HAL open science

# Spherical Perspective on Learning with Normalization Layers

Simon Roburin, Yann de Mont-Marin, Andrei Bursuc, Renaud Marlet, Patrick Pérez, Mathieu Aubry

## To cite this version:

Simon Roburin, Yann de Mont-Marin, Andrei Bursuc, Renaud Marlet, Patrick Pérez, et al.. Spherical Perspective on Learning with Normalization Layers. Neurocomputing, 2022, 487, pp.66-74. 10.1016/j.neucom.2022.02.021 . hal-04496492

## HAL Id: hal-04496492

## https://hal.science/hal-04496492v1

Submitted on 8 Mar 2024

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

arXiv:2006.13382v3 [cs.LG] 19 May 2022

# Spherical Perspective on Learning with Normalization Layers

Simon Roburin[a,c,1,*], Yann de Mont-Marin[b,1], Andrei Bursuc[c], Renaud Marlet[a,c], Patrick Pérez[c], Mathieu Aubry[a]

[a]*LIGM, Ecole des Ponts, Univ Gustave Eiffel, CNRS, Marne-la-Vallée, France*
[b]*Département d'informatique de l'ENS, PSL, Inria, Paris, France*
[c]*valeo.ai, Paris, France*

**Abstract**

Normalization Layers (NLs) are widely used in modern deep-learning architectures. Despite their apparent simplicity, their effect on optimization is not yet fully understood. This paper introduces a spherical framework to study the optimization of neural networks with NLs from a geometric perspective. Concretely, the radial invariance of groups of parameters, such as filters for convolutional neural networks, allows to translate the optimization steps on the $L_2$ unit hypersphere. This formulation and the associated geometric interpretation shed new light on the training dynamics. Firstly, the first effective learning rate expression of Adam is derived. Then the demonstration that, in the presence of NLs, performing Stochastic Gradient Descent (SGD) alone is actually equivalent to a variant of Adam constrained to the unit hypersphere, stems from the framework. Finally, this analysis outlines phenomena that previous variants of Adam act on and their importance in the optimization process are experimentally validated.

*Keywords:* Optimization, Deep Learning, Normalization Layers, Batch Normalization

## 1. Introduction

The optimization process of deep neural networks is still poorly understood. Their training involves minimizing a high-dimensional non-convex function, which has been proved to be a NP-hard problem [1]. Yet, elementary gradient-based methods show good results in practice. To improve the quality of reached minima, numerous Normalization Layers (NLs) have stemmed in the last years and become common practices. One of the most prominent is Batch Normalization (BN) [2], which improves significantly both the the training speed and the prediction performance; it has however a notable shortcoming: BN relies heavily on the batch size. To avoid the dependency on the batch size, normalization layers such as LayerNorm (LN) [3], WeightNorm (WN) [4], InstanceNorm (IN) [5], or GroupNorm (GN) [6] were introduced. Yet, the interaction of NLs with optimization remains an open research topic.
Previous studies highlighted some of the mechanisms of the interaction between BN and Stochastic

*Corresponding author.
[1]Equal contribution.

Gradient Descent (SGD), both empirically [7] and theoretically [8, 9, 10]. But none of them provides a generic framework, nor studies the interaction between NLs and the Adam optimizer [11]. In this work, we provide an extensive analysis of the relation between NLs and any order-1 optimization scheme. Moreover, we theoretically relate SGD with NLs to a variant of Adam (AdaGradG), which is of interest as Adam probably is the most commonly-used adaptive scheme for Neural Networks (NNs). A shared effect of all mentioned NLs is to make NNs invariant to positive scalings of groups of parameters. These groups of parameters may differ from one NL method to another. The core idea of this paper is precisely to focus on these groups of radially-invariant parameters and analyze their optimization projected on the $L_2$ unit hypersphere (see Figure 1), which is topologically equivalent to the quotient manifold of the parameter space by the scaling action. In fact, one could directly optimize parameters on the hypersphere as [12]. Yet, most optimization methods are still performed successfully in the original parameter space. Here we propose to study an optimization scheme for a given group of radially-invariant parameters through its image scheme on the unit hypersphere. This geometric perspective sheds light on the interaction between normalization layers and Adam.

The paper is organized as follows. In **Section 2**, we introduce our spherical framework to study the optimization of any radially-invariant model. We also define a generic optimization scheme that encompasses methods such as SGD with momentum (SGD-M) and Adam. We then derive its image step on the unit hypersphere, leading to definitions and expressions of *effective learning rate* and *effective learning direction*. These new definitions are explicit and have a clear interpretation, whereas the definition of [13] is asymptotic and the definitions of [8] and of [10] are variational. In **Section 3**, we leverage the tools of our spherical framework to demonstrate that, in presence of NLs, SGD is equivalent to AdaGradG, a combination of AdaGrad [14] (a special case of Adam without momentum) and AdamG [12] (a variant of Adam constrained to the unit hypersphere). In other words, AdaGradG is a variant of Adam without momentum and constrained to the unit hypersphere. In **Section 4**, we analyze the effective learning direction for Adam. The spherical framework highlights phenomena that previous variants of Adam [15, 12] act on. We perform an empirical study of these phenomena and show that they play a significant role in the training of convolutional neural networks (CNNs). In **Section 5**, these results are put in perspective with related work.

Our main contributions are the following:

- A framework to analyze and compare order-1 optimization schemes of radially-invariant models;
- The first explicit expression of the effective learning rate for Adam;
- The demonstration that, in the presence of NLs, standard SGD is equivalent to AdaGradG, a variant of Adam without momentum and constrained to the unit hypersphere;
- The identification and the study of geometrical phenomena that occur with Adam and that impact significantly the training of CNNs with NLs.

## 2. Spherical framework and effective quantities

In this section, we provide background on radial invariance and introduce a generic optimization scheme that encompasses SGD, SGD with momentum (SGD-M) and Adam, with and without $L_2$-regularization.

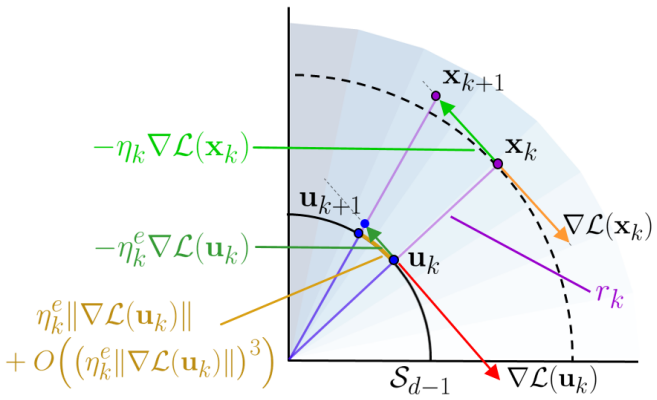



Figure 1: **Illustration of the spherical perspective for SGD.** The loss function $\mathcal{L}$ of a NN w.r.t. the parameters $\mathbf{x}_k \in \mathbb{R}^d$ of a neuron followed by a BN is radially invariant. The neuron update $\mathbf{x}_k \to \mathbf{x}_{k+1}$ in the original space, with velocity $\eta_k \nabla \mathcal{L}(\mathbf{x}_k)$, corresponds to an update $\mathbf{u}_k \to \mathbf{u}_{k+1}$ of its projection through an exponential map on the unit hypersphere $\mathcal{S}_{d-1}$ with velocity $\eta_k^e \|\nabla \mathcal{L}(\mathbf{u}_k)\|$ at order 2 (see details in Section 2.3).

Projecting the scheme update on the unit hypersphere leads to the formal definitions of effective learning rate and learning direction for any order-1 optimization scheme. This geometric perspective leads to the first explicit expression of the effective learning rate for Adam. The main notations are summarized in Figure 1.

### *2.1. Radial invariance*

We consider a family of parametric functions $\phi_{\mathbf{x}} \colon \mathbb{R}^{in} \to \mathbb{R}^{out}$ parameterized by a group of radially-invariant parameters $\mathbf{x} \in \mathbb{R}^d \setminus \{\mathbf{0}\}$, i.e., $\forall \rho > 0, \phi_{\rho \mathbf{x}} = \phi_{\mathbf{x}}$ (possible other parameters of $\phi_{\mathbf{x}}$ are omitted for clarity), a dataset $\mathcal{D} \subset \mathbb{R}^{in} \times \mathbb{R}^{out}$, a loss function $\ell : \mathbb{R}^{out} \times \mathbb{R}^{out} \to \mathbb{R}$ and a training loss function $\mathcal{L} \colon \mathbb{R}^d \to \mathbb{R}$ defined as:

$$\mathcal{L}(\mathbf{x}) \stackrel{\text{def}}{=} \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{s},\mathbf{t}) \in \mathcal{D}} \ell(\phi_{\mathbf{x}}(\mathbf{s}), \mathbf{t}). \tag{1}$$

It verifies: $\forall \rho > 0,\ \mathcal{L}(\rho \mathbf{x}) = \mathcal{L}(\mathbf{x})$. In the context of NNs, the group of radially-invariant parameters $\mathbf{x}$ can be the parameters of a single neuron in a linear layer or the parameters of a whole filter in a convolutional layer, followed by BN. See Appendix A for details, and Appendix B for the application to other normalization schemes such as WeightNorm [4], InstanceNorm [5], LayerNorm [3] or GroupNorm [6].

Please note that the loss is in fact evaluated on random batches of $\mathcal{D}$ at each optimization step. However, the study of the impact of splitting $\mathcal{D}$ into batches is out of the scope of this work. For notation purposes, we use $\mathcal{L}$ to actually denote the loss function evaluated on a random batch of the dataset.

The quotient of the parameter space by the equivalence relation associated to radial invariance is topologically equivalent to a sphere. We consider here the $L_2$ sphere $\mathcal{S}_{d-1} = \{\mathbf{u} \in \mathbb{R}^d / \|\mathbf{u}\|_2 = 1\}$ whose canonical metric corresponds to angles: $d_{\mathcal{S}}(\mathbf{u}_1, \mathbf{u}_2) = \arccos(\langle \mathbf{u}_1, \mathbf{u}_2 \rangle)$. This choice of metric is relevant to study NNs since filters in CNNs or neurons in MLPs are applied through scalar product to input data. Besides, normalization in NLs is also performed using the $L_2$ norm. Directly in WeightNorm, and indirectly in other NLs because the standard deviation can be interpreted as a centered $L_2$ norm.

Our framework relies on the decomposition of vectors into radial and tangential components. During optimization, we write the radially-invariant parameters at step $k \geq 0$ as $\mathbf{x}_k = r_k \mathbf{u}_k$ where $r_k = \|\mathbf{x}_k\|$ and $\mathbf{u}_k = \mathbf{x}_k / \|\mathbf{x}_k\|$. For any quantity $\mathbf{q}_k \in \mathbb{R}^d$ at step $k$, we write $\mathbf{q}_k^\perp = \mathbf{q}_k - \langle \mathbf{q}_k, \mathbf{u}_k \rangle \mathbf{u}_k$ its tangential component relatively to the current direction $\mathbf{u}_k$.

The following lemma states that the gradient of a radially-invariant loss function is tangential and $-1$ homogeneous:

**Lemma 1 (Gradient of a function with radial invariance).** *If $\mathcal{L} \colon \mathbb{R}^d \to \mathbb{R}$ is radially invariant and almost everywhere differentiable, then, for all $\rho > 0$ and all $\mathbf{x} \in \mathbb{R}^d$ where $\mathcal{L}$ is differentiable:*

$$\langle \nabla \mathcal{L}(\mathbf{x}), \mathbf{x} \rangle = 0 \quad \text{and} \quad \nabla \mathcal{L}(\mathbf{x}) = \rho \, \nabla \mathcal{L}(\rho \mathbf{x}). \tag{2}$$

### *2.2. Generic optimization scheme*

There is a large body of literature on optimization schemes [16, 14, 17, 11, 15]. We focus here on two of the most popular ones, namely SGD and Adam [11]. Yet, to establish general results that may apply to a variety of other schemes, we introduce here a *generic optimization update*:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \eta_k \mathbf{a}_k \oslash \mathbf{b}_k, \tag{3}$$

$$\mathbf{a}_k = \beta \mathbf{a}_{k-1} + \nabla \mathcal{L}(\mathbf{x}_k) + \lambda \mathbf{x}_k, \tag{4}$$

where $\mathbf{x}_k \in \mathbb{R}^d$ is the group of radially-invariant parameters at iteration $k$, $\mathcal{L}$ is the group's loss estimated on a batch of input data, $\mathbf{a}_k \in \mathbb{R}^d$ is a momentum, $\mathbf{b}_k \in \mathbb{R}^d$ is a division vector that can depend on the trajectory $(\mathbf{x}_i, \nabla \mathcal{L}(\mathbf{x}_i))_{i \in [\![0,k]\!]}$, $\eta_k \in \mathbb{R}$ is the scheduled trajectory-independent learning rate, $\oslash$ denotes the Hadamard element-wise division, $\beta$ is the momentum parameter, and $\lambda$ is the $L_2$-regularization parameter. We show how it encompasses several known optimization schemes.

*Stochastic gradient descent (SGD)* has proven to be an effective optimization method in deep learning. It can include $L_2$ regularization (also called weight decay) and momentum. Its updates are:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \eta_k \mathbf{m}_k, \tag{5}$$

$$\mathbf{m}_k = \beta \mathbf{m}_{k-1} + \nabla \mathcal{L}(\mathbf{x}_k) + \lambda \mathbf{x}_k, \tag{6}$$

where $\mathbf{m}_k$ is the momentum, $\beta$ is the momentum parameter, and $\lambda$ is the $L_2$-regularization parameter. It corresponds to our generic scheme (Eqs. 3-4) with $\mathbf{a}_k = \mathbf{m}_k$ and $\mathbf{b}_k = [1 \cdots 1]^\top$.

*Adam* is likely the most common adaptive scheme for NNs. Its updates are:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \eta_k \frac{\mathbf{m}_k}{1 - \beta_1^{k+1}} \oslash \sqrt{\frac{\mathbf{v}_k}{1 - \beta_2^{k+1}} + \epsilon}, \tag{7}$$

$$\mathbf{m}_k = \beta_1 \mathbf{m}_{k-1} + (1 - \beta_1)(\nabla \mathcal{L}(\mathbf{x}_k) + \lambda \mathbf{x}_k), \tag{8}$$

$$\mathbf{v}_k = \beta_2 \mathbf{v}_{k-1} + (1 - \beta_2)(\nabla \mathcal{L}(\mathbf{x}_k) + \lambda \mathbf{x}_k)^2, \tag{9}$$

where $\mathbf{m}_k$ is the momentum with parameter $\beta_1$, $\mathbf{v}_k$ is the second-order moment with parameter $\beta_2$, and $\epsilon$ prevents division by zero. (Here and in the following, the square and the square root of

a vector are to be understood as element-wise.) It corresponds to our generic scheme (Eqs. 3-4) with $\beta{=}\beta_1$ and:

$$\mathbf{a}_k = \frac{\mathbf{m}_k}{1-\beta_1}, \tag{10}$$

$$\mathbf{b}_k = \frac{1-\beta_1^{k+1}}{1-\beta_1}\sqrt{\frac{\mathbf{v}_k}{1-\beta_2^{k+1}}+\epsilon}. \tag{11}$$

*2.3. Image optimization on the hypersphere*

The radial invariance implies that the radial part of the parameter update $\mathbf{x}$ does not change the function $\phi_{\mathbf{x}}$ encoded by the model, nor does it change the loss $\mathcal{L}(\mathbf{x})$. Due to radial invariance, the parameter space projected on the unit hypersphere is topologically closer to the functional space of the network than the full parameter space. It hints that looking at optimization behaviour on the unit hypersphere might be interesting. To achieve this, we separate the quantities that can (tangential part) and cannot (radial part) change the model function. Theorem 2 formulates the spherical decomposition of our generic optimization scheme (Eqs. 3-4) in simple terms. It relates the update of radially-invariant parameters in the parameter space $\mathbb{R}^d$ and their update on $\mathcal{S}_{d-1}$ through an exponential map.

**Theorem 2 (Image step on $\mathcal{S}_{d-1}$).** *Let's consider the update of a group of radially-invariant parameters $\mathbf{x}_k$ at step $k$ following the generic optimization scheme (Eqs. 3-4) and the corresponding update of its projection $\mathbf{u}_k$ on $\mathcal{S}_{d-1}$. Under the hypothesis (H1) and (H2) formalized in Appendix C.1.1, the update of $\mathbf{u}_k$ is given by an exponential map at $\mathbf{u}_k$ with velocity $\eta_k^e \mathbf{c}_k^\perp$:*

$$\mathbf{u}_{k+1} = \mathrm{Exp}_{\mathbf{u}_k}\left(-\left[1+O\left(\|\eta_k^e\mathbf{c}_k^\perp\|^2\right)\right]\eta_k^e\mathbf{c}_k^\perp\right), \tag{12}$$

*where $\mathrm{Exp}_{\mathbf{u}_k}$ is the exponential map on $\mathcal{S}_{d-1}$, and with*

$$\mathbf{c}_k \stackrel{\text{def}}{=} r_k\mathbf{a}_k \oslash \frac{\mathbf{b}_k}{d^{-1/2}\|\mathbf{b}_k\|}, \tag{13}$$

$$\eta_k^e \stackrel{\text{def}}{=} \frac{\eta_k}{r_k^2 d^{-1/2}\|\mathbf{b}_k\|}\left(1-\frac{\eta_k\langle\mathbf{c}_k,\mathbf{u}_k\rangle}{r_k^2 d^{-1/2}\|\mathbf{b}_k\|}\right)^{-1}. \tag{14}$$

*More precisely:*

$$\mathbf{u}_{k+1} = \frac{\mathbf{u}_k - \eta_k^e\mathbf{c}_k^\perp}{\sqrt{1+(\eta_k^e\|\mathbf{c}_k^\perp\|)^2}}. \tag{15}$$

The proof is given in Appendix C.1.1 and the theorem is illustrated in the case of SGD in Figure 1. Note that for CNN training the hypothesis (H1) and (H2), empirically discussed in the appendix are typically verified. In particular, with typical values $1-\frac{\eta_k\langle\mathbf{c}_k,\mathbf{u}_k\rangle}{r_k^2 d^{-1/2}\|\mathbf{b}_k\|} > 0$ (H1) is true. The other hypothesis $\eta_k^e\|\mathbf{c}_k^\perp\| < \pi$ (H2) where steps are supposed shorter than $\pi$ is also true (see Appendix C.1.2).

