# CWY Parametrization: a Solution for Parallelized Optimization of Orthogonal and Stiefel Matrices

**Valerii Likhosherstov**[*1] **Jared Davis**[*2,3] **Krzysztof Choromanski**[4,5] **Adrian Weller**[1,6]
[1]University of Cambridge [2]DeepMind [3]Stanford University [4]Google Brain [5]Columbia University
[6]The Alan Turing Institute [*]Equal contribution

## Abstract

We introduce an efficient approach for optimization over orthogonal groups on highly parallel computation units such as GPUs or TPUs. As in earlier work, we parametrize an orthogonal matrix as a product of Householder reflections. However, to overcome low parallelization capabilities of computing Householder reflections sequentially, we propose employing an accumulation scheme called the compact WY (or CWY) transform – a compact parallelization-friendly matrix representation for the series of Householder reflections. We further develop a novel Truncated CWY (or T-CWY) approach for Stiefel manifold parametrization which has a competitive complexity and, again, yields benefits when computed on GPUs and TPUs. We prove that our CWY and T-CWY methods lead to convergence to a stationary point of the training objective when coupled with stochastic gradient descent. We apply our methods to train recurrent neural network architectures in the tasks of neural machine translation and video prediction.

## 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 can be helpful for several reasons. A series of works addresses the problems of exploding or vanishing gradients in recurrent neural networks (RNNs) (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; Lezcano-Casado and Martínez-Rubio, 2019). Further, orthogonality appears to improve forward and backward information propagation in deep convolutional neural networks where convolutions are parametrized by a Stiefel manifold—a general class of orthogonal matrices (Huang et al., 2018; Bansal et al., 2018; Li et al., 2020). The norm-preserving property of an orthogonal linear operator helps to gain control over the Lipschitz constant of the deep architecture and, therefore, can enhance adversarial robustness of the model and its generalization capabilities both in theory and practice (Cisse et al., 2017). Orthogonality is also useful when designing invertible constructions for flow-based generative modelling (Van Den Berg et al., 2018).

Yet there is a lack of an orthogonal optimization method which is compatible with the industry-standard use of highly-parallel devices (GPU or TPU) for computations. Indeed, existing approaches for training an $N \times N$ orthogonal matrix can be grouped into two categories (see Table 1):

- Algorithms involving expensive operation of $N \times N$-sized matrix inversion or exponent (Wisdom et al., 2016; Lezcano-Casado and Martínez-Rubio, 2019; Helfrich et al., 2018) resulting in at least $O(N^2 \log N)$ parallel complexity (Tuma, 2020).
- Algorithms decomposing the orthogonal operator into a set of $L < N$ linear operators applied sequentially (Jing et al., 2016; Mhammedi et al., 2017), not taking full advantage of parallel matrix multiplication on GPU and TPU (Schatz et al., 2016; Tuma, 2020), and resulting in at least $O(L)$ parallel complexity.

Hence, there is a critical gap, with no method which works when a) cubic time is prohibitive and b) large $L$ for non-cubic approaches is slow while small $L$ seri-

Proceedings of the 24[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR: Volume 130. 

ously restricts model capacity.

We present a new approach to optimization over orthogonal matrices, focusing on computational efficiency. We employ the compact WY (or CWY) transform, a scheme for the composition of several Householder reflections (Householder, 1958). Our proposed approach has several advantages:

1. While in exact arithmetic being equivalent to decomposition into Householder reflections (Mhammedi et al., 2017), the parallel complexity of the algorithm **is only $\boldsymbol{O(\log(LN))}$** with $O(L^2 \log L)$ preprocessing (see Table 1) which makes it especially efficient when executed on GPU or TPU. We observe $20\times$ speedup in practice compared to sequential Householder reflections (Mhammedi et al., 2017) (see Table 2) and **1-3 orders of magnitude speedups compared to matrix exponential and Cayley map** (Figure 1c).

2. We introduce an extension for parametrizing Stiefel manifolds – nonsquare generalizations of orthogonal matrices. The extension scheme, named "Truncated CWY" (or T-CWY), is to our knowledge **a novel parametrization of the Stiefel manifold which requires the smallest number of floating point operations (FLOPs)** among methods for Stiefel optimization (see Table 2).

3. Finally, we prove that SGD based on CWY or T-CWY leads to a gradient norm convergence to zero with $o(K^{-0.5+\epsilon})$ rate for any $\epsilon > 0$ where $K$ is an iteration index.

We evaluate CWY on standard benchmarks (Copying task, Pixel-by-pixel MNIST) and neural machine translation. We evaluate T-CWY on the task of video prediction. All theoretical results are proven in Appendix F.

# 2 RELATED WORK

We discuss orthogonality in the motivating example of RNN gradient explosion and vanishing. Then we review orthogonal optimization methods and their properties, summarized in Tables 1 and 2.

## 2.1 Gradient Explosion and Vanishing

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

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

for $t = 1, \dots, T$. Here $x_1, \dots, x_T \in \mathbb{R}^K$ are the states of an observed sequence $X = \{x_1, \dots, x_T\}$ from the training set, $h_0, \dots, h_T \in \mathbb{R}^N$ 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 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:

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

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 $\|W\|_2 = \sup_{\|h\|_2=1} \|Wh\|_2 > 1$ (*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 this problem, Arjovsky et al. (2016) proposed using an orthogonal or unitary matrix $W$, that is to set either $W = Q \in \mathcal{O}(N)$ or $W = Q \in \mathcal{U}(N)$. Here $\mathcal{O}(N) = \{Q \in \mathbb{R}^{N \times N} \,|\, Q^\top Q = I\}$ is called the *orthogonal group*, $\mathcal{U}(N) = \{Q \in \mathbb{C}^{N \times N} \,|\, Q^H Q = I\}$ is called the *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. $\forall h : \|Qh\|_2 = \|h\|_2$), 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

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

## 2.2 Orthogonal Optimization

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

### 2.2.1 Parametrization

This is a family of methods constructing $Q$ as a function of unconstrained parameters, on which standard gradient descent can be performed.

