathtt{id} + g_m$), and x_{M+1} is then classified. To keep the greedily-trained modules from overfitting and destroying information needed by deeper modules, we propose to penalize their kinetic energy to force them to preserve the geometry of the problem as much as possible. The total discrete kinetic energy of a ResNet (for a single point x_0) is $\sum \|g_m(x_m)\|^2$, since a ResNet can be seen as an Euler scheme for an ODE with velocity field g (Weinan 2017):

$$x_{m+1} = x_m + g_m(x_m) \iff \partial_t x_t = g_t(x_t) \tag{2}$$

That ResNets are already biased towards small displacements and therefore low kinetic energy and that this bias is desirable and should be encouraged has been observed in many works (Jastrzebski et al. 2018; Zhang et al. 2019; Hauser 2019; De and Smith 2020; Karkar et al. 2020). Using the notations from (1), if each module T_k is made up of M ResBlocks, i.e. has the form $(\mathtt{id} + g_{M-1}) \circ ... \circ (\mathtt{id} + g_0)$, we propose to penalize its kinetic energy over its input points by adding it to the loss L in the target of the greedy problems (1). We denote ψ_m^x the position of an input x after m ResBlocks. Given $\tau > 0$ used to weight the regularization, we solve, for $0 \leq k \leq K$, problems

$$(T_k^\tau, F_k^\tau) \in \tag{3}$$
$$\arg\min_{T,F} \sum_{x \in \tilde{\rho}_0} (L(F, T(G_{k-1}^\tau(x)) + \frac{1}{2\tau} \sum_{m=0}^{M-1} \|g_m(\psi_m^x)\|^2)$$
$$\text{s.t. } T = (\mathtt{id} + g_{M-1}) \circ ... \circ (\mathtt{id} + g_0)$$
$$\psi_0^x = G_{k-1}^\tau(x), \psi_{m+1}^x = \psi_m^x + g_m(\psi_m^x)$$

where $G_k^\tau = T_k^\tau \circ ... \circ T_0^\tau$ for $0 \leq k \leq K$ and $G_{-1}^\tau = \mathtt{id}$. The final network is now $F_K^\tau \circ G_K^\tau$. Intuitively, we can think that this biases the modules towards moving the points as little as possible, thus at least keeping the performance of the previous module. In our experiments, we will mostly focus on block-wise training, i.e. the case $M = 1$ where each T_k^τ is a single residual block, as it is more challenging.

Regularity result

The Appendix gives the necessary background on optimal transport (OT) theory to prove a regularity result for our method. We start by moving to a continuous viewpoint. We denote $\rho_0 = \rho_0^\tau$ the data distribution of which $\tilde{\rho}_0$ is a sample and $\mathcal{L}$ the distribution-wide loss that arises from the pointwise loss L. As expressed in (2), a residual network can be seen as an Euler discretization of a differential equation. Problem (3) is then the discretization of problem

$$(T_k^\tau, F_k^\tau) \in \tag{4}$$
$$\arg\min_{T,F} \mathcal{L}(F, T_\sharp \rho_k^\tau) + \frac{1}{2\tau} \int_0^1 \|v_t\|^2_{L^2((\phi_t^\cdot)_\sharp \rho_k^\tau)} \, \mathrm{d}t$$
$$\text{s.t. } T = \phi_1^\cdot, \ \partial_t \phi_t^x = v_t(\phi_t^x), \ \phi_0^\cdot = \mathrm{id}$$

where $\rho_{k+1}^\tau = (T_k^\tau)_\sharp \rho_k^\tau$ and g_m is the discretization of vector field v_t at time $t = m/M$. Here, data distributions ρ_k^τ are pushed forward through the maps T_k^τ which correspond to the flow at $t = 1$ of the kinetically-regularized velocity field v_t. Given the equivalence between the Monge OT problem (12) and the OT problem in dynamic form (14) in the Appendix, problem (4) is equivalent to

$$(T_k^\tau, F_k^\tau) \in \tag{5}$$
$$\arg\min_{T,F} \mathcal{L}(F, T_\sharp \rho_k^\tau) + \frac{1}{2\tau} \int_\Omega \|T(x) - x\|^2 \, \mathrm{d}\rho_k^\tau(x)$$

where points are moved instantly through T instead of infinitesimally through velocity filed v. This equivalent formulation leads to another discretization (10) and implementation of the method more easily applicable in practice to non-residual architectures by simply penalizing the difference between the module's output and its input. We can show that problem (5) indeed has a solution and that T_k^τ is necessarily an optimal transport between its input and output distributions, which means that it comes with some regularity. We assume that the minimization in F is over a compact set $\mathcal{F}$, that ρ_k^τ is absolutely continuous, that $\mathcal{L}$ is continuous and non-negative and that Ω is compact.

Theorem 1. *Problems* (5) *and* (4) *have a minimizer* (T_k^τ, F_k^τ) *such that* T_k^τ *is an optimal transport map. And for any minimizer* (T_k^τ, F_k^τ), T_k^τ *is an optimal transport map.*

The proof is in the Appendix. Optimal transport maps have regularity properties under some boundedness assumptions. Given Theorem 2 in the Appendix taken from (Figalli 2017), T_k^τ is η-Hölder continuous almost everywhere and if the optimization algorithm we use to solve the discretized problem (3) returns an approximate solution pair $(\tilde{F}_k^\tau, \tilde{T}_k^\tau)$ such that $\tilde{T}_k^\tau$ is an ϵ-optimal transport map, i.e. $\|\tilde{T}_k^\tau - T_k^\tau\|_\infty \leq \epsilon$, then we have (using the triangle inequality) the following stability property of the neural module $\tilde{T}_k^\tau$:

$$\|\tilde{T}_k^\tau(x) - \tilde{T}_k^\tau(y)\| \leq 2\epsilon + C\|x - y\|^\eta \tag{6}$$

for almost every $x, y \in \mathrm{supp}(\rho_k^\tau)$ and a constant $C > 0$. The experimental advantages of using such networks have been shown in (Karkar et al. 2020). Naively composing these stability bounds on T_k^τ and $\tilde{T}_k^\tau$ allows to get stability bounds for the composition networks G_K^τ and $\tilde{G}_K^\tau = \tilde{T}_K^\tau \circ .. \circ \tilde{T}_0^\tau$.

Link with the minimizing movement scheme

The Appendix gives a background on gradient flows and the minimizing movement scheme following (Ambrosio, Gigli, and Savare 2005; Santambrogio 2016). Given a compact set $\Omega \subset \mathbb{R}^d$ and a lower semi-continuous function $\mathcal{L} : \mathbb{W}_2(\Omega) \to \mathbb{R} \cup \{\infty\}$, the minimizing movement scheme is a discretized gradient flow that is well-defined in non-Euclidean metric spaces and can, under some conditions, minimize $\mathcal{L}$ starting from $\rho_0^\tau \in \mathcal{P}(\Omega)$. It is given by

$$\rho_{k+1}^\tau \in \arg\min_{\rho \in \mathcal{P}(\Omega)} \mathcal{L}(\rho) + \frac{1}{2\tau} W_2^2(\rho, \rho_k^\tau) \tag{7}$$