Table 1: Effective learning rate and direction for optimization schemes with $\nu = rd^{-1/2}\|\mathbf{b}\|$ (we omit here the iteration index $k$).

| Scheme | $\eta^e$ | $\mathbf{c}^\perp$ |
|---|---|---|
| SGD | $\frac{\eta}{r^2}$ | $\nabla\mathcal{L}(\mathbf{u})$ |
| SGD + $L_2$ | $\frac{\eta}{r^2(1-\eta\lambda)}$ | $\nabla\mathcal{L}(\mathbf{u})$ |
| SGD-M | $\frac{\eta}{r^2}(1-\frac{\eta\langle\mathbf{c},\mathbf{u}\rangle}{r^2})^{-1}$ | $\mathbf{c}^\perp$ |
| Adam | $\frac{\eta}{r\nu}(1-\frac{\eta\langle\mathbf{c},\mathbf{u}\rangle}{r\nu})^{-1}$ | $\mathbf{c}^\perp$ |

### *2.4. Effective quantities*

In Theorem 2, the normalized parameters update in Eq. 12 can be read $\mathbf{u}_{k+1} \approx \mathrm{Exp}_{\mathbf{u}_k}\left(-\eta_k^e \mathbf{c}_k^\perp\right)$, where $\eta_k^e$ and $\mathbf{c}_k^\perp$ can then be respectively interpreted as the learning rate and the direction of an optimization step constrained to $\mathcal{S}_{d-1}$. Since $\mathbf{a}_k$ is the momentum and, with Lemma 1, the quantity $r_k\mathbf{a}_k$ in $\mathbf{c}_k$ can be seen as *a momentum on the hypersphere*. Due to the radial invariance, only the change of parameter on the unit hypersphere corresponds to a change of model function. Hence we can interpret $\eta_k^e$ and $\mathbf{c}_k^\perp$ as *effective learning rate* and *effective learning direction*. In other words, these quantities correspond to the learning rate and direction on the hypersphere that reproduce the function update of the optimization step.

Using Theorem 2, we can derive actual effective learning rates for any optimization scheme that fits our generic framework. These expressions, summarized in Table 1 are explicit and have a clear interpretation, in contrast to learning rates in [13], which are approximate and asymptotic, and in [18, 8], which are variational and restricted to SGD without momentum only.

In particular, we provide the first explicit expression of the effective learning rate for Adam:

$$\eta_k^e = \frac{\eta_k}{r_k\nu_k}\left(1 - \frac{\eta_k\langle\mathbf{c}_k,\mathbf{u}_k\rangle}{r_k\nu_k}\right)^{-1} \tag{16}$$

where $\nu_k = r_k d^{-1/2}\|\mathbf{b}_k\|$ is homogeneous to the norm of a gradient on the hypersphere and can be related to an *second-order moment on the hypersphere* (see Appendix C.1.3 for details). Using the variable $\nu$ also simplifies the in-depth analysis in Section 4, allowing a better interpretation of formulas.

The expression of the effective learning rate of Adam, i.e., the amplitude of the step taken on the hypersphere, reveals a dependence on the dimension $d$ (through $\nu$) of the update of the considered group of radially-invariant parameters. In the case of an MLP or CNN that stacks layers with neurons or filters of different dimensions, the learning rate is thus tuned differently from one layer to another.

We can also see that for all schemes the learning rate is tuned by the dynamics of radiuses $r_k$, which follow:

$$\frac{r_{k+1}}{r_k} = \left(1 - \frac{\eta_k\langle\mathbf{c}_k,\mathbf{u}_k\rangle}{r_k^2 d^{-1/2}\|\mathbf{b}_k\|}\right)\sqrt{1 + (\eta_k^e\|\mathbf{c}_k^\perp\|)^2}. \tag{17}$$

In contrast to previous studies [8, 13], this result demonstrates that for momentum methods, $\langle\mathbf{c}_k,\mathbf{u}_k\rangle$, which involves accumulated gradients terms in the momentum as well as $L_2$ regularization, tunes the learning rate.

## 3. SGD is equivalent to AdaGradG

In this section, we leverage the tools introduced in the spherical framework of Section 2 to find a scheme constrained to the hypersphere that is equivalent to SGD. We show that, for radially-invariant models, SGD is actually an adaptive optimization method. Formally, SGD is equivalent to a special case of AdamG [12] without momentum, where AdamG is a variant of Adam adapted and constrained to the unit hypersphere. Alternatively, we can also say that SGD is equivalent to a variant of AdaGrad adapted and constrained to the hypersphere, where AdaGrad is a special case of Adam without momentum. Besides, we illustrate this theoretical equivalence empirically.

### *3.1. Equivalence between two optimization schemes*

Due to the radial invariance, the functional space of the model is encoded by $\mathcal{S}_{d-1}$. In other words, two schemes with the same sequence of groups of radially-invariant parameters on the hypersphere $(\mathbf{u}_k)_{k\geq 0}$ encode the same sequence of model functions. We say that two optimization schemes $S$ and $\tilde{S}$ are equivalent if and only if $\forall k \geq 0, \mathbf{u}_k = \tilde{\mathbf{u}}_k$. Hence, starting from the same parameters, they reach the same optimum. By using Eq. 15, we obtain the following lemma, which is useful to prove the equivalence of two given optimization schemes:

**Lemma 3 (Sufficient condition for the equivalence of optimization schemes).**

$$\left\{ \begin{array}{l} \mathbf{u}_0 = \tilde{\mathbf{u}}_0 \\ \forall k \geq 0, \eta_k^e = \tilde{\eta}_k^e, \mathbf{c}_k^\perp = \tilde{\mathbf{c}}_k^\perp \end{array} \right. \Rightarrow \forall k \geq 0, \mathbf{u}_k = \tilde{\mathbf{u}}_k. \tag{18}$$

### *3.2. A hypersphere-constrained scheme equivalent to SGD*

We now study, within our spherical framework, SGD with $L_2$ regularization, i.e., the update $\mathbf{x}_{k+1} = \mathbf{x}_k - \eta_k(\nabla\mathcal{L}(\mathbf{x}_k) - \lambda_k\mathbf{x}_k)$. From the effective learning rate expression, we know that SGD yields an adaptive behaviour because it is scheduled by the radius dynamic, which depends on gradients. In fact, the tools in our framework allow us to find that SGD is equivalent to a variant of Adam constrained to the unit hypersphere, similar to AdamG [12], and without momentum, similar to AdaGrad. AdamG [12] uses the same updates as Adam (eq. 7-9) but project the weight on the hyper-sphere after each optimization step. More precisely, SGD is equivalent to AdamG with a null momentum factor $\beta_1 = 0$ (like AdaGrad), a non-null initial second-order moment $v_0$, an offset of the scalar second-order moment $k+1 \to k$ and without the bias correction term $1 - \beta_2^{k+1}$. Dubbed AdaGradG, this scheme reads:

$$(\text{AdaGradG}) : \left\{ \begin{array}{l} \hat{\mathbf{x}}_{k+1} = \mathbf{x}_k - \eta_k \frac{\nabla\mathcal{L}(\mathbf{x}_k)}{\sqrt{v_k}}, \\ \mathbf{x}_{k+1} = \frac{\hat{\mathbf{x}}_{k+1}}{\|\hat{\mathbf{x}}_{k+1}\|}, \\ v_{k+1} = \beta v_k + \|\nabla\mathcal{L}(\mathbf{x}_k)\|^2. \end{array} \right.$$

AdaGradG, like AdamG, is an adaptive method. Unlike Adam, which is adaptive with respect to the second-order moment for each parameter, AdaGradG and AdamG are adaptive for each group of radially-invariant parameters (e.g., filters for CNNs with BN or WN). In other words, each filter is adapted individually and independently by the optimization algorithm; it is not a global scheduling.

Now if we call « equivalent at order 2 in the step » a scheme equivalence that holds when we use for $r_k$ an expression that satisfies the radius dynamic with a Taylor expansion at order 2, then we have the following theorem:

**Theorem 4 (SGD equivalent scheme on the unit hypersphere).** *For any $\lambda \geq 0, \eta > 0, r_0 > 0$, we have the following equivalence when using the radius dynamic at order 2 in $(\eta_k \|\nabla \mathcal{L}(\mathbf{u}_k)\|)^2/r_k^2$:*

$$\begin{cases} \text{(SGD)} \\ \mathbf{x}_0 = r_0 \mathbf{u}_0 \\ \lambda_k = \lambda \\ \eta_k = \eta \end{cases} \text{ is scheme-equivalent at order 2 in step with } \begin{cases} \text{(AdaGradG)} \\ \mathbf{x}_0 = \mathbf{u}_0 \\ \beta = (1-\eta\lambda)^4 \\ \eta_k = (2\beta)^{-1/2} \\ v_0 = r_0^4 (2\eta^2 \beta^{1/2})^{-1}. \end{cases}$$

*Sketch of proof.*. Starting from SGD, we first use Lemma 3 to find a strict equivalence scheme with a simpler radius dynamic. We resolve this radius dynamic with a Taylor expansion at order 2 in $(\eta_k \|\nabla \mathcal{L}(\mathbf{u}_k)\|)^2/r_k^2$. A second use of Lemma 3 finally leads to the scheme equivalence in Theorem 4. The formal complete proof can be found in Appendix C.1.4. The fact that $r_k$ is well approximated at order 2 in practice is illustrated in Appendix C.1.5 and Figure C.8.

This result is unexpected because SGD, which is not adaptive by itself, is equivalent to a second order moment adaptive method. The scheduling performed by the radius dynamics actually replicates the effect of dividing the learning rate by the second-order moment of the gradient norm: $v_k$.

For standard values of hyperparameters $\lambda < 1$ (order of magnitude of $10^{-4}$) and $\eta < 1$ (order of magnitude at most $10^{-1}$), the higher-order terms of the radius in the Taylor expansion empirically become negligible in practice.

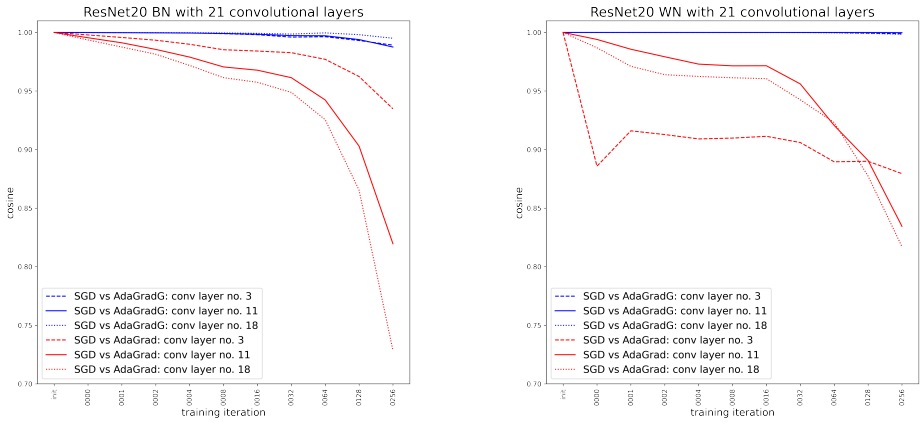
Second, with standard values of the hyper-parameters, namely learning rate $\eta < 1$ and $L_2$ regularization $\lambda < 1$, we have $\beta \leq 1$ which corresponds to a standard value for a moment factor. Interestingly, the $L_2$ regularization parameter $\lambda$ controls the memory of the past gradients' norm. If $\beta = 1$ (with $\lambda = 0$), there is no attenuation, each gradient norm has the same contribution in the order-2 moment. If $\lambda \neq 0$, there is a decay factor ($\beta < 1$) on past gradients' norm in the order-2 moment. This gives a new interpretation of the role of the $L_2$ regularization parameter $\lambda$ in SGD with NLs.

### *3.3. Empirical validation*

In order to illustrate the equivalence in Theorem 4, we experiment with learning an image classifier on CIFAR10 using different optimization schemes. We consider two architectures: ResNet20 with BN and ResNet20 with WN.

The set of parameters $\boldsymbol{\theta}$ of the above architectures can be split into two disjoint subsets: $\boldsymbol{\theta} = \mathcal{F} \cup \mathcal{R}$, where $\mathcal{F}$ is the set of groups of radially-invariant parameters and $\mathcal{R}$ is the set of remaining parameters. Note that the set of remaining parameters in $\mathcal{R}$ differs from one architecture to another: for ResNet20 BN, it includes the last linear layer as well as the scaling and bias of BN layers; for ResNet20 WN, it includes the magnitude parameters in each convolutional layer as well as the last linear layer.

For each architecture, we experiment with tracking the trajectory of parameters under different optimization schemes: SGD, AdaGradG and AdaGrad. As our analysis is restricted to radially-invariant parameters, we only track the trajectory of parameters belonging to $\mathcal{F}$, while the remaining parameters, belonging to $\mathcal{R}$, are always optimized in the same way, i.e., with SGD. For stability purposes, we finetune a previously trained architecture with SGD. The order of batches as well as the random seed for data augmentation are fixed to obtain comparable trajectories. The hyperparameters are chosen for SGD and AdaGrad so that gradient steps have the same order of




(a) ResNet20 BN

(b) ResNet20 WN

Figure 2: **Comparison of the trajectories of radially-invariant parameters using different optimization schemes**. For three randomly selected filters in each block of a ResNet20 architecture, with BN (left) or WN (right), we compute the cosine similarity between the parameter values obtained with SGD and the parameters values obtained respectively by AdaGradG and AdaGrad, at different iteration stages of a classification training on CIFAR10.

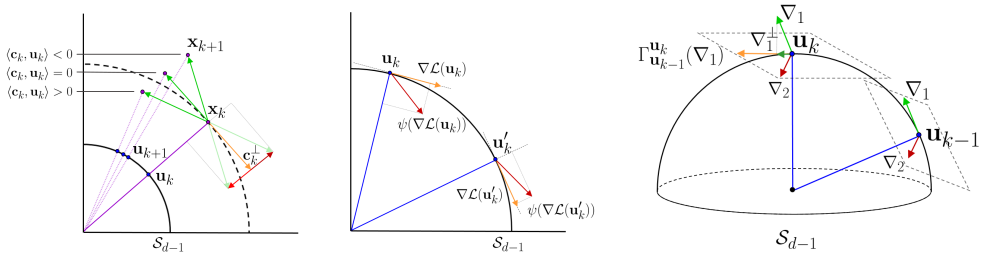
magnitude (see Appendix C.1.6 for details); the hyperparameters for AdaGradG are provided by the equivalence in Theorem 4.

In Figure 2, we show the angle between the training trajectories using SGD and AdaGradG (resp. SGD and AdaGrad), for three different filters in each block of the ResNet20 architectures. We observe that the trajectories on the $L_2$ unit hypersphere remain aligned for SGD and AdaGradG whereas, for SGD and AdaGrad, they quickly diverge. It empirically validates the equivalence mentioned in Theorem 4.

## 4. Geometric phenomena in Adam

Our framework with its geometrical interpretation reveals intriguing behaviors occurring in Adam. Indeed, since the unit hypersphere is enough to represent the functional space encoded by the network, from the perspective of manifold optimization, the optimization direction should only depend on the trajectory on that manifold. In the case of Adam, the effective direction not only depends on the trajectory on the hypersphere but also on the deformed gradients and additional radial terms. These terms are thus likely to play a role in Adam optimization.

In order to understand their role, we describe these geometrical phenomena in Section 4.1. Interestingly, previous variants of Adam, AdamW [15] and AdamG [12] are related to these phenomena. To study empirically their importance, we consider in Section 4.2 variants of Adam that first provide a direction intrinsic to the unit hypersphere, without deformation of the gradients, and then where radial terms are decoupled from the direction. The empirical study of these variants over a variety of datasets and architectures suggests that these behaviors do play a significant role in CNNs training with BN.




(a) Radial scheduling (b) Deformation anisotropy (c) Gradient-history contribution

Figure 3: (a) Effect of the radial part of $\mathbf{c}_k$ on the displacement on $\mathcal{S}_{d-1}$; (b) Example of anisotropy and sign instability for the deformation $\psi(\nabla\mathcal{L}(\mathbf{u}_k)) = \nabla\mathcal{L}(\mathbf{u}_k) \oslash \frac{|\nabla\mathcal{L}(\mathbf{u}_k)|}{d^{-1/2}\|\nabla\mathcal{L}(\mathbf{u}_k)\|}$ (where $|\cdot|$ is the element-wise absolute value) occurring in Adam's first optimization step; (c) Different contribution in $\mathbf{c}_k^\perp$ of two past gradients $\nabla_1$ and $\nabla_2$ of equal norm, depending on their orientation. Illustration of the transport of $\nabla_1$ from $\mathbf{u}_{k-1}$ to $\mathbf{u}_k$ : $\Gamma^{\mathbf{u}_k}_{\mathbf{u}_{k-1}}(\nabla_1)$ (*cf.* Appendix D.2 for details).

### *4.1. Identification of geometrical phenomena in Adam*

Here, we perform an in-depth analysis of the effective learning direction of Adam.

**(a) Deformed gradients.** Considering the quantities defined for a generic scheme in Eq. 14, $\mathbf{b}_k$ has a deformation effect on $\mathbf{a}_k$, due to the Hadamard division by $\frac{\mathbf{b}_k}{d^{-1/2}\|\mathbf{b}_k\|}$, and a scheduling effect $d^{-1/2}\|\mathbf{b}_k\|$ on the effective learning rate. In the case where the momentum factor is null $\beta_1 = 0$, the direction of the update at step $k$ is $\nabla\mathcal{L}(\mathbf{u}_k) \oslash \frac{\mathbf{b}_k}{d^{-1/2}\|\mathbf{b}_k\|}$ (Eq. 14) and the deformation $\frac{\mathbf{b}_k}{d^{-1/2}\|\mathbf{b}_k\|}$ may push the direction of the update outside the tangent space of $\mathcal{S}_{d-1}$ at $\mathbf{u}_k$, whereas the gradient itself lies in the tangent space. This deformation is in fact not isotropic: the displacement of the gradient from the tangent space depends on the position of $\mathbf{u}_k$ on the sphere. We illustrate this anisotropy in Figure 3(b).

