CWY Parametrization for Scalable Learning of Orthogonal and Stiefel Matrices

Valerii Likhosherstov 1 Jared Davis 2 Krzysztof Choromanski 2,3 Adrian Weller 1,4

Abstract

In this paper we propose a new approach for optimization over orthogonal groups. We parametrize an orthogonal matrix as a product of Householder reflections. To overcome low parallelization capabilities of computing Householder reflections sequentially, we employ an accumulation scheme called the compact WY (or CWY) transform—a compact matrix representation for the series of Householder reflections which can be computed efficiently on highly parallelizable computation units such as GPU and TPU. We further introduce the Truncated CWY (or T-CWY)—a novel approach for Stiefel manifold parametrization which has a competitive complexity estimate compared to other methods and, again, has an advantage when computed on GPU and TPU. We apply these proposed parametrizations to train recurrent neural network architectures in the tasks of neural machine translation and video prediction and demonstrate superiority in both computational and learning aspects compared to other methods from the literature.

1. Introduction

Training weight matrices in a neural network with an orthogonality constraint gives various benefits for a deep learning practitioner, including enabling control over the norm of the hidden representation and its gradient which might be helpful for several reasons. A series of works addresses the problems of exploding or vanishing gradients in recurrent neural networks (Hochreiter, 1998) by using orthogonal or unitary transition matrices (Arjovsky et al., 2016; Wisdom et al., 2016; Jing et al., 2016; Mhammedi et al., 2017; Helfrich et al., 2018; Casado & Martinez-Rubio, 2019).

Further, orthogonality improves forward and backward information propagation in deep convolutional neural networks where convolutions are parametrized by a Stiefel manifold—a general class of orthogonal matrices. Orthogonality constraints are satisfied either by parametrizing the orthogonal matrix by a set of unconstrained parameters (Huang et al., 2018), using orthogonal regularization (Bansal et al., 2018), or performing optimization directly in the manifold by Riemannian Gradient Descent (Li et al., 2020).

The norm-preserving property of orthogonal linear operator helps to gain control over the Lipschitz constant of the deep architecture and, therefore, enhances adversarial robustness of the model and its generalization capabilities both in theory and practice (Cisse et al., 2017). Orthogonality can also be useful when designing invertible, fast to compute yet flexible constructions for generative modelling in the form of normalizing flows (Tomczak & Welling, 2016).

Contribution. We present a new approach to optimization over orthogonal matrices, focusing on computational efficiency. We employ compact the WY (or CWY) transform, an aggregation scheme for the composition of several Householder reflections. The proposed approach has several advantages:

1. While in exact arithmetic being equivalent to decomposition into Householder reflections (Mhammedi et al., 2017), our approach is more computationally efficient when executed on a parallel computation unit such as GPU or TPU. By parallelizing, we manage to reach scales which are intractable when using Householder reflections explicitly.
2. The approach is compatible with a user-defined trade-off between runtime and expressiveness of the orthogonal matrices covered during optimization.
3. Since the proposed approach is based on parametrizing an orthogonal matrix by unconstrained parameters, it is compatible with general-purpose optimizers such as Adam (Kingma & Ba, 2014).
4. Finally, the approach can be extended for parametrizing Stiefel manifolds—nonsquare generalizations of orthogonal matrices. The proposed extension scheme, named “truncated CWY” (or T-CWY), is to our knowledge a novel parametrization of the Stiefel manifold with competitive...
runtime complexity.

We evaluate the CWY parametrization of orthogonal matrices for transition matrices in RNNs on the task of neural machine translation (NMT). We demonstrate the superiority of our approach both in computational efficiency and problem-solving potential compared to other methods for orthogonal optimization, and baselines such as LSTM (Hochreiter & Schmidhuber, 1997) and GRU (Cho et al., 2014). Based on T-CWY we construct a convolutional recurrent unit with guaranteed robustness to gradient explosion and evaluate it on a task of one-step-ahead video prediction. We compare with ConvLSTM (Xingjian et al., 2015), unconstrained baselines and other orthogonal optimization methods. All theoretical results are proven in Section A of the Supplementary materials.

2. Background and Related Work

In this section we outline the problem of gradient explosion and vanishing in RNNs and how it can be addressed using an orthogonal state transition matrix. Then we review earlier optimization methods with orthogonality constraints, and analyse their computational runtime complexities and solution domains (summarized in Tables 1 and 2).

2.1. Gradient Explosion and Vanishing in RNNs

The rollout of a recurrent neural network (RNN) can be formalized as a series of computations (Jordan, 1990):

$$ y_t := W h_{t-1} + b; \quad h_t := \sigma(y_t + V x_t); \quad (1) $$

for $t = 1, \ldots, T$. Here $x_1, \ldots, x_T \in \mathbb{R}^K$ are the states of an observed sequence $X = \{x_1, \ldots, x_T\}$ from the training set, $h_0, \ldots, h_T \in \mathbb{R}^N$ is a sequence of hidden states ($h_0$ is fixed and usually zero), $W \in \mathbb{R}^{N \times N}$ is a transition matrix, $b \in \mathbb{R}^N$ is a bias term, $V \in \mathbb{R}^{N \times K}$ is an input transformation matrix and $\sigma(\cdot)$ is an elementwise nonlinear function. $N$ and $K$ are the dimensions of the hidden and observed states respectively. In this work, we are interested in constraining $W$ to a restricted (orthogonal) form $Q$, which we shall make precise shortly.

Let $C$ denote an objective function to minimize. For the ease of illustration we assume that $C$ is a function of the last hidden state: $C = C(h_T)$. Then one has the following expression for gradients w.r.t. intermediate hidden states:

$$ \frac{\partial C}{\partial h_t} = \left( \prod_{k=t}^{T-1} \frac{\partial h_{k+1}}{\partial h_k} \right) \frac{\partial C}{\partial h_T} = \left( \prod_{k=t}^{T-1} J_\sigma(h_k)W^T \right) \frac{\partial C}{\partial h_T} $$

where $J_\sigma$ is the Jacobian of $\sigma(\cdot)$ applied elementwise. In practice, the expression leads to the hidden state norm increasing exponentially fast with $T - t$ when $\|W\|_2 = \sup\|h\|_2 = 1$ ($\|W\|_2 > 1$ (gradient explosion)) or decreasing exponentially fast when $\|W\|_2 < 1$ (gradient vanishing). Both effects are undesirable as they lead to unstable learning and inability to capture long-term dependencies in the data.

To alleviate the problem of gradient explosion, Arjovsky et al. (2016) proposed using an orthogonal or unitary matrix $W$, that is to say either $W = Q \in \mathcal{O}(N)$ or $W = Q \in \mathcal{U}(N)$. Here $\mathcal{O}(N) = \{Q \in \mathbb{R}^{N \times N} \mid Q^\top Q = I\}$ is called orthogonal group, $\mathcal{U}(N) = \{Q \in \mathbb{C}^{N \times N} \mid Q^H Q = I\}$ is called unitary group, $Q^H$ denotes the conjugate transpose and $I$ denotes an identity matrix, with shape inferred from the context. Since orthogonal or unitary linear operators are $l_2$-norm preserving (i.e. $\forall h : \|Q h\|_2 = \|h\|_2$), the norm of the intermediate state gradient is approximately constant when $J_\sigma(h_k) \approx I$.

Next we discuss approaches to tackle the constrained optimization problem formulated as

$$ \min_{W, V, b} C \quad \text{s.t. } W \in \mathcal{O}(N) \quad \text{or } Q \in \mathcal{U}(N). \quad (2) $$

2.2. Earlier Approaches to Orthogonality-constrained Optimization

We review two families of earlier methods to solve the constrained optimization problem (2).