**URNN** (Unitary Recurrent Neural Network, Arjovsky et al., 2016) expresses Q as $D^{(3)} H^{(2)} F^{-1} D^{(2)} \Pi H^{(1)} F D^{(1)}$, 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) parametrizes $Q = DF^{(1)}F^{(2)} \dots F^{(L)} \in \mathcal{U}(N)$ where $L \leq N$, $D$ is diagonal unitary and $F^{(i)} \in \mathbb{C}^{N \times N}$ are permuted block-diagonal with $2 \times 2$ blocks.

**HR** (Householder reflections, Mhammedi et al., 2017) decomposes $Q = H(v^{(1)}) \dots H(v^{(L)}) \in \mathcal{O}(N)$ where for each nonzero $v \in \mathbb{R}^N$, $H(v) = I - 2vv^\top / \|v\|_2^2 \in \mathcal{O}(N)$ is a *Householder reflection.*

**EXPRNN** (Exponent RNN, Lezcano-Casado and Martínez-Rubio, 2019). This method takes advantage of the fact that the matrix exponent $\exp(A)$ is a surjective mapping from the set of skew-symmetric matrices $\mathrm{Skew}(N) = \{A \in \mathbb{R}^{N \times N} \,|\, A = -A^\top\}$ to the *special orthogonal group* $\mathcal{O}^{+1}(N)$, where for $s = \pm 1$ we define $\mathcal{O}^s(N) = \{Q \in \mathcal{O}(N) \,|\, \det Q = s\}$. Notice that $\mathcal{O}(N) = \mathcal{O}^{+1}(N) \cup \mathcal{O}^{-1}(N)$.

**SCORNN** (Skew Cayley, Helfrich et al., 2018) uses the *Cayley transform* instead of matrix exponent: $Q = \mathrm{Cayley}(A) = (I + A/2)^{-1}(I - A/2)$ which is a bijective map from $\mathrm{Skew}(N)$ to $\mathcal{O}^{+1}(N) \setminus \Theta$ where $\Theta$ is a set of matrices with $-1$ eigenvalue. To cover all matrices from $\mathcal{O}(N)$, $Q$ is scaled as $\tilde{Q} = Q\tilde{D}$ where $\tilde{D}$ is a diagonal matrix with $\pm 1$ values. The number of $-1$'s in $\tilde{D}$ is a hyperparameter, which requires an additional search method. For fair comparison, we fix $\tilde{D} = I$.

**OWN** (Orthogonal Weight Normalization, Huang et al., 2018). This method considers the more general task of optimizing a function over the *Stiefel manifold* $\mathrm{St}(N, M) = \{\Omega \in \mathbb{R}^{N \times M} \,|\, \Omega^\top \Omega = I\}$ where $M \leq N$, which generalizes the set $\mathcal{O}(N)$. $\Omega$ is set as $\Omega = \tilde{V} P \Lambda^{-1/2} P^\top$, $\tilde{V} = (V - \frac{1}{N}\mathbf{1}\mathbf{1}^\top V)$ where $P \Lambda P^\top$ is an eigendecomposition of matrix $\tilde{V}^\top \tilde{V} \in \mathbb{R}^{M \times M}$ and $\mathbf{1}$ is the all-ones $N$-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 $\mathrm{St}(N, M)$ and b) points in the direction of fastest descent along the manifold. More precisely, RGD starts with a predefined matrix $\Omega^{(0)} \in \mathrm{St}(N, M)$ and makes sequential updates of the type $\Omega^{(k)} := g_k(\eta_k)$ where $\eta_k$ is a step size, $g_k : \mathbb{R} \to \mathrm{St}(N, M)$, $g_k(0) = \Omega^{(k-1)}$ and $g_k'(0)$ is the gradient $\frac{\partial f}{\partial \Omega}(\Omega^{(k-1)})$ projected onto the *tangent space* $\mathcal{T}_{\Omega^{(k-1)}}$ – a linear space approximating the Stiefel manifold $\mathrm{St}(N, M)$ at the point $\Omega^{(k-1)}$. It is known that $\mathcal{T}_\Omega = \{Z \in \mathbb{R}^{N \times M} \,|\, Z^\top \Omega \in \mathrm{Skew}(M)\}$. For a rigorous introduction to Riemannian manifolds and Riemannian Gradient Descent see (Absil et al., 2007).