**(b) Additional radial terms.** In the momentum on the sphere $\mathbf{c}_k$, quantities that are radial (resp. orthogonal) at a point on the sphere may not be radial (resp. orthogonal) at another point. To clarify the contribution of $\mathbf{c}_k$ in the effective learning direction $\mathbf{c}_k^\perp$, we perform the following decomposition (*cf.* Appendix D.1):

$$\mathbf{c}_k = (\mathbf{c}_k^{\text{grad}} + \lambda r_k^2 \mathbf{c}_k^{L_2}) \oslash \frac{\mathbf{b}_k}{d^{-1/2}\|\mathbf{b}_k\|} \quad \text{with:} \tag{19}$$

$$\mathbf{c}_k^{\text{grad}} \stackrel{\text{def}}{=} \nabla\mathcal{L}(\mathbf{u}_k) + \sum_{i=0}^{k-1} \beta^{k-i} \frac{r_k}{r_i} \nabla\mathcal{L}(\mathbf{u}_i) \tag{20}$$

$$\mathbf{c}_k^{L_2} \stackrel{\text{def}}{=} \mathbf{u}_k + \sum_{i=0}^{k-1} \beta^{k-i} \frac{r_i}{r_k} \mathbf{u}_i. \tag{21}$$

**b1. Contribution of $\mathbf{c}_k^{\text{grad}}$.** At step $k$, the contribution of each past gradient corresponds to the orthogonal part $\nabla\mathcal{L}(\mathbf{u}_i) - \langle \nabla\mathcal{L}(\mathbf{u}_i), \mathbf{u}_k \rangle \mathbf{u}_k$. It impacts the effective learning direction depending on its orientation relatively to $\mathbf{u}_k$. Two past points, although equally distant from $\mathbf{u}_k$ on the sphere and with equal gradient amplitude may thus contribute differently in $\mathbf{c}_k^\perp$ due to their orientation (*cf.* Figure 3(c)).

**b2. Contribution of $\mathbf{c}_k^{L_2}$.** Naturally, the current point $\mathbf{u}_k$ does not contribute to the effective

learning direction $\mathbf{c}_k^\perp$, unlike the history of points in $\sum_{i=0}^{k-1} \beta^{k-i} \frac{r_i}{r_k} \mathbf{u}_i$, which does. This dependency can be avoided if we decouple the $L_2$ regularization, in which case we do not accumulate $L_2$ terms in the momentum. This shows that the decoupling proposed in AdamW [15] actually removes the contribution of $L_2$ regularization in the effective learning direction.

**(c) The radius ratio** $\frac{r_k}{r_i}$ present in both $\mathbf{c}_k^{\text{grad}}$ and $\mathbf{c}_k^{L_2}$ (in inverse proportion) impacts the effective learning direction $\mathbf{c}_k^\perp$: it can differ for identical sequences $(\mathbf{u}_i)_{i \leq k}$ on the sphere but with distinct radius histories $(r_i)_{i \leq k}$. Since the radius is closely related to the effective learning rate, it means that the effective learning direction $\mathbf{c}_k^\perp$ is adjusted according to the learning rates history.

Note that AdamG [12], by constraining the optimization to the unit hypersphere and thus removing $L_2$ regularization, neutralizes all the above phenomena. However, this method has no scheduling effect allowed by the radius dynamics (*cf.* Eq.17) since it is kept constant during training.

### *4.2. Empirical study*

To study empirically the importance of the identified geometric phenomena, we perform an ablation study: we compare the performance (accuracy and training loss speed) of Adam and variants that neutralize each of them. We recall that AdamW neutralizes **(b2)** and that AdamG neutralizes all of above phenomena but loses the scheduling effect identified in Eq. 17. To complete our analysis, we use geometrical tools to design variations of Adam which neutralizes sequentially each phenomenon while preserving the natural scheduling effect in Theorem 2. We neutralize **(a)** by replacing the element-wise second-order moment, **(b1)** and **(b2)** by transporting the momentum from a current point to the new one, **(c)** by re-scaling the momentum at step $k$. The details are in Appendix D.2. The final scheme reads:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \eta_k \frac{\mathbf{m}_k}{1-\beta_1^{k+1}} \Big/ \sqrt{\frac{v_k}{1-\beta_2^{k+1}} + \epsilon}, \tag{22}$$

$$\mathbf{m}_k = \beta_1 \frac{r_{k-1}}{r_k} \Gamma_{\mathbf{u}_{k-1}}^{\mathbf{u}_k}(\mathbf{m}_{k-1}) + (1-\beta_1)(\nabla \mathcal{L}(\mathbf{x}_k) + \lambda \mathbf{x}_k), \tag{23}$$

$$v_k = \beta_2 \frac{r_{k-1}^2}{r_k^2} v_{k-1} + (1-\beta_2) d^{-1} \|\nabla \mathcal{L}(\mathbf{x}_k) + \lambda \mathbf{x}_k\|^2, \tag{24}$$

where $\Gamma_{\mathbf{u}_{k-1}}^{\mathbf{u}_k}$ is the hypersphere canonical parallel transport from $\mathbf{u}_{k-1}$ to $\mathbf{u}_k$. Implementation details are in Appendix D.3.

**Protocol.** For evaluation, we conduct experiments on two architectures: VGG16 [19] and ResNet [20] – more precisely ResNet20, a simple variant designed for small images [20], and ResNet18, a popular variant for image classification. We consider three datasets: SVHN [21], CIFAR10 and CIFAR100 [22].

Since our goal is to evaluate the significance of phenomena on radially-invariant parameters, i.e., the convolution filters followed by BN, we only apply variants of Adam including AdamG and AdamW on convolution layers. For comparison consistency, we keep standard Adam on the remaining parameters, and we use a fixed grid hyperparameter search budget and frequency for each method and each architecture (see Appendix D.3 for details).

Besides, please also remember that the impact of the scaling and bias parameters (which belongs to the remaining non radially-invariant parameters) is out of the scope of this study. Nevertheless, we additionally evaluate the variants of Adam on CNNs with BN without scaling

Table 2: **Accuracy of Adam and its variants when training with BN layers**.
The figures in this table are the mean top1 accuracy $\pm$ the standard deviation over 5 seeds on the test set for CIFAR10, CIFAR100 and on the validation set for SVHN. $^\dagger$ indicates that the original method is only used on convolutional filters while Adam is used for other parameters.

| | CIFAR10 | | | CIFAR100 | | SVHN | |
|---|---|---|---|---|---|---|---|
| Method | ResNet20 | ResNet18 | VGG16 | ResNet18 | VGG16 | ResNet18 | VGG16 |
| Adam | $90.98 \pm 0.06$ | $93.77 \pm 0.20$ | $92.83 \pm 0.17$ | $71.30 \pm 0.36$ | $68.43 \pm 0.16$ | $95.32 \pm 0.23$ | $95.57 \pm 0.20$ |
| AdamW$^\dagger$ | $90.19 \pm 0.24$ | $93.61 \pm 0.12$ | $92.53 \pm 0.25$ | $67.39 \pm 0.27$ | $71.37 \pm 0.22$ | $95.13 \pm 0.15$ | $94.97 \pm 0.08$ |
| AdamG$^\dagger$ | $91.64 \pm 0.17$ | $94.67 \pm 0.12$ | $93.41 \pm 0.17$ | $73.76 \pm 0.34$ | $70.17 \pm 0.20$ | $95.73 \pm 0.05$ | $95.70 \pm 0.25$ |
| Adam w/o (a) | $91.15 \pm 0.11$ | $93.95 \pm 0.23$ | $92.92 \pm 0.11$ | $74.44 \pm 0.22$ | $68.73 \pm 0.27$ | $95.75 \pm 0.09$ | $95.66 \pm 0.09$ |
| Adam w/o (ab) | $\mathbf{91.92} \pm 0.18$ | $\mathbf{95.11} \pm 0.10$ | $\mathbf{93.89} \pm 0.09$ | $\mathbf{76.15} \pm 0.25$ | $\mathbf{71.53} \pm 0.19$ | $\mathbf{96.05} \pm 0.12$ | $\mathbf{96.22} \pm 0.09$ |
| Adam w/o (abc) | $91.81 \pm 0.20$ | $94.92 \pm 0.05$ | $93.75 \pm 0.06$ | $75.28 \pm 0.35$ | $71.45 \pm 0.13$ | $95.84 \pm 0.07$ | $95.82 \pm 0.05$ |

and bias parameters (BN w/o affine) in Appendix D.4.2 and observe marginal performance difference in comparison to CNNs with standard BN. It hints that the normalization layer in BN plays the most important role regarding the model performance.

**Results.** In Table 2, we report quantitative results of Adam variants across architectures and datasets. To indicate that AdamW and AdamG are actually only used on convolutional filters, while Adam is used for the other parameters, we denote the experimented methods as AdamW$^\dagger$ and AdamG$^\dagger$. In addition, we compare the evolution of the training loss in Figure 4 and the top1 accuracy on the validation set in Figure 5.

We observe that each phenomenon displays a specific trade-off between generalization (accuracy on the test set) and training speed, as following. Neutralizing **(a)** has little effect on the speed over Adam, yet achieves better accuracy on the train set. Although it slows down training, neutralizing **(ab)** leads to minima with the overall best accuracy on test set in the case of BN equipped CNNS. Note that AdamW$^\dagger$ neutralizes **(b2)** with its decoupling and is the fastest method, but finds minima with overall worst generalization properties. By constraining the optimization to the hypersphere, AdamG$^\dagger$ speeds up training over the other variants. Finally, neutralizing **(c)** with Adam w/o **(abc)** brings a slight acceleration, though reaches lower accuracy than Adam w/o **(ab)**. In terms of generalization, before the first decrease of the learning rate, neutralizing **(a)** displays the same speed in terms of reached accuracy on the valid set compared to Adam. When neutralizing **(ab)** and **(abc)**, we observe a slight increase, comparable with AdamG†. After the first decrease of the learning rate, we observe the same hierarchy as in Table 2.

These results show that the geometrical phenomena revealed by our analysis in the spherical framework have a significant impact on the training of BN-equipped CNNs.

## 5. Related work

**Understanding Batch Normalization.** Albeit conceptually simple, BN has been shown to have complex implications over optimization. The argument of Internal Covariate Shift reduction [2] has been challenged and shown to be secondary to smoothing of optimization landscape [7, 23] or its modification by creating a different objective function [24], enabling of high learning rates through improved conditioning [9], or alleviating the sharpness of the Fisher information matrix[25]. In an analysis of a variant of GD, Kohler et al. [26] show that BN accelerates optimization by decoupling the optimization of the direction and length of parameters. Daneshmand

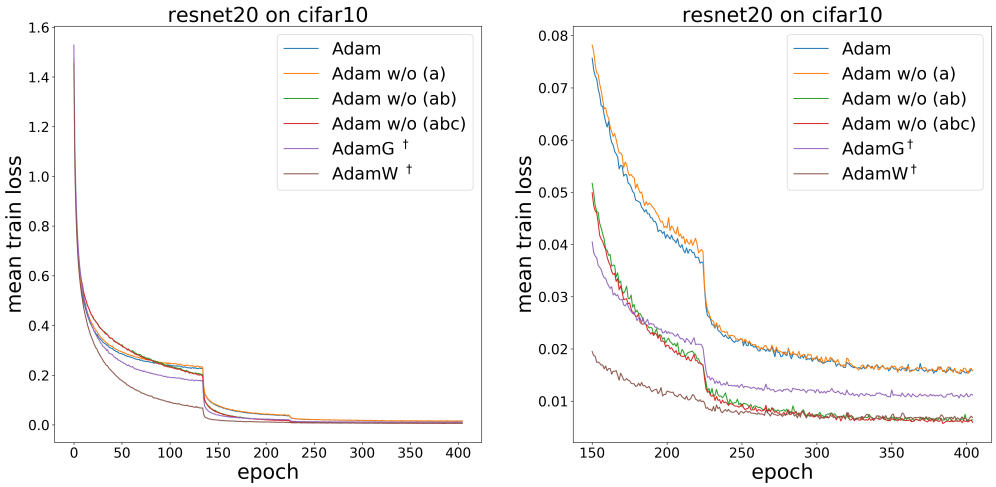



Figure 4: **Training speed comparison with ResNet20 BN on CIFAR10.** *Left:* Mean training loss over all training epochs (averaged across 5 seeds) for different Adam variants. *Right:* Zoom-in on the last epochs. Please refer to Table 2 for the corresponding accuracies.

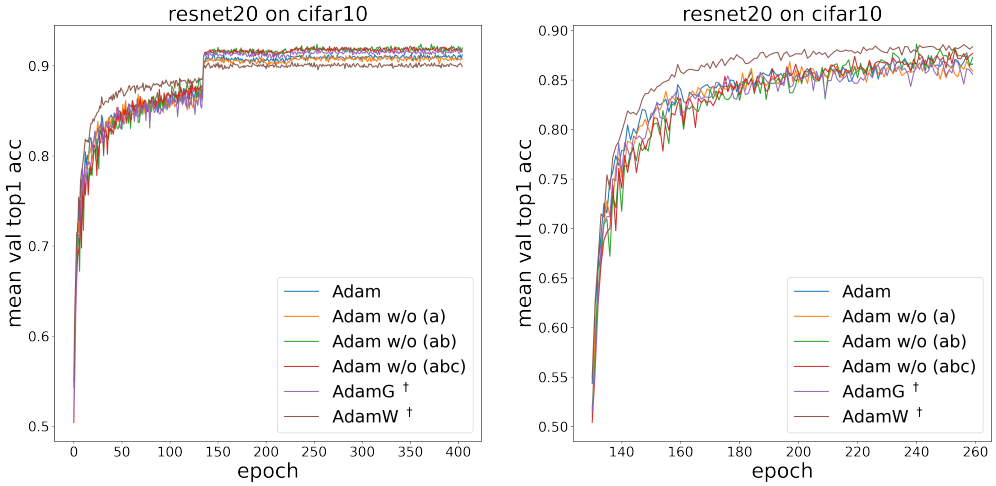



Figure 5: **Valid accuracy comparison with ResNet20 BN on CIFAR10.** *Left:* Mean valid top1 acc over all training epochs (averaged across 5 seeds) for different Adam variants. *Right:* Zoom-in on the first epochs. Please refer to Table 2 for the corresponding accuracies.

et al. [27] argue that BN is an effective strategy to ensure that activations generated by a randomly-initialized network have high rank, unlike vanilla networks where the rank in the final layers collapses with depth [28]. Arora et al. [8] demonstrate that (S)GD with BN is robust to the choice of the learning rate, with guaranteed asymptotic convergence, while a similar finding for GD with

BN is made in [29].

**Invariances in neural networks.** Cho and Lee [12] propose optimizing over the Grassmann manifold using Riemannian GD. Li and Arora [30] project weights and activations on the unit hypersphere and compute a function of the angle between them instead of inner products, and subsequently generalize these operators by scaling the angle [31]. In [32] the radial invariance is leveraged to prove that weight decay (WD) can be replaced by an exponential learning-rate scheduling for SGD with or without momentum. Arora et al. [8] investigate the radial invariance and show that radius dynamics depends on the past gradients, offering an adaptive behavior to the learning rate. Here we go further and show that SGD projected on the unit hypersphere corresponds to Adam constrained to the hypersphere, and we give an accurate definition of this adaptive behavior.

**Effective learning rate.** Due to its scale invariance, BN can adaptively adjust the learning rate [13, 12, 8, 32]. Van Laarhoven [13] shows that in BN-equipped networks, WD increases the effective learning rate by reducing the norm of the weights. Conversely, without WD, the norm grows unbounded [33], decreasing the effective learning rate. Zhang et al. [34] bring additional evidence supporting hypothesis in [13], while Hoffer et al. [18] find an exact formulation of the effective learning rate for SGD in normalized networks. In contrast with prior work, we find generic definitions of the effective learning rate with exact expressions for SGD and Adam.

## 6. Conclusion

The spherical framework introduced in this study provides a powerful tool to analyse Adam optimization scheme through its projection on the $L_2$ unit hypersphere. It allows us to give a precise definition and expression of the effective learning rate for Adam, to relate SGD to a variant of Adam, and to identify geometric phenomena which empirically impact training. The framework also brings light to existing variations of Adam, such as $L_2$-regularization decoupling. The geometry of invariance properties appears as a promising research direction toward a better understanding of their impact on optimization.

## References


[1] A. Blum, R. L. Rivest, Training a 3-node neural network is np-complete, in: Advances in Neural Information Processing Systems (NeurIPS), 1989.

[2] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: 32nd International Conference on Machine Learning (ICML), 2015.

[3] J. L. Ba, J. R. Kiros, G. E. Hinton, Layer normalization, arXiv preprint arXiv:1607.06450.

[4] T. Salimans, D. P. Kingma, Weight normalization: A simple reparameterization to accelerate training of deep neural networks, in: Advances in neural information processing systems (NeurIPS), 2016.

[5] D. Ulyanov, A. Vedaldi, V. Lempitsky, Instance normalization: The missing ingredient for fast stylization, arXiv preprint arXiv:1607.08022.

[6] Y. Wu, K. He, Group normalization, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018.

[7] S. Santurkar, D. Tsipras, A. Ilyas, A. Madry, How does batch normalization help optimization?, in: Advances in Neural Information Processing Systems (NeurIPS), 2018.

[8] S. Arora, Z. Li, K. Lyu, Theoretical analysis of auto rate-tuning by batch normalization, in: International Conference on Learning Representations (ICLR), 2019.

[9] N. Bjorck, C. P. Gomes, B. Selman, K. Q. Weinberger, Understanding batch normalization, in: Advances in Neural Information Processing Systems (NeurIPS), 2018.

[10] E. Hoffer, I. Hubara, D. Soudry, Fix your classifier: the marginal value of training the last weight layer, in: International Conference on Learning Representations (ICLR), 2018.