2.2.1. PARAMETRIZATION

Here $Q$ is constructed as a function of unconstrained parameters, on which the usual gradient descent can be performed.

**URNN** (Unitary Recurrent Neural Network, Arjovsky et al., 2016) where a unitary Q is expressed as

$$ Q = D(3) H(2) F^{-1} D(2) \Pi H(1) FD(1) \in \mathcal{U}(N), $$

where $D(1), D(2), D(3)$ are parametrized diagonal unitary matrices, $H(1), H(2)$ are parametrized Householder reflections (Householder (1958), see the definition below), $F$ is a discrete Fourier transform matrix and $\Pi$ is a random permutation matrix.

**EURNN** (Efficient Unitary RNN, Jing et al., 2016).

$$ Q = DF(1) F^{(2)} \ldots F^{(L)} \in \mathcal{U}(N), $$

where $L \leq N$, $D$ is a diagonal unitary matrix and each $F^{(i)} \in \mathbb{C}^{N \times N}$ is a block-diagonal matrix with $2 \times 2$ parametrized unitary blocks.

**HR** (Householder reflections, Mhammedi et al., 2017). An orthogonal $Q$ is written

$$ Q = H(v^{(1)}) \ldots H(v^{(L)}) \in \mathcal{O}(N), \quad (3) $$

where for each nonzero $v \in \mathbb{R}^N$, $H(v) = I - 2vv^T/\|v\|_2^2$ is an orthogonal matrix.
EXPRNN (Exponent RNN, Casado & Martínez-Rubio, 2019). This method takes advantage of the fact that the matrix exponent \( e^A \) is a surjective mapping from the set of skew-symmetric matrices \( \text{Skew}(N) = \{ A \in \mathbb{R}^{N \times N} \mid A = -A^T \} \) to the special orthogonal group \( O^{+1}(N) \), where for \( s = \pm 1 \) we define \( O^s(N) = \{ Q \in O(N) \mid \det Q = s \} \) (notice that \( O(N) = O^+(N) \cup O^-(N) \)). Skew\((N)\) is a linear space which can be parametrized by elements below the diagonal of \( A \).

SCORN (Skew Cayley, Helfrich et al., 2018). Here the Cayley transform is used instead of matrix exponent:

\[
Q = \text{Cayley}(A) = (I + A/2)^{-1}(I - A/2),
\]

which is a bijective map from \( \text{Skew}(N) \) to \( O^{+1}(N) \). Where \( \Theta \) we denote the set of orthogonal matrices with \(-1\) eigenvalue. To cover all matrices from \( O(N) \), \( Q \) is scaled as \( \tilde{Q} = QD \) where \( D \) is a diagonal matrix with \pm 1 on the diagonal. The number of \(-1\)’s in \( D \) is a hyperparameter, which requires some additional search method. Hence, for fair comparison, we fix \( D = I \).

OWN (Orthogonal Weight Normalization, Huang et al., 2018). This method considers the more general task of optimizing a function over the Stiefel manifold \( \text{St}(N, M) = \{ \Omega \in \mathbb{R}^{N \times M} \mid \Omega^T \Omega = I \} \) where \( M \leq N \), which generalizes the set \( O(N) \). The Stiefel matrix \( \Omega \) is parametrized as

\[
\Omega = \tilde{V}P\tilde{A}^{-1/2}P^T, \quad \tilde{V} = (V - \frac{1}{N}11^TV),
\]

where \( P\tilde{A}P^T \) is an eigenvalue decomposition of matrix \( \tilde{V}^T \tilde{V} \in \mathbb{R}^{M \times M} \) and \( 1 \) is the all-ones vertical \( N \)-vector.

2.2.2. RIEMANNIAN GRADIENT DESCENT (RGD)

These methods instead consider gradient descent directly on the Stiefel manifold. Rather than “straight-line” steps as in typical gradient descent, RGD goes along a curve which a) lies in \( \text{St}(N, M) \) and b) points in the direction of fastest descent along the manifold. More precisely, RGD starts with a predefined matrix \( \Omega(0) \in \text{St}(N, M) \) and makes sequential updates of the type \( \tilde{\Omega}(k) \coloneqq g_k(\eta_k) \) where \( \eta_k \) is a step size, \( g_k : \mathbb{R} \to \text{St}(N, M) \), \( g_k(0) = \Omega(k-1) \) and \( g_k(\eta) \) is the gradient \( \frac{\partial f}{\partial \Omega}(\Omega(k-1)) \) projected onto the tangent space \( T_{\Omega(k-1)} \) —a linear space approximating the Stiefel manifold \( \text{St}(N, M) \) at the point \( \Omega(k-1) \). It is known that \( T_{\Omega} = \{ Z \in \mathbb{R}^{N \times M} \mid Z^T \Omega \in \text{Skew}(M) \} \). For a rigorous introduction to Riemannian manifolds and Riemannian Gradient Descent see (Absil et al., 2007).

In a Riemannian manifold, the tangent space \( T_{\Omega} \) must have an inner product, usually chosen as either the canonical inner product \( \langle Z_1, Z_2 \rangle_1 = \text{Tr}(Z_1^T(I - \frac{1}{2}\Omega\Omega^T)Z_2) \) or Euclidean inner product \( \langle Z_1, Z_2 \rangle_2 = \text{Tr}(Z_1^T Z_2) \).

Consequently, the projection of the gradient has the form:

\[
g_k'(0) = A(k-1)\Omega(k-1), \quad A(k-1) = \hat{A}(k-1) - \hat{A}(k-1)^T,
\]

where \( \hat{A}(k-1) = \frac{\partial f}{\partial \Omega}(\Omega(k-1))\Omega(k-1)^T \) corresponds to the canonical inner product choice, and

\[
\hat{A}_2(k-1) = \hat{A}_1(k-1) - \frac{1}{2}(\Omega(k-1)^T\Omega(k-1)^T)\hat{A}_1(k-1)
\]

corresponds to the Euclidean inner product choice. Next, there is freedom in choosing the type of \( g_k(\eta) \) function. Two popular choices are 1) Cayley retraction:

\[
g_k^{\text{Cay}}(\eta) = \text{Cayley}(\eta A(k-1)^T\Omega(k-1)),
\]

and 2) QR-Decomposition-Type retraction:

\[
g_k^{\text{QR}}(\eta) = q\text{f}(\eta A(k-1)^T\Omega(k-1)),
\]

where \( q\text{f}(\cdot) \) denotes a Q matrix of the argument’s QR decomposition so that diagonal elements of the R matrix are positive. Wisdom et al. (2016) evaluate performance of complex number RGD with canonical inner product in the context of RNNs with unitary transition matrix. Also, Li et al. (2020) employ the Euclidean inner product version of RGD in the context of deep learning.

2.3. Runtime Complexity and Expressiveness

We compare the runtime complexity of different methods to train orthogonal RNNs in Table 1 (we introduce the notation \( O_L(N) \) later in this section). We measure complexity of a single forward pass and show the comparison with respect to the domain covered by optimization approach. Square brackets indicate terms which can be significantly reduced by computing matrix products in parallel. The runtime complexity and parallelization pattern of the backward pass coincides with those of the forward pass.

Row “RNN” indicates the complexity of an unconstrained recurrent neural network. We aim for complexity estimates close to that of RNN while ideally covering the set of all orthogonal matrices \( O(N) \). Mhammedi et al. (2017) show that any RNN with a unitary transition matrix can be efficiently modelled by a different network with orthogonal weights. Hence, we opt for simplification by only covering the orthogonal group \( O(N) \).

URNN, while being computationally efficient, uses \( 7N \) parameters when building unitary matrices. As noted by Wisdom et al. (2016), this is not enough to cover all matrices from \( U(N) \), which is an \( N^2 \)-dimensional manifold.