This problem has a solution because the objective is lower semi-continuous and $\mathcal{P}(\Omega)$ is compact. It is equivalent to

$$T_{k+1}^\tau \in \arg\min_{T:\Omega\to\Omega} \mathcal{L}(T_\sharp \rho_k^\tau) + \frac{1}{2\tau} W_2^2(T_\sharp \rho_k^\tau, \rho_k^\tau) \tag{8}$$

with $\rho_{k+1}^\tau = (T_k^\tau)_\sharp \rho_k^\tau$, under conditions that guarantee the existence of a transport map between ρ_k^τ and any other measure, for example $\partial\Omega$ negligible and ρ_k^τ absolutely continuous, and $\mathcal{L}$ can ensure that ρ_{k+1}^τ is also absolutely continuous. Among the functions T_{k+1}^τ that solve problem (8), is the optimal transport map from ρ_k^τ to ρ_{k+1}^τ. To solve for this optimal map, we consider the following equivalent problem

$$T_{k+1}^\tau \in \arg\min_{T:\Omega\to\Omega} \mathcal{L}(T_\sharp \rho_k^\tau) + \frac{1}{2\tau} \int_\Omega \|T(x) - x\|^2 \, \mathrm{d}\rho_k^\tau(x) \tag{9}$$

This problem is equivalent to problem (8) as it has the same minimum value, but its minimizer is now an optimal transport map between ρ_k^τ and a minimizer ρ_{k+1}^τ of (7).

Therefore, if we cast a learning problem as the minimization of a loss $\mathcal{L}$ over the space of measures starting from the input data distribution ρ_0, then this suggests solving it through the minimizing movement scheme in $\mathbb{W}_2(\Omega)$, using neural nets T_k^τ found by solving (9) that will have regularity (6) to push the successive distributions ρ_k^τ towards a minimizer of $\mathcal{L}$. We find again our regularization for module-wise training of neural nets as formulated in (5), when auxiliary networks are not necessary, for example in generative tasks. Note that the representation power of neural nets shown by universal approximation theorems is important here to get close to equivalence between (7) and (8) when restricting the optimization in (8) to neural nets.

As mentioned in the Appendix, for losses $\mathcal{L}$ that are λ-geodesically convex for $\lambda > 0$, we can show convergence of this scheme as $k \to \infty$ and $\tau \to 0$ to a minimizer of $\mathcal{L}$, potentially under more technical conditions (Santambrogio 2016). Functionals of distributions that might be λ-geodesically convex and are useful as machine learning losses (mostly for generation and transfer tasks) include $\mathcal{V}(\rho) = \int V \, \mathrm{d}\rho$ (depending on real-valued function V) (Santambrogio 2016)). Since these functionals have been explored in (Alvarez-Melis and Fusi 2021), and since their first variations are known in closed form, rendering optimization methods other than the minimizing movement scheme applicable, we choose to focus on less amenable (from this viewpoint) tasks, namely classification. In tasks such as classification, the loss is preceded by a function (a classification head) that maps the transported input to the desired dimension. Introducing this function into the minimizing movement scheme (9) leads directly to our regularized module-wise training problem as formulated in (5). This is no longer exactly a minimizing movement scheme, except in applications where we can fix the functions $F_k^\tau = F$, but the convergence discussion still suggests taking τ as small as possible and many modules T_k^τ.

Practical implementation

The module-wise problems (1) can be solved in one of two ways. One can completely train each module with its auxiliary classifier for N epochs before training the next module, which receives as input the output of the previous trained module. We call this *sequential* module-wise training. But we can also do this batch-wise, i.e. do a complete forward pass on each batch but without a full backward pass, rather a backward pass that only updates the current module T_k^τ and its auxiliary classifier F_k^τ, meaning that T_k^τ forwards its output to T_{k+1}^τ immediately after it computes it. We call this *parallel* module-wise training. It is called *decoupled* greedy training in (Belilovsky, Eickenberg, and Oyallon 2020), which shows that combining it with buffers solves all three locking problems and allows a linear training parallelization in the depth of the network. We propose a variant of sequential module-wise training that we call *multi-lap sequential* module-wise training, in which instead of training each module for N epochs, we train each module from the first to the last sequentially for N/R epochs, then go back and train from the first module to the last for N/R epochs again, and we do this for R laps. For the same total number of epochs and training time, and the same advantages (loading and training one module at a time) this provides a non-negligible improvement in accuracy over normal sequential module-wise training in most cases, as shown below. Despite our theoretical framework being that of sequential module-wise training, our method improves the test accuracy of all three layer-wise training regimes. And in all three cases, we use SGD.

Formulation (5) gives another way of implementing the regularization, especially useful for non-residual architecture if the module preserves the dimension. Its discretization is

$$(T_k^\tau, F_k^\tau) \in \arg\min_{T,F} \sum_{x \in \tilde{\rho}_0} L(F, T(G_{k-1}^\tau(x)) + \frac{1}{2\tau} \|T(G_{k-1}^\tau(x)) - G_{k-1}^\tau(x)\|^2 \tag{10}$$

Simple variations of the method work better in practice in some cases. For example, instead of using a fixed weight τ for the transport cost, we can vary it along the depth k to further constrain with a smaller τ_k the earlier modules to avoid that they overfit or the later modules to maintain the performance of earlier modules.

We might also want to regularize the networks further in earlier epochs when the data is more entangled as in (Karkar et al. 2020). To unify and formalize this varying weight $\tau_{k,i}$ across modules k and SGD iterations i, we use a scheme inspired by the method of multipliers to solve problems (3) and (10). To simplify the notations, we will instead consider the weight $\lambda_{k,i} = 2\tau_{k,i}$ given to the loss. We denote $\theta_{k,i}$

the parameters of both T_k and F_k at SGD iteration i. We also denote $L(\theta, x)$ and $T(\theta, x)$ respectively the loss and the transport regularization as functions of parameters θ and data point x. We now increase the weight $\lambda_{k,i}$ of the loss every s iterations of SGD by a value that is proportional to the current loss. Given increase factor $h > 0$, initial parameters $\theta_{k,1}$, initial weight $\lambda_{k,1} \geq 0$, learning rates (η_i) and batches (x_i), we apply for module k and $i \geq 1$:

$$\theta_{k,i+1} = \theta_{k,i} - \eta_i \nabla_\theta (\lambda_{k,i}\, L(\theta_{k,i}, x_i) + T(\theta_{k,i}, x_i))$$
$$\lambda_{k,i+1} = \lambda_{k,i} + hL(\theta_{k,i+1}, x_{i+1}) \text{ if } i \bmod s = 0 \text{ else } \lambda_{k,i}$$

The weights $\lambda_{k,i}$ will vary along the depth k even if we use the same initial weights $\lambda_{k,1} = \lambda_1$ because they will evolve differently with iterations i for each k. They will increase more slowly with i for larger k because deeper modules will have smaller loss. We are then regularizing the earlier blocks less. This method can be seen as a method of multipliers for the problem of minimizing the transport under the constraint of zero loss (a reasonable assumption as recent deep learning architectures have shown to systematically achieve near zero training loss (Zhang et al. 2017; Jacot, Hongler, and Gabriel 2018; Belkin, Ma, and Mandal 2018, 2019)). Therefore it is immediate by slightly adapting the proof of Theorem 1 or from (Karkar et al. 2020) that we are still solving a problem that admits a solution whose non-auxiliary part is an optimal transport map with the same regularity as stated above. This method works better than a simple fixed τ in some experiments, but has more hyperparameters to be tuned.