[11] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: International Conference on Learning Representations (ICLR), 2015.
[12] M. Cho, J. Lee, Riemannian approach to batch normalization, in: Advances in Neural Information Processing Systems (NeurIPS), 2017.
[13] T. van Laarhoven, L2 regularization versus batch and weight normalization, arXiv preprint arXiv:1706.05350 (2017).
[14] J. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, Journal of Machine Learning Research (JMLR).
[15] I. Loshchilov, F. Hutter, Decoupled weight decay regularization, in: International Conference on Learning Representations (ICLR), 2019.
[16] I. Sutskever, J. Martens, G. Dahl, G. Hinton, On the importance of initialization and momentum in deep learning, in: 30th International Conference on Machine Learning (ICML), Atlanta, Georgia, USA, 2013.
[17] T. Tieleman, G. Hinton, Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude, COURSERA: Neural networks for machine learning.
[18] E. Hoffer, R. Banner, I. Golan, D. Soudry, Norm matters: efficient and accurate normalization schemes in deep networks, in: Advances in Neural Information Processing Systems (NeurIPS), 2018.
[19] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: International Conference on Learning Representations (ICLR), 2015.
[20] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.
[21] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A. Y. Ng, Reading digits in natural images with unsupervised feature learning, in: NIPS Workshop on Deep Learning and Unsupervised Feature Learning, 2011.
[22] A. Krizhevsky, G. Hinton, et al., Learning multiple layers of features from tiny images, Tech. rep., University of Toronto (2009).
[23] B. Ghorbani, S. Krishnan, Y. Xiao, An investigation into neural net optimization via hessian eigenvalue density, in: 36th International Conference on Machine Learning (ICML), 2019.
[24] X. Lian, J. Liu, Revisit batch normalization: New understanding and refinement via composition optimization, in: The 22nd International Conference on Artificial Intelligence and Statistics (AISTATS), 2019.
[25] R. Karakida, S. Akaho, S.-i. Amari, The normalization method for alleviating pathological sharpness in wide neural networks, in: NeurIPS, 2019.
[26] J. Kohler, H. Daneshmand, A. Lucchi, T. Hofmann, M. Zhou, K. Neymeyr, Exponential convergence rates for batch normalization: The power of length-direction decoupling in non-convex optimization, in: AISTATS, 2019.
[27] H. Daneshmand, J. Kohler, F. Bach, T. Hofmann, A. Lucchi, Batch normalization provably avoids rank collapse for randomly initialised deep networks, in: NeurIPS, 2020.
[28] A. M. Saxe, J. L. McClelland, S. Ganguli, Exact solutions to the nonlinear dynamics of learning in deep linear neural networks, arXiv preprint arXiv:1312.6120.
[29] Y. Cai, Q. Li, Z. Shen, A quantitative analysis of the effect of batch normalization on gradient descent, in: 36th International Conference on Machine Learning (ICML), 2019.
[30] W. Liu, Y.-M. Zhang, X. Li, Z. Yu, B. Dai, T. Zhao, L. Song, Deep hyperspherical learning, in: Advances in Neural Information Processing Systems (NeurIPS), 2017.
[31] W. Liu, Z. Liu, Z. Yu, B. Dai, R. Lin, Y. Wang, J. M. Rehg, L. Song, Decoupled networks, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.
[32] Z. Li, S. Arora, An exponential learning rate schedule for deep learning, in: International Conference on Learning Representations (ICLR), 2020.
[33] D. Soudry, E. Hoffer, M. S. Nacson, S. Gunasekar, N. Srebro, The implicit bias of gradient descent on separable data, The Journal of Machine Learning Research (JMLR).
[34] G. Zhang, C. Wang, B. Xu, R. Grosse, Three mechanisms of weight decay regularization, in: International Conference on Learning Representations (ICLR), 2019.

**Simon Roburin** is a Ph.D. student in Deep Learning at Ecole National des Ponts ParisTech (ENPC) and valeo.ai research lab. He received a Bsc. in Mathematics from University of Paris as valecditorian of the class, a Bsc. in Engineerring and a M.Sc. degree in Applied Mathematics from Ecole Centrale Paris. He is also laureate of the Paris Graduate school of Mathematics from the Foundation Sciences Mathématiques de Paris. His research interests focus on Optimization as well as Robustness in Deep Learning.

**Yann de Mont-Marin** is a Ph D. student in Robotics and Machine Learning at Inria Paris. He received a a Bsc. in Engineerring and a M.Sc. degree in Applied Mathematics from Ecole Centrale Paris. He previously developed new language identification tools in the service industry. His research interests now focus on Robotic Motion as well as Machine Learning approach to Optimal Control.

**Andrei Bursuc** is Senior Research Scientist at valeo.ai in Paris, France. He completed his PhD at Mines ParisTech in 2012. He was a postdoc researcher at Inria Rennes and Inria Paris. In 2016, he moved to industry to pursue research on autonomous systems. His current research interests concern computer vision and deep learning, in particular learning with limited supervision and predictive uncertainty quantification.

**Renaud Marlet** is a Senior Researcher at École des Ponts ParisTech (ENPC) and a Principal Scientist at valeo.ai, France. He previously held positions both in academia (researcher at Inria) and in the software industry (expert at Simulog, deputy CTO of Trusted Logic). He was the head of the IMAGINE group at LIGM/ENPC (2010-2019). He is currently interested in 3D scene understanding and reconstruction.

**Patrick Pérez** is Scientific Director of Valeo.ai, a Valeo research lab on artificial intelligence for automotive applications. Before joining Valeo, Patrick Pérez has been Distinguished Scientist at Technicolor (2009-2918), researcher at Inria (1993-2000, 2004-2009) and at Microsoft Research Cambridge (2000-2004). His research revolves around machine learning for scene understanding, data mining and visual editing.

**Mathieu Aubry** received MSc degree from the Ecole Polytechnique, Ph D. from the Ecole Normale Superieure and Habilitation from Universite Paris-Est. In 2015, he spent a year as a visiting researcher at UC Berkeley and is a researcher at Ecole des Ponts. He works on unsupervised image and 3D shape analysis and developing Computer Vision approaches for Digital Humanities.

# Supplementary Material to "Spherical Perspective on Learning with Normalization Layers"

## Appendix A. Radial invariance of filters with BN

In this section, we show the radial invariance of a set of filters equipped with BN. Please note that the following notations are specific and restricted to this section.

For the sake of simplicity, we only consider the case of a convolutional layer that preserves the spatial extension of the input. We also focus on a single filter. Since all filters act independently on input data, the following calculation holds for any filter.

Let $\mathbf{x} \in \mathbb{R}^{C \times K}$ be the parameters of a single filter, where $C$ is the number of input channels and $K$ is the kernel size. During training, this layer is followed by BN and applied to a batch $\mathbf{s} \in \mathbb{R}^{B \times C \times D}$ of $B$ inputs of spatial size $D$. The output of the convolution operator $\phi$ applied to a filter $\mathbf{x} \in \mathbb{R}^{C \times K}$ and to a given batch element $\mathbf{s}_b \in \mathbb{R}^{C \times D}$, with $b \in [\![1, B]\!]$, is thus:

$$\mathbf{t}_b \stackrel{\text{def}}{=} \phi(\mathbf{x}, \mathbf{s}_b) \in \mathbb{R}^D. \tag{A.1}$$

The application $(\mathbf{x}, \mathbf{s}_b) \mapsto \phi(\mathbf{x}, \mathbf{s}_b)$ is bilinear. BN then centers and normalizes the output $\mathbf{t}$ using the mean and variance over the batch and the spatial dimension:

$$\mu = \frac{1}{BD} \sum_{b,j} t_{b,j}, \tag{A.2}$$

$$\sigma^2 = \frac{1}{BD} \sum_{b,j} \left(t_{b,j} - \mu\right)^2, \tag{A.3}$$

$$\hat{\mathbf{t}}_b \stackrel{\text{def}}{=} \left(\sigma^2 + \epsilon\right)^{-1/2} \left(\mathbf{t}_b - \mu \mathbf{1}_D\right), \tag{A.4}$$

where $\mathbf{1}_D$ denotes the all-ones vector of dimension $D$ and $\epsilon$ is a small constant.

Now if the coefficients of the filter are rescaled by $\rho > 0$, then, by bilinearity, the new output of the layer for this filter verifies:

$$\tilde{\mathbf{t}}_b = \phi(\rho \mathbf{x}, \mathbf{s}_b) = \rho \phi(\mathbf{x}, \mathbf{s}_b). \tag{A.5}$$

Since the variance of inputs is generally large in practice, for small $\epsilon$, the mean and variance are:

$$\tilde{\mu} = \rho \mu, \tag{A.6}$$

$$\tilde{\sigma}^2 \approx \rho^2 \sigma^2. \tag{A.7}$$

It can then be considered that the subsequent BN layer is invariant to this rescaling, *i.e.*, $\hat{\tilde{\mathbf{t}}}_b \approx \hat{\mathbf{t}}_b$.

## Appendix B. Extension to other normalization layers

The radial invariance for BN described above in Appendix A applies as well to InstanceNorm (IN) [5] and WeightNorm (WN) [4] as the normalization is also done with respect to channels but

without the batch dimension. Regarding LayerNorm [3] (LN), the normalization is performed over all channels and the entire weight layer can thus be rescaled too, without impacting the output. As for GroupNorm [6] (GN), it associates several channels for normalization; the radial invariance in this case concerns the corresponding group of filters.

Thanks to this general property of radial invariance, the results in this paper not only concern BN but also WN and IN. In fact, they apply as well to LN and GN when considering the suitable group of parameters. The optimization in this case concerns the proper slice of the parameter tensor of the layer, i.e., the whole tensor for LN, and the selected group of filters for GN.

## Appendix C. Results in Sections 2 and 3

In this section, we provide proofs and/or empirical results supporting the claims in Sections 2 and 3 of the paper.

In the following, the double parentheses around an equation number, e.g., ((12)), indicate that we recall an equation that was previously stated in the main paper, rather than introduce a new one, e.g., noted (C.1). Also, framed formulas actually refer to results stated in the main paper, thus with double-bracket equation numbering.

### *Appendix C.1. Proof of theorems and validity of assumptions*

#### *Appendix C.1.1. Proof of Theorem 2 (Image step on $\mathcal{S}_{d-1}$) in Section 2.3*

We recall the main theorem in Section 2.3.

**Theorem 2** (Image step on $\mathcal{S}_{d-1}$) *If the following hypothesis are verified:*

- *(H1):* $1-\frac{\eta_k\langle \mathbf{c}_k,\mathbf{u}_k\rangle}{r_k^2 d^{-1/2}\|\mathbf{b}_k\|} > 0$.
- *(H2):* $\eta_k^e\|\mathbf{c}_k^\perp\| < \pi$.

*The update of a group of radially-invariant parameters $\mathbf{x}_k$ at step $k$ following the generic optimization scheme (Eqs. 3-4) and the corresponding update of its projection $\mathbf{u}_k$ on $\mathcal{S}_{d-1}$ is given by an exponential map at $\mathbf{u}_k$ with velocity $\eta_k^e\mathbf{c}_k^\perp$:*

$$\mathbf{u}_{k+1} = \mathrm{Exp}_{\mathbf{u}_k}\left(-\left[1+O\left(\left(\eta_k^e\|\mathbf{c}_k^\perp\|\right)^2\right)\right]\eta_k^e\mathbf{c}_k^\perp\right), \tag{(12)}$$

*where $\mathrm{Exp}_{\mathbf{u}_k}$ is the exponential map on $\mathcal{S}_{d-1}$, and with*

$$\mathbf{c}_k \stackrel{\text{def}}{=} r_k\mathbf{a}_k \oslash \frac{\mathbf{b}_k}{d^{-1/2}\|\mathbf{b}_k\|}, \quad \eta_k^e \stackrel{\text{def}}{=} \frac{\eta_k}{r_k^2 d^{-1/2}\|\mathbf{b}_k\|}\left(1-\frac{\eta_k\langle \mathbf{c}_k,\mathbf{u}_k\rangle}{r_k^2 d^{-1/2}\|\mathbf{b}_k\|}\right)^{-1}. \tag{(14)}$$

*More precisely:*

$$\mathbf{u}_{k+1} = \frac{\mathbf{u}_k - \eta_k^e\mathbf{c}_k^\perp}{\sqrt{1+(\eta_k^e\|\mathbf{c}_k^\perp\|)^2}}. \tag{(15)}$$

PROOF. To simplify the calculation in the demonstration, we introduce the following notation:

$$A_k \stackrel{\text{def}}{=} \frac{\eta_k}{r_k^2 d^{-1/2}\|\mathbf{b}_k\|}. \tag{C.1}$$

We first demonstrate the expression for the radius dynamics in Eq. (17) and the precise step for $\mathbf{u}$ in Eq. (15). Then we use geometric arguments and a Taylor expansion to derive the update on the sphere stated in Eq.(12).

**Radius dynamics.** We first show Eq. (17), which we recall here using the $A_k$ notation:

$$\boxed{\frac{r_{k+1}}{r_k} = (1 - A_k \langle \mathbf{c}_k, \mathbf{u}_k \rangle) \sqrt{1 + (\eta_k^e \|\mathbf{c}_k^\perp\|)^2}.} \tag{(17)}$$

First, we rewrite the step of a generic scheme in Eqs. (3-4) along the radial and tangential directions and separate the division vector $\mathbf{b}_k$ into its deformation $\frac{\mathbf{b}_k}{d^{-1/2}\|\mathbf{b}_k\|}$ and its scalar scheduling effect $d^{-1/2}\|\mathbf{b}_k\|$, as stated in the discussion:

$$\begin{aligned} r_{k+1}\mathbf{u}_{k+1} &= r_k \mathbf{u}_k - \frac{\eta_k}{d^{-1/2}\|\mathbf{b}_k\|} \mathbf{a}_k \oslash \frac{\mathbf{b}_k}{d^{-1/2}\|\mathbf{b}_k\|} \\ &= r_k \left[ \mathbf{u}_k - \frac{\eta_k}{r_k^2 d^{-1/2}\|\mathbf{b}_k\|} r_k \mathbf{a}_k \oslash \frac{\mathbf{b}_k}{d^{-1/2}\|\mathbf{b}_k\|} \right] \\ &= r_k \left[ \mathbf{u}_k - A_k r_k \mathbf{a}_k \oslash \frac{\mathbf{b}_k}{d^{-1/2}\|\mathbf{b}_k\|} \right]. \end{aligned} \tag{C.2}$$

We can note the appearance of a new term $r_k \mathbf{a}_k$. The vector $\mathbf{a}_k$ is a gradient momentum and therefore homogeneous to a gradient. Using Lemma 1, $r_k \mathbf{a}_k$ is homogeneous to a gradient on the hypersphere and can be interpreted as the momentum on the hypersphere.

From Eq. (C.2), we introduce $\mathbf{c}_k$ (the deformed momentum on hypersphere) as in Eq. (14) and decompose it into the radial and tangential components. We have:

$$\begin{aligned} \frac{r_{k+1}}{r_k} \mathbf{u}_{k+1} &= \mathbf{u}_k - A_k \mathbf{c}_k \\ &= (1 - A_k \langle \mathbf{c}_k, \mathbf{u}_k \rangle) \mathbf{u}_k - A_k \mathbf{c}_k^\perp. \end{aligned} \tag{C.3}$$

By taking the squared norm of the equation, we obtain:

$$\frac{r_{k+1}^2}{r_k^2} = (1 - A_k \langle \mathbf{c}_k, \mathbf{u}_k \rangle)^2 + \left( A_k \|\mathbf{c}_k^\perp\| \right)^2. \tag{C.4}$$

Making the assumption that $1 - A_k \langle \mathbf{c}_k, \mathbf{u}_k \rangle > 0$, which is true in practice and discussed in the next subsection, we have:

$$\frac{r_{k+1}}{r_k} = (1 - A_k \langle \mathbf{c}_k, \mathbf{u}_k \rangle) \sqrt{1 + \left( \frac{A_k}{1 - A_k \langle \mathbf{c}_k, \mathbf{u}_k \rangle} \|\mathbf{c}_k^\perp\| \right)^2}. \tag{C.5}$$

After introducing $\eta_k^e = \frac{A_k}{(1 - A_k \langle \mathbf{c}_k, \mathbf{u}_k \rangle)}$ as in Eq. (14), we obtain the result of (17).

**Update of normalized parameters.** We then show Eq. (15):

$$\boxed{\mathbf{u}_{k+1} = \frac{\mathbf{u}_k - \eta_k^e \mathbf{c}_k^\perp}{\sqrt{1 + (\eta_k^e \|\mathbf{c}_k^\perp\|)^2}}.} \tag{(15)}$$

Combining the radius dynamics previously calculated with Eq. (C.3), we have:

$$\mathbf{u}_{k+1} = \frac{(1 - A_k\langle \mathbf{c}_k, \mathbf{u}_k\rangle)\mathbf{u}_k - A_k\mathbf{c}_k^{\perp}}{(1 - A_k\langle \mathbf{c}_k, \mathbf{u}_k\rangle)\sqrt{1 + (\eta_k^e\|\mathbf{c}_k^{\perp}\|)^2}} \tag{C.6}$$

$$= \frac{\mathbf{u}_k - \frac{A_k}{1-A_k\langle \mathbf{c}_k,\mathbf{u}_k\rangle}\mathbf{c}_k^{\perp}}{\sqrt{1 + (\eta_k^e\|\mathbf{c}_k^{\perp}\|)^2}}. \tag{C.7}$$

Hence the result (15) using the definition of $\eta_k^e$.

This result provides a unique decomposition of the generic step as a step in $\mathrm{span}(\mathbf{u}_k, \mathbf{c}_k^{\perp})$ for the normalized filter (Eq. (15)) and as a radius update (Eq. (17)).

We split the rest of the proof of the theorem in three parts.

**Distance covered on the sphere.** The distance covered on the hypersphere $\mathcal{S}_{d-1}$ by an optimization step is:

$$\mathrm{dist}_{\mathcal{S}_{d-1}}(\mathbf{u}_{k+1}, \mathbf{u}_k) = \arccos(\langle \mathbf{u}_{k+1}, \mathbf{u}_k\rangle). \tag{C.8}$$

From Eq. (15) and with Lemma 1, we also have:

$$\langle \mathbf{u}_{k+1}, \mathbf{u}_k\rangle = \frac{1}{\sqrt{1 + (\eta_k^e\|\mathbf{c}_k^{\perp}\|)^2}}. \tag{C.9}$$

Therefore, $\mathrm{dist}_{\mathcal{S}_{d-1}}(\mathbf{u}_{k+1}, \mathbf{u}_k) = \varphi(\eta_k^e\|\mathbf{c}_k^{\perp}\|)$ where $\varphi : z \mapsto \arccos\left(\frac{1}{\sqrt{1+z^2}}\right)$, which is equal to $\arctan$ on $\mathbb{R}_+$. Then a Taylor expansion at order 3 of $\arctan$ yields for $\eta_k^e\|\mathbf{c}_k^{\perp}\|$:

$$\mathrm{dist}_{\mathcal{S}_{d-1}}(\mathbf{u}_{k+1}, \mathbf{u}_k) = \eta_k^e\|\mathbf{c}_k^{\perp}\| + O\left(\left(\eta_k^e\|\mathbf{c}_k^{\perp}\|\right)^3\right). \tag{C.10}$$

The Taylor expansion validity is discussed in the next subsection.

