MODULE-WISE TRAINING OF RESIDUAL NETWORKS VIA THE MINIMIZING MOVEMENT SCHEME

Anonymous authors
Paper under double-blind review

ABSTRACT

Greedy layer-wise or module-wise training of neural networks is compelling in constrained and on-device settings, as it circumvents a number of problems of end-to-end back-propagation. However, it suffers from a stagnation problem, whereby early layers overfit and deeper layers stop increasing the test accuracy after a certain depth. We propose to solve this issue by introducing a simple module-wise regularization inspired by the minimizing movement scheme for gradient flows in distribution space. The method, which we call TRGL for Transport Regularized Greedy Learning, is particularly well-adapted to residual networks. We study it theoretically, proving that it leads to greedy modules that are regular and that successively solve the task. Experimentally, we show improved accuracy of module-wise trained networks when our regularization is added.

1 INTRODUCTION

End-to-end backpropagation is the standard training method of neural nets. But there are reasons to look for alternatives. First, 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). Secondly, it forces the training of systems of cooperative networks to be sequential and synchronous, which is not flexible enough when the networks are distributed between a central agent and clients that operate at different rates (Jaderberg et al. (2017)). Thirdly, it prohibits training the layers in parallel. These limitations follow from the three locking problems that end-to-end backpropagation suffers from: 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). Dividing the network into modules, a module being made up of one or more layers, and greedily solving module-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 batch buffers, parallel module-wise training solves all three problems (Belilovsky et al. (2020)) and allows parallel training of the modules. Module-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 module (so possibly one layer) at a time. Despite its simplicity, module-wise training has been shown to scale well (Belilovsky et al. (2020); Pyeon et al. (2021)), outperforming more complicated methods addressing the locking problems e.g. synthetic (Jaderberg et al. (2017); Czarnecki et al. (2017)) and delayed (Huo et al. (2018b;a)) gradients. We can also deduce theoretical results about networks of greedily-trained shallow modules from the existing results about shallow networks (Belilovsky et al. (2019; 2020)).

The typical setting of (sequential) module-wise training for minimizing a loss L, is, given a dataset $\mathcal{D}$, to solve one after the other, for $1 \leq k \leq K$, Problems

$$(T_k, F_k) \in \arg\min_{T,F} \sum_{x \in \mathcal{D}} L(F, T(G_{k-1}(x)) \tag{1}$$

where $G_k = T_k \circ ... \circ T_1$ for $1 \leq k \leq K$ and $G_0 = \mathtt{id}$. Here, T_k is the module (one or many layers) and it receives the output of module T_{k-1}, and F_k is an auxiliary classifier that processes the outputs of T_k so the loss can be computed. The features are x and L has access to their label to calculate the loss. See Figure 3 in Appendix E for a representation of this training. The final network trained this

way is $F_K \circ G_K$, but we can stop at any depth k and use $F_k \circ G_k$ if it performs better. In fact, when the modules are numerous and shallow, module-wise training suffers from a *stagnation problem*, whereby greedy early modules overfit and learn more discriminative features than end-to-end training (Marquez et al. (2018)), and deeper modules don't improve the test accuracy (Wang et al. (2021)), or even degrade it (see Figure 1). To tackle this issue, Wang et al. (2021) propose to maximize the mutual information that each module keeps with the input, in addition to minimizing the loss. We propose a different, transport-based perspective, leveraging the analogy between ResNets and the Euler scheme for ODEs (Weinan (2017)). To preserve input information, we minimize the kinetic energy of the modules along with the training loss. This has tight connections with the theories of gradient flows in distribution space and optimal transport. Our approach is particularly well-adapted to ResNets (He et al. (2016a;b)), which remain competitive (Wightman et al. (2021)) to this day, and similar models such as ResNeXts (Xie et al. (2017)) and Wide ResNets (Zagoruyko & Komodakis (2016)), but is easily usable on other architectures. Our contributions are the following:

- We propose a new method for module-wise training. Being a regularization, it is easier to implement and lighter than many recent methods (e.g. PredSim (Nøkland & Eidnes (2019)), InfoPro (Wang et al. (2021))) that train another auxiliary network besides the auxiliary classifier for each module.
- Theoretically, we show that our method amounts to a transport regularization, which forces the module to be an optimal transport map and makes it more regular and stable. We also show that it amounts to a discretization of the gradient flow of the loss in distribution space, which means that the modules successively minimize the loss and explains why the method avoids the accuracy collapse or stagnation observed in module-wise training.
- We propose *multi-lap sequential* training, a variant of sequential module-wise training, that for the same time and number of epochs, and the same memory usage, often performs better.
- Experimentally, we consistently improve the test accuracy of module-wise trained ResNets, both in sequential and parallel module-wise training, whereas most recent methods, with the exception of Belilovsky et al. (2019), focus on the parallel case. The method also works when the modules are few and large, and when they are numerous and shallow, when many methods focus on one setting or show problems in the other.

2 Related work

Layer-wise training has been considered as a pre-training and initialization method (Bengio et al. (2006); Marquez et al. (2018)) and was shown recently to be competitive with end-to-end training (Belilovsky et al. (2019); Nøkland & 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)). 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 & Choi (2019); Ma et al. (2020); Wang et al. (2021)) and biologically plausible losses (Sindy Löwe (2019); Nøkland & Eidnes (2019); Gupta (2020); Bernd Illing (2020); Yuwen Xiong (2020)). Paper Belilovsky et al. (2019) reports the best experimental results when solving the layer-wise problems sequentially. Methods PredSim (Nøkland & Eidnes (2019)), DGL (Belilovsky et al. (2020)), Sedona (Pyeon et al. (2021)) and InfoPro (Wang et al. (2021)) report the best results when solving the module-wise problems in parallel. Belilovsky et al. (2019; 2020) do it simply through architectural considerations regarding the auxiliary networks. However, Belilovsky et al. (2019) do not consider ResNets and PredSim state that their method does not perform well on them. Sedona applies architecture search to decide on where to split the network into up to 16 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 ResBlocks to one of up to 16 modules that are trained independently and reassembled before another random partition. More of 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 architecture from Belilovsky et al. (2019; 2020).

