Cheap Orthogonal Constraints in Neural Networks: A Simple Parametrization of the Orthogonal and Unitary Group

Mario Lezcano-Casado [1] **David Martínez-Rubio** [2]

Abstract

We introduce a novel approach to perform first-order optimization with orthogonal and unitary constraints. This approach is based on a parametrization stemming from Lie group theory through the *exponential map*. The parametrization transforms the constrained optimization problem into an unconstrained one over a Euclidean space, for which common first-order optimization methods can be used. The theoretical results presented are general enough to cover the special orthogonal group, the unitary group and, in general, any connected compact Lie group. We discuss how this and other parametrizations can be computed efficiently through an implementation trick, making numerically complex parametrizations usable at a negligible runtime cost in neural networks. In particular, we apply our results to RNNs with orthogonal recurrent weights, yielding a new architecture called EXPRNN. We demonstrate how our method constitutes a more robust approach to optimization with orthogonal constraints, showing faster, accurate, and more stable convergence in several tasks designed to test RNNs. [1]

1. Introduction

Training deep neural networks presents many difficulties. One of the most important is the exploding and vanishing gradient problem, as first observed and studied in (Bengio et al., 1994). This problem arises from the ill-conditioning of the function defined by a neural network as the number of layers increase. This issue is particularly problematic in Recurrent Neural Networks (RNNs). In RNNs the eigenvalues of the gradient of the recurrent kernel explode or vanish exponentially fast with the number of time-steps whenever the recurrent kernel does not have unitary eigenvalues (Arjovsky et al., 2016). This behavior is the same as the one encountered when computing the powers of a matrix, and results in very slow convergence (vanishing gradient) or a lack of convergence (exploding gradient).

In the seminal paper (Arjovsky et al., 2016), they note that unitary matrices have properties that would solve the exploding and vanishing gradient problems. These matrices form a group called the *unitary group* and they have been studied extensively in the fields of Lie group theory and Riemannian geometry. Optimization methods over the unitary and orthogonal group have found rather fruitful applications in RNNs in recent years (*cf.* Section 2).

In parallel to the work on unitary RNNs, there has been an increasing interest for optimization over the orthogonal group and the Stiefel manifold in neural networks (Harandi & Fernando, 2016; Ozay & Okatani, 2016; Huang et al., 2017; Bansal et al., 2018). As shown in these papers, orthogonal constraints in linear and CNN layers can be rather beneficial for the generalization of the network as they act as a form of implicit regularization. The main problem encountered while using these methods in practice was that optimization with orthogonality constraints was neither simple nor computationally cheap. We aim to close that bridge.

In this paper we present a simple yet effective way to approach problems that present orthogonality or unitary constraints. We build on results from Riemannian geometry and Lie group theory to introduce a parametrization of these groups, together with theoretical guarantees for it.

This parametrization has several advantages, both theoretical and practical:

1. It can be used with general purpose optimizers.
2. The parametrization does not create additional minima or saddle points in the main parametrization region.
3. It is possible to use a structured initializer to take advantage of the structure of the eigenvalues of the orthogonal matrix.

[1]Mathematical Institute, University of Oxford, Oxford, United Kingdom [2]Department of Computer Science, University of Oxford, Oxford, United Kingdom. Correspondence to: Mario Lezcano Casado <mario.lezcanocasado@maths.ox.ac.uk>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

[1]Implementation can be found at `https://github.com/Lezcano/expRNN`

4. Other approaches need to enforce hard orthogonality constraints, ours does not.

Most previous approaches fail to satisfy one or many of these points. The parametrization in (Helfrich et al., 2018) and (Maduranga et al., 2018) comply with most of these points but they suffer degeneracies that ours solves (*cf.* Remark in Section 4.2). We compare our architecture with other methods to optimize over $\mathrm{SO}(n)$ and $\mathrm{U}(n)$ in the remarks in Sections 3 and 4.

High-level idea The matrix exponential maps skew-symmetric matrices to orthogonal matrices transforming an optimization problem with orthogonal constraints into an unconstrained one. We use Padé approximants and the scale-squaring trick to compute machine-precision approximations of the matrix exponential and its gradient. We can implement the parametrization with negligible overhead observing that it does not depend on the batch size.

Structure of the Paper In Section 3, we introduce the parametrization and present the theoretical results that support the efficiency of the exponential parametrization. In Section 4, we explain the implementation details of the layer. Finally, in Section 5, we present the numerical experiments confirming the numerical advantages of this parametrization.

2. Related Work

Riemannian gradient descent. There is a vast literature on optimization methods on Riemannian manifolds, and in particular for matrix manifolds, both in the deterministic and the stochastic setting. Most of the classical convergence results from the Euclidean setting have been adapted to the Riemannian one (Absil et al., 2009; Bonnabel, 2013; Boumal et al., 2016; Zhang et al., 2016; Sato et al., 2017). On the other hand, the problem of adapting popular optimization algorithms like RMSPROP (Tieleman & Hinton, 2012), ADAM (Kingma & Ba, 2014) or ADAGRAD (Duchi et al., 2011) is a topic of current research (Kumar Roy et al., 2018; Becigneul & Ganea, 2019).

Optimization over the Orthogonal and Unitary groups. The first formal study of optimization methods on manifolds with orthogonal constraints (Stiefel manifolds) is found in the thesis (Smith, 1993). These ideas were later simplified in the seminal paper (Edelman et al., 1998), where they were generalized to Grassmannian manifolds and extended to get the formulation of the conjugate gradient algorithm and the Newton method for these manifolds. After that, optimization with orthogonal constraints has been a central topic of study in the optimization community. A rather in depth literature review of existing methods for optimization with orthogonality constraints can be found in (Jiang & Dai, 2015). When it comes to the unitary case, the algorithms used in practice are similar to those used in the real case, *cf.* (Manton, 2002; Abrudan et al., 2008).

Unitary RNNs. The idea of parametrizing the matrix that defines an RNN by a unitary matrix was first proposed in (Arjovsky et al., 2016). Their parametrization centers on a matrix-based fast Fourier transform-like (FFT) approach. As pointed out in (Jing et al., 2017), this representation, although efficient in memory, does not span the whole space of unitary matrices, giving the model reduced expressiveness. This second paper solves this issue in the same way it is solved when computing the FFT—using $\log(n)$ iterated butterfly operations. A different approach to perform this optimization was presented in (Wisdom et al., 2016; Vorontsov et al., 2017). Although not mentioned explicitly in either of the papers, this second approach consists of a retraction-based Riemannian gradient descent via the Cayley transform. The paper (Hyland & Rätsch, 2017) proposes to use the exponential map on the complex case, but they do not perform an analysis of the algorithm or provide a way to approximate the map nor the gradients. A third approach has been presented in (Mhammedi et al., 2017) via the use of Householder reflections. Finally, in (Helfrich et al., 2018) and the follow-up (Maduranga et al., 2018), a parametrization of the orthogonal group via the use of the Cayley transform is proposed. We will have a closer look at these methods and their properties in Sections 3 and 4.

3. Parametrization of Compact Lie Groups

For a much broader introduction to Riemannian geometry and Lie group theory see Appendix A. We will restrict our attention to the special orthogonal [2] and unitary case, but the results in this section can be generalized to any connected compact matrix Lie group equipped with a bi-invariant metric. We prove the results for general connected compact matrix Lie groups in Appendix C.

3.1. The Lie algebras of $\mathrm{SO}(n)$ and $\mathrm{U}(n)$

We are interested in the study of parametrizations of the special orthogonal group

$$\mathrm{SO}(n) = \{B \in \mathbb{R}^{n\times n} \mid B^\intercal B = \mathrm{I},\, \det(B) = 1\}$$

and the unitary group

$$\mathrm{U}(n) = \{B \in \mathbb{C}^{n\times n} \mid B^* B = \mathrm{I}\}.$$

These two sets are compact and connected Lie groups. Furthermore, when seen as submanifolds of $\mathbb{R}^{n\times n}$ (resp. $\mathbb{C}^{n\times n}$) equipped with the metric induced from the ambient space

[2] Note that we consider just matrices with determinant equal to one, since the full group of orthogonal matrices $\mathrm{O}(n)$ is not connected, and hence, not amenable to gradient descent algorithms.

$\langle X, Y\rangle = \operatorname{tr}(X^\intercal Y)$ (resp. $\operatorname{tr}(X^* Y)$), they inherit a *bi-invariant metric*, meaning that the metric is invariant with respect to left and right multiplication by matrices of the group. This is clear given that the matrices of the two groups are isometries with respect to the metric on the ambient space.

We call the tangent space at the identity element of the group the *Lie algebra* of the group. For the two groups of interest, their Lie algebras are given by

$$\mathfrak{so}(n) = \{A \in \mathbb{R}^{n\times n} \mid A + A^\intercal = 0\},$$
$$\mathfrak{u}(n) = \{A \in \mathbb{C}^{n\times n} \mid A + A^* = 0\},$$

That is, the skew-symmetric and the skew-Hermitian matrices respectively. Note that these two spaces are isomorphic to a vector space. For example, for $\mathfrak{so}(n)$, the isomorphism is given by

$$\begin{aligned}\alpha\colon \mathbb{R}^{\frac{n(n-1)}{2}} &\to \mathfrak{so}(n)\\ A &\mapsto A - A^\intercal\end{aligned}$$

where we identify $A \in \mathbb{R}^{\frac{n(n-1)}{2}}$ with an upper triangular matrix with zeros in the diagonal.

3.2. Parametrizing $\mathrm{SO}(n)$ and $\mathrm{U}(n)$

In the theory of Lie groups there exists a tight connection between the structure of the Lie algebra and the geometry of the Lie group. One of the most important tools that is used to study one in terms of the other is the *Lie exponential map*. The Lie exponential map on matrix Lie groups with a bi-invariant metric is given by the exponential of matrices. If we denote the group by G (which would be $\mathrm{SO}(n)$ or $\mathrm{U}(n)$ in this case) and its Lie algebra by $\mathfrak{g}$, we have the mapping $\exp : \mathfrak{g} \to G$ defined as

$$\exp(A) := \mathrm{I} + A + \tfrac{1}{2}A^2 + \cdots$$

This mapping is not surjective in general. On the other hand, there are particular families of Lie groups in which the exponential map is, in fact, surjective. Compact Lie groups are one of such families.