Experiments

We consider classification tasks, with L being the cross-entropy loss. For the ResBlocks, we use the architecture from (He et al. 2016a). For the auxiliary classifiers, we use the architecture from (Belilovsky, Eickenberg, and Oyallon 2019, 2020), that is a convolution followed by an average pooling and a fully connected layer. The experiments below then show that our method combines well with theirs and improves on it when using ResNets. Code is available at `github.com/block-wise/block-wise`.

The first task is training a 10-block ResNet block-wise on CIFAR100 (Krizhevsky 2009) with standard data augmentation. The network starts with an encoder, i.e. a first layer that downsamples the images into a 16×16 shape with 256 filters and that is also trained greedily with its auxiliary classifier, but without transport regularization. A further downsampling into shape $512 \times 8 \times 8$ takes place after 5 blocks via a convolutional layer that is considered part of the following block but is also not transport regularized. We use orthogonal initialization (Saxe, Mcclelland, and Ganguli 2014) with a gain of 0.05. For sequential and multi-lap sequential training, we use SGD with a learning rate of 0.007. For parallel training we use SGD with learning rate of 0.003. For sequential training, block k is trained for $50 + 10k$ epochs where $0 \leq k \leq 10$, block 0 being the encoder. This idea of increasing the number of epochs per layer along with the depth is found in (Marquez, Hare, and Niranjan 2018). For multi-lap sequential training, block k is trained for $10 + 2k$ epochs, and this is repeated for 5 laps. For parallel training, the network is trained for 300 epochs. These architectural and training choices have been made to improve the baseline test performance of vanilla block-wise training without transport regularization. For reference, we also report the end-to-end test performance of the same architecture, trained for 300 epochs with a learning rate of 0.1 that is divided by five at epochs 120, 160 and 200. For each block-wise training method, we report the highest test accuracy achieved along the depth of the greedily trained blocks, for different sizes of the train set.

In Table 1 we report the results for both methods of sequential training and end-to-end training for comparison. We see that multi-lap sequential training improves the test accuracy of sequential training by around 0.8 percentage points when the training dataset is large, but works less well on a small training set. Our method mainly improves the test accuracy of multi-lap sequential training. The improvement increases as the training set gets smaller and reaches 1 percentage point. In Table 3 we report the results for parallel training, which already performs quite close to end-to-end training in the full data regime and even better in the small data regime. Again the improvement in test accuracy from the regularization happens mostly with smaller training sets. As in (Belilovsky, Eickenberg, and Oyallon 2020), parallel greedy training performs better than sequential training for the same architecture and a somewhat shorter total training time. An observation that is confirmed in all subsequent experiments. Note that performances of around 70% in these tables for block-wise training on the full CIFAR100 dataset are comparable to, and even a little better than, the DDG method (Table 3 in (Huo et al. 2018)) and the ResIST method (Table 2 in (Dun et al. 2021)), even though they only split much deeper ResNets in two parts (DDG) or in up to 8 parts (ResIST).

We also report results for similar experiments with some variations in the architecture and on other datasets (MNIST (LeCun, Cortes, and Burges 2010) and CIFAR10 (Krizhevsky 2009)), but here we report the accuracy achieved by the last block. We see a similar pattern of a greater improvement due to the regularization as the training sets get smaller, gaining as much as 6 percentage points in some cases (Tables 5, and 7 and 8 in the Appendix). We notice that the improvement from the regularization is more important in these three tables where the architecture has not been particularly adapted to block-wise training (i.e. the networks are deeper and not very wide, a classic classifier made up of one or two linear layers is used as opposed to the classifier from (Belilovsky, Eickenberg, and Oyallon 2019)). Regularizing can then replace too much architectural fine-tuning, although it still helps, even if a little, in all cases. Finally, we include in Table 6 in the Appendix the accuracy achieved by the best block along the depth for the same experiment as in Table 5 and we notice a more important improvement in the accuracy of the last block than in the accuracy of the best block when using the regularization. The around 88% accuracy on CIFAR10 of sequential training (Tables 5 and 6) is comparable to results for sequential training in Table 2 of (Belilovsky, Eickenberg, and Oyallon 2019) (with VGG networks of comparable depth and width). We also observe that with the regularization the difference between the accuracy of the last block and that of the best block is smaller than without the regularization. The

Train size	seq	seq with reg	multi-lap seq	multi-lap seq with reg	end-to-end
50000	68.74 ± 0.45	**68.79** ± 0.56	69.48 ± 0.53	**69.95** ± 0.50	75.85 ± 0.70
25000	60.48 ± 0.15	**60.59** ± 0.14	61.33 ± 0.23	**61.71** ± 0.32	65.36 ± 0.31
12500	51.64 ± 0.33	**51.74** ± 0.26	51.30 ± 0.22	**51.89** ± 0.30	52.39 ± 0.97
5000	36.37 ± 0.33	**36.40** ± 0.40	33.68 ± 0.48	**34.61** ± 0.59	36.38 ± 0.31

Table 1: Average highest test accuracy (best block) and 95% confidence interval of 10-1 ResNet models (256 filters) over 10 runs on CIFAR100 with train sets of different sizes and different methods of training: block-wise sequential (seq), block-wise multi-lap sequential (multi-lap seq), both with and without the transport regularization (reg), and end-to-end.

seq	seq with reg	multi-lap seq	multi-lap seq with reg	end-to-end
71.47 ± 0.60	**71.98** ± 0.58	73.35 ± 0.53	**73.59** ± 0.52	75.85 ± 0.70

Table 2: Average highest test accuracy (best block) and 95% confidence interval of 2-5 ResNet models (256 filters) over 10 runs on CIFAR100 with different methods of training: module-wise sequential (seq), module-wise multi-lap sequential (multi-lap seq), both with and without the transport regularization (reg), and end-to-end.

Train	par	par with reg	end-to-end
50000	72.59 ± 0.40	**72.63** ± 0.40	75.85 ± 0.70
25000	64.84 ± 0.19	**65.01** ± 0.27	65.36 ± 0.31
12500	55.13 ± 0.24	**55.40** ± 0.35	52.39 ± 0.97
5000	39.45 ± 0.23	**40.36** ± 0.23	36.38 ± 0.31

Table 3: Average highest test accuracy (best block) and 95% confidence interval of 10-1 ResNet models (256 filters) over 10 runs on CIFAR100 with train sets of different sizes and block-wise parallel (par) training with and without the transport regularization (reg) and end-to-end training.

regularization then helps to train deeper networks block-wise.

We confirm this through the following experiment. As the network gets deeper (50 blocks trained for 10 epochs), we expect training it block-wise to become more difficult, and the improvement from the regularization increases when looking at the accuracy of the last block for sequential training (Table 13 in the Appendix). The Appendix also contains in Tables 11 and 12 results on ResNeXt-50-32×4d (Xie et al. 2017), which turns out to be difficult to train block-wise.

The second task is splitting the 10-block ResNet from the first task into two modules of 5 blocks (plus the encoder) trained module-wise on all of CIFAR100. When trained sequentially, module k is trained for $50 + 75k$ epochs for $0 \leq k \leq 2$. When trained sequentially in multiple laps, module k is trained for $10 + 15k$ epochs for 5 laps. When trained in parallel, we train for 300 epochs. Initialization and learning rates are the same. Here the highest test accuracy along the depth is always that of the second module. In Table 2 (sequential training) and Table 9 in the Appendix (parallel training) we see improvement from the regularization mostly for sequential training. We also find in Table 2 that multi-lap sequential training improves by 2 percentage points on simple sequential training. In these tables 2-5 ResNet means 2 modules of 5 blocks each trained module-wise. The 75% test accuracy on CIFAR100 in Table 9 in the Appendix for parallel training is very close to the end-to-end baseline in Table 1, beats the DDG method (Table 3 in (Huo et al. 2018)) and is comparable to the Features Replay method (Table 2 in (Huo, Gu, and Huang 2018)). To compare to InfoPro, we train the same model in parallel on CIFAR10 (Table 10 in the Appendix) and find an improvement of around 0.5 percentage points over InfoPro (Table 2 in (Wang et al. 2021)).