Besides module-wise training, methods such as DNI (Jaderberg et al. (2017); Czarnecki et al. (2017)), DDG (Huo et al. (2018b)) and FR (Huo et al. (2018a)), solve the update and backward locking problems with an eye towards parallelization by using delayed or predicted gradients, or even predicted inputs to address forward locking, which is what Sun et al. (2021) do. But they only split networks into a small number of sub-modules (less than five) that don't backpropagate to each other and observe training issues with more sub-modules (Huo et al. (2018a)). This makes them compare unfavorably to module-wise training (Belilovsky et al. (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 module-wise training is more appealing when working in a constrained setting.

Viewing ResNets as dynamic transport systems (de Bézenac et al. (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)). 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 & 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 et al. (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.

3 TRANSPORT-REGULARIZED MODULE-WISE TRAINING

In this section we state the module-wise optimization problems we solve. We show that successively solving these problems means following a minimizing movement scheme in distribution space that maximizes the separability of the embedding distributions. We then show that the solution modules exist and have some regularity as they are optimal transport maps. Appendices A and B give the necessary background on optimal transport, gradient flows and the minimizing movement scheme.

3.1 METHOD STATEMENT

To keep greedily-trained modules from overfitting and destroying information needed later, we penalize their kinetic energy to force them to preserve the geometry of the problem as much as possible. If each module is a single ResBlock (i.e. a function $T = \texttt{id}+r$), its kinetic energy is simply the squared norm of its residue $r = T-\texttt{id}$, which we add to the loss L in the target of the greedy problems (1). Given $\tau>0$ used to weight the regularization, we now solve, for $1\leq k\leq K$, Problems

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

where $G_k^\tau = T_k^\tau \circ .. \circ T_1^\tau$ for $1\leq k\leq K$ and $G_0^\tau = \texttt{id}$. 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. We focus on ResNets because they are already biased towards small displacements and that this bias is desirable and should be encouraged (Jastrzebski et al. (2018); Zhang et al. (2019); Hauser (2019); De & Smith (2020); Karkar et al. (2020)), and because $T(x)-x$ can always be computed as both terms have the same dimension. But the method can be applied to any module where this quantity can be computed.

To facilitate the theoretical analysis, we rewrite the method in a more general formulation using data distribution ρ, which can be discrete or continuous, and the distribution-wide loss $\mathcal{L}$ that arises from the point-wise loss L. Then Problem (2) is equivalent to Problem

$$(T_k^\tau, F_k^\tau) \in \arg\min_{T,F} \mathcal{L}(F, T_\sharp \rho_k^\tau) + \frac{1}{2\tau}\int_\Omega \|T(x) - x\|^2 \,\mathrm{d}\rho_k^\tau(x) \tag{3}$$

with $\rho_{k+1}^\tau = (T_k^\tau)_\sharp \rho_k^\tau$ and $\rho_1^\tau = \rho$. So data embedding distributions ρ_k^τ are pushed forward by maps T_k^τ.

3.2 LINK WITH THE MINIMIZING MOVEMENT SCHEME

We now show that solving Problems (3) successively means following a minimizing movement scheme in distribution space for minimizing $\mathcal{Z}(\mu) = \min_F \mathcal{L}(F, \mu)$, which represents the loss of the best classifier on distribution μ. If we restrict ourselves to linear classifiers, $\mathcal{Z}(\rho_k^\tau)$ represents the linear separability of our representation ρ_k^τ at module k of the input data distribution ρ. When auxiliary networks are not necessary, for example in generative tasks where the output has the same shape as the input, or when the auxiliary network F is fixed, $\mathcal{Z}$ reduces essentially to $\mathcal{L}$.

The distribution space we work in is the metric Wasserstein space $\mathbb{W}_2(\Omega) = (\mathcal{P}(\Omega), W_2)$, where $\Omega \subset \mathbb{R}^d$ is a convex compact set, $\mathcal{P}(\Omega)$ is the set of probability distributions over Ω and W_2 is the Wasserstein distance over $\mathcal{P}(\Omega)$ derived from the optimal transport problem with Euclidean cost:

$$W_2^2(\alpha, \beta) = \min_{T \text{ s.t. } T_\sharp \alpha = \beta} \int_\Omega \|T(x) - x\|^2 \, \mathrm{d}\alpha(x) \tag{4}$$

where we further assume that $\partial\Omega$ is negligible and that the distributions we are dealing with are absolutely continous. Given $\mathcal{Z} : \mathbb{W}_2(\Omega) \to \mathbb{R}$, the minimizing movement scheme is a discretized gradient flow that is well-defined in non-Euclidean metric spaces and minimizes (under some conditions) $\mathcal{Z}$ starting from $\rho_1^\tau \in \mathcal{P}(\Omega)$. It is given by

$$\rho_{k+1}^\tau \in \arg\min_{\rho \in \mathcal{P}(\Omega)} \; \mathcal{Z}(\rho) + \frac{1}{2\tau} W_2^2(\rho, \rho_k^\tau) \tag{5}$$

Equation (5) can be interpreted as a non-Euclidean version of the implicit Euler scheme for following the gradient flow of $\mathcal{Z}$, or as a Wasserstein proximal step to minimize $\mathcal{Z}$. Indeed, if $\mathcal{Z}$ is lower-semi continuous then problems (5) always admit a solution because $\mathcal{P}(\Omega)$ is compact. If $\mathcal{Z}$ is also λ-geodesically convex for $\lambda{>}0$, we have convergence of ρ_k^τ as $k{\to}\infty$ and $\tau{\to}0$ to a minimizer of $\mathcal{Z}$, potentially under more technical conditions (see Appendix B for details). Even though a machine learning loss will usually not satisfy these conditions, this analyses offers hints as to why our method avoids in practice the problem of stagnation or collapse in performance of module-wise training along the depth k, as we are making proximal local steps in Wasserstein space to minimize the loss. This convergence discussion also suggests taking τ as small as possible and many modules T_k^τ in practice.

