Dynamic Isometry As a Consequence of Weight Orthogonality for Faster and Better Convergence

Ester Hlav
Columbia University
Department of Mathematics
May 2019
New York, NY

Abstract

How does orthogonality help Neural Networks? From an optimization perspective, orthogonality limits the impact of vanishing and exploding gradient during backpropagation by achieving *dynamical isometry*. To achieve this state, we initialize the network's weight matrices as random orthogonal matrices, which consequently leads to learning in the Stiefel manifold of orthogonal matrices. However, orthogonality does not necessarily sustain throughout training and, eventually, the weight matrices can diverge from the manifold. Regularization techniques can then help to maintain the weight matrices within that manifold, either by imposing hard constraint (continuously enforcing weights to remain orthogonal) or soft constraints (enforcing singular values of weight matrices to be close to one). We explore in depth the idea of *dynamical isometry* and propose a gain-adjusted orthogonal regularizer to leverage the gain factor of initialization further. This approach was empirically found to help performance on the Sequential MNIST dataset modelled with traditional RNN.

1 Introduction

Since backpropagation was invented and started being utilized during the 80's, people began to discover both how powerful neural networks can be and how difficult it is to train them. The vanishing/exploding gradient nature of their architecture is one of the main reasons of the abandonment of neural networks by the machine learning community in the 90's and their comeback did not happen until early 2010. Since then, neural networks have proven to perform well in a wide range of tasks in fields such as Computer Vision, Natural Language Processing, Speech Recognition and Reinforcement Learning with the state-of-the-art results of Google's AlphaGo. One reason for the success of deep learning is the possibility of arbitrarily increasing the depth of neural networks to enable them learning complex representations of inputs. With sufficient resources in data abundance and strong compute power, the performance of deep neural networks can be very impressive.

Nonetheless, this success comes at a high cost of optimization instability (local minima) and divergence. Many optimizers and regularization methods have been proposed in an attempt to overcome these inevitable convergence challenges of fitting a model. In 2014, Saxe et al. [2013] first introduce in their paper on learning dynamics of deep linear networks a new method of initializing weights with random orthogonal matrices. Since then, several other researchers analyzed this approach of imposing orthogonality in neural networks and its drastic improvement in network's learning. Finally, the idea of orthogonality was extended to regularizers and how to maintain this orthogonality from initialization throughout the network's training, i.e. a regime of so-called *dynamical isometry*.

Preprint. Under review.

In this paper, we investigate the concept of *dynamic isometry* and aim to demonstrate its impact on the dynamics of learning for a neural network, i.e. attempting to achieve faster and better convergence. We study the impact of orthogonal initialization specifically on both linear as well as non-linear neural networks. Furthermore, we explore both in theory and empirically the effect of orthogonality beyond initialization (i.e. during training) by imposing hard/soft constraint, which leads to matrices within/close to the Stiefel Manifold. Finally, we propose a new soft orthogonal regularizer that benefits from a slightly positive gain in initialization to remain beyond the edge of chaos [1]. To test its performance and demonstrate its value, we run experiments on the SeqMNIST dataset for recurrent neural networks.

2 Dynamical Isometry in Neural Networks

To understand what *dynamical isometry* is, we first need to define some necessary mathematical notation. More specifically, we need to define the input-output Jacobian of a neural network as well as a characterization of its spectral density[2].

We restrict our study to a Multi-Layer Perceptron (MLP) of L layers, each of width N. All layers are connected with weight matrices $W_i \in \mathbb{R}^{N\times N}$, bias vectors b_i, pre-activations h_i and post-activations x_i for $i \in \{1, ..., L\}$.
The forward equations for a point-wise non-linear activation function $\phi : \mathbb{R} \to \mathbb{R}$ are:

$$x_i = \phi(h_i), \quad h_i = W_i x_{i-1} + b_i \tag{1}$$

We denote $h_0 \in \mathbb{R}$ the input of the neural network. From this set of equations, we can define the input-output Jacobian of this neural network.

Definition 2.1. *Input-output Jacobian*. The input-output Jacobian, *J*,

$$\mathbf{J} = \frac{\partial x_L}{\partial h_0} \tag{2}$$

is the partial derivative of the last post-activation x_L with respect to the input h_0. Since output and input have the same dimension, here $\mathbf{J} \in \mathbb{R}^{N\times N}$.

Remark. The Jacobian can be computed explicitly for an MLP, but also for a more complex architecture such as RNN.

Theorem 2.1. *Explicit form of input-output Jacobian.* We denote D_i the Jacobian of the post-activation function x_i *w.r.t.* the pre-activation function h_i. Then we can write:

$$\mathbf{J} = \prod_{i=1}^{L} D_i W_i^T, \tag{3}$$

with $D_i = diag(\phi'(h_i))$.

Proof. First, let us derive the Jacobian of the layer $i+1$ *w.r.t.* layer i denoted $\mathbf{J_i}$.

Knowing that $x_{i+1} = \phi(h_{i+1})$ and that $h_{i+1} = W_{i+1}x_i + b_{i+1}$, we use the multi-dimensional chain-rule to write:

$$\frac{\partial x_{i+1}}{\partial x_i} = \frac{\partial x_{i+1}}{\partial h_{i+1}} \frac{\partial h_{i+1}}{\partial x_i}$$

For the moment, we only focus on the first term. As $x_{i+1} \in \mathbb{R}^N$ and $h_{i+1} \in \mathbb{R}^N$, we can see the mapping f as $f : \mathbb{R}^N \to \mathbb{R}^N$ such that $x_{i+1} = f(h_{i+1})$. Hence, in the matrix form, D_{i+1} is the Jacobian of f and is a matrix $N \times N$.

Since $x_{i+1} = \phi(h_{i+1})$ with ϕ an being element-wise function, only $h_{i+1,j}$ is linked to $x_{i+1,j}$, where the second index j represents the element at index j of a vector. Finally, we obtain that

$$\forall (k, l) \in 1, .., N, \quad \frac{\partial x_{i+1,k}}{\partial h_{i+1,l}} = \phi'(h_{i+1,l})\delta_{l,k},$$

[1] Refer to Figure 1

[2] The high level derivation of these concepts can be found in Pennington et al. [2017].

with δ being the standard Kronecker delta.

We define $diag(.)$ as the operator that maps a vector $x \in \mathbb{R}^N$ to a matrix $M \in \mathbb{R}^{N \times N}$ such that $M_{i,j} = x_i \delta_{i,j}$.