Finally, we train a 16-block ResNet (with 256 initial filters and downsampling and doubling of the filters at the midpoint) divided in 4 modules of 4 blocks (a 4-4 ResNet in our notations) module-wise on TinyImageNet. Parallel module-wise training in this case consumes about 25% less memory than end-to-end training (6951MiB vs 9241MiB). We compare in Table 4 our results in this setup to those of three of the best recent parallel module-wise training methods: DGL (Belilovsky, Eickenberg, and Oyallon 2020), PredSim (Nøkland and Eidnes 2019) and Sedona (Pyeon et al. 2021), as reported in Table 2 of (Pyeon et al. 2021). The benefit of the regularization is clear as it adds 1.2 percentage points of accuracy. While our network has around 51 million parameters, our method easily beats the first two methods, even when they use a bigger ResNet152 (around 59 million parameters) divided in four. Our method beats Sedona when it uses a ResNet101 (43 million parameters) but not a ResNet152. However, Sedona is an architecture search method that first searches for the best auxiliary architectures and the best positions to split the network in four, which requires a long pre-training time, before the actual module-wise training. The methods are therefore not quite comparable and can be combined by adding the regularization at the module-wise training phase of Sedona. We report in Table 14 in the Appendix results from training these 4-4 ResNets on TinyImageNet sequentially. We find that multi-lap sequential training

par (ours)	par with reg (ours)	DGL ResNet152	PredSim ResNet152	Sedona ResNet101	Sedona ResNet152
59.56 ± 0.16	60.72 ± 0.44	57.64	51.76	59.12	64.10

Table 4: Average test accuracy and 95% confidence intervals of 4-4 ResNet models (256 filters) over 5 runs on TinyImageNet with block-wise parallel training (par) and block-wise parallel training with transport regularization (reg), compared to methods DGL, PredSim and Sedona from (Pyeon et al. 2021) that also split their networks in 4 module-wise-parallel-trained modules.

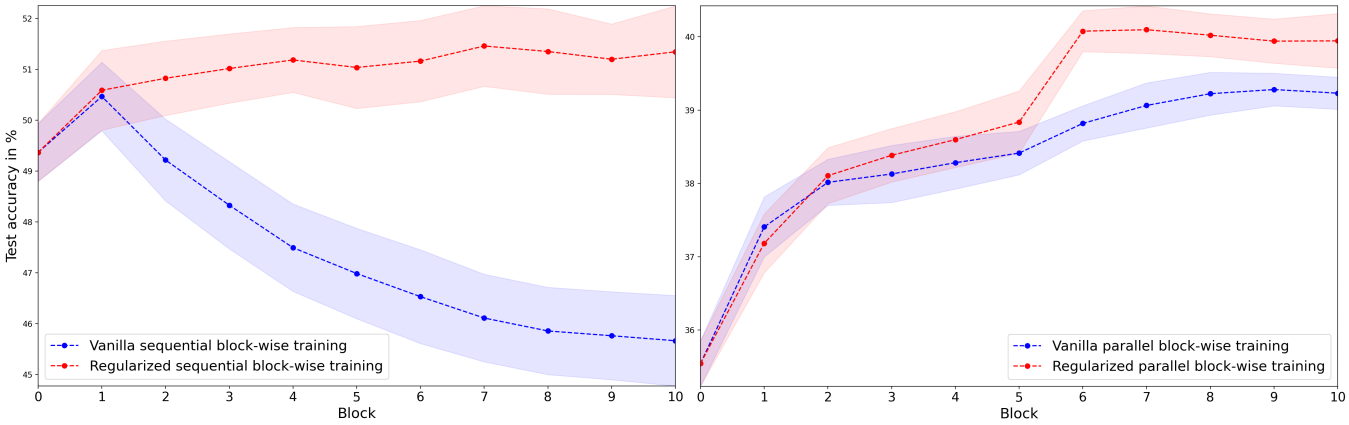

Figure 1: Highest test accuracy after each block of 10-block ResNet averaged over 10 runs with 95% confidence intervals. Left: vanilla (blue) and regularized (red) sequential block-wise training on 2% of the CIFAR10 training set. Right: vanilla (blue) and regularized (red) parallel block-wise training on 10% of the CIFAR100 training set.

Train	seq	seq with reg	end-to-end
50000	88.02 ± 0.18	**88.20** ± 0.24	91.88 ± 0.18
25000	83.95 ± 0.13	**84.28** ± 0.22	88.75 ± 0.27
10000	76.00 ± 0.39	**77.18** ± 0.34	82.61 ± 0.35
5000	67.74 ± 0.49	**69.67** ± 0.44	73.93 ± 0.67
1000	45.67 ± 0.88	**51.34** ± 0.90	50.63 ± 0.98

Table 5: Average highest test accuracy (last block) and 95% confidence interval of 10-1 ResNet models (256 filters) over 10 runs on CIFAR10 with train sets of different sizes and block-wise sequential (seq) training with and without the transport regularization (reg) and end-to-end training.

improves over simple sequential training by 1 percentage point and that the regularization improves the performance of multi-lap sequential training by almost 1 percentage point.

In Figure 1, we look at the test accuracy of each block after block-wise training with and without the regularization. On the left, from experiments with sequential block-wise training from Table 5 on a train set of 1000 CIFAR10 images, we see a large decline in accuracy after the first block (block 0 being the encoder) that our method avoids. On the right, from experiments with parallel block-wise training from Table 3 on a train set of 5000 CIFAR100 images, we see a steeper increase in test accuracy along the blocks with our method. We see the same pattern in Figure 2 in the Appendix from experiments with multi-lap sequential block-wise training from Table 1 on a train set of 5000 CIFAR100 images.

Conclusion

We introduced a kinetic energy regularization for block-wise training of ResNets that links block-wise training to gradient flows of the loss in distribution space. The method provably leads to more regular blocks and experimentally improves the test classification accuracy of block-wise sequential, parallel and multi-lap sequential (a variant of sequential training that we introduce) training, especially in small data regimes. The method can easily be adapted to layer-wise training of non-residual networks and combined with other methods that improve the accuracy of layer-wise training.

Future work can experiment with working in Wasserstein space W_p for $p \neq 2$, i.e. regularizing with a norm $\|.\|_p$ with $p \neq 2$. One can also ask how far the obtained composition network G_K is from being an OT map itself, which could provide a better stability bound than the one obtained by naively chaining the stability bounds (6) that follow from each module T_k being an OT map.

Making layer-wise training competitive allows to benefit from its practical advantages without losing too much performance in comparison with end-to-end training. These benefits include the possibility of on-device training in memory-constrained settings, which can be required for privacy reasons for example, via sequential layer-wise training that only requires loading and training one layer at a time, and training parallelism that differs from and complements data and model parallelism, via parallel layer-wise training.

References

Alvarez-Melis, D.; and Fusi, N. 2021. Dataset Dynamics via Gradient Flows in Probability Space. *ICML*.