So under the mentioned hypotheses on Ω and absolute continuity of the distributions, we have:

Proposition 3.1. *The distributions $\rho_{k+1}^\tau = (T_k^\tau)_\sharp \rho_k^\tau$, where the functions T_k^τ are found by solving (3) and $\rho_1^\tau = \rho$ is the data distribution, coincide with the minimizing movement scheme (5) for $\mathcal{Z} = \min_F \mathcal{L}(F, .)$.*

Proof. The minimizing movement scheme (5) is equivalent to taking $\rho_{k+1}^\tau = (T_k^\tau)_\sharp \rho_k^\tau$ where

$$T_k^\tau \in \arg\min_{T:\Omega\to\Omega} \mathcal{Z}(T_\sharp \rho_k^\tau) + \frac{1}{2\tau} W_2^2(T_\sharp \rho_k^\tau, \rho_k^\tau) \tag{6}$$

under conditions that guarantee the existence of a transport map between ρ_k^τ and any other measure, and absolute continuity of ρ_k^τ suffices, and the loss can ensure that ρ_{k+1}^τ is also absolutely continuous. Among the functions T_k^τ that solve problem (6), is the optimal transport map from ρ_k^τ to ρ_{k+1}^τ. To solve specifically for this optimal transport map, we have to solve the equivalent Problem

$$T_k^\tau \in \arg\min_T \mathcal{Z}(T_\sharp \rho_k^\tau) + \frac{1}{2\tau} \int_\Omega \|T(x) - x\|^2 \, \mathrm{d}\rho_k^\tau(x) \tag{7}$$

Problems (6) and (7) have the same minimum value, but the minimizer of (7) is now an optimal transport map between ρ_k^τ and ρ_{k+1}^τ. This is immediate from the definition (4) of the W_2 distance. Equivalently minimizing first over the auxiliary F in (3), and from the definition of $\mathcal{Z}$, Problems (3) and (7) are equivalent, which concludes. □

When solving over neural networks in practice, their representation power shown by universal approximation theorems is important here to get close to equivalence between (5) and (6) when restricting the optimization to neural networks.

3.3 REGULARITY RESULT

We can show that Problem (3) has a solution and that the solution module T_k^τ is an optimal transport map 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, that Ω is convex and compact and that $\partial\Omega$ is negligible.

Theorem 3.2. *Problem (3) has 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 Appendix C. OT maps have regularity properties under some boundedness assumptions. Given Theorem A.1 in Appendix A and 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 (2) 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 module $\tilde{T}_k^\tau$:

$$\|\tilde{T}_k^\tau(x) - \tilde{T}_k^\tau(y)\| \leq 2\epsilon + C\|x - y\|^\eta \tag{8}$$

for almost every $x, y \in \text{supp}(\rho_k^\tau)$ and a constant $C > 0$. Karkar et al. (2020) show that these networks generalize better and overfit less in practice. 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 $\tilde{G}_k^\tau = \tilde{T}_k^\tau \circ .. \circ \tilde{T}_1^\tau$.

4 PRACTICAL IMPLEMENTATION

4.1 MULTI-BLOCK MODULES

For simplicity, we have presented in (2) the case where each module is a single ResBlock. However, in practice, we often split the network into modules that are made-up of many ResBlocks each. We show here that regularizing the kinetic energy of such modules still amounts to a transport regularization, which means that Theorem 3.2, the regularity bound (8) and the link with gradient flows still apply.

If each module T_k is made up of M ResBlocks, i.e. applies $x_{m+1} = x_m + r_m(x_m)$ for $0 \leq m < M$, then its total discrete kinetic energy for a single data point x_0 is the sum of its squared residue norms $\sum \|r_m(x_m)\|^2$, since a ResNet can be seen as a discrete Euler scheme for an ordinary differential equation (Weinan (2017)) with velocity field r:

$$x_{m+1} = x_m + r_m(x_m) \iff \partial_t x_t = r_t(x_t) \tag{9}$$

and $\sum \|r_m(x_m)\|^2$ is then the discretization of the total kinetic energy $\int_0^1 \|r_t(x)\|^2 \,\mathrm{d}t$ of the ODE. If ψ_m^x denotes the position of a point x after m ResBlocks, then regularizing the kinetic energy of multi-block modules means solving

$$(T_k^\tau, F_k^\tau) \in \arg\min_{T,F} \sum_{x \in \tilde{\rho}_0} (L(F, T(G_{k-1}^\tau(x)) + \frac{1}{2\tau} \sum_{m=0}^{M-1} \|r_m(\psi_m^x)\|^2) \tag{10}$$
$$\text{s.t. } T = (\mathtt{id} + r_{M-1}) \circ ... \circ (\mathtt{id} + r_0),\ \psi_0^x = G_{k-1}^\tau(x),\ \psi_{m+1}^x = \psi_m^x + r_m(\psi_m^x)$$

We also want to minimize this sum of squared residue norms instead of $\|T(x) - x\|^2$ (the two no longer coincide) as it works better in practice, which we assume is because it offers a better and more localized control of the transport. As expressed in (9), a ResNet can be seen as an Euler discretization of a differential equation and this new Problem (10) is then the discretization of Problem

$$(T_k^\tau, F_k^\tau) \in \arg\min_{T,F}\ \mathcal{L}(F, T_\sharp \rho_k^\tau) + \frac{1}{2\tau} \int_0^1 \|v_t\|^2_{L^2((\phi_t^\cdot)_\sharp \rho_k^\tau)} \,\mathrm{d}t \tag{11}$$
$$\text{s.t. } T = \phi_1^\cdot,\ \partial_t \phi_t^x = v_t(\phi_t^x),\ \phi_0^\cdot = \mathrm{id}$$

where $\rho_{k+1}^\tau = (T_k^\tau)_\sharp \rho_k^\tau$ and r_m is the discretization of vector field v_t at time $t = m/M$. Here, distributions ρ_k^τ are pushed forward through the maps T_k^τ which correspond to the flow ϕ at time $t = 1$ of the kinetically-regularized velocity field v_t. We recognize in the second term in the target of (11) the optimal transport problem in its dynamic formulation (Benamou & Brenier (2000)), and given the equivalence between the Monge OT problem (4) and the dynamic OT problem (16) in Appendix A, Problem (11) is in fact equivalent to the original continuous formulation (3), and the theoretical results in Section 3 follow immediately (see also the proof in Appendix C).

4.2 SOLVING THE MODULE-WISE PROBLEMS

The module-wise problems 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 et al. (2020), which shows that combining it with batch 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 module-wise training regimes.