Theorem 3.1. *The Lie exponential map on a connected, compact Lie group is surjective.*

Proof. We give a short self-contained proof of this classical result in Appendix C. We give an alternative, less abstract proof of this fact for the groups $\mathrm{SO}(n)$ and $\mathrm{U}(n)$ as a corollary of Proposition 3.2 in Appendix D. □

Both $\mathrm{SO}(n)$ and $\mathrm{U}(n)$ are compact and connected, so this result applies to them. As such, the exponential of matrices gives a complete parametrization of these groups.

3.3. From Riemannian to Euclidean optimization

In this section we describe some properties of the exponential parametrization which make it a sound choice for optimization with orthogonal constraints in neural networks.

Fix G to be $\mathrm{SO}(n)$ or $\mathrm{U}(n)$ equipped with the metric[3] $\langle X, Y\rangle = \operatorname{tr}(X^* Y)$ and let $\mathfrak{g}$ be its Lie algebra (the space of skew-symmetric or skew-Hermitian matrices). The exponential parametrization satisfies the following properties.

It can be used with general purpose optimizers. The exponential parametrization allows us to *pullback* an optimization problem from the group G back to the Euclidean space. If we have a problem

$$\min_{B\in G} f(B) \tag{1}$$

this is equivalent to solving

$$\min_{A\in\mathfrak{g}} f(\exp(A)). \tag{2}$$

We noted in Section 3.1 that $\mathfrak{g}$ is isomorphic to a Euclidean vector space, and as such we can use regular gradient descent optimizers like ADAM or ADAGRAD to approximate a solution to problem (2).

A rather natural question to ask is whether using gradient-based methods to approximate the solution of problem (2) would give a sensible solution to problem (1), given that precomposing with the exponential map might change the geometry of the problem. If the parametrization is, for example not locally unique, this might degrade the gradient flow and affect the performance of the gradient descent algorithm. In this section we will show theoretically that this parametrization has rather desirable properties for a parametrization of a manifold. We will confirm that these properties have a positive effect on the convergence of the gradient descent algorithms when compared with other parametrizations when applied to real problems in Section 5.

It does not change the minimization problem. It is clear that a minimizer $\widehat{B}$ for problem (1) and a minimizer $\widehat{A}$ for problem (2) will be related by the equation $\widehat{B} = \exp(\widehat{A})$, since the exponential map is surjective, so if we find a solution to the second problem we will have a solution to the first one.

It acts as a change of metric on the group. If the parametrization did not induce a change of metric on the manifold it could mean that it would induce saddle points, which would potentially slow down the convergence of the optimization algorithm.

A map $\phi\colon \mathcal{M} \to \mathcal{N}$ with $\mathcal{N}$ a Riemannian manifold induces a metric on a differentiable manifold $\mathcal{M}$ whenever it is an

[3] Note that in the real case we have that $A^* = A^\intercal$.

immersion, that is, its differential is injective. The Lie exponential is not just an immersion, it is bi-analytic on an open neighborhood around the origin. The image of this neighborhood is sufficiently large to cover almost all the Lie group.

Proposition 3.2. *Let G be $\mathrm{SO}(n)$ or $\mathrm{U}(n)$. The exponential map is analytic, invertible, with analytic inverse on a bounded open neigborhood V of the origin and $\exp(V)$ covers almost all G in the sense that the whole group lies in the closure of $\exp(V)$.*

Proof. See Appendix D. □

This proposition assures that, as long as the optimization problem stays in the neighborhood V, the representation of the matrices in G is unique, so this parametrization is not creating spurious minima. Furthermore, given that it is a diffemorphism, it is not creating saddle points on V either. Additionally, on this neighborhood, we have the adjoint of $\mathrm{d}\exp$ with respect to the metric, that is,

$$\langle \mathrm{d}\exp(X), Y\rangle = \langle X, \mathrm{d}\exp^*(Y)\rangle.$$

This is the map that induces the new metric on G, through the pushforward of the canonical metric from the Lie algebra into the Lie group. As such, the optimization process using our parametrization can be seen as Riemannian gradient descent using this new metric, and all the existent results developed for optimization over manifolds apply to this setting.

Remark. We saw empirically that whenever the initialization of the skew-symmetric matrix starts in V, the optimization path throughout all the training epochs does not leave V. For this reason, in practice the exponential parametrization behaves as a change of metric on the Lie group.

The induced metric is different to the classic one. The standard first order optimization technique to solve problem (1) is given by Riemannian gradient descent (Absil et al., 2009). In the Riemannian setting, we have the *Riemannian exponential map* $\exp_B$ which maps lines that pass through the origin on the tangent space T_BG to geodesics on G that pass through B. In the special orthogonal or unitary case, when we choose the metric induced by the canonical metric on the ambient space, for a function defined on the ambient space, this translates to the update rule

$$B \leftarrow B\exp(-\eta B^* \operatorname{grad} f(B))$$

for a learning rate $\eta > 0$, where $\exp$ is the exponential of matrices and $\operatorname{grad} f(B)$ denotes the gradient of the function restricted to G. We deduce this formula in Example C.1.

Computing the Riemannian exponential map exactly is computationally expensive in many practical situations. For this reason, approximations are in order. Retractions are of particular interest.

Definition 3.3 (Retraction). A retraction r for a manifold $\mathcal{M}$ is defined as a family of functions $r_x \colon T_x\mathcal{M} \to \mathcal{M}$ for every $x \in \mathcal{M}$ such that

$$r_x(0) = x \qquad \text{and} \qquad (\mathrm{d}r_x)_0 = \mathrm{Id}.$$

In other words, retractions are a first order approximation of the Riemannian exponential map. A study of the convergence properties of first and second-order optimization algorithms when using retractions can be found in (Boumal et al., 2016). In the case of G, we have that a way to form retractions is to choose a function $\phi \colon \mathfrak{g} \to G$ such that it is a first order approximation of the exponential of matrices and its image lies in G. Then, the update rule is given by

$$B \leftarrow B\phi(-\eta B^* \operatorname{grad} f(B)).$$

Remark. For the special orthogonal and the unitary group, one such function is the *Cayley map*

$$\phi(A) = (\mathrm{I} + \tfrac{1}{2}A)(\mathrm{I} - \tfrac{1}{2}A)^{-1}.$$

This justifies theoretically the optimization methods used in (Wisdom et al., 2016; Vorontsov et al., 2017) and extends their work, given that all their architectures can still be applied with different retractions for these manifolds. In Section 4.2 we give examples of more involved retractions, and in Section 4.3 we explain why it is computationally cheap to use machine-accuracy approximants to compute the exponential map both in our approach and in the Riemannian gradient descent approach. Examples of other retractions and a deeper treatment of these objects can be found in Appendix B.

The update rule for the exponential parametrization induces a retraction-like map for $A \in \mathfrak{g}$

$$e^A \leftarrow \exp(A - \eta\nabla(f \circ \exp)(A)),$$

where the gradient is the gradient with respect to the Euclidean metric, that is, the regular gradient, given that $f \circ \exp$ is defined on a Euclidean space. A natural question that arises is whether this new update rule defines a retraction. It turns out that this map is not a retraction for $\mathrm{SO}(n)$ or $\mathrm{U}(n)$.

Proposition 3.4. *The step-update map induced by the exponential parametrization is not a retraction for $\mathrm{SO}(n)$ if $n > 2$ nor for $\mathrm{U}(n)$ if $n > 1$.*

Proof. It is a corollary of Theorem C.12, where we give necessary and sufficient conditions for this map to be a retraction when defined on a compact, connected matrix Lie group. □

This tells us that the metric induced by the $\log$ map on $\mathrm{SO}(n)$ and $\mathrm{U}(n)$ is intrinsically different to the canonical metric on these manifolds when seen as submanifolds of $\mathbb{R}^{n\times n}$ (resp. $\mathbb{C}^{n\times n}$). In particular, it changes the geodesic flow defined by the metric.

4. Numerical Implementation

As an application of this framework we show how to model an orthogonal (or unitary) recurrent neural network with it, that is, an RNN whose recurrent matrix is orthogonal (or unitary). We also show how to implement numerically the ideas of the last section.

4.1. Exponential RNN Architecture

Given a sequence of inputs $(x_t) \subseteq \mathbb{R}^d$, we define an orthogonal exponential RNN (EXPRNN) with hidden size $p > 0$ as

$$h_{t+1} = \sigma(\exp(A)h_t + Tx_{t+1})$$

where $A \in \mathrm{Skew}(p)$, $T \in \mathbb{R}^{p\times d}$, and σ is some fixed nonlinearity. In our experiments we chose the `modrelu` nonlinearity, as introduced in (Arjovsky et al., 2016). Note that generalizing this architecture to the complex unitary case simply accounts for considering A to be skew-Hermitian rather than skew-symmetric. We stayed with the real case because we did not observe any improvement in the empirical results when using the complex case.

4.2. Approximating the exponential of matrices

There is a myriad of methods to approximate the exponential of a matrix (Moler & Van Loan, 2003). Riemannian gradient descent over $\mathrm{SO}(n)$ requires that the result of the approximation is orthogonal. If not, the error would accumulate after each step making the resulting matrix deviate from the orthogonality constraint exponentially fast in the number of steps. On the other hand, the approximation of the exponential in our parametrization does not require orthogonality. This allows many other approximations of the exponential function. The requirement is removed because the orthogonal matrix is implicitly represented as an exponential of a skew-symmetric matrix. The loss of orthogonality in Riemannian gradient descent is due to storing an orthogonal matrix and updating it directly.

Padé approximants. Padé approximants are rational approximations of the form $\exp(A) \approx p_n(A)q_n(A)^{-1}$ for polynomials p_n, q_n of degree n. A Padé approximant of degree n agrees with the Taylor expansion of the exponential to degree $2n$. The Cayley transform is the Padé approximant of degree 1. These methods and their implementations are described in detail in (Higham, 2009).

Scale-squaring trick. The error of the Padé approximant scales as $\mathcal{O}(\|A\|^{2n+1})$. If $\|A\| > 1$ and we have an approximant ϕ, the scale-squaring trick accounts for computing $\phi(\frac{A}{2^k})^{2^k}$ for the first $k \in \mathbb{N}$ such that $\frac{\|A\|}{2^k} < \frac{1}{2}$. Most types of approximants, like Padé's or a truncated Taylor expansion of the exponential, can be coupled with the scale-squaring trick to reduce the error (Higham, 2009).