Ambrosio, L.; Gigli, N.; and Savare, G. 2005. *Gradient Flows in Metric Spaces and in the Space of Probability Measures*. Birkhäuser Basel.

Ansari, A. F.; Ang, M. L.; and Soh, H. 2021. Refining Deep Generative Models via Discriminator Gradient Flow. In *ICLR*.

Arbel, M.; Korba, A.; Salim, A.; and Gretton, A. 2019. Maximum Mean Discrepancy Gradient Flow. In *NeurIPS*.

Belilovsky, E.; Eickenberg, M.; and Oyallon, E. 2019. Greedy Layerwise Learning Can Scale to ImageNet. In *ICML*.

Belilovsky, E.; Eickenberg, M.; and Oyallon, E. 2020. Decoupled Greedy Learning of CNNs. In *ICML*.

Belkin, M.; Ma, S.; and Mandal, S. 2018. To Understand Deep Learning We Need to Understand Kernel Learning. In *ICML*, 540–548.

Belkin, M.; Ma, S.; and Mandal, S. 2019. Reconciling modern machine-learning practice and the classical bias–variance trade-off. In *PNAS*, volume 116, 15849–15854.

Benamou, J.; and Brenier, Y. 2000. A computational fluid mechanics solution to the Monge-Kantorovich mass transfer problem. *Numerische Mathematik*.

Bengio, Y.; Lamblin, P.; Popovici, D.; and Larochelle, H. 2006. Greedy Layer-Wise Training of Deep Networks. In *NeurIPS*.

Bernd Illing, G. B., Wulfram Gerstner. 2020. Towards Truly Local Gradients with CLAPP: Contrastive, Local and Predictive Plasticity. *arXiv*.

Chen, R. T. Q.; Rubanova, Y.; Bettencourt, J.; and Duvenaud, D. 2018. Neural Ordinary Differential Equations. In *NeurIPS*.

Czarnecki, W. M.; Świrszcz, G.; Jaderberg, M.; Osindero, S.; Vinyals, O.; and Kavukcuoglu, K. 2017. Understanding Synthetic Gradients and Decoupled Neural Interfaces. In *ICML*.

De, S.; and Smith, S. L. 2020. Batch Normalization Biases Residual Blocks Towards the Identity Function in Deep Networks. In *NeurIPS*.

de Bézenac, E.; Ayed, I.; and Gallinari, P. 2019. Optimal Unsupervised Domain Translation. *arXiv*.

Dun, C.; Wolfe, C. R.; Jermaine, C. M.; and Kyrillidis, A. 2021. ResIST: Layer-Wise Decomposition of ResNets for Distributed Training. *arXiv*.

Figalli, A. 2017. *The Monge-Ampere Equation and Its Applications*. Zurich lectures in advanced mathematics. European Mathematical Society.

Finlay, C.; Jacobsen, J.-H.; Nurbekyan, L.; and Oberman, A. M. 2020. How To Train Your Neural ODE. In *ICML*.

Gao, Y.; Jiao, Y.; Wang, Y.; Wang, Y.; Yang, C.; and Zhang, S. 2019. Deep Generative Learning via Variational Gradient Flow. In *ICML*.

Gupta, S. K. 2020. A More Biologically Plausible Local Learning Rule for ANNs. *arXiv*.

Hauser, M. 2019. On Residual Networks Learning a Perturbation from Identity. *arXiv*.

He, K.; Zhan, X.; Ren, S.; and Sun, J. 2016a. Identity Mappings in Deep Residual Networks. In *ECCV*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016b. Deep Residual Learning for Image Recognition. In *CVPR*.

Huang, F.; Ash, J.; Langford, J.; and Schapire, R. 2018. Learning Deep ResNet Blocks Sequentially Using Boosting Theory. In *ICML*.

Huo, Z.; Gu, B.; and Huang, H. 2018. Training Neural Networks using Features Replay. In *NeurIPS*.

Huo, Z.; Gu, B.; Yang, Q.; and Huang, H. 2018. Decoupled Parallel Backpropagation with Convergence Guarantee. In *ICML*.

Jacot, A.; Hongler, C.; and Gabriel, F. 2018. Neural Tangent Kernel: Convergence and Generalization in Neural Networks. In *NeurIPS*, 8580–8589.

Jaderberg, M.; Czarnecki, W. M.; Osindero, S.; Vinyals, O.; Graves, A.; Silver, D.; and Kavukcuoglu, K. 2017. Decoupled Neural Interfaces using Synthetic Gradients. In *ICML*.

Jastrzebski, S.; et al. 2018. Residual Connections Encourage Iterative Inference. In *ICLR*.

Karkar, S.; Ayed, I.; de Bézenac, E.; and Gallinari, P. 2020. A Principle of Least Action for the Training of Neural Networks. In *ECML-PKDD*.

Krizhevsky, A. 2009. Learning Multiple Layers of Features from Tiny Images. *University of Toronto Technical Report*.

LeCun, Y.; Cortes, C.; and Burges, C. J. 2010. MNIST handwritten digit database. *yann.lecun.com/exdb/mnist*.

Liutkus, A.; Imşekli, U.; Majewski, S.; Durmus, A.; and Stoter, F.-R. 2019. Sliced-Wasserstein Flows: Nonparametric Generative Modeling via Optimal Transport and Diffusions. In *ICML*.

Lu, Y.; Zhong, A.; Li, Q.; and Dong, B. 2018. Beyond Finite Layer Neural Networks: Bridging Deep Architectures and Numerical Differential Equations. In *ICML*.

Ma, W.-D. K.; Lewis, J.; and Kleijn, W. B. 2020. The HSIC Bottleneck: Deep Learning without Back-Propagation. In *AAAI*.

Mandar Kulkarni, S. K. 2016. Layer-wise Training of Deep Networks Using Kernel Similarity. In *DLPR workshop, ICPR*.

Marquez, E. S.; Hare, J. S.; and Niranjan, M. 2018. Deep Cascade Learning. *IEEE Transactions on Neural Networks and Learning Systems*.

Mostafa, H.; Ramesh, V.; and Cauwenberghs, G. 2018. Deep Supervised Learning Using Local Errors. *Front. Neurosci.*

Nguyen, T. T.; and Choi, J. 2019. Layer-wise Learning of Stochastic Neural Networks with Information Bottleneck. *Entropy*, 21.

Nøkland, A.; and Eidnes, L. H. 2019. Training Neural Networks with Local Error Signals. In *ICML*.

Pyeon, M.; Moon, J.; Hahn, T.; and Kim, G. 2021. SEDONA: Search for Decoupled Neural Networks toward Greedy Block-wise Learning. In *ICLR*.

Santambrogio, F. 2015. *Optimal Transport for Applied Mathematicians*. Birkhäuser.

Santambrogio, F. 2016. Euclidean, Metric, and Wasserstein Gradient Flows: an overview. *arXiv*.

Saxe, A. M.; Mcclelland, J. L.; and Ganguli, S. 2014. Exact solutions to the nonlinear dynamics of learning in deep linear neural network. In *ICLR*.

Sindy Löwe, B. V., Peter O'Connor. 2019. Putting An End to End-to-End: Gradient-Isolated Learning of Representations. In *NeurIPS*.

Stich, S. U. 2019. Local SGD Converges Fast and Communicates Little. In *ICLR*.