In a Riemannian manifold, the tangent space $\mathcal{T}_\Omega$ must have an inner product, usually chosen as either the *canonical inner product* $\langle Z_1, Z_2 \rangle_1 = \mathrm{Tr}(Z_1^\top (I - \frac{1}{2}\Omega\Omega^\top) Z_2)$ or *Euclidean inner product* $\langle Z_1, Z_2 \rangle_2 = \mathrm{Tr}(Z_1^\top Z_2)$. Consequently, the projection of the gradient has the form: $g_k'(0) = A^{(k-1)}\Omega^{(k-1)}$, $\quad A^{(k-1)} = \widehat{A}_i^{(k-1)} - \widehat{A}_i^{(k-1)\top}$ where $\widehat{A}_1^{(k-1)} = \frac{\partial f}{\partial \Omega}(\Omega^{(k-1)})\Omega^{(k-1)\top}$ corresponds to the canonical inner product choice, and $\widehat{A}_2^{(k-1)} = \widehat{A}_1^{(k-1)} - \frac{1}{2}\Omega^{(k-1)}\Omega^{(k-1)\top}\widehat{A}_1^{(k-1)}$ 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* $g_k^{\mathrm{Cay}}(\eta) = \mathrm{Cayley}(\eta A^{(k-1)})\Omega^{(k-1)}$ and 2) *QR-decomposition retraction* $g_k^{\mathrm{QR}}(\eta) = \mathrm{qf}(\eta A^{(k-1)}\Omega^{(k-1)})$ where $\mathrm{qf}(\cdot)$ denotes a Q matrix of the argument's QR decomposition so that diagonal elements of the R matrix are positive. Wisdom et al. (2016); Li et al. (2020) evaluate performance of RGD in the context of deep learning.

### 2.3 Runtime Complexity

We compare the serial and parallel runtime complexity of different methods to train orthogonal RNNs in Table 1 (we introduce the notation $\mathcal{O}_L(N)$ later in this section). We also show the domain covered by each optimization approach.

Row "RNN" indicates the complexity of an unconstrained RNN. Mhammedi et al. (2017) show that any RNN with a unitary transition matrix can be modelled by a different network with orthogonal weights. Hence, we opt for simplification by only covering the orthogonal group $\mathcal{O}(N)$. As noted by (Wisdom et al., 2016), URNN parametrization is not enough to cover all matrices from $\mathcal{U}(N)$, which is an $N^2$-dimensional manifold.

RGD, SCORNN and EXPRNN employ a costly $O(N^3)$ operation of matrix exponent or Cayley transform. Note that the limitation of EXPRNN covering only $\mathcal{O}^{+1}(N)$ can be alleviated, since a matrix $Q \in \mathcal{O}^s(N)$ can be parametrized by $\widehat{Q} \in \mathcal{O}^{-s}(N)$ obtained by inverting one of $Q$'s rows.

EURNN enables a tradeoff between computational complexity and unitary matrix coverage. Matrix-vector product with $F^{(i)}$ can be efficiently computed in serial time $O(N)$ (parallel $O(1)$). Next, by choosing bigger $L$, we can increase the family of supported unitary matrices at the cost of additional computation time. Eventually, when $L = N$, all unitary matrices are covered. Similar properties hold for HR decomposition – applying a Householder reflection to a vector is an $O(N)$ (parallel $O(\log N)$) operation and the following theorem holds:

**Theorem 1** (adapted from Mhammedi et al., 2017)**.**

Table 1: Comparison of runtime complexity required for a forward pass through RNN. To report parallel complexity we use that a) a product of $d_1 \times d_2$ and $d_2 \times d_3$-sized matrix takes $O(\log(d_1 d_2 d_3))$ time (distribution over $O(d_1 d_2 d_3)$ processes) (Schatz et al., 2016) and b) finding an inverse of $d_1 \times d_1$-sized matrix takes $d_1^2 \log d_1$ time (distribution over $O(d_1)$ processes) (Tuma, 2020). All complexities are in $O(\cdot)$ notation, terms related to $Vx_t$ computation are omited (serial $TKN$ and parallel $T \log(KN)$ additional term). The *Cheap Gradient Principle* (Griewank and Walther, 2008) states that serial complexity of the backward pass coincides with that of the forward pass (can be extended to parallel complexity, see Bischof, 1991; Juedes and Griewank, 1990).

| METHOD | SERIAL TIME | PARALLEL TIME | SOLUTION DOMAIN |
|---|---|---|---|
| RNN | $TN^2$ | $T \log N$ | — |
| URNN | $TN \log N$ | $TN \log N$ | $\mathcal{U}(N)$'s subset |
| SCORNN | $TN^2 + N^3$ | $T \log N + N^2 \log N$ | $\mathcal{O}^{+1}(N) \setminus \Theta$ |
| RGD for $\mathcal{U}(N)$ | $TN^2 + N^3$ | $T \log N + N^2 \log N$ | $\mathcal{U}(N)$ |
| EXPRNN | $TN^2 + N^3$ | $T \log N + N^3$ | $\mathcal{O}^{+1}(N)$ |
| EURNN, $L$ iter. | $TLN$ | $TL$ | $\mathcal{U}(N)$ when $L = N$ |
| HR, $L$ refl. | $TLN$ | $TL \log N$ | $\mathcal{O}_L(N)$ |
| CWY, $L$ refl. (ours) | $TLN + L^2 N + L^3$ | $T \log(LN) + L^2 \log L$ | $\mathcal{O}_L(N)$ |

*Let $Q \in \mathcal{O}^s(N)$ where $s = (-1)^N$. Then there exist nonzero $v^{(1)}, \dots, v^{(N)} \in \mathbb{R}^N$ s.t. $Q = H(v^{(1)}) \dots H(v^{(N)})$.*

Although EURNN and HR methods don't have an $O(N^3)$ term in runtime complexity, they cannot be parallelized in $L$, the number of sequentially applied operators $F^{(i)}$ or $H(v^{(i)})$. This becomes a problem when $N$ is big and, thus, bigger $L$ is needed to obtain good expressiveness. We use the notation $\mathcal{O}_L(N)$ for the set of orthogonal matrices which can be obtained with $L$ Householder reflections: $\mathcal{O}_L(N) = \{H(v^{(1)}) \dots H(v^{(L)}) \,|\, \forall i : v^{(i)} \in \mathbb{R}^N \setminus \{\mathbf{0}\}\}$.