Remark. Given that the Cayley transform is a degree 1 Padé approximant of the exponential, if we choose this approximant without the scale-squaring trick we essentially recover the parametrization proposed in (Helfrich et al., 2018). The Cayley transform suffers from the fact that, if the optimum has -1 as an eigenvalue, the weights of the corresponding skew-symmetric matrix will tend to infinity. The parametrization in (Helfrich et al., 2018) is the Cayley transform multiplied by a diagonal matrix D, but the parametrization still has the same problem, it just moves it to a different eigenvalue. Proposition 3.2 assures that the exponential parametrization does not suffer from this problem.

The follow-up work (Maduranga et al., 2018) mitigates this problem learning the diagonal of D as well, but by doing so it loses the local unicity of the parametrization. Proposition 3.2 assures that the exponential parametrization is not only locally unique, but also differentiable with differentiable inverse, thus inducing a metric.

Remark. It is straightforward to show that a degree n Padé approximant combined with the scaling-squaring trick also maps the skew-symmetric matrices to the special orthogonal matrices. This constitutes a much more precise retraction than the Cayley map at almost no extra computational cost. This observation can be used to improve the precision of the method proposed in (Wisdom et al., 2016; Vorontsov et al., 2017).

Exact approximation. Combining the methods above we can get an efficient approximation of the exponential to machine-precision. The best one known to the authors is based on the paper (Al-Mohy & Higham, 2009b). It accounts for an efficient use of the scaling-squaring trick and a Padé approximant. This is the algorithm that we use on the experiments section to approximate the exponential.

Exact gradients. A problem often encountered in practice is that of biased gradients. Even though an approximation might be good, it can significantly bias the gradient. An example of this would be trying to approximate the function $f \equiv 0$ on $[0, 1]$ by the functions $f_n(x) = \frac{\sin(2\pi nx)}{n}$. Even though $f_n \to f$ uniformly, their derivatives do not converge to zero. This problem is rather common when using involved parametrizations, for example, those coming from Chebyshev polynomials. On the other hand, the gradient can be implemented separately using a machine-precision formula.

Proposition 4.1. *Let $A \in \mathrm{Skew}(n)$. For a function $f\colon \mathbb{R}^{n\times n} \to \mathbb{R}$, denote the matrix $B = e^A$. We have that*

$$\nabla(f\circ\exp)(A) = B(\mathrm{d}\exp)_{-A}(\tfrac{1}{2}(\nabla f(B)^\intercal B - B^\intercal \nabla f(B))).$$

Proof. It follows from the discussion in Example C.1 and Proposition C.11 in the supplementary material. □

The differential of the exponential of matrices $(\mathrm{d}\exp)_A$ can be approximated to machine-precision computing the exponential of a $2n \times 2n$ matrix (Al-Mohy & Higham, 2009a). We use this algorithm in conjunction with Proposition 4.1 to implement the gradients.

4.3. Parametrizations are computationally cheap.

At first sight, one may think that an exact computation of the exponential and its gradient in neural networks is rather expensive. This is not the case when the exponential is just used as a parametrization. The value—and hence the gradient—of a parametrization does not depend on the training examples used to compute the stochastic gradient. For this reason, in order to compute the gradient of a function $\nabla(f \circ \phi)(A)$ with $B = \phi(A)$, we can first let the auto-differentiation engine compute the stochastic gradient of f with respect to B, that is, $\nabla f(B)$. The value $\nabla f(B)$ depends on the batch size and the number of appearances of B as a subexpression in the neural network (think of a recurrent kernel in an LSTM). We can use $\nabla f(B)$ to compute—just once per batch—the gradient $\nabla(f \circ \phi)(A)$, for example with the formula given in Proposition 4.1 for $\phi = \exp$. This allows the user to implement rather complex parametrizations, like the one we showed, without a noticeable runtime penalty. For instance, for an RNN with batch size b, sequences of average length ℓ, and a hidden size of n, in each iteration one needs to compute bln matrix-vector products at the cost of $O(n^2)$ operations each. The overhead incurred using the exponential parametrization is the computation of two matrix exponentials that run in $O(n^3)$, which is negligible in comparison. In practice, with an EXPRNN of size 512, we did not observe any noticeable time penalty when using this parametrization trick with respect to not imposing orthogonality constraints at all.

4.4. Intialization

For the initialization of the layer with a matrix $A_0 \in \mathrm{Skew}(p)$, we drew ideas from both (Henaff et al., 2016) and (Helfrich et al., 2018). Both initializations sample blocks of the form

$$\begin{pmatrix} 0 & s_i \\ -s_i & 0 \end{pmatrix}.$$

for s_i i.i.d. and then form A_0 as a block-diagonal matrix with these blocks.

The *Henaff* initialization consists of sampling $s_i \sim \mathcal{U}[-\pi, \pi]$. This defines a block-diagonal orthogonal matrix e^A with uniformly distributed blocks on the corresponding torus of block-diagonal 2×2 rotations. We sometimes found that the sampling presented in (Helfrich et al., 2018) performed better. This initialization, which we call *Cayley*, accounts for sampling $u_i \sim \mathcal{U}[0, \frac{\pi}{2}]$ and then setting $s_i = -\sqrt{\frac{1-\cos(u_i)}{1+\cos(u_i)}}$, thus biasing the eigenvalues towards 0.

We chose as the initial vector $h_0 = 0$ for simplicity, as we did not observe any empirical improvement when using the initialization given in (Arjovsky et al., 2016).

5. Experiments

In this section we compare the performance of our parametrization for orthogonal RNNs with the following approaches:

- Long short-term memory (LSTM). (Hochreiter & Schmidhuber, 1997)
- Unitary RNN (URNN). (Arjovsky et al., 2016)
- Efficient Unitary RNN (EURNN). (Jing et al., 2017)
- Cayley Parametrization (SCORNN). (Helfrich et al., 2018)
- Riemannian Gradient Descent (RGD). (Wisdom et al., 2016)

We use three tasks that have become standard to measure the performance of RNNs and their ability to deal with long-term dependencies. These are the copying memory task, the pixel-permuted MNIST task, and the speech prediction on the TIMIT dataset (Arjovsky et al., 2016; Wisdom et al., 2016; Henaff et al., 2016; Mhammedi et al., 2017; Helfrich et al., 2018).

In Appendix E, we enumerate the hyperparameters used for the experiments. The sizes of the hidden layer were chosen to match the number of learnable parameters of the other architectures.

Remark. We found empirically that having a learning rate for the orthogonal parameters that is 10 times larger than that of the non-orthogonal parameters yields a good performance in practice.

For the other experiments, we executed the code that the other authors provided with the best hyperparameters that they reported and a batch of 128. The results for EURNN are those reported in (Jing et al., 2017), and for RGD and URNN are those reported in (Helfrich et al., 2018).

The code with the exact configuration and seeds to replicate these results, and a plug-and-play implementation of EXPRNN and the exponential framework can be found in `https://github.com/Lezcano/expRNN`.

5.1. Copying memory task

The copying memory task was first proposed in (Hochreiter & Schmidhuber, 1997). The task can be defined as fol-

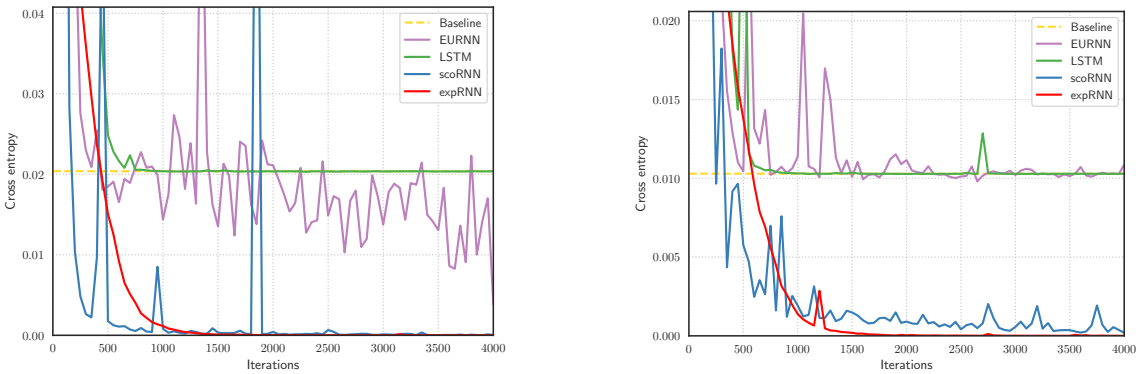

Figure 1. Cross entropy of the different algorithms in the copying problem for $L = 1000$ (left) and $L = 2000$ (right).

lows. Let $\mathcal{A} = \{a_k\}_{k=1}^{N}$ be an alphabet and let `<blank>`, `<start>` be two symbols not contained in $\mathcal{A}$. For a sequence length of K and a spacing of length L, the input sequence would be K ordered characters $(b_k)_{k=1}^{K}$ sampled i.i.d. uniformly at random from $\mathcal{A}$, followed by L repetitions of the character `<blank>`, the character `<start>` and finally $K - 1$ repetitions of the character `<blank>` again. The output for this sequence would be $K + L$ times the `<blank>` character and then the sequence of characters $(b_k)_{k=1}^{K}$. In other words, the system has to recall the initial K characters and reproduce them after detecting the input of the character `<start>`, which appears L time-steps after the end of the input characters. For example, for $N = 4$, $K = 5$, $L = 10$, if we represent `<blank>` with a dash and `<start>` with a colon, and the alphabet $\mathcal{A} = \{1, \dots, 4\}$, the following sequences could be an element of the dataset:

Input: `14221----------:----`
Output: `----------------14221`

The loss function for this task is the cross entropy. The standard baseline for this task is the output of $K + L$ `<blank>` symbols, followed by the remaining K symbols being output at random. This strategy yields a cross entropy of $K \log(N)/(L + 2K)$.

We observe that the training of SCORNN is unstable, which is probably due to the degeneracies explained in the remark in Section 4. In the follow-up paper (Maduranga et al., 2018), SCURNN presents the same instabilities as its predecessor. As explained in Section 4, EXPRNN does not suffer of this, and can be observed in our experiments as a smoother convergence. In the more difficult problem, $L = 2000$, EXPRNN is the only architecture that is able to fully converge to the correct answer.

5.2. Pixel-by-Pixel MNIST

In this task we use the MNIST dataset of hand-written numbers (LeCun & Cortes, 2010) of images of size 28×28, only this time the images are flattened and are processed as an array of 784 pixels, which is treated as a stream that is fed to the RNN, as described in (Le et al., 2015). In the unpermuted task, the stream is processed in a row-by-row fashion, while in the permuted task, a random permutation of the 784 elements is chosen at the beginning of the experiment, and all the pixels of all the images in the experiment are permuted according to this permutation. The final output of the RNN is processed as the encoding of the number and used to solve the corresponding classification task.