**Exponential map on the sphere.** Given a Riemannian manifold $\mathcal{M}$, for a point $\mathbf{u} \in \mathcal{M}$ there exists an open set $\mathcal{O}$ of the tangent space $\mathcal{T}_{\mathbf{u}}\mathcal{M}$ containing $\mathbf{0}$, such that for any tangent vector $\mathbf{w} \in \mathcal{O}$ there is a unique geodesic (a path minimizing the local distance on $\mathcal{M}$ when conserving the tangent velocity) $\gamma : [-1, 1] \to \mathcal{M}$ that is differentiable and such that $\gamma(0) = \mathbf{u}$ and $\gamma'(0) = \mathbf{w}$. Then, the exponential map of $\mathbf{w}$ from $\mathbf{u}$ is defined as $\mathrm{Exp}_{\mathbf{u}}(\mathbf{w}) = \gamma(1)$.

In the case of the manifold $\mathcal{S}_{d-1}$, the geodesics are complete (they are well defined for any point $\mathbf{u} \in \mathcal{S}_{d-1}$ and any velocity $\mathbf{w} \in \mathcal{T}_{\mathbf{u}}\mathcal{S}_{d-1}$) and are the great circles: for any $\mathbf{u} \in \mathcal{S}_{d-1}$ and any $\mathbf{w} \in \mathcal{T}_{\mathbf{u}}\mathcal{S}_{d-1}$, the map $\psi : t \in \mathbb{R} \mapsto \mathrm{Exp}_{\mathbf{u}}(t\mathbf{w}))$ verifies $\psi(\mathbb{R}) = \mathcal{S}_{d-1} \cap \mathrm{span}(\{\mathbf{u}, \mathbf{w}\})$ which is a great circle passing through $\mathbf{u}$ with tangent $\mathbf{w}$. Furthermore, since the circumference of the great circle is $2\pi$, we have that for any $\mathbf{p} \in \mathcal{S}_{d-1}\backslash\{-\mathbf{u}\}$ there is a unique $\mathbf{w}$ verifying $\|\mathbf{w}\| < \pi$ such that $\mathbf{p} = \mathrm{Exp}_{\mathbf{u}}(\mathbf{w})$ and we have:

$$\mathrm{dist}_{\mathcal{S}_{d-1}}(\mathbf{u}, \mathbf{p}) = \|\mathbf{w}\| \text{ and } \langle \mathbf{p}, \mathbf{w}\rangle \geq 0. \tag{C.11}$$

**Optimization step as an exponential map.** We will use the previously stated differential geometry properties to prove:

$$\boxed{\mathbf{u}_{k+1} = \mathrm{Exp}_{\mathbf{u}_k}\left(-\left[1 + O\left(\left(\eta_k^e\|\mathbf{c}_k^{\perp}\|\right)^2\right)\right]\eta_k^e\mathbf{c}_k^{\perp}\right).} \tag{(12)}$$

For an optimization step we have:

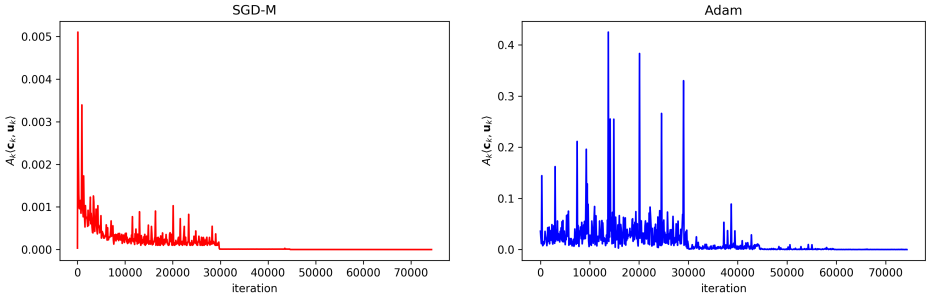



Figure C.6: **Tracking of $A_k\langle \mathbf{c}_k, \mathbf{u}_k\rangle$ for SGD-M and Adam.** The above graphs show the maximum of the absolute value of $A_k\langle \mathbf{c}_k, \mathbf{u}_k\rangle$ for all filters in all layers of a ResNet20 CIFAR trained on CIFAR10 and optimized with SGD-M (left) or Adam (right). The quantity is always small compared to 1. Therefore we may assume that $1 - A_k\langle \mathbf{c}_k, \mathbf{u}_k\rangle \geq 0$.

- by construction, $\mathbf{c}_k^\perp \in \mathcal{T}_{\mathbf{u}_k}\mathcal{S}_{d-1}$;
- from Eq. (15), $\mathbf{u}_{k+1} \in \mathcal{S}_{d-1} \cap \mathrm{span}(\{\mathbf{u}_k, \mathbf{c}_k^\perp\})$;
- from Eq. (15), $\langle \mathbf{u}_{k+1}, \mathbf{c}_k^\perp\rangle \leq 0$.

Then, there exists $\alpha$ that verifies $\|\alpha \mathbf{c}_k^\perp\| < \pi$ such that:

$$\mathbf{u}_{k+1} = \mathrm{Exp}_{\mathbf{u}_k}\left(\alpha \mathbf{c}_k^\perp\right). \tag{C.12}$$

From Eq. (C.11), because of the inequality $\langle \mathbf{u}_{k+1}, \mathbf{c}_k^\perp\rangle \leq 0$, we have $\alpha < 0$. We also have that $\|\alpha \mathbf{c}_k^\perp\| = \mathrm{dist}_{\mathcal{S}_{d-1}}(\mathbf{u}_{k+1}, \mathbf{u}_k)$. Then, using the distance previously calculated in Eq. (C.10), we have:

$$|\alpha|\|\mathbf{c}_k^\perp\| = \eta_k^e \|\mathbf{c}_k^\perp\| + O\left(\left(\eta_k^e \|\mathbf{c}_k^\perp\|\right)^3\right), \tag{C.13}$$

$$|\alpha| = \eta_k^e \left[1 + O\left(\left(\eta_k^e \|\mathbf{c}_k^\perp\|\right)^2\right)\right]. \tag{C.14}$$

Combining the sign and absolute value of $\alpha$, we get the final exponential map expression:

$$\mathbf{u}_{k+1} = \mathrm{Exp}_{\mathbf{u}_k}\left(-\left[1 + O\left(\left(\eta_k^e \|\mathbf{c}_k^\perp\|\right)^2\right)\right] \eta_k^e \mathbf{c}_k^\perp\right), \tag{(12)}$$

$$\approx \mathrm{Exp}_{\mathbf{u}_k}\left(-\eta_k^e \mathbf{c}_k^\perp\right). \tag{C.15}$$

Note that we implicitly assume here that $|\alpha|\|\mathbf{c}_k^\perp\| \approx \eta_k^e \|\mathbf{c}_k^\perp\| < \pi$, which is discussed in the next subsection.

*Appendix C.1.2. Assumptions in Theorem 2 and validity*

*Sign of* $1 - A_k\langle \mathbf{c}_k, \mathbf{u}_k\rangle$.. We tracked the maximum of the quantity $A_k\langle \mathbf{c}_k, \mathbf{u}_k\rangle$ for all the filters of a ResNet20 CIFAR trained on CIFAR10 and optimized with SGD-M or Adam (see Appendix Appendix D.3 for implementation details). As can be seen on Figure C.6, this quantity is always small compared to 1, making $1 - A_k\langle \mathbf{c}_k, \mathbf{u}_k\rangle$ always positive in practice. The order of magnitude of this quantity is roughly the same for different architectures and datasets.

*Taylor expansion..* We tracked the maximum of the quantity $\eta_k^e \|\mathbf{c}_k^\perp\|$ for all the filters of a ResNet20 CIFAR trained on CIFAR10 and optimized with SGD-M or Adam. The observed values justify the Taylor expansion and validate the assumption $|\alpha|\|\mathbf{c}_k^\perp\| \approx \eta_k^e \|\mathbf{c}_k^\perp\| < \pi$. (cf. Figure C.7). The order of magnitude of this quantity is roughly the same for other different architectures and datasets.

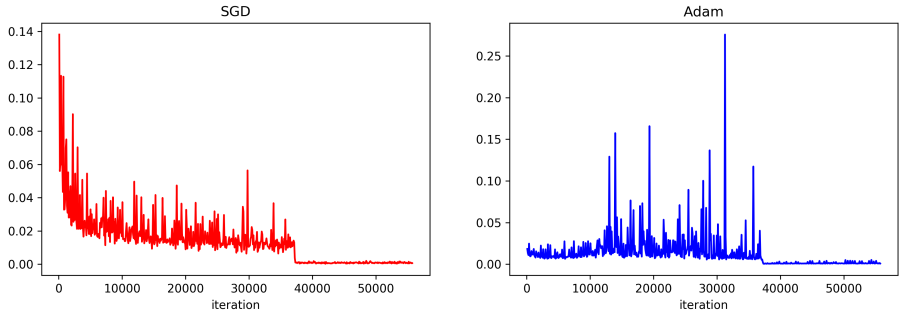



Figure C.7: **Tracking of $\eta_k^e \|\mathbf{c}_k^\perp\|$ for SGD-M and Adam.** The above graphs show the maximum of the absolute value of $\eta_k^e \|\mathbf{c}_k^\perp\|$ for all filters in all layers of a ResNet20 CIFAR trained on CIFAR10 and optimized with SGD-M (left) or Adam (right).

*Appendix C.1.3. $\nu_k$, order 2 moment on the hypersphere for Adam*

**Scheduling effect of Adam division vector.** With Eq. (D.14) and using Lemma 1, we can give the expression of the second-order moment on the sphere, defined as $\nu_k = r_k d^{-1/2} \|\mathbf{b}_k\|$:

$$\boxed{\nu_k = d^{-1/2} \frac{1-\beta_1^{k+1}}{1-\beta_1} \left( \frac{1-\beta_2}{1-\beta_2^{k+1}} \right)^{1/2} \left( \sum_{i=0}^{k} \beta_2^{k-i} \frac{r_k^2}{r_i^2} \|\nabla \mathcal{L}(\mathbf{u}_i) + \lambda r_i^2 \mathbf{u}_i\|^2 \right)^{1/2}.} \tag{C.16}$$

*Appendix C.1.4. Proof of Theorem 4 (SGD equivalent scheme on the unit hypersphere) in Section 3.2*

We prove the following theorem:

**Theorem 4** (SGD equivalent scheme on the unit hypersphere.) *For any $\lambda > 0, \eta > 0, r_0 > 0$, we have the following equivalence at order 2 in the radius dynamics:*

$$\begin{cases} \text{(SGD)} \\ \mathbf{x}_0 = r_0 \mathbf{u}_0 \\ \lambda_k = \lambda \\ \eta_k = \eta \end{cases} \quad \textit{is scheme-equivalent at order 2 to} \quad \begin{cases} \text{(AdaGradG)} \\ \mathbf{x}_0 = \mathbf{u}_0 \\ \beta = (1-\eta\lambda)^4 \\ \eta_k = (2\beta)^{-1/2} \\ \upsilon_0 = r_0^4 (2\eta^2 \beta^{1/2})^{-1} \end{cases}$$

PROOF. As summarized in Table 1, the expressions of the effective learning rates and directions for SGD are $\mathbf{c}_k^\perp = r_k \nabla \mathcal{L}(\mathbf{x}_k) = \nabla \mathcal{L}(\mathbf{u}_k)$ and $\eta_k^e = \frac{\eta_k}{r_k^2(1-\eta_k\lambda_k)}$.

**Equivalence with SGD and $L_2$ regularization.** We look for conditions leading to an equivalence between SGD with $L_2$ regularization and SGD without $L_2$ regularization. Using Lemma 3, the equality of effective directions is trivial and the equality of effective learning rates for any step $k$ yields the following equivalence:

$$\begin{cases} \text{(SGD)} \\ \tilde{\mathbf{x}}_0 = r_0 \mathbf{u}_0 \\ \tilde{\lambda}_k = \lambda \\ \tilde{\eta}_k = \eta \end{cases} \quad \text{is scheme-equivalent to} \quad \begin{cases} \text{(SGD)} \\ \mathbf{x}_0 = r_0 \mathbf{u}_0 \\ \lambda_k = 0 \\ \eta_k = \eta(1-\eta\lambda)^{-2k-1} \end{cases} \tag{C.17}$$

$L_2$ regularization is equivalent to an exponential scheduling of the learning rate, as found in [32]. Here, we provide a proof in a constructive manner. We are going to use Lemma 3 and find a sufficient condition to have:

$$\begin{cases} \text{(i) } \mathbf{u}_0 = \tilde{\mathbf{u}}_0 \\ \text{(ii) } \forall k \geq 0, \eta_k^e = \tilde{\eta}_k^e, \mathbf{c}_k^\perp = \tilde{\mathbf{c}}_k^\perp. \end{cases}$$

Equation (i) is trivially satisfied by simply taking the same starting point: $\tilde{\mathbf{x}}_0 = \mathbf{x}_0$.

Regarding (ii), because effective directions are the same and only depend on $\mathbf{u}_k$, we only need a sufficient condition on $\eta_k^e$. For effective learning rates, using Eq. ((17)) and expressions in Table 1, we have:

$$\eta_k^e = \tilde{\eta_k^e} \Leftrightarrow \frac{\eta_k}{r_k^2} = \frac{\tilde{\eta}_k}{\tilde{r}_k^2(1 - \tilde{\eta}_k \lambda)}. \tag{C.18}$$

Since $\tilde{\eta_k} = \eta$, we obtain:

$$\text{(C.18)} \Leftrightarrow \eta_k = \left(\frac{r_k}{\tilde{r}_k}\right)^2 \frac{\eta}{(1 - \eta\lambda)}.$$

Therefore:

$$\frac{\eta_{k+1}}{\eta_k} = \left(\frac{r_{k+1}\tilde{r}_k}{\tilde{r}_{k+1} r_k}\right)^2 = \left(\frac{r_{k+1}/r_k}{\tilde{r}_{k+1}/\tilde{r}_k}\right)^2.$$

By using the radius dynamics in Eq. (17) for the two schemes, SGD and SGD with $L_2$ regularization, and by the equality of effective learning rates and directions, we have:

$$\begin{aligned} \frac{\eta_{k+1}}{\eta_k} &= \left(\frac{\sqrt{1 + (\eta_k^e \|\mathbf{c}_k^\perp\|)^2}}{(1 - \eta\lambda)\sqrt{1 + (\tilde{\eta_k^e} \|\tilde{\mathbf{c}_k^\perp}\|)^2}}\right)^2 \\ &= (1 - \eta\lambda)^{-2}. \end{aligned}$$

By taking Eq. (C.18) for $k = 0$, because $r_0 = \tilde{r}_0$ we have: $\eta_0 = \eta(1 - \eta\lambda)^{-1}$. Combining the previous relation and the initialization case, we derive by induction that $\eta_k = \eta(1 - \eta\lambda)^{-2k-1}$ is a sufficient condition. We can conclude, using Lemma 3, the equivalence stated in Eq. (C.17).

**Resolution of the radius dynamics.** Without $L_2$ regularization, the absence of radial component in $\mathbf{c}_k$ makes the radius dynamics simple:

$$r_{k+1}^2 = r_k^2 + \frac{(\eta_k \|\nabla \mathcal{L}(\mathbf{u}_k)\|)^2}{r_k^2}. \tag{C.19}$$

With a Taylor expansion at order 2, we can show that for $k \geq 1$ the solution

$$r_k^2 = \sqrt{2 \sum_{i=0}^{k-1} (\eta_i \|\nabla \mathcal{L}(\mathbf{u}_i)|)^2 + r_0^4}$$

satisfies the previous equation. Indeed using the expression at step $k+1$ gives:

$$\begin{aligned}
r_{k+1}^2 &= \sqrt{2\sum_{i=0}^{k-1}(\eta_i\|\nabla\mathcal{L}(\mathbf{u}_i)|)^2 + r_0^4 + 2(\eta_k\|\nabla\mathcal{L}(\mathbf{u}_k)\|)^2} \\
&= r_k^2\sqrt{1+2\frac{(\eta_k\|\nabla\mathcal{L}(\mathbf{u}_k)\|)^2}{r_k^4}} \\
&= r_k^2\left(1+(1/2)2\frac{(\eta_k\|\nabla\mathcal{L}(\mathbf{u}_k)\|)^2}{r_k^4} + o\left(\frac{(\eta_k\|\nabla\mathcal{L}(\mathbf{u}_k)\|)^2}{r_k^4}\right)\right) \\
&= r_k^2 + \frac{(\eta_k\|\nabla\mathcal{L}(\mathbf{u}_k)\|)^2}{r_k^2} + o\left(\frac{(\eta_k\|\nabla\mathcal{L}(\mathbf{u}_k)\|)^2}{r_k^2}\right).
\end{aligned}$$

Using $\eta_k = \eta(1-\eta\lambda)^{-2k-1}$, introducing $\beta = (1-\eta\lambda)^4$, omitting the $o\left(\frac{(\eta_k\|\nabla\mathcal{L}(\mathbf{u}_k)\|)^2}{r_k^2}\right)$ and injecting the previous solution in the effective learning rate, we obtain the closed form:

$$\begin{aligned}
\eta_k^e &= \frac{\eta(1-\eta\lambda)^{-2k-1}}{\sqrt{2\sum_{i=0}^{k-1}\eta^2(1-\eta\lambda)^{-4i-2}\|\nabla\mathcal{L}(\mathbf{u}_i)\|^2 + r_0^4}} \\
&= \frac{(2\beta)^{-\frac{1}{2}}}{\sqrt{\sum_{i=0}^{k-1}\beta^{(k-1)-i}\|\nabla\mathcal{L}(\mathbf{u}_i)\|^2 + \beta^k\frac{r_0^4}{2\eta^2\beta^{\frac{1}{2}}}}}.
\end{aligned} \tag{C.20}$$

**AdaGradG.** The AdaGradG scheme is constrained on the hypersphere thanks to the normalization; the radius is therefore constant and equal to 1. The absence of radial component in the update gives: $\mathbf{c}_k^\perp = \nabla\mathcal{L}(\mathbf{u}_k)$ and $\eta_k^e = \frac{\eta_k}{\sqrt{v_k}}$. Thus, the resolution of the induction on $v_k$ leads to the the closed form:

$$\eta_k^e = \frac{\eta_k}{\sqrt{\sum_{i=0}^{k-1}\beta^{(k-1)-i}\|\nabla\mathcal{L}(\mathbf{u}_i)\|^2 + \beta^k v_0}}. \tag{C.21}$$

Hence the final theorem, when identifying the closed-form expressions of effective learning rates and using Lemma 3.