Table 2 summarizes the runtime complexity of Stiefel manifold optimization approaches. OWN requires an eigenvalue decomposition of a dense $M \times M$-sized matrix which is a cubic operation. See Appendix Section A for additional discussion of RGD-based methods' runtime complexity.

# 3 EFFICIENT $\mathcal{O}(N)$ and $\mathrm{St}(N, M)$ PARAMETRIZATION

We define the CWY transform and demonstrate its utility for RNN training. Next, we introduce a novel T-CWY map, and for both transforms prove stochastic-optimization convergence guarantees.

## 3.1 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 in experiments.

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

$$H(v^{(1)}) \dots H(v^{(L)}) = I - US^{-1}U^\top, \tag{3}$$

*where $U = \left[ v^{(1)} / \|v^{(1)}\|_2 \dots v^{(L)} / \|v^{(L)}\|_2 \right] \in \mathbb{R}^{N \times L}$ , and $S = \frac{1}{2} I + \mathrm{striu}(U^\top U)$ where $\mathrm{striu}(\cdot)$ returns an argument matrix with all diagonal and lower-triangular elements zeroed out.*

We store $v^{(1)}, \dots, v^{(L)}$ as learnable parameters. An efficient way to do a forward pass with CWY-based RNN is as follows. 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: $u_t := U^\top h_{t-1}$, $v_t := S^{-1} u_t$, $y_t := h_{t-1} - Uv_t + b$, 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. The latter approach is asymptotically efficient when $L < N$, while when $L = N$ we precompute the transition matrix (3) into $Q$ and then perform the RNN rollout as usual.

The better parallelization pattern of CWY comes with a price of an $L^2 \log L$ term related to inverting the $S$ matrix. In practice, we find that for moderate $L$ this addition is comparable to the rollout cost, considering also that $S$ is upper-triangular and, hence, takes less FLOPs to invert (Hunger, 2005).

Table 2: Complexity of performing a gradient step when optimizing over $\Omega \in \mathrm{St}(N, M)$. In the notation "RGD-A-B" "A" is C or E for canonical or Euclidean inner product choice respectively, and "B" is C or QR for Cayley or QR retraction respectively. The term related to computing the objective function and $\Omega$'s gradient is omitted. Parallel complexity is reported in $O(\cdot)$ notation while FLOPs are reported for the forward pass with exact constants in the leading terms. The backward pass requires only a constant time more operations (the *Cheap Gradient Principle*, Griewank and Walther, 2008). To report parallel complexity we use the same assumptions as for Table 1. In our estimations we use that a) a product of $d_1 \times d_2$ and $d_2 \times d_3$-sized matrix takes $2d_1d_2d_3$ FLOPs (Hunger, 2005), b) an inverse of $d_1 \times d_1$-sized dense and upper-triangular matrix takes $d_1^3$ and $d_1^3/3$ FLOPs respectively (Hunger, 2005), c) QR decomposition of a $d_1 \times d_2$-sized matrix, $d_1 \geq d_2$, takes $2d_2^2(d_1 - \frac{1}{3}d_2)$ FLOPs (Hammarling and Lucas, 2008) and d) eigendecomposition of a $d_1 \times d_1$-sized positive semi-definite matrix (as it is in OWN) coincides with its SVD which requires $\frac{8}{3}d_1^3$ FLOPs (Trefethen and Bau, 1997). Since $N \geq M$, T-CWY needs the smallest number of FLOPs.

| APPROACH | PARALLEL TIME | INVERTED MATRIX SIZE | FLOPs |
|---|---|---|---|
| RGD-C-QR | $M \log(MN)$ | — | $10NM^2 - 2M^3/3$ |
| RGD-E-QR | $M \log(MN)$ | — | $14NM^2 - 2M^3/3$ |
| RGD-C-C | $\log(MN) + M^2 \log M$ | $2M \times 2M$ | $28NM^2 + 16M^3$ |
| RGD-E-C | $\log(MN) + M^2 \log M$ | $3M \times 3M$ | $72NM^2 + 25M^3$ |
| OWN | $\log(MN) + M^3$ | — | $4NM^2 + 14M^3/3$ |
| T-CWY (ours) | $\log(MN) + M^2 \log M$ | $M \times M$ upper-triangular | $\mathbf{4NM^2 + 7M^3/3}$ |

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

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

**Theorem 3.** *Consider $M < N$ and a function $\gamma_{N,M} : (\mathbb{R}^N \setminus \{\mathbf{0}\})^M \to \mathbb{R}^{N \times M}$ defined as follows. For $v^{(1)}, \dots v^{(M)} \in \mathbb{R}^N$ construct a matrix $U = \begin{bmatrix} v^{(1)}/\|v^{(1)}\|_2 & \dots & v^{(M)}/\|v^{(M)}\|_2 \end{bmatrix} \in \mathbb{R}^{N \times M}$ and assign $\gamma_{N,M}(v^{(1)}, \dots v^{(M)}) = \begin{bmatrix} I & \mathbf{0} \end{bmatrix}^\top - US^{-1}U_1^\top \in \mathbb{R}^{N \times M}$ where $U_1$ is an upper $M \times M$ submatrix of $U$ and $S = \frac{1}{2}I + striu(U^\top U)$. Then $\gamma_{N,M}$ is a surjective mapping to $St(N, M)$.*

In other words, Theorem 3 states that Stiefel matrices can be parametrized by taking $M$ first columns of a $N \times N$ CWY-parametrized matrix with $L = M$, but without forming this $N \times N$ matrix explicitly. Computational complexity of T-CWY is indicated in Table 2. T-CWY is fully-parallelizable in $N$ with the number of floating point operations smaller than for any other approach due to the inverted matrix $S$ size $M \times M$ and upper-triangular structure (Hunger, 2005).