To map parameters to an orthogonal matrix, SCORN and EXPRNN use the matrix exponent or Cayley transform at every step. Hence, they require an additional \( O(N^3) \) term in the complexity estimate. Similarly, RGD must apply the
Although EURNN and HR methods don’t have an \( O(N^3) \) term in runtime complexity, at each timestep they require \( L \) consecutive matrix-vector products which cannot be run in parallel. This becomes a problem when \( N \) is big and, thus, bigger \( L \) is needed to obtain good expressiveness of the orthogonal RNN. We use the notation \( O_L(N) \) for the set of orthogonal matrices which can be obtained with \( L \) Householder reflections: \( O_L(N) = \{ H(v^{(1)}) \cdots H(v^{(L)}) \mid \forall i : v^{(i)} \in \mathbb{R}^N \setminus \{0\} \} \).

Table 2 summarizes the runtime complexity of Stiefel manifold optimization approaches. Both OWN and RGD cover all matrices from \( \text{St}(N,M) \) when \( M < N \). OWN requires an eigenvalue decomposition of dense \( M \times M \)-sized matrix which is a cubic operation. RGD with Cayley retraction (6-7) requires to invert \( N \times N \)-sized matrix \( \eta_k \) thus becoming cubic in \( N \). To make the computation more tractable, Tagare (2011) proposes to use the Sherman-Morrison-Woodbury formula which reduces the size of the inverted matrix to \( 2M \times 2M \) when the canonical inner product is chosen for RGD. A straightforward extension of Tagare’s approach to the Euclidean inner product would require to invert \( 3M \times 3M \)-sized matrix. We put derivations of Tagare (2011) and their extension for Euclidean inner product into Section B of the Supplementary materials.

\[ \text{Table 2. Stiefel manifold runtime complexity of performing a single gradient step when optimizing over } \Omega \in \text{St}(N,M). \text{ In the notation “RGD-A-B” “A” is CAN or EUCL for canonical or Euclidean inner product choice respectively, and “B” is CAY or QR for Cayley or QR-Decomposition-Type retraction respectively. The term related to computing the objective function and } \Omega \text{’s gradient is omitted.} \]
3. Proposed Approach: Simple and Efficient Parametrization Scheme

We show how a series of sequential Householder reflections can be computed in parallel using the compact WY (CWY) transform. Next, we propose an extension of the CWY transform and a novel parametrization of the Stiefel manifold. Finally, we propose a novel convolutional recurrent unit to handle spatio-temporal data.

3.1. Accumulating Householder Reflections through the Compact WY (CWY) Transform

We suggest an alternative algorithm to compute the composition of $L$ Householder reflections. Our approach can compute a series of reflections in parallel on GPU or TPU thus increasing the effectiveness of RNN rollout in terms of floating point operations per second (FLOPS). The approach is called the compact WY (CWY) transform (Joffrain et al., 2006), and to our knowledge, has not been applied previously in machine learning. Mhammedi et al. (2017) used CWY only for theoretical reasoning about backpropagation—they used the explicit Householder series (3) in experiments.

**Lemma 2** (adapted from Joffrain et al., 2006). Let $v^{(1)}, \ldots, v^{(L)} \in \mathbb{R}^N$ be nonzero vectors. Then it holds that

$$H(v^{(1)}) \cdots H(v^{(L)}) = I - US^{-1}U^\top,$$  \hspace{1cm} (8)

where $U = \begin{bmatrix} v^{(1)}_{\|v^{(1)}\|_2} & \cdots & v^{(L)}_{\|v^{(L)}\|_2} \end{bmatrix} \in \mathbb{R}^{N \times L}$, and

$$S = \frac{1}{2} I + \text{striu}(U^\top U),$$  \hspace{1cm} (9)

where striu returns an argument matrix with all diagonal and lower-triangular elements zeroed out.

We store $v^{(1)}, \ldots, v^{(L)}$ as learnable parameters and suggest two ways of computing the CWY transform in practice depending on user-specified $L$:

**Full expressiveness**, $L = N$. Here we precompute the transition matrix (8) into $Q$, then perform the RNN rollout as usual. See the complexity estimate of this scenario in Table 1, row $L \leq N$ REFLECTIONS. Since the right hand side of (8) takes $O(N^3)$ time to compute, the complexity estimate coincides with SCORN, RGD and EXPRNN, while empirically CWY is better at modelling the RNN transition matrix as we show in experimental results (Section 4).

**Flexible expressiveness**, $L < N$. Here we don’t compute and store $Q = I - US^{-1}U^\top$ explicitly. Instead, before each RNN rollout, we precompute $U$ and $S^{-1}$ and expand Equation (1, left) into the following computations:

$$u_t := U^\top h_{t-1}; \quad v_t := S^{-1}u_t; \quad y_t := h_{t-1} - Uv_t + b.$$  \hspace{1cm} (10)

which has two matrix-vector products with matrices of size $L \times N$ and $N \times L$. Altogether this results in the complexity estimate shown in Table 1, row $L < N$ REFLECTIONS.

The better parallelization pattern of the CWY transform comes with a price of an additional $L^3$ term related to inverting the $S$ matrix. In practice, we find that for moderate $L$ this addition is comparable to the rollout cost. Observe that for the CWY approach, the inverted matrix $S$ is upper-triangular. Hence, it takes constant time less floating operations to invert $S$ (Hunger, 2005) than to find inverse or exponent of a dense matrix of the same size, as in EXPRNN, SCORN and RGD with the Cayley transform.

3.2. CWY Extension: Truncated CWY (T-CWY)

We extend our approach and propose, to our knowledge, a novel parametrization of the Stiefel manifold $St(N,M)$ which we call the truncated CWY (T-CWY) transform. The next lemma suggests a way to parametrize any Stiefel manifold $St(N,M)$ with $M < N$ by $\mathbb{R}^{N \times M}$ minus a zero-measure set.

**Lemma 3.** Consider $M < N$ and a function $\gamma_{N,M} : \left(\mathbb{R}^{N} \setminus \{0\}\right)^M \to \mathbb{R}^{N \times M}$ defined as follows. For $v^{(1)}, \ldots, v^{(M)} \in \mathbb{R}^N$ construct a matrix

$$U = \begin{bmatrix} v^{(1)}_{\|v^{(1)}\|_2} & \cdots & v^{(M)}_{\|v^{(M)}\|_2} \end{bmatrix} \in \mathbb{R}^{N \times M},$$  \hspace{1cm} (11)

and assign

$$\gamma_{N,M}(v^{(1)}, \ldots, v^{(M)}) = \begin{bmatrix} I \quad 0 \end{bmatrix} - US^{-1}U_1^\top \in \mathbb{R}^{N \times M},$$  \hspace{1cm} (12)

where $U_1$ is an upper $M \times M$ submatrix of $U$ and

$$S = \frac{1}{2} I + \text{striu}(U^\top U).$$  \hspace{1cm} (13)

Then $\gamma_{N,M}$ is a surjective mapping to $St(N,M)$.

The runtime complexity of the T-CWY map is indicated in Table 2. The asymptotic behaviour matches Cayley-RGD and OWN, while the number of floating point operations is constant time smaller due to the inverted matrix $S$ size and upper-triangular structure.