*Appendix C.1.5. Validity of the assumptions in Theorem 4*

**Validity of the Taylor expansion.** For a CNN trained with SGD optimization, we tracked the quantity $(\eta_k\|\nabla\mathcal{L}(\mathbf{u}_k)\|)^2/r_k^2$, which is the variable of the Taylor expansion. As can be seen in Figure C.8, the typical order of magnitude is $10^{-2}$, justifying the Taylor expansion.

A quick formal analysis also suggests the validity of this hypothesis. Thanks to the expression of $\eta_k = (1-\eta\lambda)^{-2i-k}\eta$ shown in the previous section, if we replace $\|\nabla\mathcal{L}(\mathbf{u}_k)\|$ by a constant for asymptotic analysis, the comparison becomes:

$$(1-\eta\lambda)^{-4k-2} \ll (1-\eta\lambda)^{-2}\frac{1-(1-\eta\lambda)^{-4k}}{1-(1-\eta\lambda)^{-4}} \tag{C.22}$$

$$1 \ll \frac{1-(1-\eta\lambda)^{4k}}{(1-\eta\lambda)^{-4}-1}. \tag{C.23}$$

It is asymptotically true.

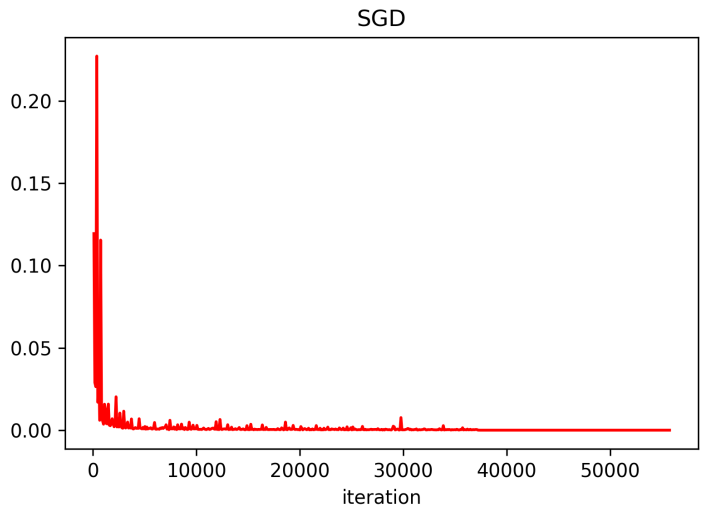



Figure C.8: **Validity of Taylor expansion**. We tracked the maximum value of $(\eta_k \|\nabla \mathcal{L}(\mathbf{u}_k)\|)^2/r_k^2$ for all filters in all layers of a ResNet20 CIFAR trained on CIFAR10 with SGD. The order of magnitude of the gradient is roughly the same for other architectures or datasets. It empirically validates the approximation by the Taylor expansion.

*Appendix C.1.6. Implementation details of weight trajectory tracking*

Due to the high non-convexity of the optimization landscape, we choose to start from a relatively stable point in the parameter space. The finetuning of each architecture (ResNet20 BN, ResNet20 BN w/o affine and ResNet WN) starts from previously trained architectures on CIFAR10 via a simple SGD with an initial learning rate of $10^{-1}$, a $L_2$-regularization parameter of $10^{-4}$ and a momentum parameter of $0.9$. The training is performed during 200 epochs, and the learning rate is multiplied by $0.1$ at epochs $80$, $120$ and $160$.

Then we track the trajectory obtained with SGD, AdaGrad and AdaGradG. The effective learning rate for SGD is fixed to $10^{-2}$ and the $L_2$-regularization parameter is set to $10^{-3}$ during finetuning. It gives us the following equivalent parameters for AdaGradG: order-2 moment parameter $\beta \approx 0.99996$ and learning rate $\eta \approx 0.71$. Since the effective direction is the same for both SGD and AdaGrad (Adam without momentum), in order to have the same order of magnitude for the gradient steps we need to have effective learning rates of same order of magnitude. From Table 1, in the case of SGD we have $\eta^e_{k\,\text{SGD}} = \frac{\eta_k}{r_k^2}$, and in the case of AdaGrad we have $\eta^e_{k\,\text{AdaGrad}} = \frac{\eta_k}{r_k \nu_k} = \frac{\eta_k}{r_k^2 d^{-1/2}\|\mathbf{b}_k\|} = \eta^e_{k\,\text{SGD}} \frac{1}{d^{-1/2}\|\mathbf{b}_k\|}$.We track the quantity $\frac{1}{d^{-1/2}\|\mathbf{b}_k\|}$ during training, which is roughly in the order of magnitude of $10^{-1}$. Therefore, to have gradient steps of equivalent order of magnitude between SGD and AdaGrad, we have to choose a learning rate of $10^{-3}$ for AdaGrad.

## Appendix D. Results in Section 4 (Geometric phenomena in Adam optimization)

### *Appendix D.1. Results in Section 4.2 (Identification of geometrical phenomena in Adam)*

**Decomposition of the effective direction.** We decompose the effective direction as a gradient term and an $L_2$ regularization term:

$$\boxed{\mathbf{c}_k^{\text{grad}} = \nabla\mathcal{L}(\mathbf{u}_k) + \sum_{i=0}^{k-1} \beta^{k-i} \frac{r_k}{r_i} \nabla\mathcal{L}(\mathbf{u}_i),} \tag{(21)}$$

$$\boxed{\mathbf{c}_k^{L_2} = \mathbf{u}_k + \sum_{i=0}^{k-1} \beta^{k-i} \frac{r_i}{r_k} \mathbf{u}_i.} \tag{(21)}$$

Note that these expressions highlight the main terms at step $k$ and the dependency on $r_i$.

Developing the recurrence in Eq (4), we obtain:

$$\mathbf{a}_k = \sum_{i=0}^{k} \beta^{k-i} \left(\nabla\mathcal{L}(\mathbf{x}_i) + \lambda \mathbf{x}_i\right). \tag{D.1}$$

Using Lemma 1 and decomposing on $\nabla\mathcal{L}(\mathbf{u}_i)$ and $\mathbf{u}_i$, we have:

$$\mathbf{a}_k = \sum_{i=0}^{k} \beta^{k-i} \left(\frac{1}{r_i}\nabla\mathcal{L}(\mathbf{u}_i) + \lambda r_i \mathbf{u}_i\right) \tag{D.2}$$

$$= \frac{1}{r_k}\left(\sum_{i=0}^{k} \beta^{k-i} \left(\frac{r_k}{r_i}\nabla\mathcal{L}(\mathbf{u}_i) + \lambda r_k r_i \mathbf{u}_i\right)\right). \tag{D.3}$$

Thus:

$$r_k \mathbf{a}_k = \sum_{i=0}^{k} \beta^{k-i} \frac{r_k}{r_i} \nabla\mathcal{L}(\mathbf{u}_i) + \lambda r_k^2 \sum_{i=0}^{k} \beta^{k-i} \frac{r_i}{r_k} \mathbf{u}_i, \tag{D.4}$$

which leads to the expression of $\mathbf{c}_k^{\text{grad}}$ and $\mathbf{c}_k^{L_2}$ when we define $\mathbf{c}_k \stackrel{\text{def}}{=} r_k \mathbf{a}_k \oslash \frac{\mathbf{b}_k}{d^{-1/2}\|\mathbf{b}_k\|}$ (Eq. (14)).

### *Appendix D.2. Results in Section 4.2 (Empirical study)*

**Clarification on Adam without deformation of gradients (a).** Following Theorem 2, the division vector $\mathbf{b}_k$ has two contributions in the decomposition:

- a deformation in $\mathbf{c}_k$ applied to $\mathbf{a}_k$: $\mathbf{c}_k = r_k \mathbf{a}_k \oslash \frac{\mathbf{b}_k}{d^{-1/2}\|\mathbf{b}_k\|}$;
- a scheduling effect in the effective learning rate $d^{-1/2}\|\mathbf{b}_k\|$ (Eq. (14)).

The goal is to find a new division vector $\mathsf{S}(\mathbf{b}_k)$ that does not create a deformation while preserving the scheduling effect of $\mathbf{b}_k$ in the effective learning rate. This means:

$$\frac{\mathsf{S}(\mathbf{b}_k)}{d^{-1/2}\|\mathsf{S}(\mathbf{b}_k)\|} = [1 \cdots 1]^\top, \tag{D.5}$$

$$d^{-1/2}\|\mathsf{S}(\mathbf{b}_k)\| = d^{-1/2}\|\mathbf{b}_k\|. \tag{D.6}$$

This leads to $\mathsf{S}(\mathbf{b}_k) = d^{-1/2}\|\mathbf{b}_k\|[1 \cdots 1]^\top$.

In the case of $\beta_1 = 0$, $\mathbf{a}_k = \nabla\mathcal{L}(\mathbf{x}_k)$, for any $\mathbf{b}_k$. When we apply the standardization, we obtain:

$$\mathbf{c}_k = r_k \nabla\mathcal{L}(\mathbf{x}_k) \oslash \frac{\mathsf{S}(\mathbf{b}_k)}{d^{-1/2}\|\mathsf{S}(\mathbf{b}_k)\|} = \nabla\mathcal{L}(\mathbf{u}_k) \oslash [1 \cdots 1]^\top = \nabla\mathcal{L}(\mathbf{u}_k). \tag{D.7}$$

The direction lies in the tangent space because, by Lemma 1, the gradient belongs to it.

In the generic scheme, using the standardization gives:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \eta_k \mathbf{a}_k \oslash \mathsf{S}(\mathbf{b}_k) \tag{D.8}$$

$$= \mathbf{x}_k - \eta_k \mathbf{a}_k \oslash (d^{-1/2}\|\mathbf{b}_k\|[1 \cdots 1]^\top) \tag{D.9}$$

$$= \mathbf{x}_k - \eta_k \mathbf{a}_k / (d^{-1/2}\|\mathbf{b}_k\|). \tag{D.10}$$

This means that the standardization consists in replacing the Hadamard division by $\mathbf{b}_k$ with a scalar division by $d^{-1/2}\|\mathbf{b}_k\|$.

In the case of Adam, we recall that:

$$\mathbf{b}_k = \frac{1-\beta_1^{k+1}}{1-\beta_1}\sqrt{\frac{\mathbf{v}_k}{1-\beta_2^{k+1}} + \epsilon}\,. \tag{(11)}$$

Omitting $\epsilon$ for simplicity we have:

$$d^{-1/2}\|\mathbf{b}_k\| = \frac{1-\beta_1^{k+1}}{1-\beta_1}\left(\frac{1}{1-\beta_2^{k+1}}\right)^{\frac{1}{2}} d^{-1/2}\|\sqrt{\mathbf{v}_k}\|. \tag{D.11}$$

Let us calculate $\|\sqrt{\mathbf{v}_k}\|$. Developing the recursion of $\mathbf{v}_k$, as defined in Eq. (9), leads to:

$$\mathbf{v}_k = (1-\beta_2)\sum_{i=0}^{k} \beta_2^{k-i}\left(\nabla\mathcal{L}(\mathbf{x}_i) + \lambda\mathbf{x}_i\right)^2, \tag{D.12}$$

$$\sqrt{\mathbf{v}_k} = \sqrt{1-\beta_2}\sqrt{\sum_{i=0}^{k} \beta_2^{k-i}\left(\nabla\mathcal{L}(\mathbf{x}_i) + \lambda\mathbf{x}_i\right)^2}, \tag{D.13}$$

where the square and the square-root are element-wise operations. Hence, if we take the square

norm:

$$
\begin{aligned}
\|\sqrt{\mathbf{v}_k}\|^2 &= (1-\beta_2)\sum_{j=1}^{d}\left(\sqrt{\sum_{i=0}^{k}\beta_2^{k-i}\left(\nabla\mathcal{L}(\mathbf{x}_i)+\lambda\mathbf{x}_i\right)^2}\right)_j^2 \\
&= (1-\beta_2)\sum_{j=1}^{d}\sum_{i=0}^{k}\beta_2^{k-i}\left(\nabla\mathcal{L}(\mathbf{x}_i)+\lambda\mathbf{x}_i\right)_j^2 \\
&= (1-\beta_2)\sum_{i=0}^{k}\beta_2^{k-i}\sum_{j=1}^{d}\left(\nabla\mathcal{L}(\mathbf{x}_i)+\lambda\mathbf{x}_i\right)_j^2 \\
&= (1-\beta_2)\sum_{i=0}^{k}\beta_2^{k-i}\|\nabla\mathcal{L}(\mathbf{x}_i)+\lambda\mathbf{x}_i\|^2, \qquad \text{(D.14)}
\end{aligned}
$$

where the $j$ subscript denotes the $j$-th element of the vector. It is exactly the order-2 moment of the gradient norm.

Therefore, we define the scalar $v_k$:

$$
v_k = \beta_2 v_{k-1} + (1-\beta_2)d^{-1}\|\nabla\mathcal{L}(\mathbf{x}_k)+\lambda\mathbf{x}_k\|^2, \qquad \text{(D.15)}
$$

which is the order-2 moment of the gradient norm with a factor $d^{-1}$. It verifies $\sqrt{v_k} = d^{-1/2}\|\sqrt{\mathbf{v}_k}\|$, needed for the scalar division stated in Eq. (D.11). By applying the bias correction, it gives the formula given in the paper of Adam w/o (a):

$$
\mathbf{x}_{k+1} = \mathbf{x}_k - \eta_k \frac{\mathbf{m}_k}{1-\beta_1^{k+1}} \Big/ \sqrt{\frac{v_k}{1-\beta_2^{k+1}}+\epsilon}, \qquad \text{(D.16)}
$$

$$
\mathbf{m}_k = \beta_1\mathbf{m}_{k-1} + (1-\beta_1)(\nabla\mathcal{L}(\mathbf{x}_k)+\lambda\mathbf{x}_k), \qquad \text{(D.17)}
$$

$$
v_k = \beta_2 v_{k-1} + (1-\beta_2)d^{-1}\|\nabla\mathcal{L}(\mathbf{x}_k)+\lambda\mathbf{x}_k\|^2. \qquad \text{(D.18)}
$$

Note that the previous demonstration makes the factor $d^{-1}$ appear in $v_k$ to have exactly the scheduling effect of Adam without the deformation.

**Clarification on Adam without deformed gradients and no additional radial terms (ab).** We introduce the rescaling and transport transformation of the momentum to neutralize the identified effects on the effective direction (cf. Section 4.2). The resulting, new $\mathbf{c}_k$ is orthogonal to $\mathbf{u}_k$ and does not contribute in the effective learning rate tuning with its radial part.

To avoid gradient history leaving the tangent space and thus neutralize **(b)**, we perform a parallel transport of the momentum $\mathbf{a}_{k-1}$ from the corresponding point on the sphere $\mathbf{u}_{k-1}$ to the new point $\mathbf{u}_k$ denoted as $\Gamma_{\mathbf{u}_{k-1}}^{\mathbf{u}_k}(\mathbf{a}_{k-1})$ at each iteration $k \geq 1$. Figure 3(c) illustrates the transport of a gradient. The parallel transport between two points associates each vector of the tangent space of the first point to a vector of the second tangent space by preserving the scalar product with the derivatives along the geodesic. Consequently, the gradients accumulated in the resulting momentum now lie in the tangent space of $\mathbf{u}_k$ at each step. This neutralizes the additional radial terms phenomena from $\mathbf{c}_k^{\text{grad}}$. Since $\mathbf{u}_{k-1}$, $\mathbf{u}_k$ and $\mathbf{a}_k$ are coplanar, the transport of the momentum on the hypersphere can be expressed as a rotation:

$$
\mathsf{T}(\mathbf{a}_{k-1}) \stackrel{\text{def}}{=} \Gamma_{\mathbf{u}_{k-1}}^{\mathbf{u}_k}(\mathbf{a}_{k-1}) = \langle\mathbf{u}_{k-1},\mathbf{u}_k\rangle\mathbf{a}_{k-1} - \langle\mathbf{a}_{k-1},\mathbf{u}_k\rangle\mathbf{u}_{k-1}, \qquad \text{(D.19)}
$$

$$
\mathbf{a}_k = \beta\mathsf{T}(\mathbf{a}_{k-1}) + \nabla\mathcal{L}(\mathbf{x}_k) + \lambda\mathbf{x}_k. \qquad \text{(D.20)}
$$

Although the transport operation is strictly defined on the tangent space only, the scalar product formulation enables its extension to the whole space. The transformation is linear and $\mathsf{T}(\mathbf{u}_{k-1}) = 0$. We thus have:

$$\mathsf{T}(\mathbf{a}_{k-1} - \lambda \mathbf{u}_{k-1}) = \mathsf{T}(\mathbf{a}_{k-1}). \tag{D.21}$$

In the previous formulation, we see that the $L_2$ component is not transported and does not contribute in the new momentum. Finally, the momentum only contains the contribution of the current $L_2$ regularization. This means that the $\mathsf{RT}$ transformation decouples the $L_2$ regularization and thus neutralizes the additional radial terms from $\mathbf{c}_k^{L_2}$.