In this experiment we observed that EXPRNN is able to saturate the capacity of the orthogonal RNN model for this task much faster than any other parametrization, as per Table 1. We conjecture that coupling the exponential parametrization with an LSTM cell or a GRU cell would yield a superior architecture. We leave this for future research.

5.3. TIMIT Speech Dataset

We performed speech prediction on audio data with our model. We used the TIMIT speech dataset (S Garofolo et al., 1992) which is a collection of real-world speech recordings. The task accounts for predicting the log-magnitude of incoming frames of a short-time Fourier transform (STFT) as it was first proposed in (Wisdom et al., 2016).

We use the separation in train / test proposed in the original paper, having 3640 utterances for the training set, a validation set of size 192, and a test set of size 400. The validation / test division and the whole preprocessing of the dataset was done according to (Wisdom et al., 2016). The preprocessing goes as follows: The data is sampled at 8kHz and then cut into time frames of the same size. These frames are then transformed into the log-magnitude Fourier space and finally, they are normalized according to a per-training

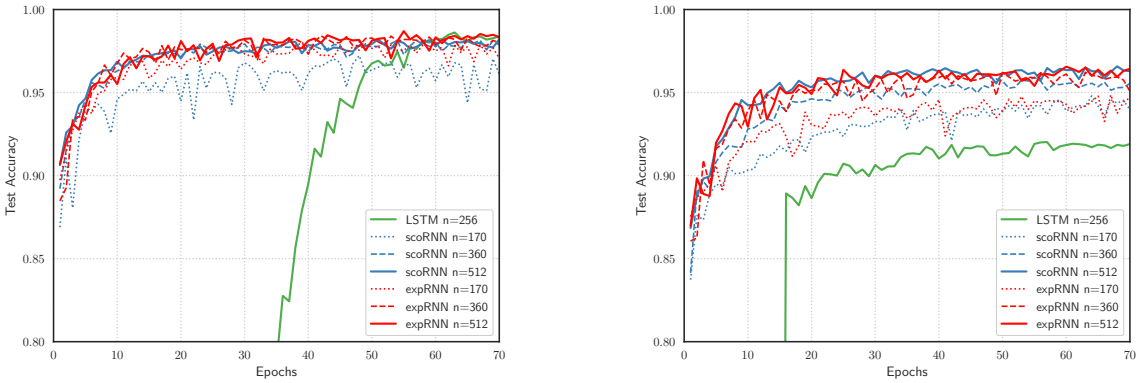

Figure 2. Test losses for several models on pixel-by-pixel MNIST (left) and P-MNIST (right).

Table 1. Best test accuracy at the MNIST and P-MNIST tasks.

MODEL	N	# PARAMS	MNIST	P-MNIST
EXPRNN	170	$\approx 16K$	0.980	0.949
EXPRNN	360	$\approx 69K$	0.984	0.962
EXPRNN	512	$\approx 137K$	**0.987**	**0.966**
SCORNN	170	$\approx 16K$	0.972	0.948
SCORNN	360	$\approx 69K$	0.981	0.959
SCORNN	512	$\approx 137K$	0.982	0.965
LSTM	128	$\approx 68K$	0.819	0.795
LSTM	256	$\approx 270K$	0.888	0.888
LSTM	512	$\approx 1058K$	0.919	0.918
RGD	116	$\approx 9K$	0.947	0.925
RGD	512	$\approx 137K$	0.973	0.947
URNN	512	$\approx 9K$	0.976	0.945
URNN	2170	$\approx 69K$	0.984	0.953
EURNN	512	$\approx 9K$	–	0.937

Table 2. Test MSE at the end of the epoch with the lowest validation MSE for the TIMIT task.

MODEL	N	# PARAMS	VAL. MSE	TEST MSE
EXPRNN	224	$\approx 83K$	5.34	5.30
EXPRNN	322	$\approx 135K$	**4.42**	**4.38**
EXPRNN	425	$\approx 200K$	5.52	5.48
SCORNN	224	$\approx 83K$	9.26	8.50
SCORNN	322	$\approx 135K$	8.48	7.82
SCORNN	425	$\approx 200K$	7.97	7.36
LSTM	84	$\approx 83K$	15.42	14.30
LSTM	120	$\approx 135K$	13.93	12.95
LSTM	158	$\approx 200K$	13.66	12.62
EURNN	158	$\approx 83K$	15.57	18.51
EURNN	256	$\approx 135K$	15.90	15.31
EURNN	378	$\approx 200K$	16.00	15.15
RGD	128	$\approx 83K$	15.07	14.58
RGD	192	$\approx 135K$	15.10	14.50
RGD	256	$\approx 200K$	14.96	14.69

set, test set, and validation set basis.

The results for this experiment are shown in Table 2. Again, the exponential parametrization beats—by a large margin—other methods of parametrization over the orthogonal group, and also the LSTM architecture. The results in Table 2 are those reported in (Helfrich et al., 2018).

As a side note, we must say that the results in this experiment should be interpreted under the following fact: We had access to two of the implementations for the tests for the other architectures regarding this experiment, and neither of them correctly handled sequences with different lengths present in this experiment. We suspect that the other implementations followed a similar approach, given that the results that they get are of the same order. In particular, the implementation released by Wisdom, which is the only publicly available implementation of this experiment, divides by a larger number than it should when computing the average MSE of a batch, hence reporting a lower MSE than the correct one. Even in this unfavorable scenario, our parametrization is able to get results that are twice as good—the MSE loss function is a quadratic function—as those from the other architectures.

6. Conclusion and Future Work

In this paper we have presented three main ideas. First, a simple approach based on classic Lie group theory to perform optimization over compact Lie groups, in particular $\mathrm{SO}(n)$ and $\mathrm{U}(n)$, proving its soundness and providing empirical evidence of its superior performance. Second, an implementation trick that allows for the implementation of arbitrary parametrizations at a negligible runtime cost. Finally, we sketched how to improve some existing methods to perform optimization on Lie groups using Riemannian

gradient descent. Any of these three ideas is of independent interest and could have more applications within neural networks.

The investigation of how to couple these ideas with the LSTM architecture to improve its performance is left for future work.

Additionally, it could be of interest to see how orthogonal constraints help with learning in deep feed forward networks. In order to make this last point formal, one would have to generalize the results presented here to *homogeneous Riemannian manifolds*, like the Stiefel manifold.

Acknowledgements

We would like to thank the help of Prof. Raphael Hauser and Jaime Mendizabal for the useful conversations, Daniel Feinstein for the proofreading, Prof. Terry Lyons for the computing power, and Kyle Helfrich for helping us setting up the experiments.

The work of MLC was supported by the Oxford-James Martin Graduate Scholarship and the "la Caixa" Banking Foundation (LCF/BQ/EU17/11590067). The work of DMR was supported by EP/N509711/1 from the EPSRC MPLS division, grant No 2053152.

References

Abrudan, T. E., Eriksson, J., and Koivunen, V. Steepest descent algorithms for optimization under unitary matrix constraint. *IEEE Transactions on Signal Processing*, 56 (3):1134–1147, 2008.

Absil, P.-A., Mahony, R., and Sepulchre, R. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

Al-Mohy, A. H. and Higham, N. J. Computing the fréchet derivative of the matrix exponential, with an application to condition number estimation. *SIAM Journal on Matrix Analysis and Applications*, 30(4):1639–1657, 2009a.

Al-Mohy, A. H. and Higham, N. J. A new scaling and squaring algorithm for the matrix exponential. *SIAM Journal on Matrix Analysis and Applications*, 31(3):970–989, 2009b.

Arjovsky, M., Shah, A., and Bengio, Y. Unitary evolution recurrent neural networks. In *International Conference on Machine Learning*, pp. 1120–1128, 2016.

Bansal, N., Chen, X., and Wang, Z. Can we gain more from orthogonality regularizations in training deep cnns? *arXiv preprint arXiv:1810.09102*, 2018.

Becigneul, G. and Ganea, O.-E. Riemannian adaptive optimization methods. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=r1eiqi09K7.

Bengio, Y., Simard, P., and Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

Bonnabel, S. Stochastic gradient descent on riemannian manifolds. *IEEE Transactions on Automatic Control*, 58 (9):2217–2229, 2013.

Boumal, N., Absil, P.-A., and Cartis, C. Global rates of convergence for nonconvex optimization on manifolds. *IMA Journal of Numerical Analysis*, 2016.

do Carmo, M. *Riemannian Geometry*. Mathematics (Boston, Mass.). Birkhäuser, 1992. ISBN 9783764334901. URL https://books.google.co.uk/books?id=uXJQQgAACAAJ.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

Edelman, A., Arias, T. A., and Smith, S. T. The geometry of algorithms with orthogonality constraints. *SIAM journal on Matrix Analysis and Applications*, 20(2):303–353, 1998.

Hall, B. *Lie Groups, Lie Algebras, and Representations: An Elementary Introduction*. Graduate Texts in Mathematics. Springer International Publishing, 2015. ISBN 9783319134673. URL https://books.google.es/books?id=didACQAAQBAJ.

Harandi, M. and Fernando, B. Generalized backpropagation, étude de cas: Orthogonality. *arXiv preprint arXiv:1611.05927*, 2016.

Helfrich, K., Willmott, D., and Ye, Q. Orthogonal recurrent neural networks with scaled Cayley transform. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1969–1978, Stockholmsmssan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/helfrich18a.html.

Henaff, M., Szlam, A., and LeCun, Y. Recurrent orthogonal networks and long-memory tasks. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning-Volume 48*, pp. 2034–2042. JMLR. org, 2016.

Higham, N. J. The scaling and squaring method for the matrix exponential revisited. *SIAM review*, 51(4):747–764, 2009.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Huang, L., Liu, X., Lang, B., Yu, A. W., Wang, Y., and Li, B. Orthogonal weight normalization: Solution to optimization over multiple dependent stiefel manifolds in deep neural networks. *arXiv preprint arXiv:1709.06079*, 2017.

Hyland, S. L. and Rätsch, G. Learning unitary operators with help from u (n). In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Jiang, B. and Dai, Y.-H. A framework of constraint preserving update schemes for optimization on stiefel manifold. *Mathematical Programming*, 153(2):535–575, 2015.

