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

[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.

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 [1] 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 $\langle X, Y\rangle = \mathrm{tr}(X^\intercal Y)$ (resp. $\mathrm{tr}(X^* Y)$), they inherit a *bi-invariant metric*, meaning that the metric is invariant with respect to left and right multiplication by matrices of the

[1] 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.

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

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

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[2] $\langle X, Y\rangle = \mathrm{tr}(X^*Y)$ 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 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.

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

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), Unitary RNN (URNN), Efficient Unitary RNN (EURNN), Cayley Parametrization (SCORNN), Riemannian Gradient Descent (RGD) which can be found in (Hochreiter & Schmidhuber, 1997), (Arjovsky et al., 2016), (Jing et al., 2017), (Helfrich et al., 2018) and (Wisdom et al., 2016) respectively.

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 follows. 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

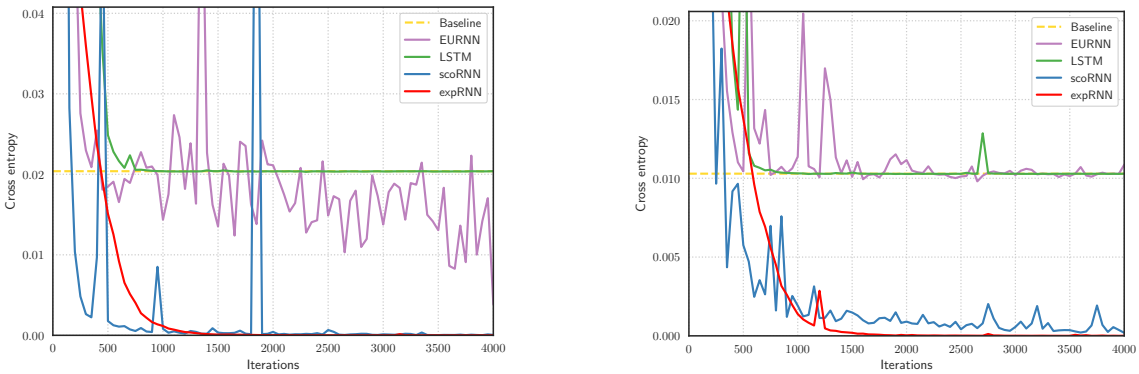

Figure 1. Cross entropy of the different algorithms in the copying problem for $L = 1000$ (left) and $L = 2000$ (right).

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: `----------14211`

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.

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	$\approx 16K$	0.947	0.925
RGD	512	$\approx 270K$	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

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 orig-

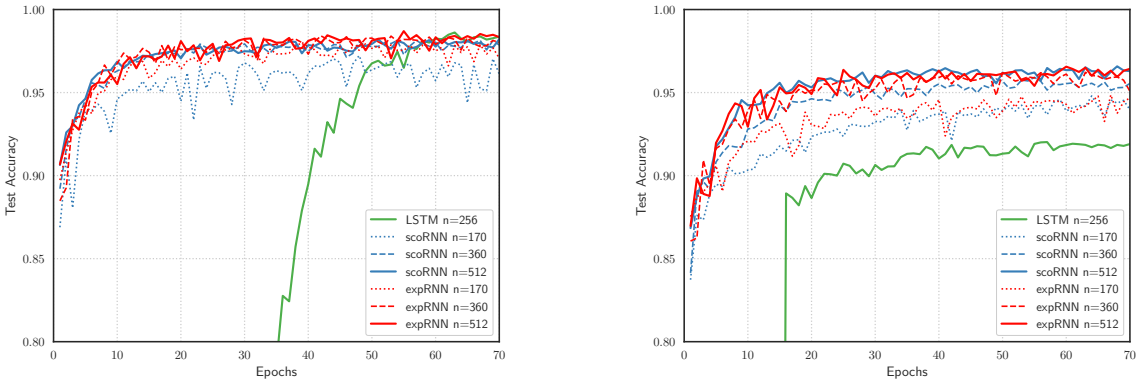

Figure 2. Test losses for several models on pixel-by-pixel MNIST (left) and P-MNIST (right).

inal 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 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).

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

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, Stockholmsmässan, 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.