Tang, Y.; Teng, Q.; Zhang, L.; Min, F.; and He, J. 2021. Layer-wise Training Convolutional Neural Networks with Smaller Filters for Human Activity Recognition Using Wearable Sensors. *IEEE Sensors Journal*.

Teng, Q.; Wang, K.; Zhang, L.; and He, J. 2020. The Layer-Wise Training Convolutional Neural Networks Using Local Loss for Sensor-Based Human Activity Recognition. *IEEE Sensors Journal*.

Villani, C. 2008. *Optimal Transport: Old and New*. Springer-Verlag.

Wang, Y.; Ni, Z.; Song, S.; and Le Yang, G. H. 2021. Revisiting Locally Supervised Learning: an Alternative to End-to-end Training. In *ICLR*.

Weinan, E. 2017. A Proposal on Machine Learning via Dynamical Systems. *Commun. Math. Stat*.

Wightman, R.; Touvron, H.; and Jégou, H. 2021. ResNet strikes back: An improved training procedure in timm. *arXiv*.

Xie, S.; et al. 2017. Aggregated Residual Transformations for Deep Neural Networks. In *CVPR*.

Yuwen Xiong, R. U., Mengye Ren. 2020. LoCo: Local Contrastive Representation Learning. In *NeurIPS*.

Zagoruyko, S.; and Komodakis, N. 2016. Wide Residual Networks. In *BMVC*.

Zhang, C.; Bengio, S.; Hardt, M.; Recht, B.; and Vinyals, O. 2017. Understanding deep learning requires rethinking generalization. In *ICLR*.

Zhang, J.; et al. 2019. Towards Robust ResNet: A Small Step but a Giant Leap. In *IJCAI*.

Appendix

Background on optimal transport

The Wasserstein space $\mathbb{W}_2(\Omega)$ with Ω a convex and compact subset of $\mathbb{R}^d$ is the space $\mathcal{P}(\Omega)$ of probability measures over Ω, equipped with the distance W_2 given by the solution to the optimal transport problem

$$W_2^2(\alpha,\beta) = \min_{\gamma\in\Pi(\alpha,\beta)} \int_{\Omega\times\Omega} \|x-y\|^2 \,\mathrm{d}\gamma(x,y) \qquad (11)$$

where $\Pi(\alpha,\beta)$ is the set of probability distribution over $\Omega\times\Omega$ with first marginal α and second marginal β, i.e. $\Pi(\alpha,\beta) = \{\gamma\in\mathcal{P}(\Omega\times\Omega) \mid \pi_{1\sharp}\gamma = \alpha,\ \pi_{2\sharp}\gamma = \beta\}$ where $\pi_1(x,y) = x$ and $\pi_2(x,y) = y$. The optimal transport problem can be seen as looking for a transportation plan minimizing the cost of displacing some distribution of mass from one configuration to another. This problem indeed has a solution in our setting and W_2 can be shown to be a geodesic distance (see for example (Santambrogio 2015; Villani 2008)). If α is absolutely continuous and $\partial\Omega$ is α-negligible then the problem in (11) (called the Kantorovich problem) has a unique solution and is equivalent to the Monge problem, i.e.

$$W_2^2(\alpha,\beta) = \min_{T \text{ s.t. } T_\sharp\alpha=\beta} \int_\Omega \|T(x)-x\|^2 \,\mathrm{d}\alpha(x) \qquad (12)$$

and this problem has a unique solution $T^\star$ linked to the solution $\gamma^\star$ of (11) through $\gamma^\star = (\mathrm{id}, T^\star)_\sharp\alpha$. Another equivalent formulation of the optimal transport problem in this setting is the dynamical formulation (Benamou and Brenier 2000). Here, instead of directly pushing samples of α to β using T, we can equivalently displace mass, according to a continuous flow with velocity $v_t : \mathbb{R}^d \to \mathbb{R}^d$. This implies that the density α_t at time t satisfies the *continuity equation* $\partial_t\alpha_t + \nabla\cdot(\alpha_t v_t) = 0$, assuming that initial and final conditions are given by $\alpha_0 = \alpha$ and $\alpha_1 = \beta$ respectively. In this case, the optimal displacement is the one that minimizes the total action caused by v :

$$\begin{aligned} W_2^2(\alpha,\beta) = &\min_v \int_0^1 \|v_t\|^2_{L^2(\alpha_t)} \,\mathrm{d}t \qquad (13)\\ &\text{s.t. } \partial_t\alpha_t + \nabla\cdot(\alpha_t v_t) = 0,\ \alpha_0 = \alpha, \alpha_1 = \beta \end{aligned}$$

Instead of describing the density's evolution through the continuity equation, we can describe the paths ϕ_t^x taken by particles at position x from α when displaced along the flow v. Here ϕ_t^x is the position at time t of the particle that was at $x\sim\alpha$ at time 0. The continuity equation is then equivalent to $\partial_t\phi_t^x = v_t(\phi_t^x)$. See chapters 4 and 5 of (Santambrogio 2015) for details. Rewriting the conditions as necessary, Problem (13) becomes

$$\begin{aligned} W_2^2(\alpha,\beta) = &\min_v \int_0^1 \|v_t\|^2_{L^2((\phi_t^\cdot)_\sharp\alpha)} \,\mathrm{d}t \qquad (14)\\ &\text{s.t. } \partial_t\phi_t^x = v_t(\phi_t^x),\ \phi_0^\cdot = \mathrm{id}, (\phi_1^\cdot)_\sharp\alpha = \beta \end{aligned}$$

and the optimal transport map $T^\star$ that solves (12) is in fact $T^\star(x) = \phi_1^x$ for ϕ that solves the continuity equation together with the optimal $v^\star$ from (14). We refer to (Santambrogio 2015; Villani 2008) for these results on optimal transport.

Optimal transport maps have some regularity properties under some boundedness assumptions. We mention the following result from (Figalli 2017):

Theorem 2. *Let α and β be absolutely continuous measures on $\mathbb{R}^d$ and T the optimal transport map between α and β for the Euclidean cost. Suppose there are bounded open sets X and Y, such that the density of α (respectively of β) is null on X^c (respectively Y^c) and bounded away from zero and infinity on X (respectively Y).*

Then there exists two relatively closed sets of null measure $A\subset X$ and $B\subset Y$, such that T is η-Hölder continuous from $X\setminus A$ to $Y\setminus B$, i.e. $\forall\, x,y\in X\setminus A$ we have

$$\|T(x)-T(y)\| \le C\|x-y\|^\eta \text{ for constants } \eta, C > 0$$

Proof of Theorem 1

Proof. Take a minimizing sequence $(\tilde{F}_i, \tilde{T}_i)$, i.e. such that $\mathcal{C}(\tilde{F}_i, \tilde{T}_i) \to \min \mathcal{C}$, where $\mathcal{C} \geq 0$ is the target function in (5) and denote $\beta_i = \tilde{T}_{i\sharp}\rho_k^\tau$. Then by compacity $\tilde{F}_i \to F^\star$ and $\beta_i \rightharpoonup \beta^\star$ in duality with $\mathcal{C}_b(\Omega)$ by Banach-Alaoglu. There exists $T^\star$ an optimal transport map between ρ_k^τ and $\beta^\star$. Then $\mathcal{C}(F^\star, T^\star) \leq \lim \mathcal{C}(\tilde{F}_i, \tilde{T}_i) = \min \mathcal{C}$ by continuity of $\mathcal{L}$ and because

$$\begin{aligned}\int_\Omega \|T^\star(x) - x\|^2 \,\mathrm{d}\rho_k^\tau(x) &= W_2^2(\rho_k^\tau, \beta^\star) \\ &= \lim W_2^2(\rho_k^\tau, \beta_i) \\ &\leq \lim \int_\Omega \|\tilde{T}_i(x) - x\|^2 \,\mathrm{d}\rho_k^\tau(x)\end{aligned}$$

as W_2 metrizes weak convergence of measures. We take $(F_k^\tau, T_k^\tau) = (F^\star, T^\star)$. It is also immediate that for any minimizing pair, the transport map has to be optimal. Taking a minimizing sequence $(\tilde{F}_i, \tilde{v}^i)$ and the corresponding induced maps $\tilde{T}_i$ we get the same result for problem (4). The two problems are equivalent by the equivalence between problems (12) and (14). □