### 3.3 SGD Convergence Analysis

Consider a function $f : \mathcal{O}(N) \to \mathbb{R}$ (e. g. an empirical risk) which is accessed through its stochastic proxy $\widetilde{f}$ (e. g. a minibatch loss). We prove a standard result (Bonnabel, 2013; Bottou et al., 2016) stating that CWY-based stochastic optimization can get arbitrarily close to a stationary point where $\nabla f = \mathbf{0}$. For convenience we formulate our results in terms of a Householder decomposition which is equivalent to CWY.

**Theorem 4.** *Let $f : \mathbb{R}^{N \times N} \to \mathbb{R}$ be a differentiable function with Lipschitz-continuous gradients on $\mathcal{O}(N)$: $\forall X', X'' \in \mathcal{O}(N) : \|\nabla f(X') - \nabla f(X'')\|_F \leq M_1 \|X' - X''\|_F$ for some $M_1 > 0$ ($\|\cdot\|_F$ denotes Frobenius norm). Let $\widetilde{f} : \mathbb{R}^{N \times N} \to \mathbb{R}$ be a stochastic differentiable function such that $\forall X \in \mathcal{O}(N)$ : $\mathbb{E}\nabla \widetilde{f}(X) = \nabla f(X)$ and suppose there exists $M_2 > 0$ such that $\forall X \in \mathcal{O}(N) : \mathbb{E}\|\nabla \widetilde{f}(X)\|_F^2 \leq M_2$. Consider a sequence $\{(v^{(k,1)} \in \mathbb{R}^N, \dots, v^{(k,L)} \in \mathbb{R}^N)\}_{k=0}^{\infty}$ where $v^{(0,1)}, \dots, v^{(0,L)} \in \mathbb{R}^N$ are deterministic and nonzero and for all $k > 0, 1 \leq l \leq L$: $v^{(k,l)} = v^{(k-1,l)} - k^{-0.5} \nabla_{v^{(k-1,l)}} \widetilde{f}(H(v^{(k-1,1)}) \dots H(v^{(k-1,L)}))$. Then all $\{v^{(k,l)}\}$ are well-defined and for any $\epsilon > 0$,*

$$\min_{0 \leq k' < K} \sum_{l=1}^{L} \mathbb{E}\|\nabla_{v^{(k',l)}} f(H(v^{(k',1)}) \times \dots \times H(v^{(k',L)}))\|_2^2 = o(K^{-0.5+\epsilon}).$$

Observe that an identical result holds for T-CWY parametrization. Indeed, using notation of Theorem 3 for any $f : \mathrm{St}(N, M) \to \mathbb{R}$ $f(\gamma_{N,M}(v^{(1)}, \dots, v^{(M)})) = f((H(v^{(1)}) \dots H(v^{(M)}))_{:,:M})$. Gradient Lipschitz-continuity of $f$ and bounded variance of $\widetilde{f}$ hold for composite functions $f((\cdot)_{:,:M})$ and $\widetilde{f}((\cdot)_{:,:M})$ which are plugged into Theorem 4 to get analogous result for T-CWY. The proof of Theorem 4, as well as a high-level sketch to help intuition, can be found in Appendix F.4.

### 3.4 Convolutional Non-Exploding Recurrent Unit (ConvNERU)

Based on the proposed Stiefel matrix parametrization, we introduce a *convolutional non-exploding recurrent unit* (ConvNERU) – a recurrent module which is provably resistant to gradient and hidden state explosion. Given a sequence of images $X_1, \dots, X_T \in \mathbb{R}^{h \times w \times f_{in}}$, our proposed module is the following modification of (1): $Y_t := \mathcal{K} * G^{(t-1)} + B, \quad G^{(t)} := \sigma(Y_t + \mathcal{K}^{in} * X_t)$ where $G^{(0)}, \dots, G^{(T)} \in \mathbb{R}^{h \times w \times f_{out}}$ 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, "$*$" denotes convolution operation and $\mathcal{K} \in \mathbb{R}^{q \times q \times f_{out} \times f_{out}}, \mathcal{K}^{in} \in \mathbb{R}^{q \times q \times f_{in} \times f_{out}}$ are convolution kernels with $q$ being kernel size. Denote by $\widehat{\mathcal{K}}$ 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 $\widehat{\mathcal{K}}_{lqf_{out}+pf_{out}+i,j} = \mathcal{K}_{l,p,i,j}$. We equip ConvNERU with a constraint $(q\widehat{\mathcal{K}}) \in \mathrm{St}(q^2 f_{out}, f_{out})$ which is implemented by T-CWY parametrization. In Appendix Section B, we theoretically show that ConvNERU is resistant to norm explosion.

## 4 EXPERIMENTS

We evaluate CWY on standard benchmarks and a neural machine translation setup. Then, we evaluate T-CWY and ConvNERU on a video prediction setup.