This enables us to rewrite the Jacobian post-to-pre-activation in a simpler form as

$$\frac{\partial x_{i+1}}{\partial h_{i+1}} = D_{i+1} = diag(\phi'(h_{i+1})).$$

The second term of the Jacobian of the layer $i+1$ *w.r.t.* the layer i is simply the Jacobian of h_{i+1} *w.r.t.* x_i. Since $h_{i+1} = W_{i+1}x_i + b_{i+1}$, simple matrix calculus yields

$$\frac{\partial h_{i+1}}{\partial x_i} = W_{i+1}^T$$

As a result, we can write the Jacobian of the layer $i+1$ *w.r.t.* layer i as

$$\mathbf{J_i} = \frac{\partial x_{i+1}}{\partial x_i} = D_{i+1} W_{i+1}^T.$$

Subsequently, we can compute the input-output Jacobian by applying the chain rule and obtain

$$\mathbf{J} = \prod_{i=1}^{L} \frac{\partial x_i}{\partial x_{i-1}} \frac{\partial x_0}{\partial h_0} \tag{4}$$

By construction of the neural network, $x_0 = h_0$ so $\frac{\partial x_0}{\partial h_0} = I$. Or it could be seen that x_0 is undefined and thus starts the product at $i = 2$, which is equivalent.

We can now use our formula of the Jacobian of the layer $i+1$ *w.r.t.* layer i to conclude that:

$$\mathbf{J} = \prod_{i=1}^{L} D_i W_i^T$$

□

The input-output Jacobian $\mathbf{J}$ is of great importance for us to understand the dynamics within the neural network. Indeed, for computing the gradient of the loss function w.r.t. any weight matrix W_i, a part of the product within $\mathbf{J}$ will be present. As we perform back-propagation at deeper and deeper layers, (*i.e.* the layers close to the input and far from the output), we obtain gradients that are more and more related to the input-output Jacobian.

Vanishing and exploding gradient tends to commonly occur at the deep layers of neural network, *i.e.* at those that are close to the input and far from the output. At the early stages of the training, where the network is in the "discovery" phase and only starts to learn, having vanishing gradients at deep layer implies that nothing is being learnt by the network in bottom layers. Thus, top layers have to learn from randomly initialized transformation of inputs induced by the randomly initialized bottom layers (since deep layers impact all top layers). This can drastically affect the convergence during training, with either an under-fitting problem or a very slow learning. Also, potential exploding gradients at deep layer can be catastrophic for the learning of the network, as those large deep updates impact all subsequent layers, and can then make the optimization procedure of the neural network diverge.

In such case, the norm of the Jacobian matrix acts as a proxy for the vanishing/exploding gradients. More precisely, the norm preserving nature of the Jacobian is an important property that we seek to obtain and, more importantly, maintain in a neural network. Since the Jacobian is a matrix, we can regard it can be seen as a linear transformation, and a linear transformation that preserves norm is called an *isometry*.

Definition 2.2. *Isometry for matrices*. For any linear transformation $M \in \mathbb{R}^{N\times K}$, the transformation is said to be a *linear isometry* if and only if

$$\forall v \in \mathbb{R}^{\mathbb{K}}, \quad \|Mv\| = \|v\| \,.\ ^{3}$$

Definition 2.3. *Condition number for matrices*. A matrix $M \in \mathbb{R}^{N\times K}$ is a linear transformation that maps $\mathbb{R}^K$ to $\mathbb{R}^N$. For any consistent norm $\|.\|$ operating on the matrix space induced by $\mathbb{R}^{N\times K}$, we can define the condition number of that matrix as:

$$\kappa(M) = \left\|M^{-1}\right\| \|M\| \,. \tag{5}$$

In a standard Euclidean space, this scalar is equivalent to the ratio of the highest singular value to the smallest singular value of M. Noting $\sigma(M)$ the spectral density of M with lower bound $\sigma_{min}(M)$ and higher bound $\sigma_{max}(M)$, we obtain

$$\kappa(M) = \frac{\sigma_{max}(M)}{\sigma_{min}(M)}. \tag{6}$$

A matrix has a condition number of exactly 1 if it is a linear isometry. The condition number of a matrix is thus a good proxy of its norm-preserving nature.

The concept of *dynamical isometry* can be therefore understood as a process of determining where the input-output Jacobian is close to a isometry and, thus, norm-preserving. This enables us to quantify the vanishing/exploding impact of a neural network related to one single object. More specifically, monitoring the condition number is equivalent to tracking the eigenvalue spectrum of the input-output Jacobian. In conclusion, we can expect the input-output Jacobian to be well-conditioned if its eigenvalue spectrum is centered on 1 (unit mean) and has a small variance.

In the following steps, we aim to derive the distribution of the spectrum of $\mathbf{J}$.

As $N \to \infty$, the empirical distribution of pre-activation $h_i = W_i x_{i-1} + b_i$ converges to $N(0, q_i)$. This is namely due that each $h_{i,k}$ is a weighted sum of a large number of uncorrelated random variable. Intuitively, that comes comes from the independence of h_{i-1}, W_i and b_i . We obtain a recursive equation for the Var(h_i):

$$q_i = \sigma_w^2 \mathbb{E}_h[\phi(\sqrt{q_{i-1}}h)^2] + \sigma_b^2 \text{ with } h \sim N(0,1), \tag{7}$$

where σ_w^2 equals to the variance of weights, σ_b^2 the variance of bias, and the initial condition $q_0 = \frac{1}{N}\sum_{i=1}^{N} h_{i,j}^2$. The recursion has a fixed point q^*:

$$q^* = \sigma_w^2 \mathbb{E}_h[\phi(\sqrt{q^*}h)^2] + \sigma_b^2 \tag{8}$$

Proof. We assume assume that all weight matrices W_i have a variance of σ_w^2, and b_i have a variance of σ_b^2. This is especially true at initialization, when all weights are initialized for some random matrices/vectors with a certain variance. We assume the weights and biases to be independent from each other, which at least at initialization is always true.

Furthermore, as those random weights and biases comes from layer i, they are completely independent of the pre-activation h_{i-1} at the precedent layer. Hence, we assume $Var(W_i) = \sigma_w^2$, $Var(b_i) = \sigma_b^2$ and finally $W_i \perp\!\!\!\perp h_{i-1} \perp\!\!\!\perp b_i$. Then adding that $h_i = W_i\phi(h_{i-1}) + b_i$, we get naturally that:

$$q_{i+1} = Var(h_{i+1}) = Var(W_{i+1}\phi(h_i) + b_{i+1}) = \sigma_w^2 \mathbb{E}_h(\phi(\sqrt{q_i}h)^2) + \sigma_b^2 \text{ with } h \sim N(0,1)$$

□

Also, we assume that the network should be close to that regime. If we pick the input h_0 to be close to q_*, then the neural network will start at the fixed point. If not, then after a few layers, the pre-activation variance should converge to this fixed point.

[3] This can also be mathematically formulated within the framework of conditioning of matrices.