Jing, L., Shen, Y., Dubcek, T., Peurifoy, J., Skirlo, S., LeCun, Y., Tegmark, M., and Soljačić, M. Tunable efficient unitary neural networks (eunn) and their application to rnns. In *International Conference on Machine Learning*, pp. 1733–1741, 2017.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kumar Roy, S., Mhammedi, Z., and Harandi, M. Geometry aware constrained optimization techniques for deep learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

Le, Q. V., Jaitly, N., and Hinton, G. E. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*, 2015.

LeCun, Y. and Cortes, C. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/, 2010. URL `http://yann.lecun.com/exdb/mnist/`.

Maduranga, K. D. G., Helfrich, K., and Ye, Q. Complex unitary recurrent neural networks using scaled cayley transform. *arXiv preprint arXiv:1811.04142*, 2018.

Manton, J. H. Optimization algorithms exploiting unitary constraints. *IEEE Transactions on Signal Processing*, 50 (3):635–650, 2002.

Mhammedi, Z., Hellicar, A., Rahman, A., and Bailey, J. Efficient orthogonal parametrisation of recurrent neural networks using householder reflections. In *International Conference on Machine Learning*, pp. 2401–2409, 2017.

Milnor, J. Curvatures of left invariant metrics on lie groups. *Adv. Math.*, 21:293–329, 1976.

Moler, C. and Van Loan, C. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM review*, 45(1):3–49, 2003.

Ozay, M. and Okatani, T. Optimization on submanifolds of convolution kernels in cnns. *arXiv preprint arXiv:1610.07008*, 2016.

S Garofolo, J., Lamel, L., M Fisher, W., Fiscus, J., S. Pallett, D., L. Dahlgren, N., and Zue, V. Timit acoustic-phonetic continuous speech corpus. *Linguistic Data Consortium*, 11 1992.

Sato, H., Kasai, H., and Mishra, B. Riemannian stochastic variance reduced gradient. *arXiv preprint arXiv:1702.05594*, 2017.

Smith, S. T. *Geometric Optimization Methods for Adaptive Filtering*. PhD thesis, Harvard University, Cambridge, MA, USA, 1993. UMI Order No. GAX93-31032.

Tieleman, T. and Hinton, G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4 (2):26–31, 2012.

Vorontsov, E., Trabelsi, C., Kadoury, S., and Pal, C. On orthogonality and learning recurrent networks with long term dependencies. In *International Conference on Machine Learning*, pp. 3570–3578, 2017.

Wisdom, S., Powers, T., Hershey, J., Le Roux, J., and Atlas, L. Full-capacity unitary recurrent neural networks. In *Advances in Neural Information Processing Systems*, pp. 4880–4888, 2016.

Zhang, H., Reddi, S. J., and Sra, S. Riemannian svrg: Fast stochastic optimization on riemannian manifolds. In *Advances in Neural Information Processing Systems*, pp. 4592–4600, 2016.

A. Riemannian Geometry and Lie Groups

In this section we aim to give a short summary of the basics of classical Riemannian geometry and Lie group theory needed for our proofs in following sections. The standard reference for classical Riemannian geometry is do Carmo's book (do Carmo, 1992). An elementary introduction to Lie group theory with an emphasis on concrete examples from matrix theory can be found in (Hall, 2015).

A.1. Riemannian geometry

A Riemannian manifold is an n-dimensional smooth manifold $\mathcal{M}$ equipped with a smooth metric $\langle \cdot, \cdot \rangle_p \colon T_p\mathcal{M} \times T_p\mathcal{M} \to \mathbb{R}$ which is a positive definite inner product for every $p \in \mathcal{M}$. We will omit the dependency on the point p whenever it is clear from the context. Given a metric, we can define the length of a curve γ on the manifold as $L(\gamma) := \int_a^b \sqrt{\langle \gamma'(t), \gamma'(t) \rangle}\, \mathrm{d}t$. The distance between two points is the infimum of the lengths of the piece-wise smooth curves on $\mathcal{M}$ connecting them. When the manifold is connected, this defines a distance function that turns the manifold into a metric space.

An *affine connection* ∇ on a smooth manifold is a bilinear application that maps two vector fields X, Y to a new one $\nabla_X Y$ such that it is linear in X, and linear and Leibnitz in Y.

Connections give a notion of variation of a vector field along another vector field. In particular, the covariant derivative is the restriction of the connection to a curve. In particular, we can define the notion of parallel transport of vectors. We say that a vector field Z is *parallel* along a curve γ if $\nabla_{\gamma'} Z = 0$ where $\gamma' := \mathrm{d}\gamma(\frac{\mathrm{d}}{\mathrm{d}t})$. Given initial conditions $(p, v) \in T\mathcal{M}$ there exists locally a unique parallel vector field Z along γ such that $Z(p) = v$. $Z(\gamma(t))$ is sometimes referred to as the *parallel transport of* v *along* γ.

We say that a connection is *compatible with the metric* if for any two parallel vector fields X, Y along γ, their scalar product is constant. In other words, the connection preserves the angle between parallel vector fields. We say that a connection is *torsion-free* if $\nabla_X Y - \nabla_Y X = [X, Y] := XY - YX$. In a Riemannian manifold, there exists a unique affine connection such that it is compatible with the metric and that is also torsion-free. We call this distinguished connection the *Levi-Civita connection*.

A geodesic is defined as a curve such that its tangent vectors are covariantly constant along itself, $\nabla_{\gamma'} \gamma' = 0$. It is not true in general that given two points in a manifold there exists a geodesic that connects them. However, the *Hopf-Rinow theorem* states that this is indeed the case if the manifold is connected and complete as a metric space. The manifolds that we will consider are all connected and complete.

At every point p in our manifold we can define the *Riemannian exponential map* $\exp_p \colon T_p\mathcal{M} \to \mathcal{M}$, which maps a vector v to $\gamma(1)$ where γ is the geodesic such that $\gamma(0) = p$, $\gamma'(0) = v$. In a complete manifold, another formulation of the *Hopf-Rinow theorem* says that the exponential map is defined on the whole tangent space for every $p \in \mathcal{M}$. We also have that the Riemannian exponential map maps diffeomorphically a neighborhood around zero on the tangent space to a neighborhood of the point on which it is defined.

A map between two Riemannian manifolds is called a (local) isometry if it is a (local) diffeomorphism and its differential respects the metric.

A.2. Lie groups

A Lie group is a smooth manifold equipped with smooth group multiplication and inverse. Examples of Lie groups are the Euclidean space equipped with its additive group structure and the general linear group GL of a finite dimensional vector space given by the invertible linear endomorphisms of the space equipped with the composition of morphisms. We say that a Lie group is a matrix Lie group if it is a closed subgroup of some finite-dimensional general linear group.

Lie groups act on themselves via the left translations given by $L_g(x) = gx$ for $g, x \in G$. A vector field X is called *left invariant* if $(\mathrm{d}L_g)(X) = X \circ L_g$. A left invariant vector field is uniquely determined by its value at the identity of the group. This identification gives us a way to identify the tangent space at a point of the group with the tangent space at the identity. We call the tangent space at the identity *the Lie algebra of* G and we denote it by $\mathfrak{g}$.

For every vector $v \in \mathfrak{g}$ there exists a unique curve $\gamma \colon \mathbb{R} \to G$ such that γ is the integral curve of the left-invariant vector field defined by v such that $\gamma(0) = e$. This curve is a Lie group homomorphism and we call it the *Lie exponential*. It is also the integral curve of the right-invariant vector field with initial vector v.

We say that $c_g(x) = gxg^{-1}$ for $g, x \in G$ is an *inner automorphism of* G. Its differential at the identity is the *adjoint representation of* G, $\mathrm{Ad}\colon G \to \mathrm{GL}(\mathfrak{g})$ defined as $\mathrm{Ad}_g(X) := (\mathrm{d}c_g)_e(X)$ for $g \in G$, $X \in \mathfrak{g}$. The differential at the identity of Ad is called the *adjoint representation of* $\mathfrak{g}$, $\mathrm{ad}\colon \mathfrak{g} \to \mathrm{End}(\mathfrak{g})$ defined as $\mathrm{ad}_X(Y) := (\mathrm{d}\,\mathrm{Ad})_e(X)(Y)$. We say that $\mathrm{ad}_X(Y)$ is the *Lie bracket of* $\mathfrak{g}$ and we denote it by $[X, Y]$. For a matrix Lie group we have $\mathrm{Ad}_g(X) = gXg^{-1}$ and $\mathrm{ad}_X(Y) = XY - YX$.

A (complex) representation of a group is a continuous group homomorphism $\rho\colon G \to \mathrm{GL}(n, \mathbb{C})$. An injective representation is called *faithful*. The inclusion is a faithful representation for any matrix Lie group. On a compact Lie group, $\rho(g)$ is diagonalizable for every $g \in G$.

A Riemannian metric on G is said to be bi-invariant if it turns left and right translations into isometries. We have that every compact Lie group admits a bi-invariant metric. An example of a bi-invariant metric on the group of orthogonal matrices with positive determinant $\mathrm{SO}(n)$ is that inherited from $\mathbb{R}^{n\times n}$, namely the canonical metric $\langle X, Y\rangle = \mathrm{tr}(X^\intercal Y)$. The same happens in the unitary case, but changing the transpose for a conjugate transpose $\langle X, Y\rangle = \mathrm{tr}(X^* Y)$. Furthermore, every Lie group that admits a bi-invariant metric is a homogeneous Riemannian manifold—there exists an isometry between that takes any point to any other point—, and hence, complete.

B. Retractions

We take a deeper look into the concept of a retraction, which helps understanding the correctness of the approach used to optimize on $\mathrm{SO}(n)$ and $\mathrm{U}(n)$ presented in (Wisdom et al., 2016; Vorontsov et al., 2017).

The concept of a retraction is a relaxation of that of the Riemannian exponential map.

Definition B.1 (Retraction). A retraction is a map

$$\begin{aligned} r\colon T\mathcal{M} &\to \mathcal{M} \\ (x, v) &\mapsto r_x(v) \end{aligned}$$

such that

$$r_x(0) = x \qquad \text{and} \qquad (\mathrm{d}r_x)_0 = \mathrm{Id}$$

where Id is the identity map.

In other words, when $\mathcal{M}$ is a Riemannian manifold, r is a first order approximation of the Riemannian exponential map.