### 4.1 Standard Tasks and Time Comparison

We evaluate orthogonal RNN with CWY parametrization on standard benchmarks, aimed to test the ability of RNN to capture long-term dependencies in the data:

1. *Copying task.* The input contains 10 digits sampled uniformly from $\{1, \dots, 8\}$, then $\mathcal{T}$ zeros, one "9" (*start*) and 9 zeros. The output consists of $\mathcal{T}+10$ zeros and 10 first digits from the input. Hence, the goal of RNN is to copy the random input prefix after observing $\mathcal{T}$ zeros. The goal is to beat a no-memory *baseline*, which outputs $\mathcal{T}+10$ zeros and 10 randomly sampled digits from $\{1, \dots, 8\}$ independently of the input. The cross-entropy of this baseline is $10 \log 8/(\mathcal{T}+20)$.

2. *Pixel-by-pixel MNIST.* The input contains images of digits from MNIST (LeCun et al., 2010), flattened into sequences of length 784. The goal is to classify the digit using the last hidden state of the RNN.

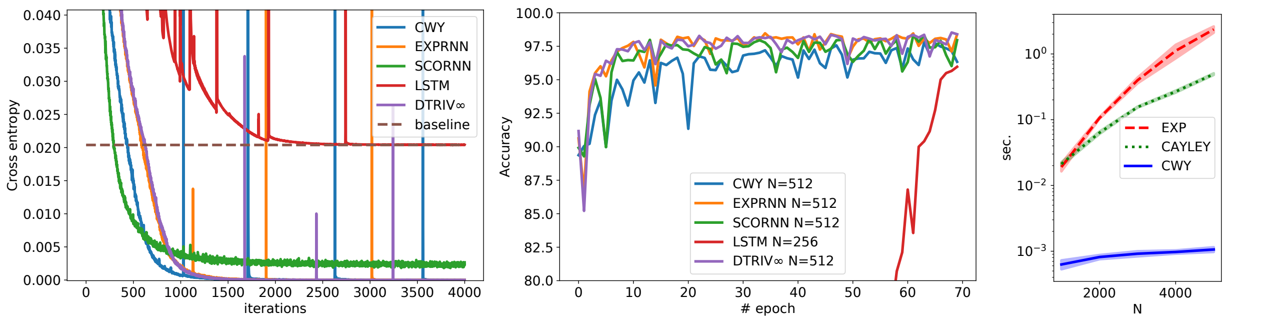
For both experiments we reuse the publicly available code from (Lezcano-Casado and Martínez-Rubio, 2019) in PyTorch (Paszke et al., 2017), **without tuning any hyperparameters**, changing random initializations or seeds, etc. Figures 1a, 1b (a-b) demonstrates the results of plugging CWY directly into the code. In the Copying task with $\mathcal{T} = 1000, L = N = 190$, CWY is converging to zero cross entropy faster, than EXPRNN and DTRIV$\infty$ (Lezcano Casado, 2019), while SCORNN fails to converge to zero and LSTM (Hochreiter and Schmidhuber, 1997) cannot beat the baseline. In the Pixel-by-pixel MNIST, CWY ($L = N$) shows competitive performance, going beyond 95% accuracy and matching the results of Mhammedi et al. (2017). See details and additional experimental results (Copying task with $\mathcal{T} = 2000$ and permuted MNIST) in Appendix C.

In addition to standard benchmarks, we perform a time comparison for computing CWY, exponential parametrization and Cayley map (Figure 1c), where the argument is a random matrix. See Appendix C for details. We conduct experiments on GPU and use the following methods from PyTorch 1.7: `torch.matrix_exp` implementing a state-of-the-art algorithm for matrix exponential (Bader et al., 2019), `torch.solve` for Cayley map and `torch.triangular_solve` for CWY. We observe that for a range of matrix sizes CWY is 1-3 orders of magnitude faster than other parametrizations. While we used full CWY ($L = N$) for this comparison, $L < N$ would lead to further speedups.

### 4.2 Neural Machine Translation

We train an orthogonal RNN-based seq2seq model with attention mechanism (Bahdanau et al., 2014) to translate sentence pairs between a given source and target language. See Appendix Section D for additional architectural and experimental details. We focus on the English-to-Spanish dataset within the Tatoeba corpus (Artetxe and Schwenk, 2019), a publicly available dataset with over 100,000 sentence pairs. We compare several variants of orthogonal RNNs with absolute value nonlinearities which are exact norm-preserving (Dorobantu et al., 2016) and compare them against GRUs and LSTMs used as RNN units in a seq2seq architecture. All variants of RNN have hidden dimension $N = 1024$. For the CWY and non-orthogonal variants, we conduct experiments with the Adam optimizer (see Table 3).

We find that standard RNNs underperform LSTMs and GRUs (Cho et al., 2014), but that parametrization-based orthogonal RNN variants are able to achieve comparable performance. 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. We find that even the full-orthogonal CWY scheme with $L = N$ runs faster in practice than other orthogonal approaches. A sweet-spot




(a) (b) (c)

Figure 1: **(a)** Copying task, $\mathcal{T} = 1000$. **(b)** Pixel-by-pixel MNIST, test accuracy. **(c)** parametrization time comparison, mean and standard error over 10 samples.

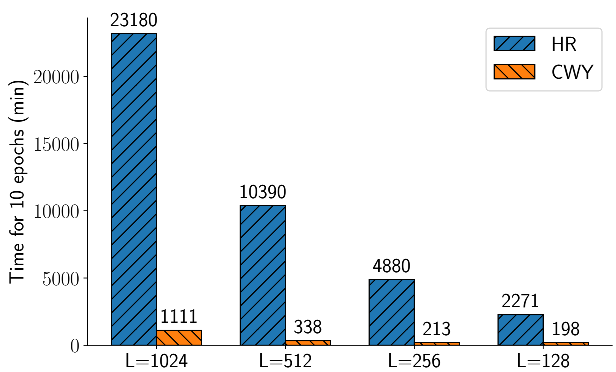



Figure 2: The CWY and HR methods are numerically equivalent; however, the parametrization of the CWY allows us to perform projections much more efficiently, leading to dramatic improvements in training time and, thereby, practical viability. The experiment is conducted on a Tensor Processing Unit (TPU).