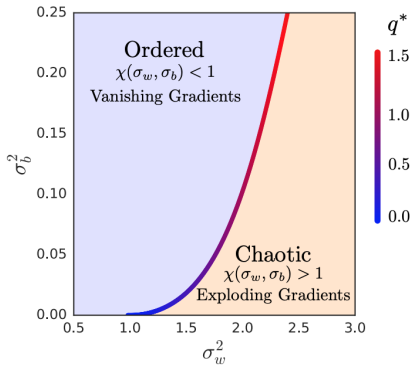

Figure 1: Order-chaos transition for $\phi = tanh$. From Pennington et al. [2017].

Pennington et al. [2017][4] claim that Poole et al. [2016] and Schoenholz et al. [2016] showed an important result on the mean of the singular values of the layer-to-layer Jacobian $\mathbf{J_i}$, noted χ. Namely:

$$\chi(q_*) = \sigma_w^2 \mathbb{E}(\phi'(\sqrt{q_*}h)^2)$$

From this, Pennington et al. [2017][1] claim that Poole et al. [2016] and Schoenholz et al. [2016] proved that the mean of the input-output Jacobian is the product of the means of the layer-per-layer Jacbobian. Therefore, they claim that the mean of the squared singular values of $\mathbf{J}$ is simply χ^L.

As we assume L to be large, the conditioning of the input-output Jacobian will depend drastically on the value of χ. As χ depends on q_* which itself depends on σ_w and σ_b, we can define a critical line $\chi(\sigma_w, \sigma_b) = 1$ between a chaotic phase ($\chi > 1$) of exploding gradient and an ordered phase ($\chi < 1$) of vanishing gradients.

The authors then look into the case of a hyperbolic tangent activation function and how the critical curve behaves for different σ_w and σ_b. The results can be seen in Figure 1.

We start an in-depth analysis of deep linear networks and investigate their learning dynamics, referring to the work of Saxe et al. [2013]. Subsequently, we approach non-linear networks and explore how different activation functions achieve (or fail to achieve) dynamical isometry.

3 Orthogonal Weight Initialization in Linear and Non-linear Networks

Saxe et al. [2013] started this interesting sub-field of deep learning on introspection of dynamical isometry by studying deep linear networks. They found that dynamical isometry can be achieved by random orthogonal matrices and, thus, proposed it as a new initialization technique. For a deep linear network, they obtain a training time (number of epochs required to achieve a certain degree of accuracy) that is independent of depth, unlike Gaussian initialization that scales linearly with depth.

In the case of a deep linear network, we have $\phi(x) = x$ that allows for a great simplification of the input-output Jacobian to

$$\mathbf{J} = \prod_{i=1}^{L} W_i^T.$$

Also, equation (8) gets reduced to $q_* = q_* \sigma_w + \sigma_b$, which for a non-bias network can be further simplified to obtain the only solution at $(\sigma_w, \sigma_b) = (1, 0)$.

In the case of random orthogonal matrices, *i.e.* $WW^T = I$, the input-output Jacobian is clearly orthogonal as well as the product of only orthogonal matrices. Since orthogonal matrices have the property of all their singular values being one, they achieve perfect dynamical isometry.

[4] After long research none of those proofs were found in the corresponding papers.

For random Gaussian matrices, Pennington et al. [2017] use a result of Wishart matrix product to derive the squared singular value density of $\mathbf{J}$. They specifically find that s^2_{max} scales linearly with the depth.

As seen in Figure 2, it is empirically found by Saxe et al. [2013] that random orthogonal matrices allow a constant time for training a deep linear network, while for random Gaussian matrices, training time scales linearly with the depth.

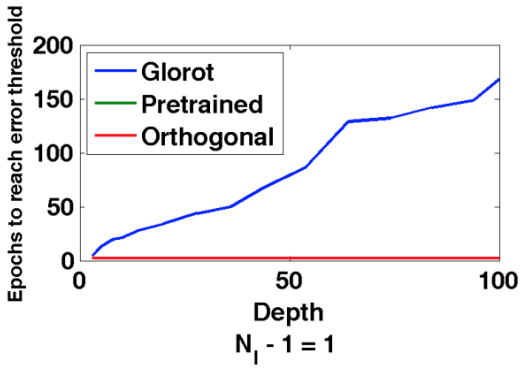

Figure 2: Time of training to reach a certain accuracy threshold on MNIST for different initialized deep linear network. From Saxe et al. [2013].

This simple analysis on deep linear networks makes us confident that dynamical isometry exists, that it can impact significantly training and finally that orthogonal initialization is a relatively cheap and efficient scheme [5].

While orthogonal initialisation and isometry seems to be achievable in a linear case, the next question is its applicability to non-linear networks.

We define $p(q^*)$ the probability that a neuron (after a non-linear activation function ϕ) is in linear regime, *i.e.* that $\phi'(x) = 1$. We will only inspect the non-linearities ReLU and hard-tanh. Both of them have in common that they are linear and then constant elsewhere with $\phi'(x) = 0$.

Hence it is easy to get that in both scenarios $\chi = \sigma_w^2 p(q_*)$. With the assumption that with large width, $h_i \sim N(0, q_*)$ or ReLU, $p(q)* = \frac{1}{2}$, while for hard-tanh $p(q_*) = erf(\frac{1}{\sqrt{2q_*}})$.

In term of conditioning of $\mathbf{J}$, ReLU networks can achieve partial isometry with $\chi = 1$ if and only if $\sigma_w^2 = 2$, which then corresponds to only one fixed point q_* at $(\sigma_w, \sigma_b) = (\sqrt{2}, 0)$. For hard-tanh, isometry is achieved on an entire curve of (σ_w, σ_b) (as seen in Figure 1).

However, this analysis takes only in consideration the mean of the singular spectrum of the input-output Jacobian, and we would like to look instead at its highest squared singular values. Pennington et al. [2017] give a method using random matrix theory and the *Stieltjes transform* S to link S_{WW^T} and $\sigma(WW^T)$.

Then doing a distinction between random Gaussian matrix and random Orthogonal matrix, they get interesting results.

For the Gaussian case, $s^2_{max} = (\sigma_w^2 p(q_*))^L (\frac{eL}{p(q_*)} + \mathcal{O}(1))$. As well, using similar methods, it can be shown that $\sigma^2_{JJ^T} = \frac{L}{p(q_*)}$. Therefore, with Gaussian initialization, whether looking at the spectral density of the Jacobian in term of variance or maximum value, the ill-conditioning seems to grow linearly with the depth L. Also, as $p(q_*)$ is between 0 and 1, no value can be taken so that we obtain a good conditioning independent of depth. Therefore, no Gaussian initialized non-linear network can achieve dynamical isometry.