Background on gradient flows

We follow (Santambrogio 2016; Ambrosio, Gigli, and Savare 2005) for this background on gradient flows. Given a function $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ and an initial point $x_0 \in \mathbb{R}^d$, a *gradient flow* is a curve $x : [0, \infty[\to \mathbb{R}^d$ that solves the Cauchy problem

$$\begin{cases} x'(t) = -\nabla \mathcal{L}(x(t)) \\ x(0) = x_0 \end{cases} \tag{15}$$

A solution exists and is unique if $\nabla \mathcal{L}$ is Lipschitz or $\mathcal{L}$ is convex. Given $\tau > 0$ and $x_0^\tau = x_0$ define a sequence $(x_k^\tau)_k$ through the *minimizing movement scheme*:

$$x_{k+1}^\tau \in \arg\min_{x \in \mathbb{R}^d} \mathcal{L}(x) + \frac{1}{2\tau}\|x - x_k^\tau\|^2 \tag{16}$$

$\mathcal{L}$ lower semi-continous and $\mathcal{L}(x) \geq C_1 - C_2\|x\|^2$ guarantees existence of a solution of (16) for τ small enough. $\mathcal{L}$ λ-convex meets these conditions and also provides uniqueness of the solution because of strict convexity of the target. See (Santambrogio 2015, 2016; Ambrosio, Gigli, and Savare 2005).

We interpret the point x_k^τ as the value of a curve x at time $k\tau$. We can then construct a curve x^τ as the piecewise constant interpolation of the points x_k^τ. We can also construct a curve $\tilde{x}^\tau$ as the affine interpolation of the points x_k^τ.

If $\mathcal{L}(x_0) < \infty$ and $\inf \mathcal{L} > -\infty$ then (x^τ) and $(\tilde{x}^\tau)$ converge uniformly to the same curve x as τ goes to zero (up to extracting a subsequence). If $\mathcal{L}$ is $\mathcal{C}^1$, then the limit curve x is a solution of (15) (i.e. a gradient flow of $\mathcal{L}$). If $\mathcal{L}$ is not differentiable then x is solution of the problem defined using the subdifferential of $\mathcal{L}$, i.e. x satisfies $x'(t) \in -\partial\mathcal{L}(x(t))$ for almost every t.

If $\mathcal{L}$ is λ-convex with $\lambda > 0$, then the solution to (15) converges exponentially to the unique minimizer of $\mathcal{L}$ (which exists by coercivity). So taking $\tau \to 0$ and $k \to \infty$, we tend towards the minimizer of $\mathcal{L}$.

The advantage of the minimizing movement scheme (16) is that it can be adapted to metric spaces by replacing the Euclidean distance by the metric space's distance. In the (geodesic) metric space $\mathbb{W}_2(\Omega)$ with Ω convex and compact, for $\mathcal{L} : \mathbb{W}_2(\Omega) \to \mathbb{R} \cup \{\infty\}$ lower semi-continuous for the weak convergence of measures in duality with $\mathcal{C}(\Omega)$ (equivalent to lower semi-continuous with respect to the distance W_2) and $\rho_0^\tau = \rho_0 \in \mathcal{P}(\Omega)$, the minimizing movement scheme (16) becomes

$$\rho_{k+1}^\tau \in \arg\min_{\rho \in \mathcal{P}(\Omega)} \mathcal{L}(\rho) + \frac{1}{2\tau} W_2^2(\rho, \rho_k^\tau) \tag{17}$$

This problem has a solution because the objective is lower semi-continuous and the minimization is over $\mathcal{P}(\Omega)$ which is compact by Banach-Alaoglu.

We can construct a piecewise constant interpolation between the measures ρ_k^τ, or a geodesic interpolation where we travel along a geodesic between ρ_k^τ and ρ_{k+1}^τ in $\mathbb{W}_2(\Omega)$, constructed using the optimal transport map between these measures. Again, if $\mathcal{L}(x_0) < \infty$ and $\inf \mathcal{L} > -\infty$ then both interpolations converge uniformly to a limit curve $\tilde{\rho}$ as τ goes to zero. Under further conditions on $\mathcal{L}$, mainly λ-geodesic convexity (i.e. λ-convexity along geodesics) for $\lambda > 0$, we can prove stability and convergence of $\tilde{\rho}(t)$ to a minimizer of $\mathcal{L}$ as $t \to \infty$, see (Santambrogio 2015, 2016; Ambrosio, Gigli, and Savare 2005).

Additional experiments

Train	seq	seq with reg	end-to-end
50000	88.14 ± 0.14	**88.34** ± 0.22	91.88 ± 0.18
25000	84.15 ± 0.17	**84.46** ± 0.22	88.75 ± 0.27
10000	76.62 ± 0.40	**77.47** ± 0.35	82.61 ± 0.35
5000	69.60 ± 0.43	**70.22** ± 0.50	73.93 ± 0.67
1000	51.59 ± 0.91	**52.06** ± 0.71	50.63 ± 0.98

Table 6: Average highest test accuracy (best block) and 95% confidence interval of 10-1 ResNet (256 filters) over 10 runs on CIFAR10 with train sets of different sizes and block-wise sequential (seq) training with and without the transport regularization (reg) and end-to-end training.

Train	par	par with reg	end-to-end
60000	99.07 $\pm$ 0.04	**99.08** $\pm$ 0.04	99.30 $\pm$ 0.03
30000	98.90 $\pm$ 0.05	**98.93** $\pm$ 0.06	99.22 $\pm$ 0.03
12000	98.52 $\pm$ 0.06	**98.59** $\pm$ 0.06	98.96 $\pm$ 0.06
6000	98.05 $\pm$ 0.09	**98.16** $\pm$ 0.07	98.62 $\pm$ 0.06
1500	96.34 $\pm$ 0.12	**96.91** $\pm$ 0.07	97.19 $\pm$ 0.08
1200	95.80 $\pm$ 0.12	**96.58** $\pm$ 0.09	96.88 $\pm$ 0.09
600	91.35 $\pm$ 0.99	**95.16** $\pm$ 0.15	95.30 $\pm$ 0.17
300	89.81 $\pm$ 0.73	**92.86** $\pm$ 0.24	92.87 $\pm$ 0.28
150	81.84 $\pm$ 1.22	**87.48** $\pm$ 0.42	87.82 $\pm$ 0.59

Table 7: Average highest test accuracy (last block) and 95% confidence interval of 20-1 ResNet (32 filters, fixed encoder, same classifier) over 20/50 runs on MNIST with block-wise parallel (par) training with and without the transport regularization (reg) and end-to-end training.