4.3 VARYING THE REGULARIZATION WEIGHT

The discussion in Section 3.2 suggests taking a fixed weight τ for the transport cost that is as small as possible. However, instead of using a fixed τ, we might want to 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 accuracy of earlier modules. We might also want to regularize the networks further in earlier epochs when the data is more entangled. 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 (2) 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 $W(\theta, x)$ 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 $i{\geq}1$:

$$\begin{cases} \theta_{k,i+1} &= \theta_{k,i} - \eta_i \nabla_\theta (\lambda_{k,i}\, L(\theta_{k,i}, x_i) + W(\theta_{k,i}, x_i)) \\ \lambda_{k,i+1} &= \lambda_{k,i} + hL(\theta_{k,i+1}, x_{i+1}) \text{ if } i \bmod s = 0 \text{ else } \lambda_{k,i} \end{cases} \tag{12}$$

The weights $\lambda_{k,i}$ will vary along modules 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. 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 et al. (2018); Belkin et al. (2018; 2019)). Therefore it is immediate by slightly adapting the proof of Theorem 3.2 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 many experiments, but has more hyperparameters to tune. In practice, we found that fixing a value of τ between 0.5 and 5 for the first half of the modules and twice as big for the second half is a simple heuristic that works well.

5 EXPERIMENTS

We test our method on classification tasks, L being the cross-entropy loss. We call vanilla greedy module-wise training without our regularization VanGL, which we include as a baseline in all tables for ablation study purposes. We call our method TRGL for Transport-Regularized Greedy Learning. For the auxiliary classifiers, we use the architecture from Belilovsky et al. (2019; 2020), that is a convolutional layer followed by an average pooling layer and a fully connected layer.

5.1 PARALLEL MODULE-WISE TRAINING WITH FEW MODULES

To compare with other methods, we focus here on parallel module-wise training with few modules, as it performs better than sequential training and is explored more in recent papers. The first experiment is training in parallel a ResNet-152 divided into 4 modules on TinyImageNet. We compare in Table 1 our results in this setup to three of the best recent parallel module-wise training methods: DGL (Belilovsky et al. (2020)), PredSim (Nøkland & Eidnes (2019)) and Sedona (Pyeon et al. (2021)) from Table 2 in Pyeon et al. (2021). The good performance of VanGL comes from the auxiliary architecture, and the regularization allows to be more accurate than Sedona without using their architecture search phase that splits the network into 4 uneven modules and chooses the auxiliary architecture. As in their paper, parallel module-wise training with 4 modules does better than end-to-end training on TinyImageNet. Module-wise training in this case consumes 20% less memory than E2E training.

Table 1: Test accuracy of parallel TRGL and VanGL with 4 modules, compared to methods DGL, PredSim and Sedona from Pyeon et al. (2021) that also split a ResNet-152 into 4 module-wise-parallel-trained modules, and E2E training, on TinyImageNet.

Parallel VanGL (ours)	Parallel TRGL (ours)	DGL	PredSim	Sedona	E2E
63.65	**64.35**	57.64	51.76	64.1	62.32

In Table 2, we compare our method using a ResNet-101 split into 2 modules trained in parallel to the two delayed gradient methods DDG and FR from Huo et al. (2018a) on CIFAR100 (Krizhevsky (2009)). Here again, parallel module-wise training is competitive with end-to-end training.

Table 2: Test accuracy of parallel TRGL and VanGL with 2 modules, compared to methods DDG and FR from Huo et al. (2018a) that also split a ResNet-101 into 2 module-wise-parallel-trained modules, and E2E training, on CIFAR100 (average and 95$\%$ confidence interval over 3 runs).

Parallel VanGL (ours)	Parallel TRGL (ours)	DDG	FR	E2E
$77.31 \pm .27$	**77.87** $\pm .44$	75.75	76.90	76.57 ± 1.02

To show that our method works well with all types of module-wise training when using 2 modules, we train the same ResNet-101 split in 2 modules on CIFAR100, this time sequentially and multi-lap sequentially. Results are in Table 3 below. We see that our idea of multi-lap sequential training adds one percentage point of accuracy to sequential training, and that the regularization further improves the accuracy by about half a percentage point.

.

Table 3: Test accuracy of sequential (Seq) and multi-lap sequential (MLS) TRGL and VanGL with 2 modules on CIFAR100 using ResNet-101 (average of 2 runs).

Seq VanGL	Seq TRGL	MLS VanGL	MLS TRGL
73.31	73.61	74.34	**74.78**

5.2 SEQUENTIAL FULL BLOCK-WISE TRAINING

We now focus on full block-wise training, meaning that each module is a single ResBlock, mostly sequentially. We propose here to use shallower and initially wider ResNets with a downsampling and 256 filters initially and a further downsampling and doubling of the number of filters at the midpoint, no matter the depth. In these ResNets, we use the ResBlock from He et al. (2016a) with two convolutional layers. If such a network is divided in K modules of M ResBlocks each, we call the network a $K-M$ ResNet. These wider shallower architectures are well-adapted to layer-wise

training as seen in Belilovsky et al. (2019). We check in Table 6 in Appendix D that this architecture works well with parallel module-wise training by comparing favorably on CIFAR10 (Krizhevsky (2009)) a 2-7 ResNet with DGL, InfoPro (Wang et al. (2021)) and DDG Huo et al. (2018b). The 2-7 ResNet has 45 millions parameters, which is about the same as the ResNet-110 divided in two used by the other methods, and performs better when trained in parallel.

We now train a 10-block ResNet block-wise on CIFAR100 (a 10-1 ResNet in our notations). We report even the small improvements in accuracy to show that our method works in all settings (parallel or sequential with many or few splits), which other methods don't do. 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 et al. (2018). For multi-lap sequential training, block k is trained for $10{+}2k$ epochs, and this is repeated for 5 laps. In block-wise training, the last block does not always perform the best and we report the accuracy of the best block. In Table 4, we see that MLS training improves the test accuracy of sequential training by around 0.8 percentage points when the training dataset is full, but works less well on small training sets. Of the two, the regularization mainly improves the test accuracy of MLS training. The improvement increases as the training set gets smaller and reaches 1 percentage point. That is also the case for parallel module-wise training, which already performs quite close to end-to-end training in the full data regime and much better in the small data regime. Combining the multi-lap trick and the regularization improves the performance of sequential training by 1.2 percentage points. These results are averaged over 10 runs and the confidence intervals are provided in Tables 7 and 8 in Appendix D.