3.3. Convolutional Non-exploding Recurrent Unit (ConvNERU)

Based on the suggested Stiefel matrix parametrization, we propose a novel convolutional recurrent module for modelling sequential data with spatial structure. Given a sequence of images $X_1, \ldots, X_T \in \mathbb{R}^{H\times W\times F_{in}}$, a proposed convolutional non-exploding recurrent unit (ConvNERU) is the following modification of (1):

$$Y_t := K \ast G^{(t-1)} + B; \quad G^{(t)} := \sigma(Y_t + K^{in} \ast X_t);$$  \hspace{1cm} (14)
where $G^{(0)}, \ldots, G^{(T)} \in \mathbb{R}^{h \times w \times f_{out}}$ are hidden states, $B \in \mathbb{R}^{h \times w \times f_{out}}$ is a bias tensor which is parametrized by $b \in \mathbb{R}^{f_{out}}$ so that $b = B_{i,j}$ for any $i,j$, $\sigma$ is an element-wise nonlinearity, “$\ast$” denotes convolution operation and $K \in \mathbb{R}^{q \times q \times f_{out} \times f_{out}}$, $K^{in} \in \mathbb{R}^{q \times q \times f_{in} \times f_{out}}$ are convolution kernels with $q$ being kernel size.

Denote by $\hat{K}$ a $(q^2 f_{out} \times f_{out})$-sized matrix such that for any $l, p \leq q$ and $i, j \leq f_{out}$ it holds that $\hat{K}_{lqf_{out}+p,f_{out}+i,j} = K_{l,p,i,j}$. We equip ConvNERU with a constraint $q\hat{K} \in \text{St}(q^2 f_{out} \times f_{out})$ which is implemented by T-CWY parametrization of the $q\hat{K}$ matrix. In Section C of the Supplementary materials we theoretically show that ConvNERU is resistant to gradient explosion.

4. Experiments

4.1. Neural Machine Translation

4.1.1. Motivation

We consider the task of supervised sequence-to-sequence modeling with Recurrent Neural Networks in the Natural Language Processing (NLP) context. Many NLP tasks such as Machine Translation require models to consider contextual connections across long sequences. However, as has been shown (Hochreiter, 1998). RNN architectures can suffer from vanishing/exploding gradient as the recurrent kernel is applied repeatedly during unrolling.

Augmented RNN architectures such as LSTMs (Hochreiter & Schmidhuber, 1997) leverage custom gating procedures to combat gradient propagation issues and help to retain critical context across long sequences. These cells enable the LSTM to ’memorize’ information but such gating mechanisms have limitations. In particular, the gating schemes of GRUs and LSTMs employ more computations per time step and require additional parameters that increase memory and computational demands.

However, by efficiently parametrizing standard RNN weight matrices as matrices on the orthogonal group, we aim to similarly retain context, but with faster time step updates, fewer parameters, and lower memory requirements.

4.1.2. Preprocessing and Experimental Setup

To demonstrate the usefulness of orthogonality, we train an orthogonal RNN-based Encoder-Decoder Seq2Seq model with Attention (Bahdanau et al., 2014) to translate sentence pairs between a given source and target language. See Section E of Supplementary materials for architecture illustration and hyperparameter information. We focus on the English-to-Spanish dataset within the Tatoeba corpus (Artetxe & Schwenk, 2019), a publicly available dataset with over 100,000 sentence pairs.

4.1.3. Experimental Results

From the translation tasks, we take aligned bi-texts between the source and target languages and, as preprocessing, remove accents and return word pairs in the form [English, Spanish]. The resulting dataset has an average sequence length of $\approx 17$ for both the input and target sequences.

Using a single Tensor Processing Unit (TPU) per model, we train several models from scratch, with no pre-training, on 80,000+ sentence pairs and test on the remaining 20,000+ pairs from the full 100,000+ pair dataset to compare their learning capabilities and stability. Given that we evaluate all models on the same corpus and that our goal is to benchmark across architectures, we elected to employ no pre-training and examine/compare cross-entropy loss directly.

Table 3. Tatoeba Spa-to-Eng NMT results. We compare the CWY with standard RNNs, GRUs and LSTMs, and other orthogonality approaches. We find that the CWY achieves the best test performance while preserving speed and requiring the fewest parameters. Note also that there is a sweet-spot for parameter $L$ for test loss minimization ($L = 128$). We conjecture that this illustrates the trade-off between the capacity of the model (which increases with larger values of $L$) and the landscape of the objective function (that simplifies with smaller values of $L$).

| Model       | Test CE Loss | Time (min) | Params |
|-------------|--------------|------------|--------|
| RNN         | 0.73         | 148        | $\approx 25M$ |
| GRU         | 0.56         | 173        | $\approx 32M$ |
| LSTM        | 0.55         | 232        | $\approx 37M$ |
| SCORN       | 0.58         | 1780       | $\approx 25M$ |
| RGD for O(N)| 2.01         | 1780       | $\approx 25M$ |
| EXPRNN      | 0.59         | 2960       | $\approx 25M$ |

| CWY, $L=1024$ | 0.56 | 1111 | $\approx 25M$ |
| CWY, $L=512$  | 0.66 | 338  | $\approx 24M$ |
| CWY, $L=256$  | 0.64 | 213  | $\approx 23M$ |
| CWY, $L=128$  | 0.50 | 198  | $\approx 23M$ |
| CWY, $L=64$   | 0.60 | 175  | $\approx 23M$ |

As noted by Li et al. (2020), all standard orthogonal baseline architectures for Riemannian Gradient Descent are not trivially compatible with momentum based optimizers, so we leverage SGD. However, for the CWY and non-orthogonal variants, we conduct experiments with both Adam and standard SGD, and show results using Adam, finding that this optimizer produces the best results.
We find that standard RNNs underperform LSTMs and GRUs, but that parametrization-based orthogonal RNN variants are able to achieve comparable performance (see Table 3). Among orthogonal RNN methods, our CWY approaches achieve the lowest test cross-entropy, whilst requiring the fewest parameters and, via our efficient parametrization, retaining training speed comparable to LSTMs and GRUs. Interestingly, we observed that there is a sweet-spot for choosing \( L \) to minimize test loss which, as we conjecture, is caused by a trade-off between architecture expressiveness (that increases as \( L \) increases) and the landscape of the objective function to be optimized (that simplifies as \( L \) decreases).

As mentioned in Section 3.1, in exact arithmetic our CWY is equivalent to the explicit Householder reflections approach leveraged by (Joffrain et al., 2006); however, our approach achieves far superior speed, as illustrated in Figure 1. The enhanced speed of our CWY variants, when paired with the flexibility that it grants w.r.t. the optimizer, makes this approach a compelling alternative to LSTMs and GRUs.

![Figure 1](image.png)

Figure 1. The CWY and HR methods are numerically equivalent; however, the parametrization of the CWY allows us to perform projections much more efficiently, leading to dramatic improvements in training time and, thereby, practical viability.

### 4.2. Video Prediction with ConvNERU

#### 4.2.1. Motivation

We demonstrate performance of ConvNERU in the task of one-step-ahead video prediction on the KTH action dataset. As a baseline we chose ConvLSTM (Xingjian et al., 2015) which is an adaptation of LSTM for modelling spatiotemporal data (e.g. with linear transforms substituted by convolutions). ConvLSTM and its modifications are employed in various applications dealing with spatiotemporal data, e.g. precipitation forecasting (Xingjian et al., 2015; Wang et al., 2017), video prediction (Finn et al., 2016; Lee et al., 2018; Ebert et al., 2017), video segmentation (Pfeuffer et al., 2019) and salient object detection (Song et al., 2018). In addition, our goal is to compare with other methods for Stiefel optimization and justify the need of Stiefel constraints in the first place.

#### 4.2.2. KTH Action Dataset and Preprocessing

We conduct experiments on the KTH action dataset (Schnadt et al., 2004) containing grey scale video recordings of 25 persons each performing 6 types of actions: walking, jogging, running, boxing, hand waving and hand clapping. All videos, 4 seconds in average, are recorded with a static camera with 25 fps frame rate and frame size of 160 × 120 pixels. We crop and resize each frame into \( 128 \times 128 \) pixels and then reshape each frame into \( 64 \times 64 \times 4 \) by moving groups of \( 2 \times 2 \) pixels into channel dimension. Since each video sequence has a different number of frames, we employ zero padding during batch construction. We use persons with indices 1-12 for training, 13-16 for validation and 17-25 for testing (see result split sizes in Section F of the Supplement).