For the orthogonal case, an analysis of the Stieltjes transform of WW^T yields that $s^2_{max} = (\sigma_w^2 p(q_*))^L \frac{1-p(q_*)}{p(q_*)} \frac{L^L}{(L-1)^{L-1}}$. The only point where s^2_{max} does not either explode or vanish is

[5] In Saxe et al. [2013], a proof is given of the expected learning time for a linear network.

when $\chi = \sigma_w^2 p(q_*) = 1$ where it then behaves as $s_{max}^2 = \frac{1-p(q_*)}{p(q_*)}(eL - \frac{e}{2}) + \mathcal{O}(\frac{1}{L})$. As well, at critical point, *i.e.* if $\chi = \sigma_w^2 p(q_*) = 1$, then $\sigma_{JJ^T}^2 = \frac{1-p(q_*)}{p(q_*)} L$ for large L.

Keeping in mind those 2 equations, we can see that for ReLU networks, where $p(q_*) = \frac{1}{2}$, the linear scaling with L cannot be counter attack. Therefore dynamical isometry is not achievable for ReLU network, even in an orthogonal setting. With a tanh network however, with $p(q_*)$ non-constant and simply bounded within [0,1], it is possible to find such values to achieve isometry. Indeed for $p(q_*) \approx 1 - \frac{1}{L}$.

Interestingly enough, this counter-balance for large depth corresponds to increasing the amount of neurons that behave in a linear regime. Many experiments are conducted in Pennington et al. [2017] to confirm this result empirically. The most conclusive is the following, in Figure 3.

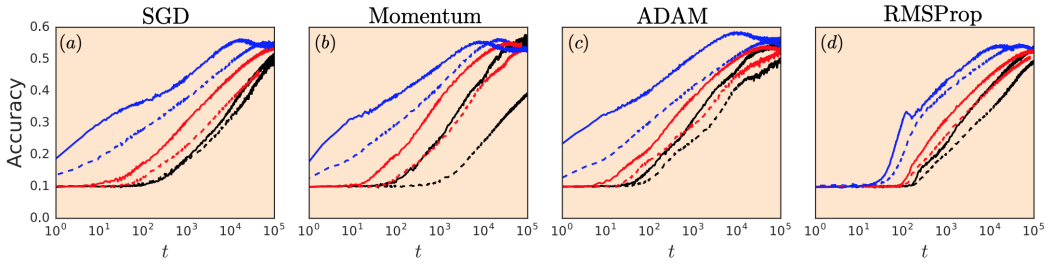

Figure 3: Generalization performance on the test set as a proxy for learning dynamics for networks of depth 200 and width 400 trained on CIFAR-10 with 4 different optimizer. Blue is tanh with $\sigma_w^2 = 1.05$, red is tanh with $\sigma_w^2 = 2$, and black is ReLU with $\sigma_w^2 = 2$. The solid lines represent orthogonal initialization and dashed lines are Gaussian initialized networks. From Pennington et al. [2017].

The ordering of the curves is relatively clear, regardless of optimizers: orthogonal initialization (which achieve isometry) is consistently better than Gaussian initialization. As well ReLU networks are much slower since they do not achieve isometry in any possible configuration.

Therefore, it is relatively interesting that orthogonal neural nets with hard-tanh activation functions can be working in a dynamical isometry, whereas ReLU network just cannot work in an isometry setting.

Now that it is clear that dynamical isometry can be reached within Neural Networks under certain initialization scheme and activation functions, and that it can be drastically beneficial for learning, we can ask our-self how to remain in this isometry during training. Indeed orthogonal initialization of weight matrices does not guarantee orthogonality and dynamical isometry after a first epoch of updates. This is where Orthogonal regularization comes as a potential solution.

4 Orthogonal Regularization with Hard and Soft Constraint

The idea of persisting in that orthogonal subspace by constraint and to keep dynamical isometry is related to the Stiefel manifold $\mathcal{V}_k(\mathbb{R}^n) = \{W \in \mathbb{R}^{n \times k} | WW^T = \mathrm{I}_{\mathbb{R}^{k \times k}}\}$.

Indeed, when initialized to random orthogonal matrices, the weights matrices of the Neural Network are all on the Stiefel manifold. Then, after a normal batch gradient update, the weight matrices naturally tend to get out of the manifold. Different methods exist to enforce hard constraints on the weight matrices, but they generally tend to be related to either applying a traditional update and project back to the manifold, or project the gradient within the manifold to then update directly within the Stiefel manifold.

A standard method is to apply the traditional update of the weight matrix, and then perform a QR decomposition in order to project that matrix onto the Steifel manifold. For instance, Vorontsov et al. [2017] use the Cayley transform to update weights matrices and enforce strictly orthogonality. It is however quite unclear from their paper what is the underlying intuition, except that the Cayley transform of a matrix and its gradient gives a new orthogonal matrix. Then, in Huang et al. [2017],

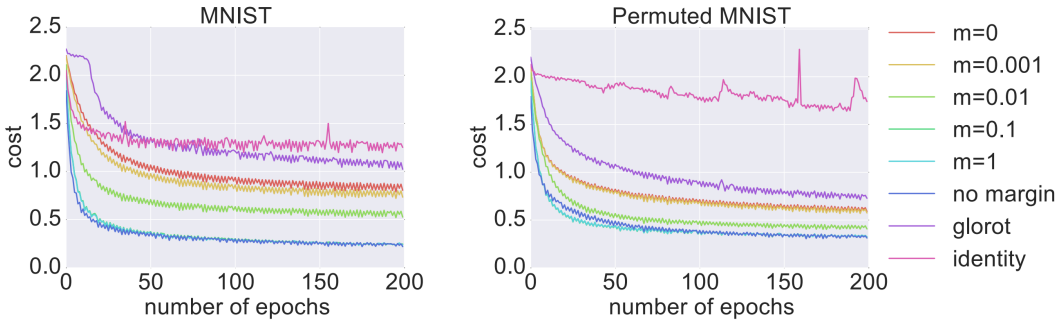

Figure 4: Sequential MNIST RNN with hard-constraint using *Cayley* transform to maintain singular values within a margin around 1: $\sigma(W) \in [1-m, 1+m]$. Best results obtained **without margin**! From Vorontsov et al. [2017]

the authors propose an Orthogonal Linear Module that optimizes a proxy matrix V such that $W = \phi(V) \in V_k(\mathbb{R}^n)$ and claim that it suffers less instability in convergence than Riemannian optimization method, for instance the Cayley transform.

One aspect to keep in mind regarding those methods, is that almost all involve some sort of SVD computation or matrix inverse computation that can yield numerical instability as well as very large computation, which is considerably slowing down training. Indeed, for each weight matrix of dimension $N \times N$, the cost of such operation is $\mathcal{O}(n^3)$.