Table 3: Tatoeba Spa-to-Eng NMT results. We report perplexity (PP) on a test set (a smaller value indicates a better result). Time is reported for 10 epochs. CWY achieves the best performance while preserving speed and requiring the fewest parameters. There is a sweet-spot for the test loss ($L = 128$).

| **MODEL** | **TEST PP** | **TIME (MIN.)** | **PARAMS** |
|---|---|---|---|
| RNN | 1.66 | 148 | ≈ 25M |
| GRU | 1.47 | 173 | ≈ 32M |
| LSTM | 1.46 | 232 | ≈ 37M |
| SCORNN | 1.49 | 1780 | ≈ 25M |
| RGD | 4.03 | 1780 | ≈ 25M |
| EXPRNN | 1.51 | 2960 | ≈ 25M |
| CWY L=1024 | 1.47 | 1111 | ≈ 25M |
| CWY L=512 | 1.58 | 338 | ≈ 24M |
| CWY L=256 | 1.56 | 213 | ≈ 23M |
| CWY L=128 | **1.41** | **198** | ≈ **23M** |
| CWY, L=64 | 1.52 | 175 | ≈ 23M |

parameter value $L = 128$ 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 $L$). As mentioned before, 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 Table 2. The enhanced speed of our CWY variants, when paired with the optimizer-choice flexibility, makes this approach a compelling alternative to LSTMs and GRUs.

### 4.3 Video prediction with ConvNERU

We demonstrate performance of T-CWY and 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), a convolutional adaptation of LSTM. In addition, our goal is to compare with other methods for Stiefel optimization and justify the need for Stiefel constraints.

We conduct experiments on the KTH action dataset (Schüldt et al., 2004) containing grey scale video recordings of 25 people, each performing 6 types of actions: walking, jogging, running, boxing, hand waving and hand clapping. We do separate evaluations for each action type to evaluate how the model learns different types of dynamics. As a video-prediction architecture we apply a simplified version of (Lee et al., 2018; Ebert et al., 2017) where we try different types of recurrent block design (see further). We opt for minimizing the $l_1$-loss $|\widehat{\mathcal{I}} - \mathcal{I}|$ ($l_1$-loss) during training where $\widehat{\mathcal{I}}, \mathcal{I}$ denote predicted and ground-truth frame respec-

Table 4: KTH action dataset test results. The indicated metric is average per-frame $l_1$-loss. Video frames are in grey scale with brightness ranged in $[0, 1]$. The GPU memory is evaluated for the "Box[ing]" class which has the longest sequences. We do not report the last two columns for the "Zeros" method which is only aimed to demonstrate the importance of recurrent connections.

| METHOD | WALK | JOG | RUN | BOX | WAVE | CLAP | # PARAMS | GPU MEMORY |
|---|---|---|---|---|---|---|---|---|
| ConvLSTM | 223.3 | 266.8 | 297.8 | 188.9 | 157.9 | 162.3 | $\approx$ 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 | $\approx$ 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-C-C | 135.8 | 155.7 | 170.7 | 172.9 | 160.3 | 144.5 | As above | As above |
| RGD-E-C | 143.3 | 152.5 | 173.7 | 171.9 | 172.9 | 142.6 | As above | As above |
| RGD-C-QR | 143.1 | 155.0 | 171.5 | 173.1 | 150.2 | 142.7 | As above | As above |
| RGD-E-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 |

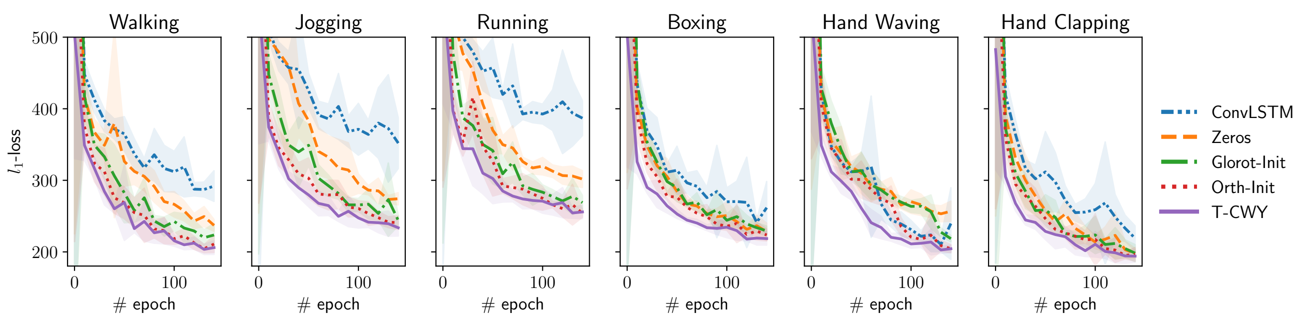



Figure 3: Validation $l_1$-loss. Mean and standard error across each 10 epochs is reported.