Train	par	par with reg	end-to-end
50000	85.98 $\pm$ 0.28	**86.02** $\pm$ 0.26	93.11 $\pm$ 0.19
25000	80.94 $\pm$ 0.25	**81.09** $\pm$ 0.32	89.10 $\pm$ 0.29
10000	72.49 $\pm$ 0.46	**73.01** $\pm$ 0.31	80.52 $\pm$ 0.46
5000	62.31 $\pm$ 0.54	**64.06** $\pm$ 0.57	69.44 $\pm$ 0.88
500	38.61 $\pm$ 0.47	**41.44** $\pm$ 0.44	40.40 $\pm$ 0.60

Table 8: Average highest test accuracy (last block) and 95% confidence interval of 20-1 ResNet (100 filters, fixed encoder, same classifier) over 10 runs on CIFAR10 with block-wise parallel (par) training with and without the transport regularization (reg) and end-to-end training.

par	par with reg	end-to-end
75.23 $\pm$ 0.51	**75.37** $\pm$ 0.49	75.85 $\pm$ 0.70

Table 9: Average highest test accuracy and 95% confidence interval of 2-5 ResNet (256 filters) over 10 runs on CIFAR100 with block-wise parallel (par) training with and without the transport regularization (reg) and end-to-end training.

par	par with reg	end-to-end
93.90 $\pm$ 0.13	**93.93** $\pm$ 0.15	94.10 $\pm$ 0.34

Table 10: Average highest test accuracy and 95% confidence interval of 2-5 ResNet (256 filters) over 10 runs on CIFAR10 with block-wise parallel (par) training with and without the transport regularization (reg) and end-to-end training.

par	par with reg	end-to-end
57.86 $\pm$ 0.49	**57.93** $\pm$ 0.51	72.97 $\pm$ 1.18

Table 11: Average highest test accuracy (best block) and 95% confidence interval of ResNeXt over 10 runs on CIFAR100 with block-wise parallel (par) training with and without the transport regularization (reg) and end-to-end training.

seq	seq with reg	multi-lap seq	multi-lap seq with reg	end-to-end
52.29 $\pm$ 0.53	**52.42** $\pm$ 0.65	52.59 $\pm$ 0.63	**52.84** $\pm$ 0.65	72.97 $\pm$ 1.18

Table 12: Average highest test accuracy (best block) and 95% confidence interval of ResNeXt over 10 runs on CIFAR100 with different methods of training: block-wise sequential (seq), block-wise multi-lap sequential (multi-lap seq), both with and without the transport regularization (reg), and end-to-end.

seq	seq with reg	multi-lap seq	multi-lap seq with reg	end-to-end
63.40 $\pm$ 0.46	**63.86** $\pm$ 0.56	62.59 $\pm$ 0.64	**63.24** $\pm$ 0.50	63.34 $\pm$ 2.41

Table 13: Average highest test accuracy (last block) and 95% confidence interval of 50-1 ResNet (256 filters) over 10 runs on CIFAR100 with different methods of training: block-wise sequential (seq), block-wise multi-lap sequential (multi-lap seq), both with and without the transport regularization (reg), and end-to-end.

seq	seq with reg	multi-lap seq	multi-lap seq with reg	end-to-end
55.53 $\pm$ 1.39	**55.58** $\pm$ 1.27	56.57 $\pm$ 0.08	**57.39** $\pm$ 0.53	64.27 $\pm$ 1.28

Table 14: Average highest test accuracy and 95% confidence interval of 4-4 ResNet models (256 filters) over 3 runs on TinyImageNet with different methods of training: block-wise sequential (seq), block-wise multi-lap sequential (multi-lap seq), both with and without the transport regularization (reg), and end-to-end.

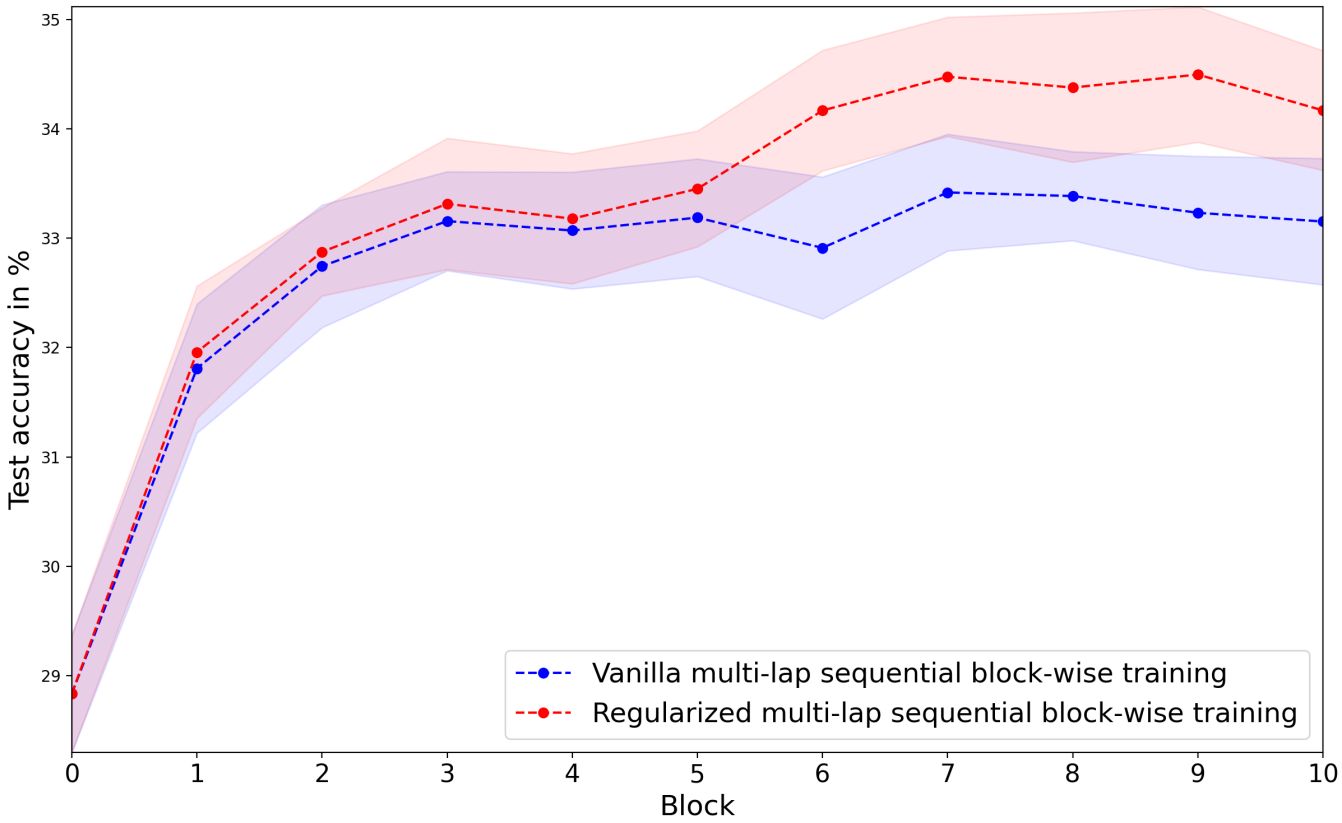

Figure 2: Highest test accuracy after each block of 10-block ResNet with multi-lap sequential block-wise training with (red) and without (blue) the transport regularization on 10% of the CIFAR100 training set averaged over 10 runs with 95% confidence intervals.