**Clarification on Adam without deformed gradients, no additional radial terms and no radius ratio (abc).** To avoid the ratio $\frac{r_k}{r_i}$ in the effective learning direction and thus to cancel (**c**), we rescale the momentum in the update by the factor $\frac{r_{k-1}}{r_k}$ at each iteration $k \geq 1$. From Lemma. 1, we obtain:

$$\mathsf{R}(\mathbf{a}_{k-1}) \stackrel{\text{def}}{=} \frac{r_{k-1}}{r_k}\mathbf{a}_{k-1} \tag{D.22}$$

$$\mathbf{a}_k = \beta \mathsf{R}(\mathbf{a}_{k-1}) + \nabla \mathcal{L}(\mathbf{x}_k) + \lambda \mathbf{x}_k \tag{D.23}$$

$$= \frac{1}{r_k}\Big(\sum_{i=0}^{k} \beta^{k-i}(\nabla \mathcal{L}(\mathbf{u}_k) + \lambda r_k r_i \mathbf{u}_i)\Big). \tag{D.24}$$

Note that now, the factor $\frac{r_k}{r_i}$ is not contained anymore in the gradient contribution of $\mathbf{c}_k = r_k \mathbf{a}_k$, which neutralizes the radius ratio phenomenon.

We can note that $\mathsf{R}$ and $\mathsf{T}$ are commutative and that we can combine them in a simple concise scalar expression:

$$\mathsf{RT}(\mathbf{a}_{k-1}) \stackrel{\text{def}}{=} \frac{\langle \mathbf{x}_k, \mathbf{x}_{k-1}\rangle \mathbf{a}_{k-1} - \langle \mathbf{x}_k, \mathbf{a}_{k-1}\rangle \mathbf{x}_{k-1}}{\langle \mathbf{x}_k, \mathbf{x}_k\rangle}, \tag{D.25}$$

$$\mathbf{a}_k = \beta \mathsf{RT}(\mathbf{a}_{k-1}) + \nabla \mathcal{L}(\mathbf{x}_k) + \lambda \mathbf{x}_k. \tag{D.26}$$

This new momentum leads to $\mathbf{c}_k = \mathbf{c}_k^{\mathsf{RT}} + r_k^2 \lambda \mathbf{u}_k$ with $\langle \mathbf{c}_k, \mathbf{u}_k\rangle = \lambda r_k^2$ and $\mathbf{c}_k^{\perp} = \mathbf{c}_k^{\mathsf{RT}}$. The latter relies only on the trajectory on the hypersphere and always lies in the tangent space:

$$\mathbf{c}_k^{\mathsf{RT}} = \beta \Gamma_{\mathbf{u}_{k-1}}^{\mathbf{u}_k}(\mathbf{c}_{k-1}^{\mathsf{RT}}) + \nabla \mathcal{L}(\mathbf{u}_k). \tag{D.27}$$

The final Adam w/o (abc) scheme reads:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \eta_k \frac{\mathbf{m}_k}{1-\beta_1^{k+1}} \Big/ \sqrt{\frac{v_k}{1-\beta_2^{k+1}} + \epsilon}, \tag{D.28}$$

$$\mathbf{m}_k = \beta_1 \mathsf{RT}(\mathbf{m}_{k-1}) + (1-\beta_1)(\nabla \mathcal{L}(\mathbf{x}_k) + \lambda \mathbf{x}_k), \tag{D.29}$$

$$v_k = \beta_2 \frac{r_{k-1}^2}{r_k^2} v_{k-1} + (1-\beta_2) d^{-1} \|\nabla \mathcal{L}(\mathbf{x}_k) + \lambda \mathbf{x}_k\|^2. \tag{D.30}$$

We also rescale the introduced scalar $v_k$ at each step with the factor $\frac{r_{k-1}^2}{r_k^2}$. This removes the radius from the gradient contribution of the scheduling $\nu^{\mathsf{R}} = r_k v_k$, in contrast with $\nu_k$ from Eq. (C.16). The new scheduling effect reads:

$$\boxed{\nu_k^{\mathsf{R}} = d^{-1/2} \frac{1-\beta_1^{k+1}}{1-\beta_1} \sqrt{\frac{1-\beta_2}{1-\beta_2^{k+1}}} \Big(\sum_{i=0}^{k} \beta_2^{k-i} \|\nabla \mathcal{L}(\mathbf{u}_i) + \lambda r_i r_k \mathbf{u}_i\|^2\Big)^{1/2}}.$$

**Algorithm 1 Adam w/o (abc)** and its algorithm illustrated for a filter $\mathbf{x} \in \mathbb{R}^d$ followed by BN. Steps that are different from Adam are shown in highlight. For non-convolutional layers we use standard Adam.

**Require:** $\beta_1, \beta_2 \in [0, 1)$; $\lambda, \eta \in \mathbb{R}$; $\mathcal{L}(\mathbf{x})$

**initialize** step $k \leftarrow -1$; $\mathbf{m}_k \leftarrow 0$; $v_k \leftarrow 0$; $\mathbf{x} \in \mathbb{R}^d$

**while** *stopping criterion not met* **do**

  $k \leftarrow k + 1$

  $\mathbf{g} \leftarrow \nabla\mathcal{L}(\mathbf{x}_k) + \lambda\mathbf{x}_k$

  $\mathbf{m}_k \leftarrow \beta_1\mathbf{m}_{k-1} + (1 - \beta_1)\mathbf{g}$

  $v_k \leftarrow \beta_2 v_{k-1} + (1 - \beta_2)d^{-1}\mathbf{g}^\top\mathbf{g}$

  $\hat{\mathbf{m}} \leftarrow \mathbf{m}_k/(1 - \beta_1^{k+1})$

  $\hat{v} \leftarrow v_k/(1 - \beta_2^{k+1})$

  $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k - \eta\hat{\mathbf{m}}/(\sqrt{\hat{v}} + \epsilon)$

  $\mathbf{m}_k \leftarrow \mathbf{m}_k(\mathbf{x}_{k+1}^\top\mathbf{x}_k\mathbf{m}_k - \mathbf{m}_k^\top\mathbf{x}_{k+1}\mathbf{x}_k)/(\mathbf{x}_{k+1}^\top\mathbf{x}_{k+1})$

  $v_k \leftarrow v_k(\mathbf{x}_k^\top\mathbf{x}_k/\mathbf{x}_{k+1}^\top\mathbf{x}_{k+1})$

**return** resulting parameters $\mathbf{x}_k$

*Appendix D.3. Training and implementation details*

To assess empirically the significance of the above phenomena in the context of CNNs with BN and BN w/o affine, we evaluate the different variants of AdamW, AdamG, Adam w/o (a), w/o (ab), w/o (abc) over a variety of datasets and architectures.

Note that the set of parameters $\boldsymbol{\theta}$ of a CNN with NLs can be split in two disjoint subsets: $\boldsymbol{\theta} = \mathcal{F} \cup \mathcal{R}$, where $\mathcal{F}$ is the set of groups of radially-invariant parameters and $\mathcal{R}$ the remaining parameters. As demonstrated in Appendix A, the subset $\mathcal{F}$ includes parameters of all filters followed by BN. Since we are only interested in comparing optimization on $\mathcal{F}$, Adam variants w/o (a), w/o (ab), w/o (abc), AdamW AdamG are applied only to the optimization of the parameters in $\mathcal{F}$ whereas the ones in $\mathcal{R}$ are optimized with the original Adam scheme. The algorithm of Adam w/o (abc) is illustrated in Algorithm 1.

For each optimization scheme, each dataset and each architecture, the same grid search range and budget was performed while mini-batch size was fixed. We used a mini-batch size of 128 for SVHN, CIFAR10 and CIFAR100. The learning rates $\eta$ varied in $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$, the weight decay in $10^{-3} \cdot \{0, \frac{1}{128}, \frac{1}{64}, \frac{1}{32}, \frac{1}{32}, \frac{1}{16}, \frac{1}{8}, \frac{1}{4}\}$ (similar to [15]), the momentum was fixed to $0.9$ ($\beta_1$ for variants of Adam) and the order-two moment $\beta_2$ in $\{0.99, 0.999, 0.9999\}$ (as in [11]).

We used the same step-wise learning rate scheduler for each method. For SVHN, CIFAR10 and CIFAR100, models were trained for 405 epochs, and the learning rate multiplied by 0.1 at epochs 135, 225 and 315.

The optimization schemes introduced in this paper do not change the complexity in time of the algorithm. During the update of parameters in a layer, we only do a temporary copy of the parameter tensor just before the update to perform the RT transformation. This temporary copy is flushed after the RT transformation. Nothing permanent is stored in the optimizer.

Note that, for each architecture and each dataset, the same learning rate was systematically found for each method while the momentum factor was fixed at $0.9$ (cf. Table D.3). Best other hyperparameters, i.e., $L_2$ regularization and order-2 moment, are shown in Table D.4.

*Appendix D.4. Additional empirical results*

*Appendix D.4.1. Batch Normalization*

In this section, we observe the mean loss training curves associated to Adam, AdamW, AdamG, Adam w/o (a), Adam w/o (ab), Adam w/o (abc) on datasets CIFAR10, CIFAR100 and SVHN with architecture ResNet20, ResNet18 or VGG16 with BN, corresponding to the accuracies given in Table 2. The parameter setting are specified in Table D.4. These curves are illustrated in Figures D.9 D.10-D.11 D.11 D.12 D.13 D.14 D.15 D.16 D.17 D.18 D.19 D.20. The case of ResNet20 is illustrated in Figure 4 of the paper.

Table D.3: **Best learning rate and momentum factor.** We systematically found the same learning rate for each dataset and architecture while the momentum factor was fixed to 0.9.

| Method | $\eta_0$ | $\beta$, $\beta_1$ |
|---|---|---|
| Adam | 0.001 | 0.9 |
| AdamW | 0.001 | 0.9 |
| AdamG | 0.01 | 0.9 |
| Adam w/o (a) | 0.001 | 0.9 |
| Adam w/o (ab) | 0.001 | 0.9 |
| Adam w/o (abc) | 0.001 | 0.9 |

Table D.4: **Best $L_2$ regularization ($\lambda$) and order-2 moment factors ($\beta_2$).**

| Setup | | | Adam | AdamW | AdamG | Adam w/o (a) | Adam w/o (ab) | Adam w/o (abc) |
|---|---|---|---|---|---|---|---|---|
| CIFAR10 | ResNet20 | $\lambda \times 10^4$ | 2.5 | 5 | 1.25 | 0.31 | 1.25 | 5 |
| | | $\beta_2$ | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 |
| | ResNet18 | $\lambda \times 10^4$ | 2.5 | 0.08 | 0.63 | 2.5 | 1.25 | 0.16 |
| | | $\beta_2$ | 0.999 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 |
| | VGG16 | $\lambda \times 10^4$ | 2.5 | 0.31 | 2.5 | 0.63 | 0.00 | 0.31 |
| | | $\beta_2$ | 0.999 | 0.999 | 0.999 | 0.999 | 0.99 | 0.999 |
| CIFAR100 | ResNet18 | $\lambda \times 10^4$ | 1.25 | 1.25 | 1.25 | 1.25 | 1.25 | 0.00 |
| | | $\beta_2$ | 0.999 | 0.99 | 0.99 | 0.99 | 0.99 | 0.999 |
| | VGG16 | $\lambda \times 10^4$ | 0.63 | 0.16 | 0.63 | 0.63 | 1.25 | 0.08 |
| | | $\beta_2$ | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 |
| SVHN | ResNet18 | $\lambda \times 10^4$ | 0.00 | 0.08 | 5 | 0.31 | 5 | 0.08 |
| | | $\beta_2$ | 0.999 | 0.999 | 0.99 | 0.99 | 0.999 | 0.999 |
| | VGG16 | $\lambda \times 10^4$ | 0.00 | 0.31 | 5 | 0.08 | 2.5 | 2.5 |
| | | $\beta_2$ | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.999 |

*Appendix D.4.2. Batch Normalization without scaling and bias parameters*

In this last section, we observe the mean loss training curves associated to Adam, AdamW, AdamG, Adam w/o (a), Adam w/o (ab), Adam w/o (abc) on datasets CIFAR10, CIFAR100 and SVHN with architecture ResNet20, ResNet18 or VGG16 with BN w/o affine, corresponding to the accuracies given in Table D.5.

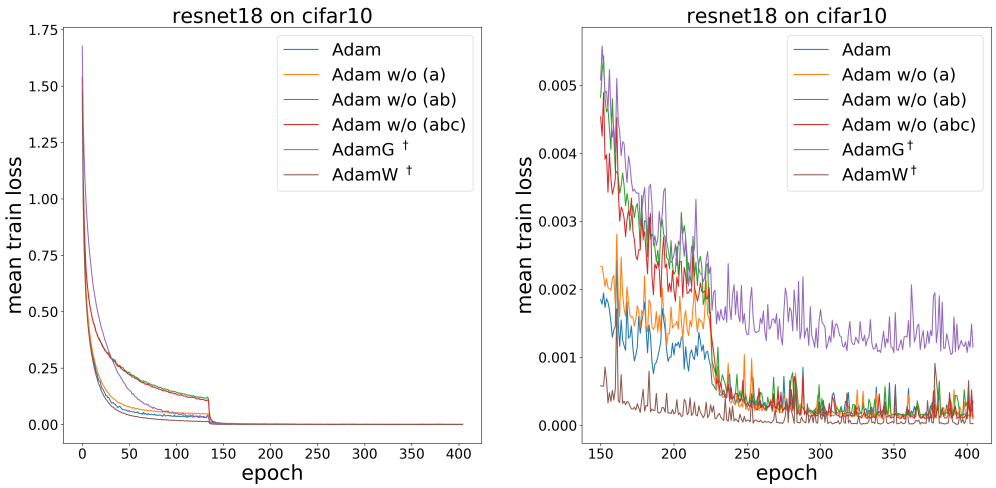



Figure D.9: **Training speed comparison with ResNet18 BN on CIFAR10.** *Left:* Mean training loss over all training epochs (averaged across 5 seeds) for different Adam variants. *Right:* Zoom-in on the last epochs. Please refer to Table 2 for the corresponding accuracies.

Table D.5: **Accuracy of Adam and its variants when training with BN w/o affine layers**. The figures in this table are the mean top1 accuracy $\pm$ the standard deviation over 5 seeds on the test set for CIFAR10, CIFAR100 and on the validation set for SVHN. $^\dagger$ indicates that the original method is only used on convolutional filters while Adam is used for other parameters.

| | CIFAR10 | | | CIFAR100 | | SVHN | |
|---|---|---|---|---|---|---|---|
| Method | ResNet20 | ResNet18 | VGG16 | ResNet18 | VGG16 | ResNet18 | VGG16 |
| Adam | 90.41 $\pm$ 0.06 | 93.67 $\pm$ 0.15 | 92.62 $\pm$ 0.15 | 71.60 $\pm$ 0.22 | 68.28 $\pm$ 0.19 | 95.29 $\pm$ 0.11 | 95.56 $\pm$ 0.18 |
| AdamW$^\dagger$ | 90.36 $\pm$ 0.11 | 93.7 $\pm$ 0.16 | 93.03 $\pm$ 0.12 | 70.11 $\pm$ 0.31 | 69.68 $\pm$ 0.12 | 89.83 $\pm$ 0.28 | 95.63 $\pm$ 0.11 |
| AdamG$^\dagger$ | 91.12 $\pm$ 0.09 | 93.62 $\pm$ 0.14 | 93.20 $\pm$ 0.20 | 69.96 $\pm$ 0.34 | 70.07 $\pm$ 0.23 | 95.12 $\pm$ 0.09 | 95.62 $\pm$ 0.21 |
| Adam w/o (a) | 91.15 $\pm$ 0.11 | 93.98 $\pm$ 0.18 | 93.12 $\pm$ 0.14 | 75.43 $\pm$ 0.13 | 70.01 $\pm$ 0.24 | 95.75 $\pm$ 0.09 | 95.64 $\pm$ 0.10 |
| Adam w/o (ab) | **91.38** $\pm$ 0.08 | **94.63** $\pm$ 0.08 | 93.45 $\pm$ 0.06 | **75.68** $\pm$ 0.22 | 71.74 $\pm$ 0.15 | **95.77** $\pm$ **0.08** | **95.78** $\pm$ 0.07 |
| Adam w/o (abc) | 91.11 $\pm$ 0.11 | 94.02 $\pm$ 0.10 | **93.56** $\pm$ 0.09 | 75.38 $\pm$ 0.21 | **72.08** $\pm$ 0.22 | 95.51 $\pm$ 0.08 | 95.69 $\pm$ 0.09 |

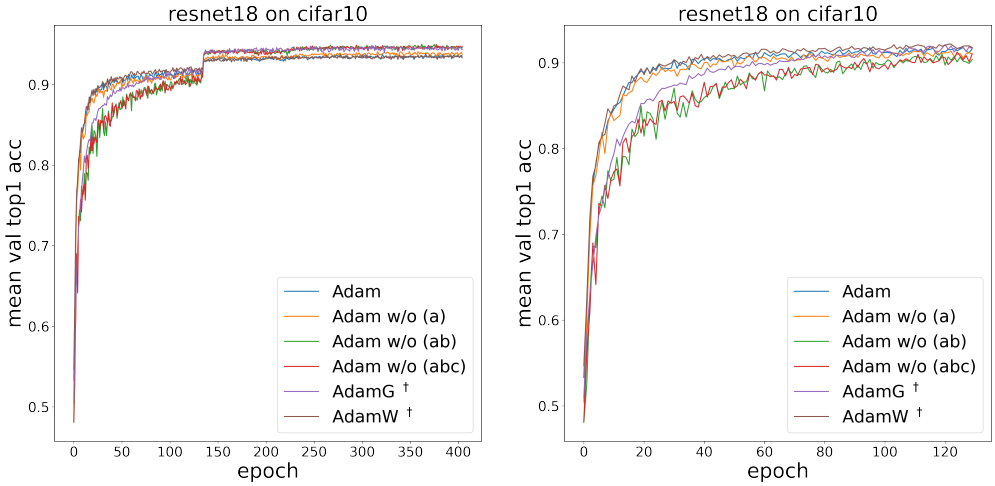



Figure D.10: **Accuracy comparison on the validation set with ResNet18 BN on CIFAR10.** *Left:* Mean training loss over all training epochs (averaged across 5 seeds) for different Adam variants. *Right:* Zoom-in on the first epochs. Please refer to Table 2 for the corresponding accuracies.