We do separate evaluations for each action type to evaluate how the model learns different types of dynamics.

#### 4.2.3. Experimental Setup

Given a sequence of known frames \( \mathcal{I}^{(1)}, \ldots, \mathcal{I}^{(t)} \in [0, 1]^{64 \times 64 \times 4} \), the network outputs a prediction \( \hat{\mathcal{I}}^{(t+1)} \) of the next frame \( \mathcal{I}^{(t+1)} \). The network is designed as a recurrent block composed of several convolutional recurrent units stacked together with the sequence \( \{\mathcal{I}^{(i)}\}_{i=1}^{t} \) passed to the input. In order to increase the receptive field of the recurrent architecture while maintaining a tractable training procedure, we adapt a simplified version of the video prediction architecture from (Lee et al., 2018; Ebert et al., 2017).

Namely, we stack several recurrent units with a bottleneck structure (hidden sizes \( 32 \times 32 \times 32 \rightarrow 16 \times 16 \times 64 \rightarrow 8 \times 8 \times 128 \rightarrow 16 \times 16 \times 64 \rightarrow 32 \times 32 \times 32 \) and skip connections. We alternate recurrent layers with strided convolutions and then deconvolutions. After each convolution and deconvolution we place a ReLU nonlinearity, as well as using ReLU as the recurrent nonlinearity \( \sigma \). See Section G in the Supplement for an illustration of the architecture.

In the proposed architecture a prediction \( \hat{\mathcal{I}}^{(t+1)} \) is conditioned upon \( \mathcal{I}^{(t)} \) through bottleneck and skip connections and conditioned upon \( \{\mathcal{I}^{(i')}\}_{t<i} \) through recurrent temporal connections. We opt for minimizing the \( l_1 \)-loss \( \|\hat{\mathcal{I}}^{(t+1)} - \mathcal{I}^{(t+1)}\| \) (Mean Absolute Error) during training. We compare different designs of recurrent unit used in the full architecture:

1. **ConvLSTM** was used in the original variant of the architecture (Lee et al., 2018; Ebert et al., 2017). As in the original papers, we find that the model works best when instance normalization (Ulyanov et al., 2016) is added after each convolution and before the nonlinearity, including convolutions inside ConvLSTM. We don’t use instance normalization with other model variants.

2. **Zeros** indicates a model with recurrence of type (13) with
Table 4. KTH action dataset test results. The indicated metric is average per-frame $l_1$-loss $|\hat{\mathcal{J}}(t+1) - \mathcal{J}(t+1)|$ computed on the test set. Video frames are in grayscale with brightness ranged in $[0, 1]$. The GPU memory is evaluated for the "Boxing" class which has longest sequences. We don’t report last columns for “Zeros” method which is only aimed to demonstrate importance of recurrent connections.

| METHOD       | WALKING | JOGGING | RUNNING | BOXING | WAVING | CLAPPING | # PARAMS | GPU MEMORY |
|--------------|---------|---------|---------|--------|--------|----------|----------|------------|
| ConvLSTM     | 223.3   | 266.8   | 297.8   | 188.9  | 157.9  | 162.3    | ≈ 3.26 M | 8.7 GB     |
| Zeros        | 160.3   | 176.1   | 203.8   | 179.0  | 197.2  | 147.4    | —        | —          |
| Glorot-Init  | 145.8   | 161.5   | 182.1   | 179.9  | 164.5  | 145.4    | ≈ 0.72 M | 3.5 GB     |
| Orth-Init    | 139.9   | 153.2   | 175.0   | 173.3  | 150.8  | 144.0    | As above | As above   |
| RGD-CAN-CAY  | 135.8   | 155.7   | 170.7   | 172.9  | 160.3  | 144.5    | As above | As above   |
| RGD-EUCL-CAY | 143.3   | 152.5   | 173.7   | 171.9  | 172.9  | 142.6    | As above | As above   |
| RGD-CAN-QR   | 143.1   | 155.0   | 171.5   | 173.1  | 150.2  | 142.7    | As above | As above   |
| RGD-EUCL-QR  | 135.5   | 153.9   | 169.6   | 169.9  | 160.4  | 142.5    | As above | As above   |
| RGD-ADAM     | 142.6   | 157.3   | 177.8   | 176.8  | 159.1  | 145.2    | As above | As above   |
| OWN          | 137.5   | 155.0   | 177.7   | 171.3  | 149.8  | 142.5    | As above | As above   |
| T-CWY        | **134.6** | **149.8** | **166.7** | **166.2** | **147.8** | **141.2** | As above | As above   |

Figure 2. Validation $l_1$-loss $|\hat{\mathcal{J}}(t+1) - \mathcal{J}(t+1)|$ depending on the training epoch. Mean and standard error across each 10 epochs is reported.

transition kernel $K$ zeroed out so that the model is only using a single previous frame for prediction. We include this variant to evaluate importance of recurrent connections.

3. **Glorot-Init** is an unconstrained model with recurrence of type (13) where $K$ initialized through Glorot uniform initialization (Glorot & Bengio, 2010).

4. **Orth-Init** indicates using unconstrained recurrence of type (13) with $q\hat{K}$ initialized as a Stiefel matrix. For that we initialize a matrix with Glorot uniform, find its QR decomposition and take Q matrix as initializer of $q\hat{K}$.

5. **RGD-**.* indicates Stiefel RGD for optimizing $q\hat{K}$ with various combinations of inner product and retractor (consistent with notation in Table 2).

6. **RGD-Adam** is ADAM adaptation of RGD proposed in (Li et al., 2020) applied to optimization of $q\hat{K}$.

7. Finally, **OWN** and **T-CWY** indicate ConvNERU with $q\hat{K}$ matrix parametrized by OWN and T-CWY respectively.

4.2.4. **Experimental Results**

We opt for batch size of 3, recurrent kernel size $q = 3$, learning rate of $10^{-3}$. For all unconstrained parameters we use Adam optimizer. Our experiments are implemented in TensorFlow and run on a single Nvidia Tesla P100 GPU for each experiment. For each experiment we run 150 epochs and choose the model showing smallest validation loss value for testing. Table 4 demonstrates test $l_1$-loss, number of parameters and maximal GPU memory consumption. Additionally, Figure 2 demonstrates validation $l_1$-loss depending on epoch number for a subgroup of evaluated methods.

We see from the figure that in most cases, with the same learning rate, ConvLSTM cannot outperform “Zeros” baseline which has no recurrence and, hence, doesn’t face an issue of gradient explosion or vanishing. Among the versions of ConvNERU and its unconstrained analogs, we observe that T-CWY performs best on both validation and test set, showing the fastest descent to problem solution. While performing better than ConvLSTM, T-CWY also has several times less parameters and is using much less GPU memory.
5. Conclusion

We introduced an efficient scheme for parametrizing orthogonal groups $O(N)$ and Stiefel manifolds $\text{St}(N,M)$, and compared to earlier approaches. The proposed $O(N)$-parametrization scheme is especially efficient when working with large-scale orthogonal matrices on a parallelized computation unit such as GPU or TPU. We empirically demonstrated strong performance in real-life applications. In future work, we plan to explore opportunities for flexible expressiveness of Stiefel matrices parametrized by T-CWY, in a similar fashion as we proposed for the CWY.