Table 4: Average test accuracy of 10-1 ResNet over 10 runs on CIFAR100 with different train sizes and sequential (Seq), multi-lap sequential (MLS) and parallel (Par) TRGL and VanGL, and E2E.

	Seq		MLS		Par		E2E
Train size	VanGL	TRGL	VanGL	TRGL	VanGL	TRGL	
50000	68.74	**68.79**	69.48	**69.95**	72.59	**72.63**	75.85
25000	60.48	**60.59**	61.33	**61.71**	64.84	**65.01**	65.36
12500	51.64	**51.74**	51.30	**51.89**	55.13	**55.40**	52.39
5000	36.37	**36.40**	33.68	**34.61**	39.45	**40.36**	36.38

We report further results for full block-wise training on MNIST (LeCun et al. (2010)) and CIFAR10, but now we report the accuracy of the last block. We see again greater improvement due to the regularization as the training set gets smaller, gaining as much as 6 percentage points (Table 5, and Tables 9 and 10 in Appendix D). The 88% accuracy of sequential training on CIFAR10 in Table 5 is the same as for sequential training in table 2 of Belilovsky et al. (2019), which is the best method for layer-wise sequential training available, with VGG networks of comparable depth and width.

Table 5: Average last block test accuracy and 95% confidence interval of 10-1 ResNet over 10 runs on CIFAR10 with different train sizes and sequential (Seq) TRGL and VanGL, compared to E2E.

Train	Seq VanGL	Seq TRGL	E2E
50000	$88.02 \pm .18$	**88.20** $\pm .24$	$91.88 \pm .18$
25000	$83.95 \pm .13$	**84.28** $\pm .22$	$88.75 \pm .27$
10000	$76.00 \pm .39$	**77.18** $\pm .34$	$82.61 \pm .35$
5000	$67.74 \pm .49$	**69.67** $\pm .44$	$73.93 \pm .67$
1000	$45.67 \pm .88$	**51.34** $\pm .90$	$50.63 \pm .98$

Finally, 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 performance after the first block (block 0 being the encoder) that the regularization completely avoids. On the right, from experiments with parallel block-wise training from Table 8 on a train set of 5000 CIFAR100 images,

we see a steeper increase in test accuracy along the blocks with the regularization. We see the same pattern in Figure 2 in Appendix D from experiments with multi-lap sequential block-wise training from Table 7 on a train set of 5000 CIFAR100 images.

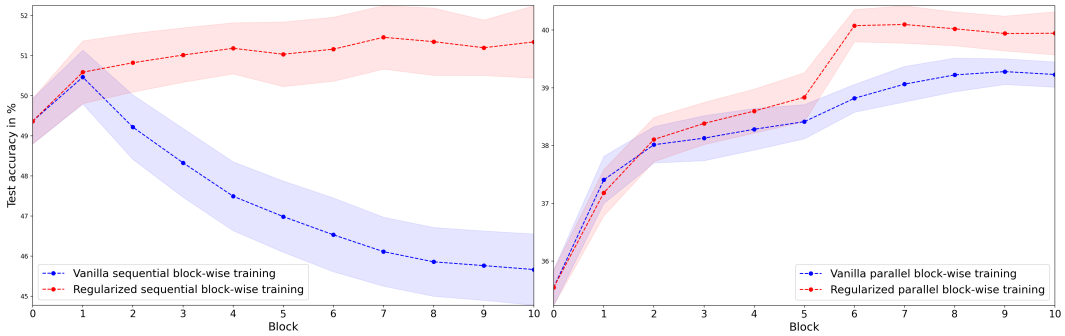

Figure 1: Highest test accuracy after each block of 10-1 ResNet models averaged over 10 runs with 95% confidence intervals. Left: sequential vanilla (VanGL, in blue) and regularized (TRGL, in red) block-wise training on 2% of the CIFAR10 training set. Right: parallel vanilla (VanGL, in blue) and regularized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

6 Conclusion

We introduced a transport regularization for module-wise training of ResNets that links module-wise training to gradient flows of the loss in distribution space. Our method provably leads to more regular modules and experimentally consistently improves the test accuracy of module-wise and block-wise sequential, parallel and multi-lap sequential (a variant of sequential training that we introduce) training, especially in small data regimes. Through this simple method that does not complexify the architecture or the implementation, we aim at making module-wise training competitive with end-to-end training while benefiting from its computational advantages: reduced memory usage and parallelism that is complementary to model and data parallelism in the case of parallel module-wise training, and training only one module at a time in constrained settings such as training on mobile devices in the case of sequential module-wise training. The method can easily be implemented for layer-wise training of non-residual networks and combined with other methods of layer-wise training. Future work can also 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 (8) that follow from each module T_k being an OT map.

Reproducibility Statement

The code is available at `https://github.com/block-wise/module-wise-training` and implementation details are in Appendix E.

References

David Alvarez-Melis and Nicolò Fusi. Dataset dynamics via gradient flows in probability space. *ICML*, 2021.

Luigi Ambrosio, Nicola Gigli, and Giuseppe Savare. *Gradient Flows in Metric Spaces and in the Space of Probability Measures*. Birkhäuser Basel, 2005.

Abdul Fatir Ansari, Ming Liang Ang, and Harold Soh. Refining deep generative models via discriminator gradient flow. In *ICLR*, 2021.

Michael Arbel, Anna Korba, Adil Salim, and Arthur Gretton. Maximum mean discrepancy gradient flow. In *NeurIPS*, 2019.

Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Greedy layerwise learning can scale to imagenet. In *ICML*, 2019.

Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Decoupled greedy learning of cnns. In *ICML*, 2020.

M. Belkin, S. Ma, and S. Mandal. To understand deep learning we need to understand kernel learning. In *ICML*, pp. 540–548, 2018.

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

J.D. Benamou and Y. Brenier. A computational fluid mechanics solution to the monge-kantorovich mass transfer problem. *Numerische Mathematik*, 2000.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *NeurIPS*, 2006.