It is clear that the exponential map is a retraction. For manifolds embedded in the Euclidean space with the metric induced by that of the Euclidean space, the following proposition gives us a simple way to construct a rather useful family of retractions—those used in projected gradient descent.

Proposition B.2. *Let $\mathcal{M}$ be an embedded submanifold of $\mathbb{R}^n$, then for a differentiable surjective projection $\pi\colon \mathbb{R}^n \to \mathcal{M}$, that is, $\pi \circ \pi = \pi$, the map*

$$\begin{aligned} r\colon T\mathcal{M} &\to \mathcal{M} \\ (x, v) &\mapsto \pi(x + v) \end{aligned}$$

is a retraction, where we are implicitly identifying $T_x\mathcal{M} \subseteq T_x\mathbb{R}^n \cong \mathbb{R}^n$.

Proof. From π being a surjective projection we have that $\pi(x) = x$ for every $x \in \mathcal{M}$, which implies the first condition of the definition of retraction.

Another way of seeing the above is saying that $\pi|_{\mathcal{M}} = \mathrm{Id}$. This implies that, for every $x \in \mathcal{M}$, $(\mathrm{d}\pi)_x = \mathrm{Id}|_{T_x\mathcal{M}}$. By the chain rule, since the differential of $v \mapsto x + v$ is the identity as well we get the second condition. □

This proposition lets us see projected Riemannian gradient descent as an specific case of Riemannian gradient descent with a specific retraction. A corollary of this proposition that allows r to be defined just in those vectors of the form $x + v$ with $(x, v) = T\mathcal{M}$ lets us construct specific examples of retractions:

Example B.3 (Sphere). The function

$$r_x(v) = \frac{x + v}{\|x + v\|}$$

for $v \in T_x\mathbb{S}^n$ is a retraction.

Example B.4 (Special orthogonal group). Recall that for $A \in \mathrm{SO}(n)$,

$$T_A(\mathrm{SO}(n)) = \{X \in \mathbb{R}^{n\times n} \mid A^\intercal X + X^\intercal A = 0\},$$

then, for an element of $T\,\mathrm{SO}(n)$ we can define the map given by Proposition B.2. In this case, the projection $\pi(X)$ for a matrix with singular value decomposition $X = U\Sigma V^\intercal$ is $\pi(X) = UV^\intercal$.

This projection is nothing but the orthogonal projection from $\mathbb{R}^{n\times n}$ onto $\mathrm{SO}(n)$ when equipped with the canonical metric.

The two examples above are examples of orthogonal projections. The manifolds being considered are embedded into a Euclidean space and they inherit its metric. The projections here are orthogonal projections on the ambient space. On the other hand, Proposition B.2 does not require the projections to be orthogonal.

Different examples of retractions can be found in (Absil et al., 2009), Example 4.1.2.

C. Comparing Riemannian gradient descent and the exponential parametrization

The proofs in this section are rather technical and general so that they apply to a wide variety of manifolds such as $\mathrm{SO}(n)$, $\mathrm{U}(n)$, or the symplectic group. Even though we do not explore applications of optimizing over other compact matrix Lie groups, we state the results in full generality. Having this in mind, we will first motivate this section with the concrete example that we study in the applications of this paper: $\mathrm{SO}(n)$.

Example C.1. Let $f\colon \mathbb{R}^{n\times n} \to \mathbb{R}$ be a function defined on the space of matrices. Our task is to solve the problem

$$\min_{B\in\mathrm{SO}(n)} f(B).$$

A simple way to approach this problem would be to apply Riemannian gradient descent to it. Let A be a skew-symmetric matrix an let $B = e^A \in \mathrm{SO}(n)$, where e^A denotes the exponential of matrices. We will suppose that we are working with the canonical metric on $\mathbb{R}^{n\times n}$, namely $\langle X, Y\rangle = \mathrm{tr}(X^\intercal Y)$. We will denote by $\nabla f(B)$ the gradient of the function f at the matrix B, and by $\mathrm{grad}\, f(B) \in T_B\,\mathrm{SO}(n)$ the gradient associated to the restriction $f|_{\mathrm{SO}(n)}$ with respect to the induced metric.

Riemannian gradient descent works by following the geodesic defined by the direction $-\mathrm{grad}\, f(B)$ at the point B. In the words of the Riemannian exponential map at B, if we have a learning rate $\eta > 0$, the update rule will be given by

$$B \leftarrow \exp_B(-\eta\,\mathrm{grad}\, f(B)).$$

The tangent space to $\mathrm{SO}(n)$ at a matrix B is

$$T_B\,\mathrm{SO}(n) = \{X \in \mathbb{R}^{n\times n} \mid B^\intercal X + X^\intercal B = 0\}$$

and it is easy to check that the orthogonal projection with respect to the canonical metric onto this space is given by

$$\begin{aligned}\pi_B\colon \mathbb{R}^{n\times n} &\to T_B\,\mathrm{SO}(n)\\ X &\mapsto \tfrac{1}{2}(X - BX^\intercal B)\end{aligned}$$

Since the gradient on the manifold is just the tangent component of the gradient on the ambient space, we have that

$$\mathrm{grad}\, f(B) = \tfrac{1}{2}(\nabla f(B) - B\nabla f(B)^\intercal B),$$

and since multiplying by an orthogonal matrix constitutes an isometry, in order to compute the Riemannian exponential map we can transport the vector from T_BG to $T_\mathrm{I}G$, compute the exponential at the identity using the exponential of matrices and then transport the result back. In other words,

$$\exp_B(X) = B\exp(B^\intercal X) \qquad \forall X \in T_B\,\mathrm{SO}(n).$$

Putting everything together, the Riemannian gradient descent update rule for $\mathrm{SO}(n)$ is given by

$$B \leftarrow e^A e^{-\eta B^\intercal\,\mathrm{grad}\, f(B)}.$$

The other update rule that we have is the one given by the exponential parametrization

$$B \leftarrow e^{A - \eta \nabla (f \circ \exp)(A)}.$$

This is nothing but the gradient descent update rule applied to the problem

$$\min_{A \in \mathrm{Skew}(n)} f(\exp(A)).$$

Both of these rules follow geodesic flows for two metrics whenever A is in a neighborhood of the identity on which $\exp$ is a diffeomorphism (*cf.*, Appendix D). A natural question that arises is whether these metrics are the same. A less restrictive question would be whether this second optimization procedure defines a retraction or whether their gradient flow is completely different.

In the sequel, we will see that for $\mathrm{SO}(n)$ these two optimization methods give raise to two rather different metrics. We will explicitly compute the quantity $\nabla(f \circ \exp)(A)$, and we will give necessary and sufficient conditions equivalent under which these two optimization methods agree.

C.1. Optimization on Lie Groups with Bi-invariant Metrics

In this section we expose the theoretical part of the paper. The first part of this section is classic and can be found, for example, in Milnor (Milnor, 1976). We present it here for completeness. The results Proposition C.11 and Theorem C.12 are novel.

Remark. Throughout this section the operator $(-)^*$ will have two different meanings. It can be either the pullback of a form along a function or the adjoint of a linear operator on a vector space with an inner product. Although the two can be distinguished in many situations, we will explicitly mention to which one we are referring whenever it may not be clear from the context. Note that when we are on a matrix Lie group equipped with the product $\langle X, Y \rangle = \mathrm{tr}(X^* Y)$, the adjoint of a linear operator is exactly its conjugate transpose, hence the notation.

When we deal with an abstract group we will denote the identity element as e. If the group is a matrix Lie group, we will sometimes refer to it as I.

We start by recalling the definition of our object of study.

Definition C.2 (Invariant metric on a Lie Group). A Riemannian metric on a Lie group G is said to be left (resp. right) invariant if it makes left (resp. right) translations into isometries. Explicitly, it is so if for every $g \in G$, and the metric α we have that $\alpha_g = L_{g^{-1}}^* \alpha_e$ (resp. $\alpha_g = R_{g^{-1}}^* \alpha_e$).

A bi-invariant metric is a metric that is both left and right-invariant.

We can construct a bi-invariant metric on a Lie group by using the *averaging trick*.

Proposition C.3 (Bi-invariant metric on compact Lie groups). *A compact Lie group G admits a bi-invariant metric.*

Proof. Let n be the dimension of G and let μ_e be a non-zero n-form at $\mathfrak{g}$. This form is unique up to a multiplicative constant. We can then extend it to the whole G by pulling it back along R_g defining $\mu_g := R_{g^{-1}}^* \mu_e$. This makes it into a right-invariant n-form on the manifold, which we call the *right Haar measure*.

Let $(\cdot, \cdot)$ be an inner product on $\mathfrak{g}$. We can turn this inner product into an Ad-invariant inner product on $\mathfrak{g}$ by averaging it over the elements of the group using the right Haar measure

$$\langle u, v \rangle = \int_G (\mathrm{Ad}_g(u), \mathrm{Ad}_g(v))\, \mu(\mathrm{d}g).$$

Note that this integral is well defined since G is compact. The Ad-invariance follows from the right-invariance of μ

$$\langle \mathrm{Ad}_h(u), \mathrm{Ad}_h(v) \rangle = \int_G (\mathrm{Ad}_{gh}(u), \mathrm{Ad}_{gh}(v))\, \mu(\mathrm{d}g) = \langle u, v \rangle.$$

Finally, we can extend this product to the whole group by pulling back the inner product along L_g, that is, if we denote the metric by α, $\alpha_g = L_{g^{-1}}^* \alpha_e$. This automatically makes it into a left-invariant metric. But since it is Ad-invariant at the

identity, we have that for every $g, h \in G$

$$R_g^* \alpha_{hg} = R_g^* L_{g^{-1}h^{-1}}^* \alpha_e = \mathrm{Ad}_{g^{-1}}^* L_{h^{-1}}^* \alpha_e = L_{h^{-1}}^* \alpha_e = \alpha_h$$

and the metric is also right-invariant, finishing the proof. □

If the group is abelian, the construction above is still valid without the need of the averaging trick, since Ad is the identity map, so every inner product is automatically Ad-invariant.

It turns out that these examples and their products exhaust all the Lie groups that admit a bi-invariant metric. We include this result for completeness, even though we will not use it.

Theorem C.4 (Classification of groups with bi-invariant metrics). *A Lie group admits a bi-invariant metric if and only if it is isomorphic to $G \times H$ with G compact and H abelian.*

Proof. (Milnor, 1976) Lemma 7.5. □

Lie groups, when equipped with a bi-invariant metric are rather amenable from the Riemannian geometry perspective. This is because it is possible to reduce many computations on them to matrix algebra, rather than the usual systems of differential equations that one encounters when dealing with arbitrary Riemannian manifolds.