6. Acknowledgements

Adrian Weller acknowledges support from the David MacKay Newton research fellowship at Darwin College, The Alan Turing Institute under EPSRC grant EP/N510129/1 and U/B/000074, and the Leverhulme Trust via CFI.

References

Absil, P.-A., Mahony, R., and Sepulchre, R. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, Princeton, NJ, USA, 2007. ISBN 0691132984, 9780691132983.

Arjovsky, M., Shah, A., and Bengio, Y. Unitary evolution recurrent neural networks. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML’16, pp. 1120–1128. JMLR.org, 2016. URL http://dl.acm.org/citation.cfm?id=3045390.3045509.

Artetxe, M. and Schwenk, H. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *Transactions of the Association for Computational Linguistics*, 7:597–610, 2019.

Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Bansal, N., Chen, X., and Wang, Z. Can we gain more from orthogonality regularizations in training deep networks? In *Advances in Neural Information Processing Systems*, pp. 4261–4271, 2018.

Casado, M. L. and Martínez-Rubio, D. Cheap orthogonal constraints in neural networks: A simple parametrization of the orthogonal and unitary group. *CoRR*, abs/1901.08428, 2019. URL http://arxiv.org/abs/1901.08428.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Cisse, M., Bojanowski, P., Grave, E., Dauphin, Y., and Usunier, N. Parseval networks: Improving robustness to adversarial examples. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 854–863. JMLR. org, 2017.

Dorobantu, V., Stromhaug, P. A., and Renteria, J. Dizzyrnn: Reparameterizing recurrent neural networks for norm-preserving backpropagation. *arXiv preprint arXiv:1612.04035*, 2016.

Ebert, F., Finn, C., Lee, A. X., and Levine, S. Self-supervised visual planning with temporal skip connections. *CoRR*, abs/1710.05268, 2017. URL http://arxiv.org/abs/1710.05268.

Finn, C., Goodfellow, I., and Levine, S. Unsupervised learning for physical interaction through video prediction. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS16, pp. 6472, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.

Gallier, J. *Geometric Methods and Applications: For Computer Science and Engineering*. Texts in Applied Mathematics. Springer New York, 2011. ISBN 97814414999610. URL https://books.google.co.uk/books?id=4v5VOTZ-vMcC.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

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.

Hochreiter, S. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL http://dx.doi.org/10.1162/neco.1997.9.8.1735.
Householder, A. S. Unitary triangularization of a non-symmetric matrix. *J. ACM*, 5(4):339–342, October 1958. ISSN 0004-5411. doi: 10.1145/320941.320947. URL http://doi.acm.org/10.1145/320941.320947.

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. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Hunger, R. *Floating Point Operations in Matrix-vector Calculus*. Munich University of Technology, Inst. for Circuit Theory and Signal Processing, 2005. URL https://books.google.co.uk/books?id=EccIcgAACAAJ.

Jing, L., Shen, Y., Dubcek, T., Peurifoy, J., Skirlo, S. A., Tegmark, M., and Soljacic, M. Tunable efficient unitary neural networks (EUNN) and their application to RNN. *CoRR*, abs/1612.05231, 2016. URL http://arxiv.org/abs/1612.05231.

Joffrain, T., Low, T. M., Quintana-Ortí, E. S., Geijn, R. v. d., and Zee, F. G. V. Accumulating Householder transformations, revisited. *ACM Trans. Math. Softw.*, 32(2):169–179, June 2006. ISSN 0098-3500. doi: 10.1145/1141885.1141886. URL http://doi.acm.org/10.1145/1141885.1141886.

Jordan, M. I. Attractor dynamics and parallelism in a connectionist sequential machine. In *Artificial neural networks: concept learning*, pp. 112–127. 1990.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization, 2014.

Lee, A. X., Zhang, R., Ebert, F., Abbeel, P., Finn, C., and Levine, S. Stochastic adversarial video prediction. *CoRR*, abs/1804.01523, 2018. URL http://arxiv.org/abs/1804.01523.

Li, J., Li, F., and Todorovic, S. Efficient Riemannian optimization on the Stiefel manifold via the Cayley transform. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HJxV-ANKDH.

Mhammedi, Z., Hellicar, A., Rahman, A., and Bailey, J. Efficient orthogonal parametrisation of recurrent neural networks using Householder reflections. In *Proceedings of the 34th International Conference on Machine Learning*, Volume 70, pp. 2401–2409. JMLR. org, 2017.

Pfeuffer, A., Schulz, K., and Dietmayer, K. Semantic segmentation of video sequences with convolutional lstms. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1441–1447. IEEE, 2019.

Schldt, C., Laptev, I., and Caputo, B. Recognizing human actions: a local SVM approach. In *Proc. Int. Conf. Pattern Recognition (ICPR’04)*, Cambridge, U.K. 2004.

Song, H., Wang, W., Zhao, S., Shen, J., and Lam, K.-M. Pyramid dilated deeper convlstm for video salient object detection. In Ferrari, V., Hebert, M., Sminchisescu, C., and Weiss, Y. (eds.), *Computer Vision – ECCV 2018*, pp. 744–760, Cham, 2018. Springer International Publishing. ISBN 978-3-030-01252-6.

Tagare, H. D. Notes on optimization on Stiefel manifolds. 2011.

Tomczak, J. M. and Welling, M. Improving variational auto-encoders using Householder flow. *CoRR*, abs/1611.09630, 2016. URL http://arxiv.org/abs/1611.09630.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. S. Instance normalization: The missing ingredient for fast stylization. *CoRR*, abs/1607.08022, 2016. URL http://arxiv.org/abs/1607.08022.

Wang, Y., Long, M., Wang, J., Gao, Z., and Yu, P. S. Predrnn: Recurrent neural networks for predictive learning using spatiotemporal lstms. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 879–888. Curran Associates, Inc., 2017.

Wisdom, S., Powers, T., Hershey, J. R., Roux, J. L., and Atlas, L. Full-capacity unitary recurrent neural networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS’16, pp. 4887–4895, USA, 2016. Curran Associates Inc. ISBN 978-1-5108-3881-9. URL http://dl.acm.org/citation.cfm?id=3157382.3157643.

Xingjian, S., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., and Woo, W.-c. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*, pp. 802–810, 2015.
A. Proofs

A.1. Lemma 1

Proof. The proof proceeds by induction in \( N \). For \( N = 1 \) such \( Q \) is unique and is equal to \([-1]\). So simply take \( u_1 = [-1] \).

Now assume the statement is true for \( N = k - 1 \geq 1 \). When \( N = k > 1 \) we consider \( Q \)'s first column \( q = [q_1 \ldots q_N]^\top \) and define a vector \( v \in \mathbb{R}^{k} \) as follows:

\[
v = \begin{cases} 
    \frac{q - e^{(1)}}{\|q - e^{(1)}\|} & \text{if } |q_1| < 1 \\
    0 \ldots 0 1 \end{cases}^\top \quad \text{if } q_1 = 1 \\
    e^{(1)} & \text{if } q_1 = -1
\]

Observe that

\[
H(v)Q = \begin{bmatrix} 1 & r^\top \\
0 & Q' \end{bmatrix}
\]

for some \( r \in \mathbb{R}^{k-1} \). From the fact that \( H(v)Q \in O(k) \) we deduce:

\[
\begin{bmatrix} 1 & r \\
0 & Q' \end{bmatrix} \begin{bmatrix} 1 & r^\top \\
0 & Q' \end{bmatrix} = \begin{bmatrix} 1 & r^\top Q' + rr^\top \\
0 & Q'^\top \end{bmatrix} = I
\]

Hence, \( r = 0 \) and \( Q' \in O(k-1) \). By Sylvester determinant identity \( \det(I - 2vv^\top) = 1 - 2v^\top v = -1 \), therefore \( \det Q' = (-1)^{k-1} \). By the induction step assumption there exist nonzero \( v^{(1)}, \ldots, v^{(k-1)} \in \mathbb{R}^{k-1} \) s.t.

\[
Q' = H(v^{(1)}) \ldots H(v^{(k-1)})
\]

We define \( v^{(2)} = \begin{bmatrix} 0 & v^{(1)} \end{bmatrix}^\top, \ldots, v^{(k)} = \begin{bmatrix} 0 & v^{(k-1)} \end{bmatrix}^\top \) and obtain that

\[
H(v)Q = H(v^{(2)}) \ldots H(v^{(k)})
\]

Finally, we define \( v^{(1)} = v \), left-multiply (17) by \( H(v^{(1)}) \) and complete the induction step.