However, Vorontsov et al. [2017] also worked on a relaxed constraint that would enforce all the eigenvalues of the weight matrices to be within an interval $[1-m, 1+m]$ where m is a margin hyperparameter. This sort of *soft-hard* constraint allows to compare different values of margin and their performance. For this, they train different RNN models on sequential MNIST (treat 28×28 pixels image as a sequence of 784 time-steps) and permuted MNIST (sequential MNIST but with a permutation applied within pixels that shuffles the spacial relationship of the sequence, making it a problem that requires longer term dependencies). Except the identity (initialized with I) and Glorot, all other models are initialized with random orthogonal matrices and then regularized with a certain margin m. See results on Figure 4 below.

From this experiment, we can see that the hard regularization is under-performing in comparison to both a large margin of 1, and no regularization at all. To go even further, it appears that here regularization is all-together failing. This is a reality to take into account: in some cases any orthogonal regularization might fail as it can perturb the gradient and/or limit the capacity of the model. One possible fix that author advise to try before giving up on orthogonal regularization is to simply increase the model capacity. But we could argue that this is a bad idea, as it will slow down training, and remind ourselves again that an orthogonal regularization is supposed to help accelerate convergence and reduce training time.

Another type of constraint that is more appealing is soft orthogonality. In this scenario, we do not aim to remain on the Stiefel Manifold but rather close to it. We penalize the deviation of a matrix from the Stiefel manifold using the Frobenius norm. The *single soft constraint* can be derived as follow:

$$\lambda \left\| WW^T - \mathrm{I} \right\|_F^2, (SO) \tag{9}$$

which has an explicit gradient $4\lambda W(WW^T - \mathrm{I})$. This allow for easy computation of gradients, and overall a regularizer that does not add any significant computational cost

We recall that the *Frobenius* norm is defined as:

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}|^2} = \sqrt{\sum_{i=1}^{\min\{m,n\}} \sigma_i^2(A)} \tag{10}$$

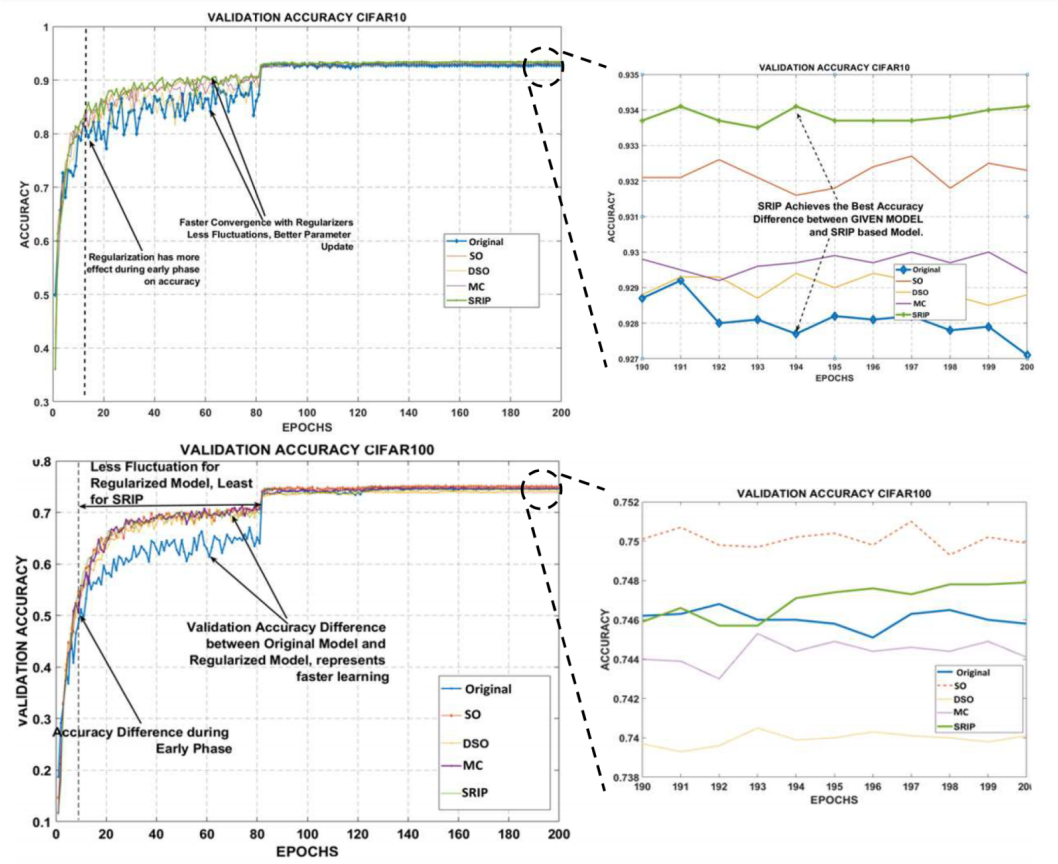

Figure 5: ResNet-110 architectures with decaying orthogonal regularization on CIFAR-10 and CIFAR-100 achieve faster convergence rate. From Bansal et al. [2018]

This norm directly penalizes the deviation of the squared singular values of W from one (the identity matrix).

Bansal et al. [2018] derive more soft constraints such as Double Soft Orthogonality Regularization, Mutual Coherence Regularization or Spectral Restricted Isometry Property Regularization. However the author find out that the Single Soft Orthogonalization is already a strong baseline and struggle to beat it.

They run an experiment involving ResNet-110 networks (with 110 layers) on CIFAR 10 and CIFAR-100 to benchmark them. The results are on Figure 5. For a very deep architecture like ResNet, it seems that all orthogonal regularization significantly help for training, especially at the beginning of training. One interesting fact is that all these regularizers are run with a decaying penalty. Therefore, they already assume that Orthogonal Regularization can hurt the expressiveness of a neural network or be an issue for final stage of training.

Ultimately, it seems that for very deep neural networks, Orthogonal Regularization can be a good idea (like for CNN), especially as it is relatively easy to implement and does not slow down training computation. However, it should always be bench-marked on model that do not have regularization before fully deciding that the model should be even regularized.

In the following experiments, we explored the possibility of a different regularizer that would enforce a stability in the gain factor of orthogonal initialization.