tively. For all unconstrained parameters we use the Adam optimizer. See Appendix Section E for more details on data preprocessing, experiment setup and architecture. We compare different designs of recurrent unit used in the full architecture. *ConvLSTM* was used in the original variant of the architecture (Lee et al., 2018; Ebert et al., 2017). *Zeros* indicates ConvNERU with transition kernel $K$ zeroed out (i.e. prediction conditioned on the previous frame only). *Glorot-Init* is a modified ConvNERU where $K$ is unconstrained initialized through Glorot uniform initialization (Glorot and Bengio, 2010). *Orth-Init* indicates a modified ConvNERU with unconstrained $q\widehat{K}$ initialized as a Stiefel matrix by QR decomposition of a random matrix. *RGD-*-** indicates Stiefel RGD for optimizing $q\widehat{K}$ with various combinations of inner product and retractor (consistent with the notation in Table 2). *RGD-Adam* is an Adam adaptation of RGD (Li et al., 2020) applied to optimization of $q\widehat{K}$. Finally, *OWN* and *T-CWY* indicate ConvNERU with $q\widehat{K}$ matrix parametrized by OWN and T-CWY respectively.

Table 4 demonstrates test $l_1$-loss, number of parameters and maximal GPU memory consumption. Additionally, Figure 3 demonstrates validation $l_1$-loss depending on epoch number for a subgroup of evaluluated 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, does not 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 while having several times less parameters and using much less GPU memory than ConvLSTM.

# 5 CONCLUSION

We introduced an efficient scheme for parametrizing orthogonal groups $\mathcal{O}(N)$ and Stiefel manifolds $\mathrm{St}(N, M)$, and compared to earlier approaches. The proposed $\mathcal{O}(N)$-parametrization scheme is 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-world applications.

## Acknowledgements

We thank Jonathan Gordon, Wessel Bruinsma and David Burt for helpful feedback on an early version of the manuscript. We further thank anonymous reviewers for their valuable feedback.

Valerii Likhosherstov acknowledges support from the Cambridge Trust and DeepMind. 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

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

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

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

P. Bader, S. Blanes, and F. Casas. Computing the matrix exponential with an optimized taylor polynomial approximation. *Mathematics*, 7(12):1174, 2019.

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

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

C. H. Bischof. Issues in parallel automatic differentiation. In *Proceedings of the 1991 International Conference on Supercomputing*, pages 146–153. ACM Press, 1991.

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

L. Bottou, F. E. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60:223–311, 2016.

K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, Oct. 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1179. URL `https://www.aclweb.org/anthology/D14-1179`.

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

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

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

J. Gallier. *Geometric Methods and Applications: For Computer Science and Engineering*. Texts in Applied Mathematics. Springer New York, 2011. ISBN 9781441999610. URL `https://books.google.co.uk/books?id=4v5VOTZ-vMcC`.

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

A. Griewank and A. Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation, Second Edition*. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 2008. ISBN 9780898717761. URL `https://books.google.co.uk/books?id=xoiiLaRxcbEC`.

S. Hammarling and C. Lucas. Updating the qr factorization and the least squares problem. 2008.

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

M. Henaff, A. Szlam, and Y. LeCun. Recurrent orthogonal networks and long-memory tasks. In *ICML*, pages 2034–2042, 2016. URL http://proceedings.mlr.press/v48/henaff16.html.

S. Hochreiter. 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.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, Nov. 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL http://dx.doi.org/10.1162/neco.1997.9.8.1735.

A. S. Householder. Unitary triangularization of a nonsymmetric matrix. *J. ACM*, 5(4):339–342, Oct. 1958. ISSN 0004-5411. doi: 10.1145/320941.320947. URL http://doi.acm.org/10.1145/320941.320947.

L. Huang, X. Liu, B. Lang, A. W. Yu, Y. Wang, and B. Li. Orthogonal weight normalization: Solution to optimization over multiple dependent Stiefel manifolds in deep neural networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

R. Hunger. *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.

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

T. Joffrain, T. M. Low, E. S. Quintana-Ortí, R. v. d. Geijn, and F. G. V. Zee. 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.

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

D. Juedes and A. Griewank. Implementing automatic differentiation efficiently. Technical report, Argonne National Lab., IL (USA). Mathematics and Computer Science Div., 1990.

Y. LeCun, C. Cortes, and C. Burges. Mnist handwritten digit database. 2010. *URL http://yann. lecun. com/exdb/mnist*, 7:23, 2010.

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

M. Lezcano Casado. Trivializations for gradient-based optimization on manifolds. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 9157–9168. Curran Associates, Inc., 2019.

M. Lezcano-Casado and D. Martínez-Rubio. Cheap orthogonal constraints in neural networks: A simple parametrization of the orthogonal and unitary group. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3794–3803, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL http://proceedings.mlr.press/v97/lezcano-casado19a.html.

J. Li, F. Li, and S. Todorovic. 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.

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

A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

M. D. Schatz, R. A. Van de Geijn, and J. Poulson. Parallel matrix multiplication: A systematic journey. *SIAM Journal on Scientific Computing*, 38(6): C748–C781, 2016.

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

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

L. Trefethen and D. Bau. *Numerical Linear Algebra*. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics, 1997. ISBN 9780898713619. URL https://books.google.co.uk/books?id=4Mou5YpRD_kC.

M. Tuma. Parallel matrix computations, May 2020. http://www.karlin.mff.cuni.cz/~mirektuma/ps/pp.pdf.

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

R. Van Den Berg, L. Hasenclever, J. M. Tomczak, and M. Welling. Sylvester normalizing flows for variational inference. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, pages 393–402. Association For Uncertainty in Artificial Intelligence (AUAI), 2018.

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

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