A.2. Lemma 2

Proof. First, observe that \( S \) is upper-triangular matrix with \( \frac{1}{2} \) on the diagonal. Hence, it is nonsingular and the lemma statement is valid. Now the proof proceeds by induction in \( L \). For \( L = 1 \) lemma is trivial. Suppose lemma is true for \( L = k - 1 \). Then the following is true:

\[
H(v^{(1)}) \ldots H(v^{(k-1)}) = I - U^\top S^{(k-1)} U^\top
\]

where \( U' = \begin{bmatrix} v^{(1)} \|v^{(1)}\| \ldots v^{(k-1)} \|v^{(k-1)}\| \end{bmatrix} \) and

\[
S' = \frac{1}{2} I + \text{striu}(U'^\top U')
\]

Then for \( L = k \) we get:

\[
H(v^{(1)}) \ldots H(v^{(k)}) = (I - U'^\top S^{(k-1)} U'^\top)H(v^{(k)}) = I - U'^\top S^{(k-1)} U'^\top - 2 \frac{v^{(k)} v^{(k)}^\top}{\|v^{(k)}\|^2} + 2 U'^\top S^{(k-1)} U'^\top \frac{v^{(k)} v^{(k)}^\top}{\|v^{(k)}\|^2}
\]

And the step of induction is completed by observing that

\[
\begin{bmatrix} S^{(k-1)} & -2 S^{(k-1)} U'^\top \frac{v^{(k)} \|v^{(k)}\|^2}{\|v^{(k)}\|^2} \\
0 & 2 \end{bmatrix} \times \begin{bmatrix} S^{(k-1)} & -2 S^{(k-1)} U'^\top \frac{v^{(k)} \|v^{(k)}\|^2}{\|v^{(k)}\|^2} \\
0 & 2 \end{bmatrix} \times \begin{bmatrix} S' & U'^\top \frac{v^{(k)} \|v^{(k)}\|^2}{\|v^{(k)}\|^2} \\
0 & \frac{1}{2} \end{bmatrix} = I
\]
A.3. Lemma 3

Proof. Similarly to Lemma 2, observe that $S$ is upper-triangular matrix with $\frac{1}{2}$ on the diagonal. Hence, it is nonsingular and the lemma statement is valid.

Observe that for any nonzero vectors $v^{(1)}, \ldots, v^{(M)} \in \mathbb{R}^N$

$$\begin{bmatrix} I & -US^{-1}U_1^T \end{bmatrix}^T \begin{bmatrix} I & -US^{-1}U_1^T \end{bmatrix} = I + U_1 \left( S^{-T}U^TUS^{-1} - S^{-1} - S^{-T} \right)U_1^T$$

$$= I + U_1 S^{-T} \left( U^T U - S \right) S^{-1} U_1^T = I$$

Hence, $\gamma_{N,M}(v^{(1)}, \ldots, v^{(M)}) \in \text{St}(N, M)$. To show surjectivity of $\gamma_{N,M}$, consider arbitarary $\Omega \in \text{St}(N, M)$. Let $q = [q_1 \ldots q_N]^T$ be $\Omega$’s first column. We consider value $v$ defined by (14). Using derivations similar to (15-16), we obtain:

$$H(v)\Omega = \begin{bmatrix} 1 & 0 \\ 0 & \Omega' \end{bmatrix}$$

where $\Omega' \in \text{St}(N - 1, M - 1)$.

Set $v^{(1)} = v$. Analogously find $v'$ for $\Omega'$ such that

$$H(v')\Omega' = \begin{bmatrix} 1 & 0 \\ 0 & \Omega'' \end{bmatrix}$$

and set $v^{(2)} = [0 \ v'^T]^T$. Repeat this procedure $M - 2$ more times to obtain:

$$H(v^{(M)})\ldots H(v^{(1)})\Omega = \begin{bmatrix} I \\ 0 \end{bmatrix} \quad (18)$$

Left-multiply (18) by $H(v^{(1)})\ldots H(v^{(M)})$:

$$\Omega = H(v^{(1)})\ldots H(v^{(M)})\begin{bmatrix} I \\ 0 \end{bmatrix}$$

Finally, apply Lemma 2 for series of Householder reflections $H(v^{(1)})\ldots H(v^{(M)})$:

$$\Omega = \left( I - US^{-1}U_1^T \right) \begin{bmatrix} I \\ 0 \end{bmatrix} = \begin{bmatrix} I \\ 0 \end{bmatrix} - US^{-1}U_1^T = \gamma_{N,M}(v^{(1)}, \ldots, v^{(M)})$$

which justifies surjectivity of $\gamma_{N,M}$. \qed

B. Stiefel RGD through Sherman-Morrison-Woodbury Formula

We adapt derivations of Tagare (2011) for canonical inner product and extend them to Euclidean inner product. The following lemma shows how to compute update $g_k(\eta_k)$ in time $O(NM^2 + M^3)$ without constructing $A^{(k-1)}$ explicitly.

Theorem 1. Consider $\Omega \in \mathbb{R}^{N \times M}$ and $A = BC^T \in \text{Skew}(N)$ for some matrices $B, C \in \mathbb{R}^{N \times D}$, $D \leq N$. Then

$$\text{Cayley}(A)\Omega = \Omega - B \left( I + \frac{1}{2} C^T B \right)^{-1} \left( C^T \Omega \right) \quad (19)$$

Proof. We first need to show that the right hand side of (19) always exists, i.e. $I + \frac{1}{2} C^T B$ is nonsingular:

$$\det(I + \frac{1}{2} C^T B) = \det(I + \frac{1}{2} BC^T) = \det(I + \frac{1}{2} A) \neq 0$$

where in the first transition we apply Sylvester’s determinant identity. $I + \frac{1}{2} A$ is nonsingular, because the spectrum of any skew-symmetric matrix is pure-imaginary (Theorem 12.9 from (Gallier, 2011)). So right hand side is well defined.
Through the application of Sherman-Morrison-Woodbury formula we deduce that

\[
\text{Cayley}(A)\Omega = \left(I + \frac{1}{2}BC^\top\right)^{-1} \left(I - \frac{1}{2}BC^\top\right)\Omega = \left(I - \frac{1}{2}B(I + \frac{1}{2}C^\top B)^{-1}C^\top\right) \left(I - \frac{1}{2}BC^\top\right)\Omega
\]

\[
= \Omega - \frac{1}{2}B \left(I + \frac{1}{2}C^\top B\right)^{-1}(I - \frac{1}{2}C^\top B)\Omega
\]

\[
= \Omega - \frac{1}{2}B(I + \frac{1}{2}C^\top B)^{-1}(2I - C^\top B + C^\top B)C^\top\Omega
\]

\[
= \Omega - B\left(I + \frac{1}{2}C^\top B\right)^{-1}(C^\top\Omega)
\]

which concludes the proof. □

For convenience denote \(G^{(k-1)} = \frac{\partial f}{\partial x_l}(\Omega^{(k-1)})\). Depending on the inner product choice we get the following cases:

1. **Canonical inner product.** Then

\[
\eta_k A^{(k-1)} = \eta_k G^{(k-1)}\Omega^{(k-1)}\top - \eta_k\Omega^{(k-1)}G^{(k-1)}\top = BC^\top
\]

where

\[
B = \eta_k \left[G^{(k-1)} \quad \Omega^{(k-1)}\right], \quad C = \left[\Omega^{(k-1)} \quad -G^{(k-1)}\right], \quad B,C \in \mathbb{R}^{N \times 2M}.
\]

2. **Euclidean inner product.** Then

\[
\eta_k A^{(k-1)} = \eta_k G^{(k-1)}\Omega^{(k-1)}\top - \eta_k\Omega^{(k-1)}G^{(k-1)}\top + \frac{\eta_k}{2}\Omega^{(k-1)}E\Omega^{(k-1)}\top = BC^\top
\]

where

\[
E = G^{(k-1)}\top\Omega^{(k-1)}\top - \Omega^{(k-1)}\top G^{(k-1)}, \quad B = \eta_k \left[G^{(k-1)} \quad \Omega^{(k-1)} \quad \frac{1}{2}\Omega^{(k-1)}E\right],
\]

\[
C = \left[\Omega^{(k-1)} \quad -G^{(k-1)} \quad \Omega^{(k-1)}\right], \quad B,C \in \mathbb{R}^{N \times 3M}.
\]

### C. Hidden State Gradients of ConvNERU

The convolution operation can be expressed as

\[
(K * G^{(t-1)})_{i,j} = \hat{K}^\top\hat{G}_{i,j}^{(t-1)}, \quad \hat{G}_{i,j}^{(t-1)} = \text{concat}\left\{\{G_{i,p}^{(t-1)} \mid i - \frac{q-1}{2} \leq l \leq i + \frac{q-1}{2} \quad j - \frac{q-1}{2} \leq p \leq j + \frac{q-1}{2}\}\right\}
\]

where \(G_{i,p}^{(t-1)} \in \mathbb{R}^{f_{out}}\) is a zero vector when \(l, p\) are pointing outside image borders (zero padding). By definition of \(\hat{G}^{(t-1)}\) and \(K * G^{(t-1)}\) we have the following chain of inequalities between Frobenius norms \(\| \cdot \|_F\):

\[
\|K * G^{(t-1)}\|_F^2 = \sum_{i,j} \|(K * G^{(t-1)})_{i,j}\|_2^2 = \sum_{i,j} \|\hat{K}^\top\hat{G}_{i,j}^{(t-1)}\|_2^2 \leq \sum_{i,j} \|\hat{K}\|_2^2 \|\hat{G}_{i,j}^{(t-1)}\|_2^2 = \|\hat{K}\|_2^2 \|\hat{G}^{(t-1)}\|_2^2 \leq \frac{q^2}{2} \|\hat{K}\|_2^2 \|G^{(t-1)}\|_2^2
\]

Assuming that \(|\sigma(x)| \leq |x|\) which holds for most popular choices of nonlinearity (ReLU, LeakyReLU, tanh), the norm of \(G^{(t)}\) cannot grow in exponential manner. The same holds for a sequence of gradients with respect to \(\{G^{(t)}\}\), since it is obtained by sequentially applying a transposed linear operator corresponding to ” convolution operation and transposition preserves the linear operator norm. This justifies the property of ConvNERU being robust to gradient explosion while allowing long-term information propagation thank to Stiefel convolution kernel. The conducted analysis is reminiscent of Lipschitz constant estimate for image classification CNNs performed by Cisse et al. (2017).
Table 5. Tatoeba Spa-to-Eng NMT results. We ran 3 seeds for each model. Below we present the average test loss across these seeds, as well as the associated standard deviation (STDDEV). We did not run additional seeds for non-CWY orthogonal parameterization approaches as these methods are slow (requiring many TPU hours to train) and our primary comparison with them was w.r.t. speed.

| MODEL | AVG TEST CE LOSS | STDDEV |
|-------|------------------|--------|
| RNN   | 0.74             | .08    |
| GRU   | 0.56             | .05    |
| LSTM  | 0.55             | .05    |
| RGD FOR O(N) | 2.01             | .14    |
| CWY, L = 1024 | 0.56             | .03    |
| CWY, L = 512  | 0.66             | .03    |
| CWY, L = 256  | 0.64             | .06    |
| CWY, L = 128  | 0.50             | .01    |
| CWY, L = 64   | 0.60             | .01    |

D. NLP Experiment Statistics

See Table 5.

E. Neural Machine Translation Architecture and Hyperparameters

See Figure 3 for the architecture illustration. In our experiments, we used a batch size of 64, an embedding dimension size of 256, and a learning rate of $10^{-2}$. For hyperparameter sweeps, we ran experiments with smaller hidden unit sizes. We also experimented with larger and smaller learning rates. Ultimately, for simplicity and clarity, we only present results using the parameters described above.

F. KTH Action Dataset Statistics

See Table 6.

Table 6. KTH action dataset statistics.

| STATISTIC                          | WALKING | JOGGING | RUNNING | BOXING | WAVING | CLAPPING |
|------------------------------------|---------|---------|---------|--------|--------|----------|
| MIN SEQUENCE LENGTH                | 62      | 42      | 26      | 42     | 62     | 24       |
| MAX SEQUENCE LENGTH                | 231     | 152     | 111     | 362    | 245    | 235      |
| MEAN SEQUENCE LENGTH               | 109.3   | 68.0    | 48.9    | 110.3  | 129.0  | 106.2    |
| TOTAL FRAMES COUNT (TRAIN SET)     | 20122   | 12730   | 9096    | 20515  | 23958  | 19529    |
| TOTAL FRAMES COUNT (VAL. SET)      | 7622    | 4551    | 3448    | 7558   | 8436   | 6415     |
| TOTAL FRAMES COUNT (TEST SET)      | 15991   | 9913    | 7018    | 15277  | 18963  | 16125    |

G. Video Prediction Architecture

See Figure 4.
Figure 3. Sketch of the architecture used for Neural Machine Translation experiments. For the ease of illustration we put 5 as maximal input and output length. $w_{in_1}, w_{out}$ are input and output word embeddings respectively. $\langle \text{eos} \rangle$ and $\langle \text{pad} \rangle$ denote embeddings of "end of sentence" and "padding" tag respectively. We use two different RNN units for the encoder rollout $h_1 \rightarrow \cdots \rightarrow h_5$ (blue) and decoder rollout $h_1 \rightarrow \cdots \rightarrow h_5$ (pink). We illustrate how the distribution of predicted output word $\hat{w}_{out}^3$ is computed, other output words are processed similarly. Given $h_d^2$, the context vector $c_3 \in \mathbb{R}^N$ is computed as $\sum_i \alpha_i h_i^e$ where $\sum_i \alpha_i = 1$, $\alpha_i \propto \exp(v^\top \tanh(W_1 h_i^e + W_2 h_d^2))$, $v \in \mathbb{R}^N, W_1, W_2 \in \mathbb{R}^{N \times N}$ are learnable parameters. Then $c_3$ is concatenated with previous word embedding (or null tag embedding for the first predicted word) and passed into decoder RNN as input. Decoder RNN output ($h_d^3$) is passed through linear layer + softmax to obtain a distribution over $\hat{w}_{out}^3$.

Figure 4. Sketch of the architecture used for video prediction experiments. Blue and grey blocks illustrate hidden representations with and without recurrent connections respectively. We compare different designs of the blue block (ConvLSTM, ConvNERU) while preserving everything else intact except of absence/presence of instance normalization as indicated in the main text. In our comparison we try different designs of blue recurrent units with everything else unchanged (unless specified).