Guillaume Bellec Bernd Illing, Wulfram Gerstner. Towards truly local gradients with clapp: Contrastive, local and predictive plasticity. *arXiv*, 2020.

R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud. Neural ordinary differential equations. In *NeurIPS*, 2018.

Wojciech Marian Czarnecki, Grzegorz Świrszcz, Max Jaderberg, Simon Osindero, Oriol Vinyals, and Koray Kavukcuoglu. Understanding synthetic gradients and decoupled neural interfaces. In *ICML*, 2017.

Soham De and Samuel L. Smith. Batch normalization biases residual blocks towards the identity function in deep networks. In *NeurIPS*, 2020.

Emmanuel de Bézenac, Ibrahim Ayed, and Patrick Gallinari. Optimal unsupervised domain translation. *arXiv*, 2019.

Chen Dun, Cameron R. Wolfe, Christopher M. Jermaine, and Anastasios Kyrillidis. Resist: Layer-wise decomposition of resnets for distributed training. *arXiv*, 2021.

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

Chris Finlay, Jorn-Henrik Jacobsen, Levon Nurbekyan, and Adam M Oberman. How to train your neural ode. In *ICML*, 2020.

Yuan Gao, Yuling Jiao, Yang Wang, Yao Wang, Can Yang, and Shunkang Zhang. Deep generative learning via variational gradient flow. In *ICML*, 2019.

Shashi Kant Gupta. A more biologically plausible local learning rule for anns. *arXiv*, 2020.

M. Hauser. On residual networks learning a perturbation from identity. *arXiv*, 2019.

K. He, X. Zhan, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*, 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016b.

Furong Huang, Jordan Ash, John Langford, and Robert Schapire. Learning deep resnet blocks sequentially using boosting theory. In *ICML*, 2018.

Zhouyuan Huo, Bin Gu, and Heng Huang. Training neural networks using features replay. In *NeurIPS*, 2018a.

Zhouyuan Huo, Bin Gu, Qian Yang, and Heng Huang. Decoupled parallel backpropagation with convergence guarantee. In *ICML*, 2018b.

Arthur Jacot, Clément Hongler, and Franck Gabriel. Neural tangent kernel: Convergence and generalization in neural networks. In *NeurIPS*, pp. 8580–8589, 2018.

M. Jaderberg, W. M. Czarnecki, S. Osindero, O. Vinyals, A. Graves, D. Silver, and K. Kavukcuoglu. Decoupled neural interfaces using synthetic gradients. In *ICML*, 2017.

Stanislaw Jastrzebski et al. Residual connections encourage iterative inference. In *ICLR*, 2018.

Skander Karkar, Ibrahim Ayed, Emmanuel de Bézenac, and Patrick Gallinari. A principle of least action for the training of neural networks. In *ECML-PKDD*, 2020.

Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto Technical Report*, 2009.

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

Antoine Liutkus, Umut Imşekli, Szymon Majewski, Alain Durmus, and Fabian-Robert Stoter. Sliced-wasserstein flows: Nonparametric generative modeling via optimal transport and diffusions. In *ICML*, 2019.

Yiping Lu, Aoxiao Zhong, Quanzheng Li, and Bin Dong. Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. In *ICML*, 2018.

Wan-Duo Kurt Ma, J.P. Lewis, and W. Bastiaan Kleijn. The hsic bottleneck: Deep learning without back-propagation. In *AAAI*, 2020.

Shirish Karande Mandar Kulkarni. Layer-wise training of deep networks using kernel similarity. In *DLPR workshop, ICPR*, 2016.

Enrique S. Marquez, Jonathon S. Hare, and Mahesan Niranjan. Deep cascade learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2018.

Thanh T. Nguyen and Jaesik Choi. Layer-wise learning of stochastic neural networks with information bottleneck. *Entropy*, 21, 2019.

Arild Nøkland and Lars Hiller Eidnes. Training neural networks with local error signals. In *ICML*, 2019.

Myeongjang Pyeon, Jihwan Moon, Taeyoung Hahn, and Gunhee Kim. Sedona: Search for decoupled neural networks toward greedy block-wise learning. In *ICLR*, 2021.

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

F. Santambrogio. Euclidean, metric, and wasserstein gradient flows: an overview. *arXiv*, 2016.

Andrew M. Saxe, James L. Mcclelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural network. In *ICLR*, 2014.

Bastiaan Veeling Sindy Löwe, Peter O'Connor. Putting an end to end-to-end: Gradient-isolated learning of representations. In *NeurIPS*, 2019.

Sebastian U. Stich. Local sgd converges fast and communicates little. In *ICLR*, 2019.

Qi Sun, Hexin Dong, Zewei Chen, Jiacheng Sun, Zhenguo Li, and Bin Dong. Layer-parallel training of residual networks with auxiliary-variable networks. *arXiv*, 2021.

Yin Tang, Qi Teng, Lei Zhang, Fuhong Min, and Jun He. Layer-wise training convolutional neural networks with smaller filters for human activity recognition using wearable sensors. *IEEE Sensors Journal*, 2021.

Qi Teng, Kun Wang, Lei Zhang, and Jun He. The layer-wise training convolutional neural networks using local loss for sensor-based human activity recognition. *IEEE Sensors Journal*, 2020.

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

Yulin Wang, Zanlin Ni, Shiji Song, and Gao Huang Le Yang. Revisiting locally supervised learning: an alternative to end-to-end training. In *ICLR*, 2021.

E Weinan. A proposal on machine learning via dynamical systems. *Commun. Math. Stat*, 2017.

Ross Wightman, Hugo Touvron, and Herve Jégou. Resnet strikes back: An improved training procedure in timm. *arXiv*, 2021.

Saining Xie et al. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.

Raquel Urtasun Yuwen Xiong, Mengye Ren. Loco: Local contrastive representation learning. In *NeurIPS*, 2020.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016.

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

Jingfeng Zhang et al. Towards robust resnet: A small step but a giant leap. In *IJCAI*, 2019.