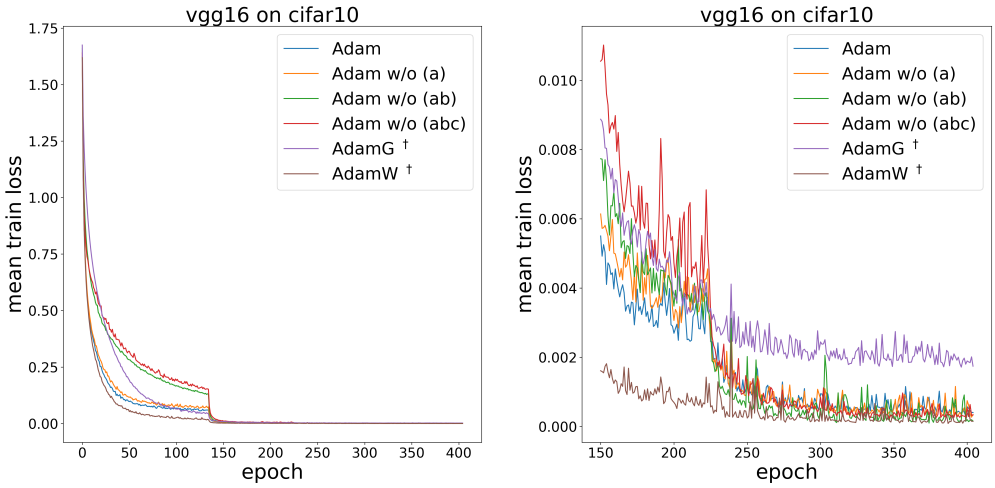



Figure D.11: **Training speed comparison with VGG16 on CIFAR10.** *Left:* Mean accuracy on the validation set over all training epochs (averaged across 5 seeds) for different Adam variants. *Right:* Zoom-in on the last epochs. Please refer to Table 2 for the corresponding accuracies.

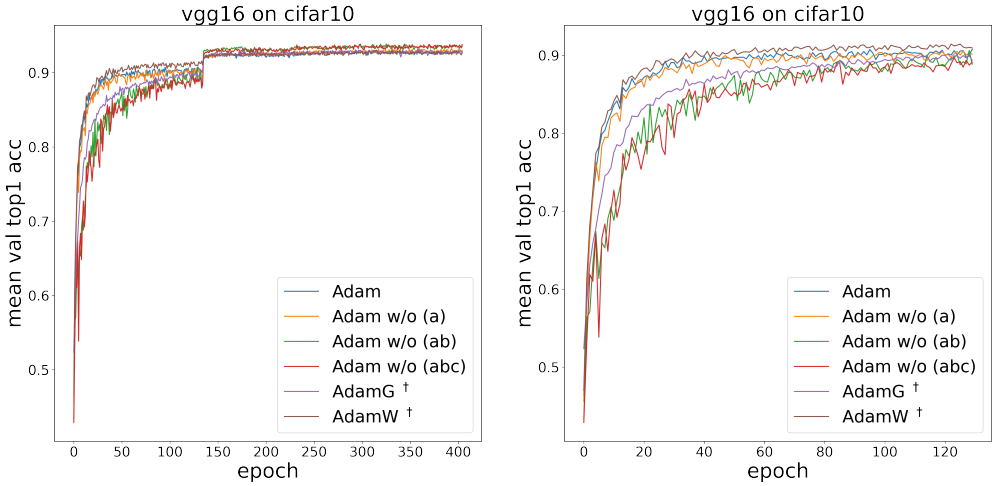



Figure D.12: **Accuracy comparison on the validation set with VGG16 BN on CIFAR10.** *Left:* Mean training loss over all training epochs (averaged across 5 seeds) for different Adam variants. *Right:* Zoom-in on the last epochs. Please refer to Table 2 for the corresponding accuracies.

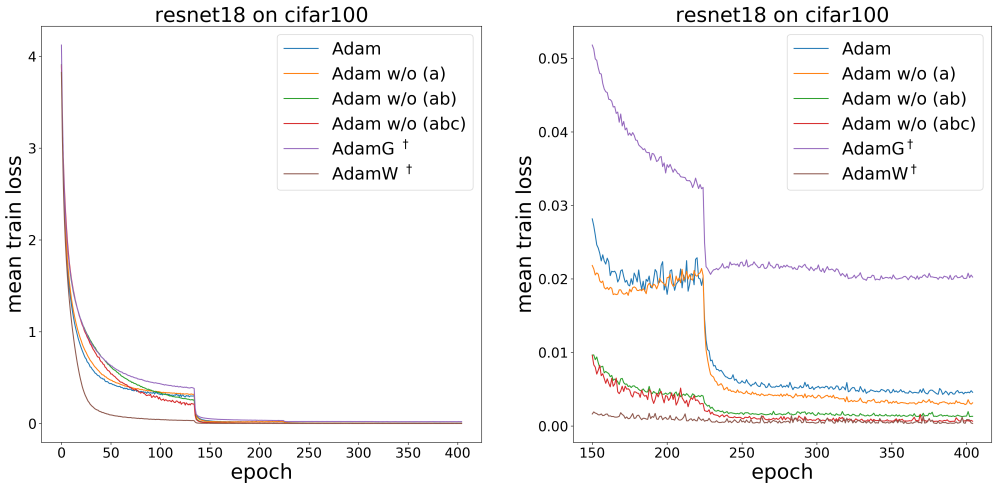



Figure D.13: **Training speed comparison with ResNet18 on CIFAR100.** *Left:* Mean training loss over all training epochs (averaged across 5 seeds) for different Adam variants. *Right:* Zoom-in on the last epochs. Please refer to Table 2 for the corresponding accuracies.

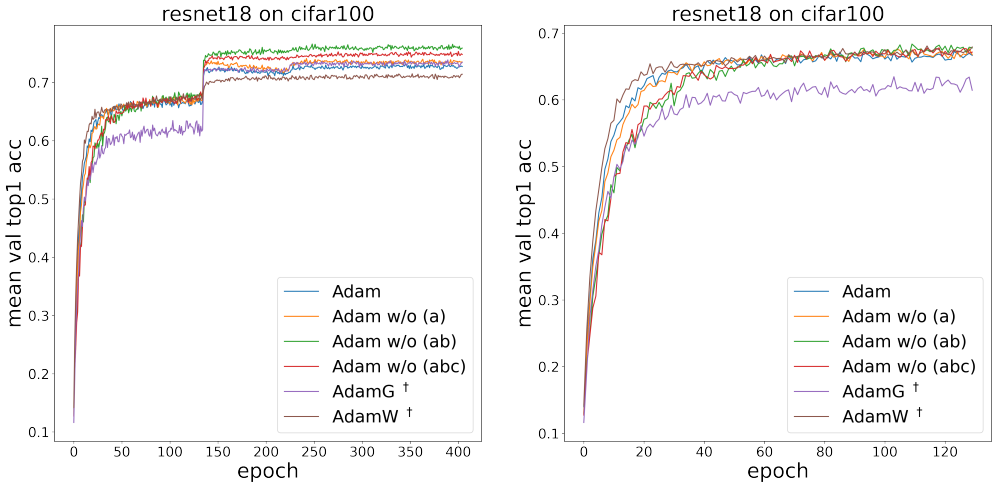



Figure D.14: **Accuracy comparison on the validation set with ResNet18 BN on CIFAR100.** *Left:* Mean training loss over all training epochs (averaged across 5 seeds) for different Adam variants. *Right:* Zoom-in on the last epochs. Please refer to Table 2 for the corresponding accuracies.

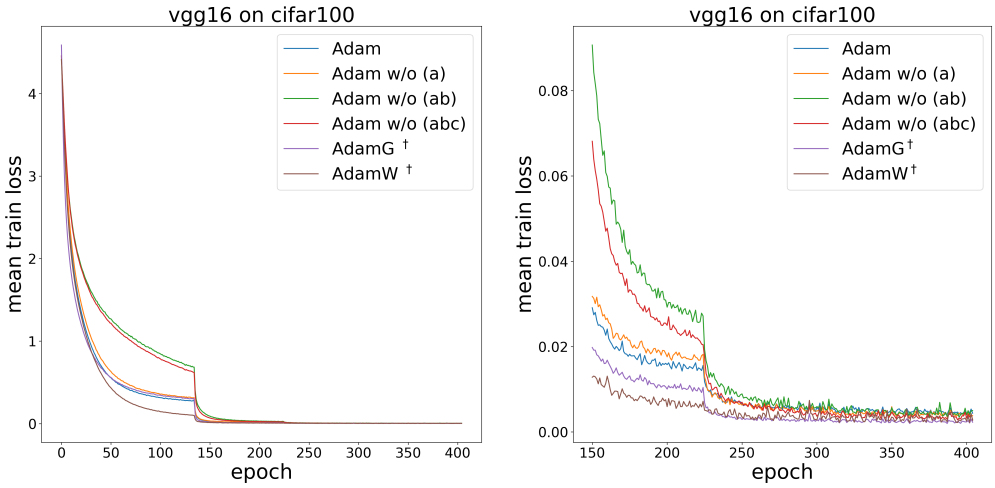



Figure D.15: **Training speed comparison with VGG16 on CIFAR100.** *Left:* Mean training loss over all training epochs (averaged across 5 seeds) for different Adam variants. *Right:* Zoom-in on the last epochs. Please refer to Table 2 for the corresponding accuracies.

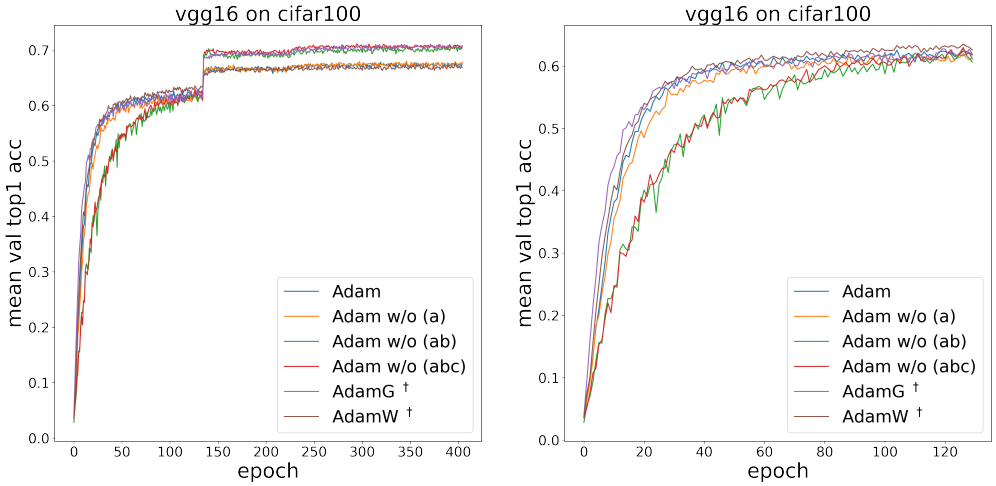



Figure D.16: **Accuracy comparison on the validation set with VGG16 BN on CIFAR100.** *Left:* Mean training loss over all training epochs (averaged across 5 seeds) for different Adam variants. *Right:* Zoom-in on the last epochs. Please refer to Table 2 for the corresponding accuracies.

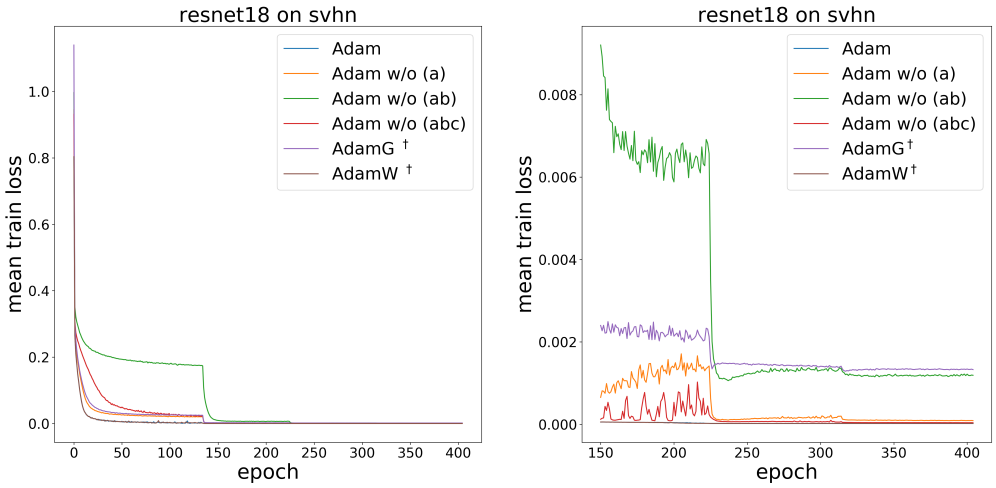



Figure D.17: **Training speed comparison with ResNet18 on SVHN.** *Left:* Mean training loss over all training epochs (averaged across 5 seeds) for different Adam variants. *Right:* Zoom-in on the last epochs. Please refer to Table 2 for the corresponding accuracies.

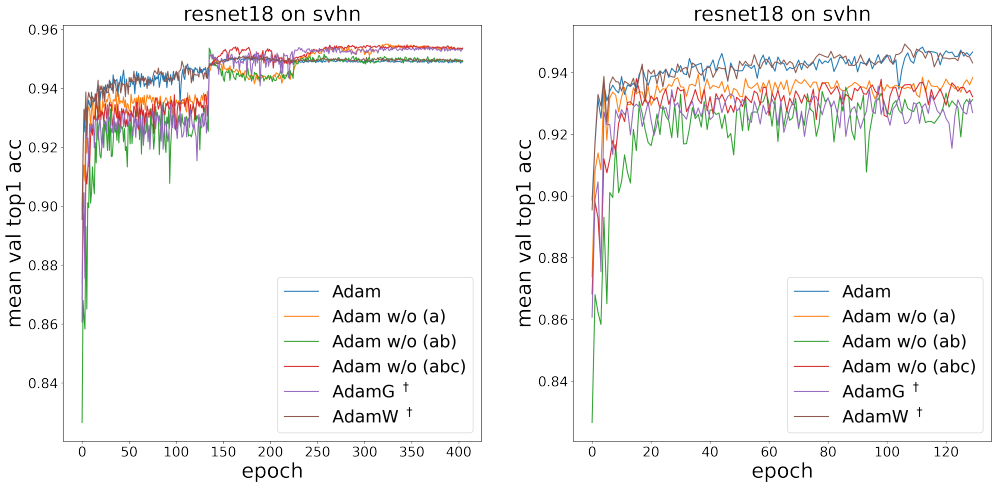



Figure D.18: **Accuracy comparison on the validation set with ResNet18 BN on SVHN.** *Left:* Mean training loss over all training epochs (averaged across 5 seeds) for different Adam variants. *Right:* Zoom-in on the last epochs. Please refer to Table 2 for the corresponding accuracies.

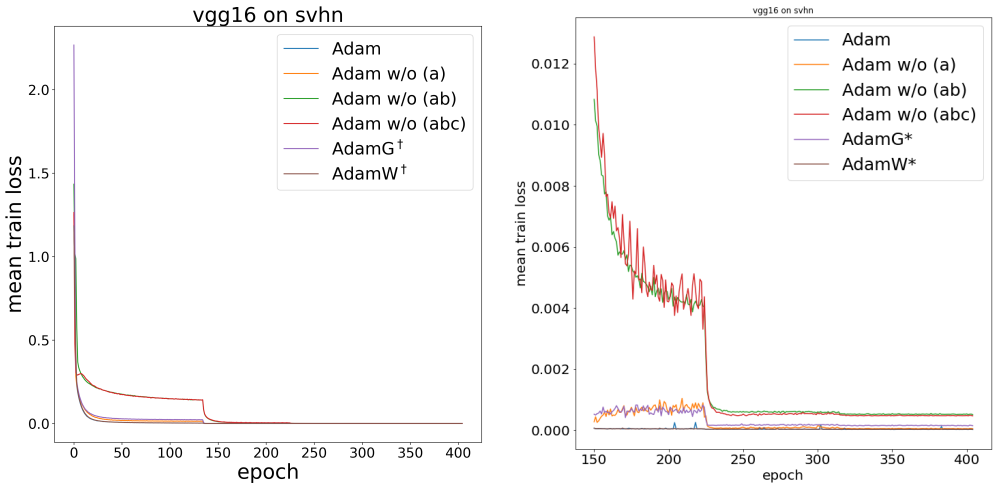



Figure D.19: **Training speed comparison with VGG16 on SVHN.** *Left:* Mean training loss over all training epochs (averaged across 5 seeds) for different Adam variants. *Right:* Zoom-in on the last epochs. Please refer to Table 2 for the corresponding accuracies.

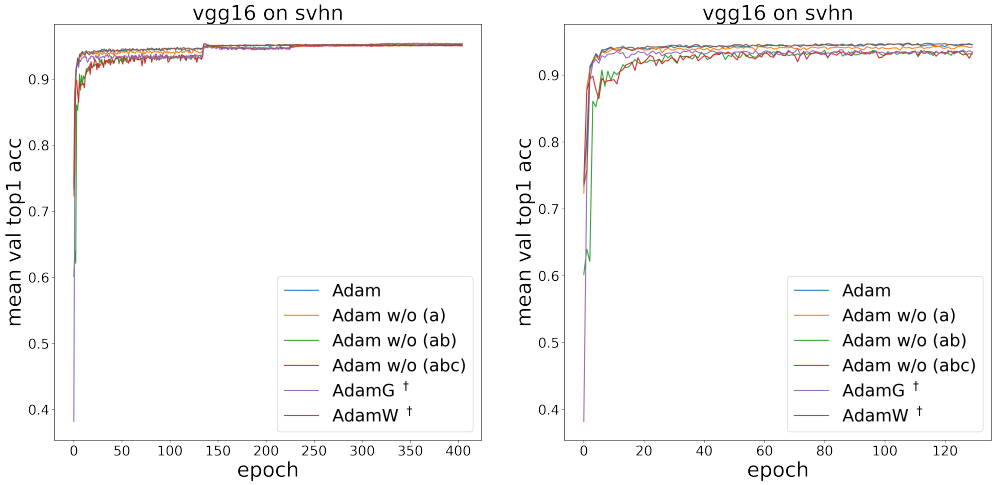



Figure D.20: **Accuracy comparison on the validation set with VGG16 BN on SVHN.** *Left:* Mean training loss over all training epochs (averaged across 5 seeds) for different Adam variants. *Right:* Zoom-in on the last epochs. Please refer to Table 2 for the corresponding accuracies.