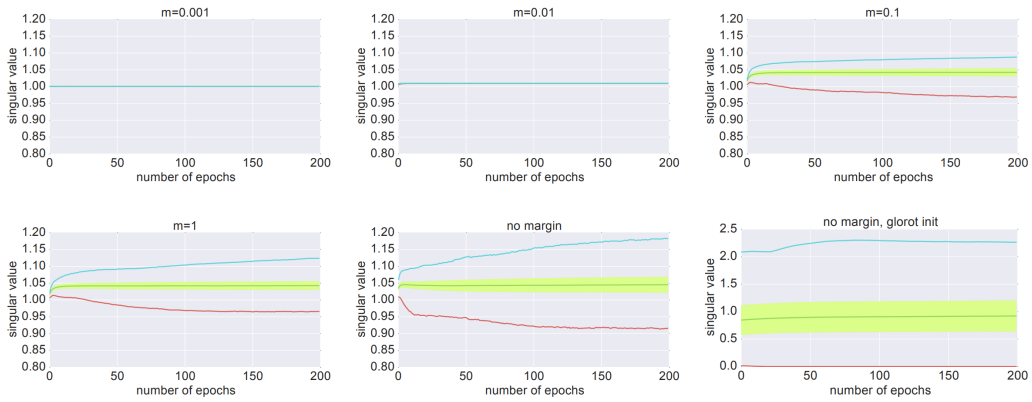

Figure 6: As described in Vorontsov et al. [2017]: "Singular value evolution on the permuted sequential MNIST task for factorized RNNs with different spectral margin sizes (m). The singular value distributions are summarized with the mean (green line, center) and standard deviation (green shading about mean), minimum (red, bottom) and maximum (blue, top) values. All models are initialized with orthogonal hidden to hidden transition matrices except for the model that yielded the plot on the bottom right, where Glorot normal initialization is used."

5 Experiments: Gain-adjusted Orthogonal Regularization in RNNs

Saxe et al. [2013] introduce a gain component as a part of their orthogonal initialization approach as a way to intensify the effect of dynamical isometry by entering beyond the edge of chaos [6]. We aim to experiment further and research the effect of imposing gain further beyond initialization, *i.e.* within orthogonal regularization during training.

The idea behind the gain factor is the following. In the layer-to-layer Jacobian, the weight matrix W_i is orthogonal and has norm of one, whereas the Jacobian D_i of the activation function is a contractive linear mapping in the case of squashing activation functions like sigmoid or tanh. This is due to the fact that the derivatives of these functions are all within 0 to 1, and hence the D_i linear mapping is norm decreasing, *i.e.* $\forall v \in \mathbb{R}^N, \|Dv\| \leq \|v\|$. Therefore, to counter the possible vanishing nature of the input-output Jacobian, we re-scale the weight matrix by a gain factor $g > 1$.

Thus, for the initialization, we randomly sample an orthogonal matrix V such that $VV^T = I$ and then re-scale it by g:[7] $W = gV$. Ultimately, $Var(W) = \mathbb{E}(WW^T) = \mathbb{E}(g^2VV^T) = g^2\mathbb{E}(I) = g^2I$.

We also note that in Vorontsov et al. [2017], the authors look at the evolution of the spectrum density of the recurrent kernel weight matrices and find it to be centered at a value slightly higher than 1, although the recurrent kernel matrix was initialized orthogonally and, hence, was orthogonal at epoch 0 (i.e. all singular values exactly one). Therefore, they run that experiment without adding gain, but probably for better convergence, the Neural Network performed in that sort of *boosted isometry* regime, right beyond the *edge of chaos*, as described in Saxe et al. [2013]. The results of this experiment can is shown in Figure 6.

To further examine the regularization methods, we propose to adjust the orthogonal initialization by the gain factor not only in the initialization, but also in the regularization phase. Building upon the apparently well-performing and smooth method of *single soft orthogonal regularization*, we show that the changes required are minimal.

Definition 5.1. (Gain-adjusted Orthogonal Regularization) We argue that if W is initialized as gV with $VV^T = \mathrm{I}$, then the gain will be lost during training with a standard orthogonal regularization.

[6] Refer to Figure 1.

[7] This quantity is also called σ_w in other precedent derivations.

Hence, we propose a "**Gain-adjusted Orthogonal Regularization**":

$$\lambda \left\| \frac{1}{g^2} W W^T - \mathrm{I} \right\|_F^2 \tag{11}$$

which has an explicit gradient $\frac{4\lambda}{g^4} W(WW^T - \mathrm{I})$.

This way, we can maintain that $VV^T = \frac{1}{g^2} WW^T \approx \mathrm{I}$, *i.e.* V orthonormal and W orthogonal with a norm of g^2.

We take on to run experiments on the following 2 datasets for RNN (that can easily have long sequences and therefore be very deep architecture):

1. sequential MNIST (row version): instead of flattening each image to a vector of size 784, we run an RNN that takes in one row per time step, constituted therefore of 28 times steps of dimension 28 each. The architecture is a RNN with architecture many-to-one and trained on cross-entropy loss. The hidden states dimensions are 64, and the batch size is set to 256. Training, validation and test input tensors are of size: $[(48000, 28, 28)(12000, 28, 28)(10000, 28, 28)]$
2. Sentiment Analysis from Netflix reviews dataset:. We apply GloVe (from Pennington et al. [2014]) as an embedding on all tokens, and pad with a sequence length of 50. The model is a CuDNN LSTM (CuDNN simply to leverage NVIDIA's GPU) with architecture many-to-one and trained on binary cross-entropy loss. The hidden states dimensions are 128, and the batch size is set to 128. Training, validation and test input tensors are of size: $[(4264, 50, 300), (533, 50, 300), (533, 50, 300)]$

As expected the code change required for creating a gain adjusted regularizer is minimal:

```
def orth_reg_gain(W, gain, reg_orth):
    return reg_orth * K.mean(K.square(1/(gain**2) * K.dot(K.transpose(W), W) - 1))
```

All models are implemented in Keras and can be found in the Jupyter notebook *OrthogonalRegRNNSV.ipynb*. The python 3 dependencies for packages are *keras, flair, torch, tqdm*. Keras makes it very easy to create a callback and thus track the singular values of the recurrent kernel weight matrices (for RNN, that is W such that $h_t = \phi(Wh_{t-1} + Ux_t + b)$).

```
class SingularValuesCallback(Callback):
    def __init__(self, layer_name):
        self.layer_name = layer_name
        self.sv = []

    def on_train_begin(self, logs=None):
        layer_dict = dict([(layer.name, layer) for layer in self.model.layers])
        self.layer = layer_dict[self.layer_name]
        self.sv.append(svd(self.layer.get_weights()[1], compute_uv=False))

    def on_epoch_end(self, epoch, logs=None):
        # compute singular values and save
        svs = svd(self.layer.get_weights()[1], compute_uv=False)
        self.sv.append(svs)
```

The final results are two folds, one will focus on the singular values tracking during training, while the other will be focusing on the performance through the training curve on validation loss. Figures 7 and 8 are the results for the RNN on Sequential MNIST, while Figures 9 and 10 are the results for the RNN on Sentiment Analysis.