The following proposition will come in handy later.

Lemma C.5. *If an inner product on $\mathfrak{g}$ is Ad-invariant then*

$$\langle Y, \mathrm{ad}_X(Z) \rangle = -\langle \mathrm{ad}_X(Y), Z \rangle \qquad \forall X, Y, Z \in \mathfrak{g}.$$

In other words, the adjoint of the map ad_X with respect to the inner product is $-\mathrm{ad}_X$. We say that ad is skew-adjoint and we write $\mathrm{ad}_X^* = -\mathrm{ad}_X$.

Proof. We have that, by definition

$$\mathrm{ad}_X(Y) = \frac{\mathrm{d}}{\mathrm{d}t}(\mathrm{Ad}_{\exp(tX)}(Y)) \mid_0,$$

so, deriving the equation

$$\langle \mathrm{Ad}_{\exp(tX)}(Y), \mathrm{Ad}_{\exp(tX)}(Z) \rangle = \langle Y, Z \rangle$$

with respect to t we get the result. □

With this result in hand, we can prove a rather useful relation between the geometry of the Lie group and its algebraic structure.

Proposition C.6. *Let G be a Lie group equipped with a bi-invariant metric. If X, Y are left-invariant vector fields, we have that their Levi-Civita connection and their sectional curvature are given by*

$$\nabla_X Y = \frac{1}{2}[X, Y]$$
$$\kappa(X, Y) = \frac{1}{4}\|[X, Y]\|^2.$$

The sectional curvature formula holds whenever X and Y are orthonormal.

Proof. For left-invariant vector fields X, Y, Z, the Koszul formula gives

$$\langle \nabla_X Y, Z \rangle = \frac{1}{2}(X\langle Y, Z \rangle + Y\langle X, Z \rangle - Z\langle X, Y \rangle + \langle [X, Y], Z \rangle - \langle [X, Z], Y \rangle - \langle [Y, Z], X \rangle).$$

The three first terms on the right vanish, since the angle between invariant vector fields is constant. Reordering the last three terms, using Lemma C.5, the fact that the Lie bracket is antisymmetric, and since invariant vector fields form a basis of the Lie algebra, the formula for the connection follows. Now, the curvature tensor is given by

$$R(X, Y)Z = \nabla_X \nabla_Y Z - \nabla_Y \nabla_X Z - \nabla_{[X,Y]} Z = \frac{1}{4}([X, [Y, Z]] - [Y, [X, Z]] - 2[[X, Y], Z]) = \frac{1}{4}[Z, [X, Y]].$$

So the sectional curvature for X, Y orthonormal is given by

$$\kappa(X,Y) = \langle R(X,Y)Y, X\rangle = \frac{1}{4}\|[X,Y]\|^2. \qquad \square$$

On a matrix Lie group equipped with a metric we have three different notions of exponential maps, namely the Lie exponential map, the Riemannian exponential map and the exponential of matrices. We will now show that if we consider the Riemannian exponential map at the identity element, these three concepts agree whenever the metric is bi-invariant.

Proposition C.7 (Equivalence of Riemannian and Lie exponential). *Let G be a Lie group equipped with a bi-invariant metric. Then, the Riemannian exponential at the identity $\exp_e$ and the Lie exponential $\exp$ agree.*

Proof. Fix a vector $X_e \in \mathfrak{g}$ and consider the curve $\gamma(t) = \exp(tX_e)$. This curve is the integral curve of the invariant vector field defined by X_e, this is $\gamma'(t) = X(\gamma(t))$. For this reason, by Proposition C.6

$$\nabla_{\gamma'}\gamma' = \frac{1}{2}[X,X] = 0.$$

so $\exp(tX_e)$ is a geodesic and the result readily follows. $\square$

Proposition C.8 (Equivalence of Lie exponential and exponential of matrices). *Let G be a matrix Lie group, that is, a closed subgroup of $\mathrm{GL}(n,\mathbb{C})$. Then the matrix exponential $\exp_M$ and the Lie exponential $\exp$ agree.*

Proof. The matrix exponential $\exp_{tX}$ can be expressed as the solution of the matrix differential equation

$$\gamma'(t) = X\gamma(t) \qquad \gamma(0) = \mathrm{I},\ t \in \mathbb{R}$$

for $X \in \mathbb{C}^{n\times n} = \mathfrak{gl}(n,\mathbb{C})$. This is exactly the equation that defines the Lie exponential map as the integral curve of a right-invariant vector field, that is, the Lie exponential. $\square$

Finally, all these equivalences give a short proof of the fact that the Lie exponential map is surjective on a connected Lie group with a bi-invariant metric.

Theorem C.9 (Lie exponential surjectivity). *Let G be a connected Lie group equipped with a bi-invariant metric. The Lie exponential is surjective.*

Proof. As the Lie exponential is defined in the whole Lie algebra, so is the map $\exp_e$. Since the metric is bi-invariant, we have that at a point $(g,v) \in TG$, $\exp_g(v) = L_g(\exp_e((\mathrm{d}L_{g^{-1}})_g(v)))$ and since left-translations are diffeomorphisms, the Riemannian exponential is defined in the whole tangent bundle. Therefore, by the Hopf-Rinow theorem, this implies that there exists a geodesic between any two points. Since the geodesics starting at the identity are given by the curves $\gamma(t) = \exp(tX_e)$ for $X_e \in \mathfrak{g}$, the result follows. $\square$

Now that we have all the necessary tools, we shall return to the problem of studying the metric induced by the exponential parametrization.

As we have seen, the problem that we are interested in solving is

$$\min_{g\in G} f(g),$$

where G is a matrix Lie group equipped with an bi-invariant metric. The exponential parametrization maps this problem back to the Lie algebra

$$\min_{X\in\mathfrak{g}} f(\exp(X)).$$

Since $\mathfrak{g}$ is a vector space, putting a basis on it we have that we can use all the classical toolbox developed for Euclidean spaces to approach a solution for this problem. In particular, in the context of neural networks, we are interested in studying first-order optimization methods that approach a solution to this problem, in particular, gradient descent methods. The gradient descent update step for this problem with learning rate $\eta > 0$ is given by

$$X \leftarrow X - \eta\nabla(f\circ\exp)(X),$$

where the gradient is defined with respect to the metric, that is, it is the vector such that

$$\mathrm{d}(f \circ \exp)_X(Y) = \langle \nabla(f \circ \exp)(X), Y \rangle.$$

To study this optimization method, we first have to make sense of the gradient $\nabla(f \circ \exp)(X)$. To do so, we will make use of the differential of the exponential map.

Proposition C.10. *The differential of the exponential map on a matrix Lie group is given by the formula*

$$(\mathrm{d}\exp)_X(Y) = e^X \sum_{k=0}^{\infty} \frac{(-\operatorname{ad}_X)^k}{(k+1)!}(Y) \qquad \forall X, Y \in \mathfrak{g}.$$

Proof. (Hall, 2015) Theorem 5.4. □

An analogous formula still holds in the general case, but the proof is more delicate. The powers of the adjoint representation are to be thought as the composition of endomorphisms on $\mathfrak{g}$. For this reason, this formula can be also expressed as

$$(\mathrm{d}\exp)_X(Y) = e^X(Y - \frac{1}{2}[X, Y] + \frac{1}{6}[X, [X, Y]] - \ldots).$$

Yet another way of looking at this expression is by defining the function

$$\begin{aligned} \phi\colon \operatorname{End}(\mathfrak{g}) &\to \operatorname{End}(\mathfrak{g}) \\ X &\mapsto \frac{1 - e^{-X}}{X} \end{aligned}$$

so that

$$(\mathrm{d}\exp)_X = \mathrm{d}L_{e^X} \circ \phi(\operatorname{ad}_X).$$

In this case, the fraction that defines ϕ is just a formal expression to refer to the formal series defined in Proposition C.10.

From this we can compute the gradient of $f \circ \exp$.

Proposition C.11. *Let $f\colon G \to \mathbb{R}$ be a function defined on a matrix Lie group equipped with a bi-invariant metric. For a matrix $A \in \mathfrak{g}$ let $B = e^A$. We have*

$$\nabla(f \circ \exp)(A) = B(\mathrm{d}\exp)_{-A}(B^{-1}\nabla f(B)) = \sum_{k=0}^{\infty} \frac{(\operatorname{ad}_A)^k}{(k+1)!}(e^{-A}\nabla f(B)).$$

Proof. Let $U \in \mathfrak{g}$. By the chain rule, we have

$$(\mathrm{d}(f \circ \exp))_A(U) = (\mathrm{d}f)_B \circ (\mathrm{d}\exp)_A(U).$$

In terms of the gradient of f with respect to the metric this is equivalent to

$$\begin{aligned} (\mathrm{d}(f \circ \exp))_A(U) &= \langle \nabla f(B), (\mathrm{d}\exp)_A(U) \rangle \\ &= \langle (\mathrm{d}\exp)_A^*(\nabla f(B)), U \rangle \end{aligned}$$

which gives

$$\nabla(f \circ \exp)(A) = (\mathrm{d}\exp)_A^*(\nabla f(B)).$$

Now we just have to compute the adjoint of the differential of the exponential function. This is now simple since

$$\begin{aligned} (\mathrm{d}\exp)_A^* &= (\mathrm{d}L_{e^A} \circ \phi(\operatorname{ad}_A))^* \\ &= \phi(\operatorname{ad}_A)^* \circ \mathrm{d}L_{e^{-A}} \\ &= \phi(\operatorname{ad}_A^*) \circ \mathrm{d}L_{e^{-A}} \\ &= \phi(\operatorname{ad}_{-A}) \circ \mathrm{d}L_{e^{-A}}, \end{aligned}$$

where the second equality follows from the product being left-invariant, the third one from ϕ being analytic and the last one from Lemma C.5. □

Now we can explicitly define the update rule for the exponential parametrization

$$\begin{aligned}\widehat{r}\colon TG &\to G\\ (e^A, U) &\mapsto \exp(A + \phi(\mathrm{ad}_{-A})(e^{-A}U))^{\cdot}\end{aligned}$$

We can then study the gradient flow induced by the exponential parametrization by means of $\widehat{r}$. If $\widehat{r}$ were a retraction, then the flow induced by the exponential parametrization would have similar properties as that of Riemannian gradient descent, as shown in (Boumal et al., 2016). It turns out that the exponential parametrization induces a different flow.