A 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

$$W_2^2(\alpha, \beta) = \min_{\gamma \in \Pi(\alpha,\beta)} \int_{\Omega \times \Omega} \|x - y\|^2 \, \mathrm{d}\gamma(x, y) \tag{13}$$

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 (13) (called the Kantorovich problem) has a unique solution and is equivalent to the Monge problem, i.e.

$$W_2^2(\alpha, \beta) = \min_{T \text{ s.t. } T_\sharp \alpha = \beta} \int_{\Omega} \|T(x) - x\|^2 \, \mathrm{d}\alpha(x) \tag{14}$$

and this problem has a unique solution $T^\star$ linked to the solution $\gamma^\star$ of (13) 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 & 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 :

$$\begin{aligned} W_2^2(\alpha, \beta) = \min_v & \int_0^1 \|v_t\|^2_{L^2(\alpha_t)} \, \mathrm{d}t \\ & \text{s.t. } \partial_t \alpha_t + \nabla \cdot (\alpha_t v_t) = 0,\ \alpha_0 = \alpha, \alpha_1 = \beta \end{aligned} \tag{15}$$

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 (15) becomes

$$\begin{aligned} W_2^2(\alpha, \beta) = \min_v & \int_0^1 \|v_t\|^2_{L^2((\phi_t^\cdot)_\sharp \alpha)} \, \mathrm{d}t \\ & \text{s.t. } \partial_t \phi_t^x = v_t(\phi_t^x),\ \phi_0^\cdot = \mathrm{id}, (\phi_1^\cdot)_\sharp \alpha = \beta \end{aligned} \tag{16}$$

and the optimal transport map $T^\star$ that solves (14) is in fact $T^\star(x) = \phi_1^x$ for ϕ that solves the continuity equation together with the optimal $v^\star$ from (16). 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 A.1. *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

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

B BACKGROUND ON GRADIENT FLOWS

We follow Santambrogio (2016); Ambrosio et al. (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

$$\begin{cases} x'(t) = -\nabla \mathcal{L}(x(t)) \\ x(0) = x_0 \end{cases} \tag{17}$$

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*:

$$x_{k+1}^\tau \in \arg\min_{x \in \mathbb{R}^d} \ \mathcal{L}(x) + \frac{1}{2\tau}\|x - x_k^\tau\|^2 \tag{18}$$

$\mathcal{L}$ lower semi-continous and $\mathcal{L}(x) \geq C_1 - C_2\|x\|^2$ guarantees existence of a solution of (18) 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 et al. (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 (17) (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 (17) 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 (18) 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 (18) becomes

$$\rho_{k+1}^\tau \in \arg\min_{\rho \in \mathcal{P}(\Omega)} \ \mathcal{L}(\rho) + \frac{1}{2\tau} W_2^2(\rho, \rho_k^\tau) \tag{19}$$

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 et al. (2005).

C Proof of Theorem 3.2

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 (3) 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

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

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 (11). Problems (3) and (11) are equivalent by the equivalence between Problems (14) and (16). □

D ADDITIONAL EXPERIMENTS

Table 6: Average test accuracy and 95% confidence interval of 2-7 ResNet over 10 runs on CIFAR10 with parallel TRGL and VanGL, compared to DGL and DDG from Belilovsky et al. (2020) and InfoPro from Wang et al. (2021) that split a ResNet-110 in 2 module-wise-parallel-trained modules.

Parallel VanGL (ours)	Parallel TRGL (ours)	DGL	DDG	InfoPro
94.01 ± .17	**94.05** ± .18	93.50	93.41	93.58

Table 7: Average highest test accuracy and 95% confidence interval of 10-1 ResNet over 10 runs on CIFAR100 with different train sizes and sequential (Seq) and multi-lap sequential (MLS) TRGL and VanGL, compared to E2E.

Train size	Seq VanGL	Seq TRGL	MLS VanGL	MLS TRGL	E2E
50000	68.74 ± .45	**68.79** ± .56	69.48 ± .53	**69.95** ± .50	75.85 ± .70
25000	60.48 ± .15	**60.59** ± .14	61.33 ± .23	**61.71** ± .32	65.36 ± .31
12500	51.64 ± .33	**51.74** ± .26	51.30 ± .22	**51.89** ± .30	52.39 ± .97
5000	36.37 ± .33	**36.40** ± .40	33.68 ± .48	**34.61** ± .59	36.38 ± .31

Table 8: Average highest test accuracy and 95% confidence interval of 10-1 ResNet over 10 runs on CIFAR100 with different train sizes and parallel (Par) TRGL and VanGL, compared to E2E.

Train	Par VanGL	Par TRGL	E2E
50000	72.59 ± .40	**72.63** ± .40	75.85 ± .70
25000	64.84 ± .19	**65.01** ± .27	65.36 ± .31
12500	55.13 ± .24	**55.40** ± .35	52.39 ± .97
5000	39.45 ± .23	**40.36** ± .23	36.38 ± .31

Table 9: Average last block test accuracy and 95% confidence interval of 20-1 ResNet (32 filters, fixed encoder, same classifier) over 20/50 runs on MNIST with different train sizes and parallel (Par) TRGL and VanGL, compared to E2E.

Train	Par VanGL	Par TRGL	E2E
60000	99.07 ± .04	**99.08** ± .04	99.30 ± .03
30000	98.90 ± .05	**98.93** ± .06	99.22 ± .03
12000	98.52 ± .06	**98.59** ± .06	98.96 ± .06
6000	98.05 ± .09	**98.16** ± .07	98.62 ± .06
1500	96.34 ± .12	**96.91** ± .07	97.19 ± .08
1200	95.80 ± .12	**96.58** ± .09	96.88 ± .09
600	91.35 ± .99	**95.16** ± .15	95.30 ± .17
300	89.81 ± .73	**92.86** ± .24	92.87 ± .28
150	81.84 ± 1.22	**87.48** ± .42	87.82 ± .59

Table 10: Average last block test accuracy and 95% confidence interval of 20-1 ResNet (100 filters, fixed encoder, same classifier) over 10 runs on CIFAR10 with different train sizes and parallel (Par) TRGL and VanGL, compared to E2E.