The different curves have the following definitions:

1. **Orthogonal_GainA_adj_RegB**: Orthogonal initialization + gain factor $g = A$ + orthogonal regularization **WITH** adjustment and penalty $\lambda = B$

2. **Orthogonal_GainA_RegB**: Orthogonal initialization + gain factor $g = A$ + orthogonal regularization **WITHOUT** adjustment and penalty $\lambda = B$
3. **Orthogonal_GainA_NoReg**: Orthogonal initialization + gain factor $g = A$ (no orthogonal regularization
4. **Orthogonal_NoReg**: Orthogonal initialization (no gain and no orthogonal regularization)
5. **NoOrthogonal**: Glorot initialization (no orthgonal initialization, no gain, no orthogonal regularization)

We start with the results on Sequential MNIST. From Figures 7 and 8, we can see that the orthogonal initialization has a large impact on the eigen-spectrum of the recurrent kernel matrix of the RNN. Indeed, the massive pink area represents a kernel matrix not well conditioned. For that matter, the training curve on the validation accuracy is in average lower for the non-orthogonal initialization than the orthogonal ones. Then, looking at the eigenvalue spectrum of the orthogonal matrices, we can see that the impact of the different regularizer in very subtle, close to insignificant. However, hen comparing the validation results, we can see that the top performer, even after 1 epoch, is a gain-adjusted regularized model (red curve). In general, we can compare the gain-adjusted vs non gain-adjusted models with same penalty (compare red vs green with red ours, and blue vs orange with blue ours). The gain-adjusted regularized models are always outperforming the non-adjusted one, giving us confidence that the modification in the regularizer can be beneficial.

Then, the results on the Sentiment Analysis task are not as conclusive. We basically observe on Figure 10 that all models are behaving the same way, and that the orthogonal regularization, like the orthogonal initialization did not convey anything in term of performance. This can be better understood by looking at Figure 9, where it is clear that the non-orthogonal matrix is behaving better than the orthogonal ones. This is potentially due to a specific behavior of LSTMs. In any case, the orthogonality altogether is clearly not useful for that task and that model.

6 Conclusion

We have researched the consequences of orthogonal initialization and regularization in neural networks and their intrinsic relationship to *dynamical isometry* - a property that can help achieve faster and better convergence. Our experiments showed that while initializing weight matrices orthogonally seems to enhance (and never hurt) performance, orthogonal regularization performs contingent upon the regularization constraint and the depth of the neural network. Soft constraints, *i.e.* soft orthogonal regularizers, are preferred over those with hard constraint for their simplicity and lower computational cost. Finally, we derived a new soft orthogonal regularizer with a gain-adjusted component, which allows to benefit from dynamical isometry right beyond the edge of chaos - thus alleviating the contracting nature of squashing activation functions - during the entire training and not only at initialization. Our experiments were demonstrated on the Sequential MNIST dataset in recurrent neural networks, where gain adjusted orthogonal regularizer achieves better performance than its non-adjusted counterpart.

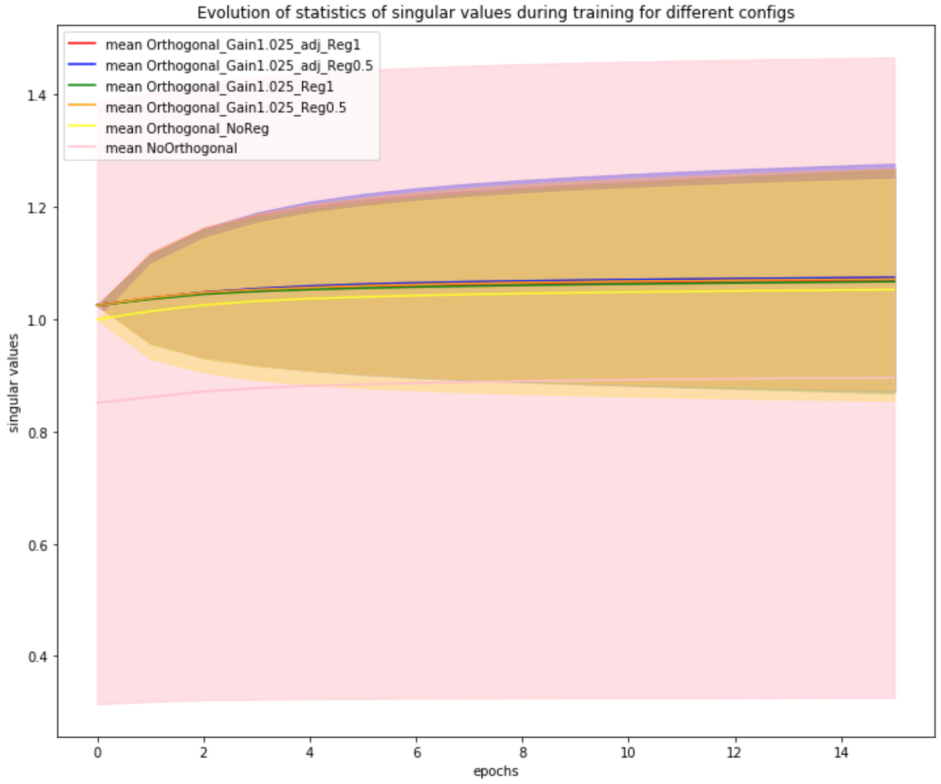

Figure 7: **Sequential MNIST** experiment: Eigenvalue spectrum of recurrent kernel matrix during training (area is mean $\pm$ std)

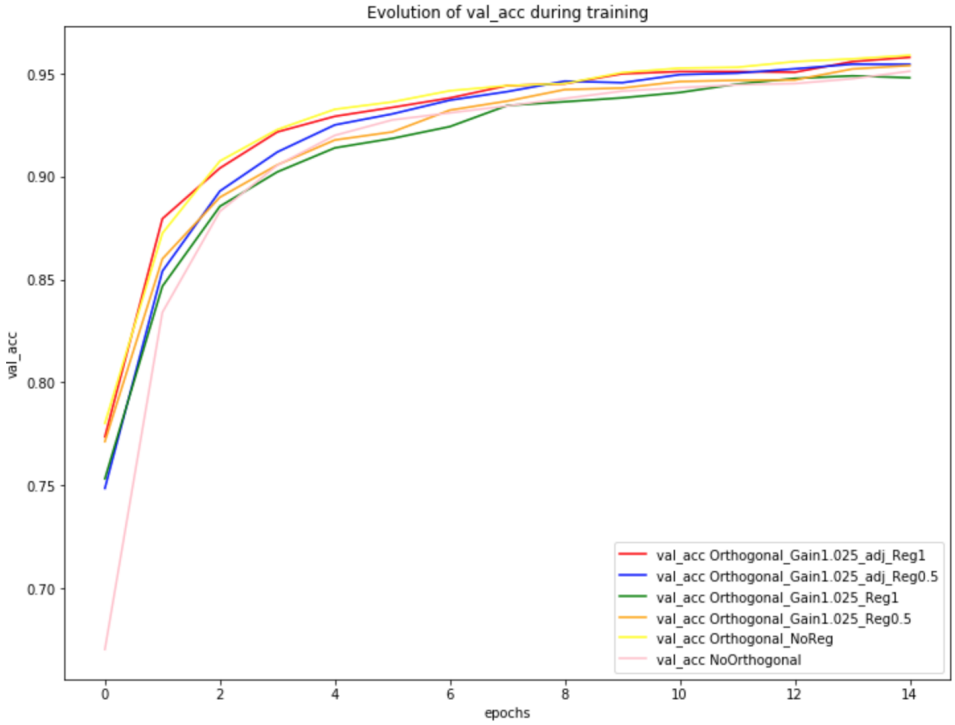

Figure 8: **Sequential MNIST** experiment: Evolution of validation accuracy for different configurations