Theorem C.12. *Let G be a connected matrix Lie group equipped with a bi-invariant metric. The function $\widehat{r}$ is a retraction if and only if G is abelian.*

Proof. It is clear that $\widehat{r}_g(0) = g$ for every $g \in G$. Let $A \in \mathfrak{g}$ and $B = e^A$ and let $U \in T_B G$. By the chain rule we have that

$$(\mathrm{d}\widehat{r}_B)_0(U) = (\mathrm{d}\exp)_A((\mathrm{d}\exp)^*_A(U)).$$

The map $\widehat{r}$ is a retraction if and only if

$$(\mathrm{d}\exp)_A((\mathrm{d}\exp)^*_A(U)) = U$$

holds for every $U \in T_B G$. This is equivalent to having

$$\langle (\mathrm{d}\exp)_A((\mathrm{d}\exp)^*_A(U)), H\rangle = \langle U, H\rangle$$

for every $H \in T_B G$. Taking adjoints and since the metric is left-invariant, using the formula for the adjoint of the differential of the exponential map computed in Proposition C.11, or equivalently

$$\langle (\mathrm{d}\exp)_{-A}(X), (\mathrm{d}\exp)_{-A}(Y)\rangle = \langle X, Y\rangle \qquad \forall X, Y \in \mathfrak{g}.$$

In other words, $\widehat{r}$ is a retraction if and only if the Lie exponential map is a local isometry.

Now, the Lie exponential maps $\mathfrak{g}$ into G, but $\mathfrak{g}$ equipped with its metric is flat, so it has constant sectional curvature zero. On the other hand, the sectional curvature of G is given by $\kappa(X, Y) = \frac{1}{4}\|[X, Y]\|^2$. Recall that a Lie group is abelian if and only if its Lie bracket is zero.

If the Lie bracket is zero, $(\mathrm{d}\exp)_A = (\mathrm{d}L_{e^A})_e$, and it is an isometry.

Conversely, if it is an isometry, it preserves the sectional curvature, so the Lie bracket has to be identically zero, hence the group is Abelian. □

In the abelian case, we do not only have that $\widehat{r}$ is a retraction, but also that the update rule for the exponential parametrization agrees with that of Riemannian gradient descent. Recall that the Riemannian gradient descent rule for a gradient U and a step-size η is given by

$$e^A e^{-\eta e^{-A} U}.$$

On an abelian group we have that $e^X e^Y = e^{X+Y}$. Furthermore, since the adjoint representation is zero, $(\mathrm{d}\exp)_A(U) = e^A U$. Putting these two things together we have

$$\widehat{r}(e^A, -\eta U) = e^{A - \eta e^{-A} U} = e^A e^{-\eta e^{-A} U}.$$

D. Maximal Normal Neighborhood of the Identity

Definition D.1 (Normal Neighborhood). Let $V \subseteq T_p\mathcal{M}$ be a neighborhood of 0 such that the Riemannian exponential map $\exp_p$ is a diffeomorphism. Then we say that $\exp_p V$ is a *normal neighborhood* of p.

Given that on a matrix Lie group $(\mathrm{d}\exp)_{\mathrm{I}} = \mathrm{Id}$, by the inverse function theorem, there exists a normal neighborhood around the identity matrix. In this section we will prove that the maximal open normal neighborhood of $\mathrm{SO}(n)$ (resp. $\mathrm{U}(n)$) covers almost all the group. By almost all the group we mean that the closure of the normal neighborhood is equal to the group. We do so by studying at which points we have that the map $\exp$ is no longer an immersion, or in other words, we look at

the points $A \in \mathfrak{g}$ at which $\det((\mathrm{d}\exp)_A) = 0$. We will prove so for the group $\mathrm{GL}(n, \mathbb{C})$, so that the arguments readily generalize to any matrix Lie group.

Recall the definition of the matrix-valued function ϕ defined on the space of endomorphisms of the Lie algebra of $\mathrm{GL}(n, \mathbb{C})$. Specifically, since $\mathfrak{gl}(n, \mathbb{C}) \cong \mathbb{C}^{n\times n}$, we have that $\mathrm{End}(\mathfrak{gl}(n, \mathbb{C})) \cong \mathbb{C}^{n^2\times n^2}$

$$\begin{aligned}\phi\colon \mathbb{C}^{n^2\times n^2} &\to \mathbb{C}^{n^2\times n^2}\\ A &\mapsto \frac{1-e^{-A}}{A} = \sum_{k=0}^{\infty} \frac{(-A)^k}{(k+1)!}\,.\end{aligned}$$

Using this function, we can factorize the differential of the exponential function on a matrix Lie group as

$$(\mathrm{d}\exp)_A = e^A \phi(\mathrm{ad}_A).$$

Let us now compute the maximal normal neighborhood of the identity. This result is classic, but the proof here is a simplification of the classical one using an approximation argument.

Theorem D.2. *Let G be a compact and connected matrix Lie group. The exponential function is analytic, with analytic inverse on a bounded open neighborhood of the origin given by*

$$U = \{A \in \mathfrak{g} \mid |\mathrm{Im}(\lambda_i(A))| < \pi\}.$$

Proof. Given that L_{e^A} is a diffemorphism, we are interested in studying when the matrix defined by the function $\phi(\mathrm{ad}_A)$ stops being full-rank and when is it injective.

First, note that if the eigenvalues of A are λ_i, then the eigenvalues of $g(A)$ with g a complex analytic function well-defined on $\{\lambda_i\}$ are $\{g(\lambda_i)\}$. This is clear for diagonalizable matrices. Since these are dense in $\mathbb{C}^{n\times n}$, given that eigenvalues are continuous functions of the matrix, it readily generalizes to arbitrary matrices.

Let $A \in \mathbb{C}^{n^2\times n^2}$ and let $\lambda_{i,j}$ for $1 \leq i, j \leq n$ be its eigenvalues. Then, ϕ is non-singular when $\phi(\lambda_{i,j}) \neq 0$ for every $\lambda_{i,j}$. Equivalently, when $\lambda_{i,j} \neq 2\pi k i$ for $k \in \mathbb{Z}\backslash\{0\}$.

Let us now compute the eigenvalues of ad_A using the same trick as above. Let $A \in \mathbb{C}^{n\times n}$ and suppose that it is diagonalizable with eigenvalues $\{\lambda_i\}$. Let u_i be the eigenvectors of A and v_i the eigenvectors of $A^\intercal$—which is also diagonalizable with the same eigenvalues—. Since

$$\mathrm{ad}_A(u_i \otimes v_j) = (\lambda_i - \lambda_j) u_i \otimes v_j$$

we have that $\{u_i \otimes v_j\}$ are the eigenvectors of ad_A with eigenvalues $\lambda_{i,j} := \lambda_i - \lambda_j$. Now, using the same continuity argument as above have that these are the eigenvalues of ad_A for every $A \in \mathbb{C}^{n\times n}$.

From all this, we draw that $(\mathrm{d}\exp)_A$ is singular whenever A has two eigenvalues that differ by a non-zero integer multiple of $2\pi i$.

Finally, on a compact matrix Lie group every matrix is diagonalizable over the complex numbers, so the exponential acts on the eigenvalues, but the complex variable function e^z is injective on $\{z \in \mathbb{C} \mid |\mathrm{Im}(z)| < \pi\}$, so the Lie exponential is injective on this domain as well. □

Let us look at the particular cases that we are interested in. If we set $G = \mathrm{SO}(n)$, we have that its Lie algebra are the skew-symmetric matrices. Skew-symmetric matrices have purely imaginary eigenvalues. Furthermore, since they are real matrices, their eigenvalues come in conjugate pairs. As such, we have that the exponential map is singular on every matrix in the boundary of the set U defined in Theorem D.2.

Special orthogonal matrices are those matrices which are similar to block-diagonal matrices with diagonal blocks of the form

$$B = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$$

for $\theta \in (-\pi, \pi]$. On $\mathrm{SO}(2n+1)$, there is an extra block with a single 1.

Similarly, skew-symmetric matrices are those matrices which are similar to block-diagonal matrices with diagonal blocks of the form

$$A = \begin{pmatrix} 0 & \theta \\ -\theta & 0 \end{pmatrix}.$$

On $\mathfrak{so}(2n+1)$ there is an extra block with a single zero.

This outlines an elementary proof of the fact that the Lie exponential is surjective on $\mathrm{SO}(n)$ and $\mathrm{U}(n)$.

In both cases this shows that the boundary of U has measure zero and that $f(\overline{U}) = G$.

Remark. The reader familiar with Lie group theory will have noticed that this proof is exactly the standard one for the surjectivity of the exponential map using the Torus theorem, where one proves that all the maximal tori in a compact Lie group are conjugated and that every element of the group lies in some maximal torus, arriving then to the same conclusion but in much more generality.

E. Hyperparameters for the Experiments

The batch size across all the experiments was 128. The learning rates for the orthogonal parameters are 10 times less those for the non-orthogonal parameters. We fixed the seed of both Numpy and Pytorch to be 5544 for all the experiments for reproducibility. This is the same seed that was used in the experiments in (Helfrich et al., 2018). In Table 3 we refer to the optimizer and learning rate for the non-orthogonal part of the neural network simply as optimizer and learning rate.

Table 3. Hyperparameters for the Experiments in Section 5.

Dataset	Size	Optimizer	Learning Rate	Orthogonal optimizer	Orthogonal Learning Rate
Copying Problem $L = 1000$	190	RMSPROP	$2 \cdot 10^{-4}$	RMSPROP	$2 \cdot 10^{-5}$
Copying Problem $L = 2000$			$2 \cdot 10^{-4}$		$2 \cdot 10^{-5}$
	170	RMSPROP	$7 \cdot 10^{-4}$	RMSPROP	$7 \cdot 10^{-5}$
MNIST	360		$5 \cdot 10^{-4}$		$5 \cdot 10^{-5}$
	512		$3 \cdot 10^{-4}$		$3 \cdot 10^{-5}$
	170		10^{-3}		10^{-4}
P-MNIST	360		$7 \cdot 10^{-4}$		$7 \cdot 10^{-5}$
	512		$5 \cdot 10^{-4}$		$5 \cdot 10^{-5}$
	224	ADAM	10^{-3}	RMSPROP	10^{-4}
TIMIT	322		$7 \cdot 10^{-4}$		$7 \cdot 10^{-5}$
	425		$7 \cdot 10^{-4}$		$7 \cdot 10^{-5}$