Train	Par VanGL	Par TRGL	E2E
50000	85.98 ± .28	**86.02** ± .26	93.11 ± .19
25000	80.94 ± .25	**81.09** ± .32	89.10 ± .29
10000	72.49 ± .46	**73.01** ± .31	80.52 ± .46
5000	62.31 ± .54	**64.06** ± .57	69.44 ± .88
500	38.61 ± .47	**41.44** ± .44	40.40 ± .60

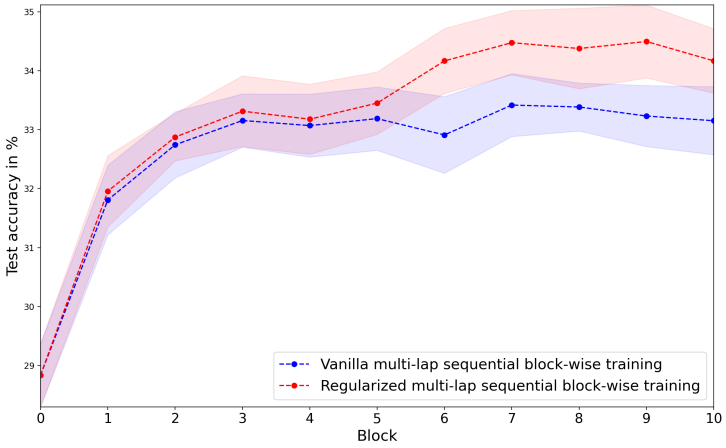

Figure 2: Highest test accuracy after each block of 10-1 ResNet averaged over 10 runs with 95% confidence intervals with multi-lap sequential vanilla (VanGL, in blue) and regularized (TRGL, in red) block-wise training on 10% of the CIFAR100 training set.

Figures 1 and 2 suggest that our regularization helps most when looking at the accuracy of the last block. We confirm this by including in Table 11 the accuracy achieved by the best block for the same experiment as in Table 5 and we notice a more important improvement from the regularization in the accuracy of the last block than in the accuracy of the best block. 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. We further confirm this through the following experiment. As the network gets deeper (50 blocks trained for 10 epochs each sequentially), we expect training it block-wise to become more difficult, and indeed the improvement from the regularization is slightly larger than usual when looking at the accuracy of the last block for both sequential training methods (Table 12). We also include in Table 13 results from block-wise training of ResNeXt-50-32×4d Xie et al. (2017), which turns out to be difficult to train block-wise.

Table 11: Average highest test accuracy and 95% confidence interval of 10-1 ResNet over 10 runs on CIFAR10 with different train sizes and sequential (Seq) TRGL and VanGL, compared to E2E.

Train	Seq VanGL	Seq TRGL	E2E
50000	88.14 ± .14	**88.34** ± .22	91.88 ± .18
25000	84.15 ± .17	**84.46** ± .22	88.75 ± .27
10000	76.62 ± .40	**77.47** ± .35	82.61 ± .35
5000	69.60 ± .43	**70.22** ± .50	73.93 ± .67
1000	51.59 ± .91	**52.06** ± .71	50.63 ± .98

Table 12: Average last block test accuracy and 95% confidence interval of 50-1 ResNet over 10 runs on CIFAR100 with sequential (Seq) and multi-lap sequential (MLS) TRGL and VanGL, compared to E2E.

Seq VanGL	Seq TRGL	MLS VanGL	MLS TRGL	E2E
63.40 ± .46	**63.86** ± .56	62.59 ± .64	**63.24** ± .50	63.34 ± 2.41

Table 13: Average test accuracy and 95% confidence interval of ResNeXt over 10 runs on CIFAR100 with sequential (Seq), multi-lap sequential (MLS) and parallel (Par) TRGL and VanGL. End-to-end training in this setting achieves an accuracy of 72.97 ± 1.18.

Seq VanGL	Seq TRGL	MLS VanGL	MLS TRGL	Par VanGL	Par TRGL
52.29 ± .53	**52.42** ± .65	52.59 ± .63	**52.84** ± .65	57.86 ± .49	**57.93** ± .51

E IMPLEMENTATION DETAILS

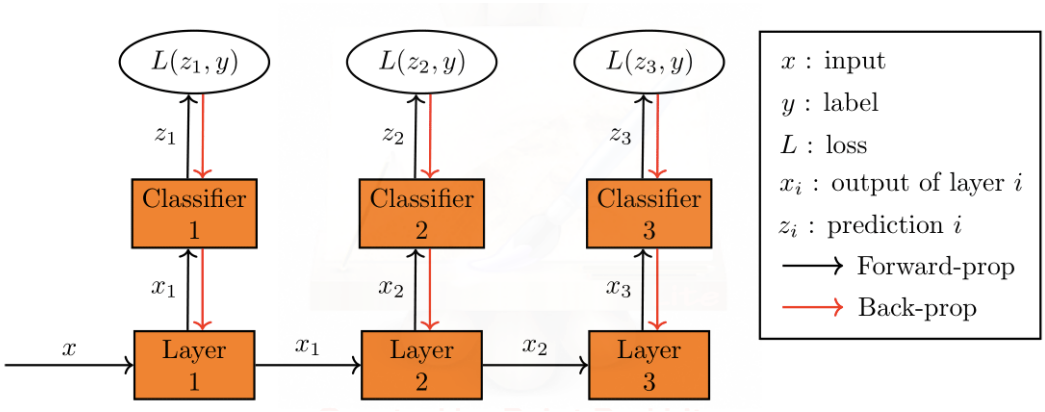

Figure 3: Module-wise training.

We use standard data augmentation and standard implementations for ResNet-101, ResNet-110 and ResNet-152 (the same as for the other methods in Section 5.1).

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 end-to-end training we use a learning rate of 0.1 that is divided by five at epochs 120, 160 and 200. Momentum is always 0.9. For parallel and end-to-end training, we train for 300 epochs. For sequential and multi-lap sequential training, the number of epochs varies per module (see Section 5.2).

For experiments in Section 5.1, we use a batch size of 256, orthogonal initialization (Saxe et al. (2014)) with a gain of 0.1, label smoothing of 0.1 and weight decay of 0.0002. For experiments in Section 5.2, we use a batch size of 128, orthogonal initialization with a gain of 0.05, no label smoothing and weight decay of 0.0001.

In Table 1, we use $\tau = 500000$ for the first two modules and then double it for the last two modules for TRGL. In Table 2, we use (12) with $\lambda_{k,1} = 1, h = 1$ and $s = 50$ for TRGL.