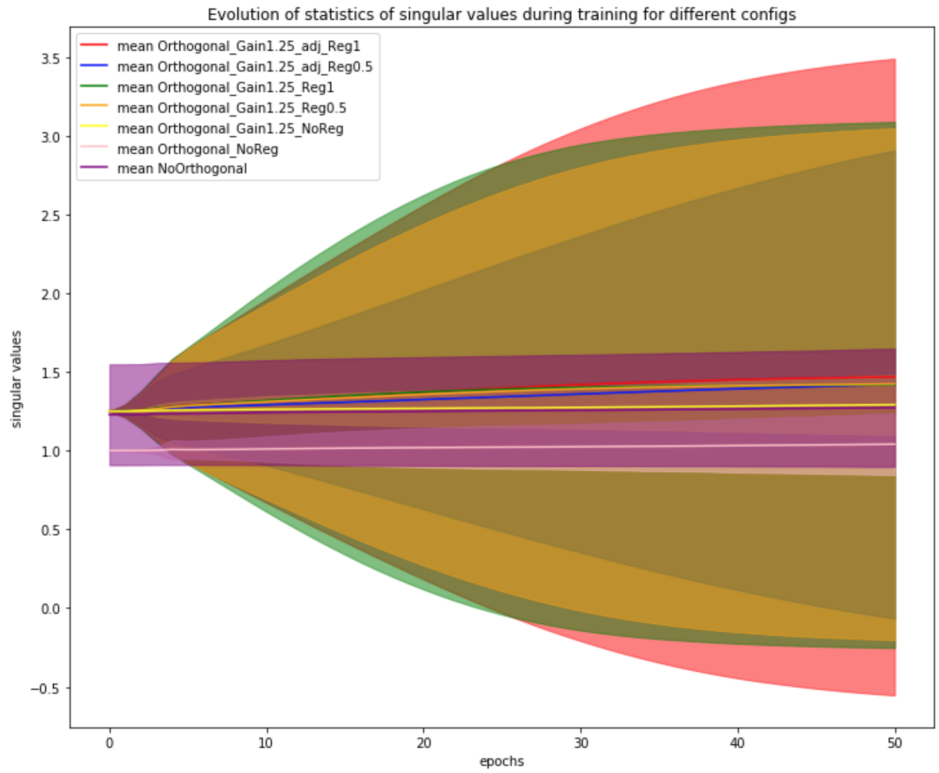

Figure 9: **Sentiment Analysis** experiment: Eigenvalue spectrum of recurrent kernel matrix during training (area is mean $\pm$ std)

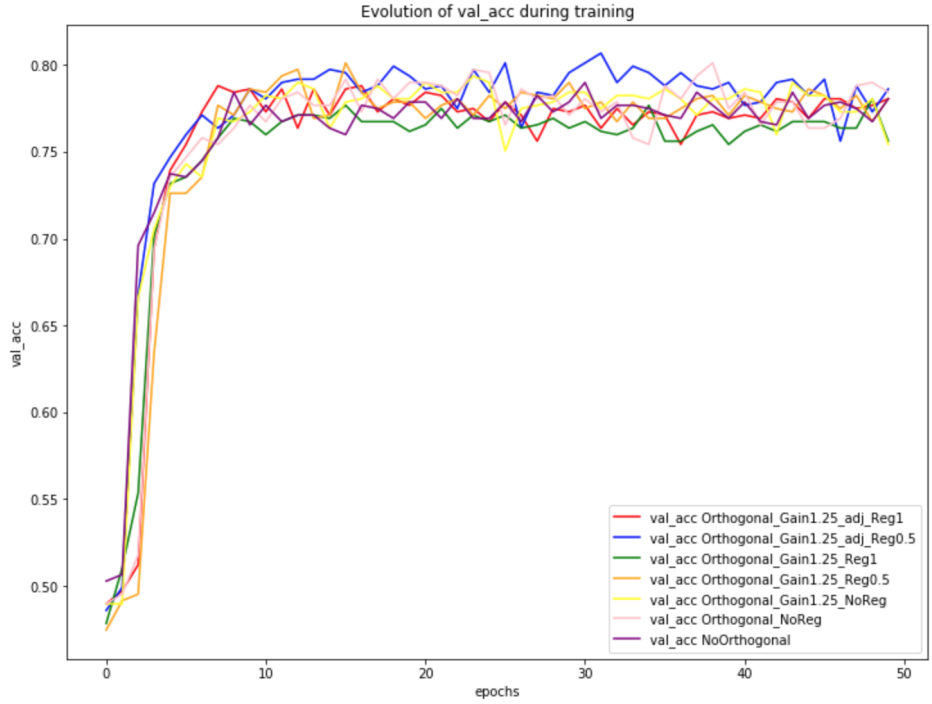

Figure 10: **Sentiment Analysis** experiment: Evolution of validation accuracy for different configurations

References

Nitin Bansal, Xiaohan Chen, and Zhangyang Wang. Can We Gain More from Orthogonality Regularizations in Training Deep CNNs? *arXiv e-prints*, art. arXiv:1810.09102, Oct 2018.

Lei Huang, Xianglong Liu, Bo Lang, Adams Wei Yu, Yongliang Wang, and Bo Li. Orthogonal Weight Normalization: Solution to Optimization over Multiple Dependent Stiefel Manifolds in Deep Neural Networks. *arXiv e-prints*, art. arXiv:1709.06079, Sep 2017.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *In EMNLP*, 2014.

Jeffrey Pennington, Samuel S. Schoenholz, and Surya Ganguli. Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. *arXiv e-prints*, art. arXiv:1711.04735, Nov 2017.

Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. *arXiv e-prints*, art. arXiv:1606.05340, Jun 2016.

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv e-prints*, art. arXiv:1312.6120, Dec 2013.

Samuel S. Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. Deep Information Propagation. *arXiv e-prints*, art. arXiv:1611.01232, Nov 2016.

Eugene Vorontsov, Chiheb Trabelsi, Samuel Kadoury, and Chris Pal. On orthogonality and learning recurrent networks with long term dependencies. *arXiv e-prints*, art. arXiv:1702.00071, Jan 2017.