PROJECTED WEIGHT REGULARIZATION TO IMPROVE NEURAL NETWORK GENERALIZATION

Guoqiang Zhang[*] *, Kenta Niwa*[†]*, and W. Bastiaan Kleijn*[‡]

[*] University of Technology Sydney, Australia
[†] NTT Media Intelligence Laboratories, Japan
[‡] Victoria University of Wellington, New Zealand

ABSTRACT

Generalization of a deep neural network (DNN) is one major concern when employing the deep learning approach for solving practical problems. In this paper we propose a new technique, named projected weight regularization (PWR), to improve the generalization capacity of a DNN model. Consider a weight matrix W from a particular neural layer in the model. Our objective is to make the eigenvalues of the matrix product WW^T have comparable or roughly the same magnitudes while allowing the DNN model to fit the training data sufficiently accurate. Intuitively speaking, by doing so, it would prevent the W matrix from matching the training data too well. Specifically, at each iteration, we first project the W matrix to a number of vectors along randomly generated directions. After that, we build an objective function of the projected vectors to regularize their behaviours towards comparable eigenvalue magnitudes of WW^T. Experimental results on training VGG16 for CIFAR10 show that PWR combined with centered weight normalization (CWN) yields promising validation performance compared to orthonormal regularisation combined with CWN.

Index Terms— DNN, projected weight regularization, CWN.

1. INTRODUCTION

How to train a deep neural network (DNN) to maximize its generalization capacity has been a challenging task. The training process may be affected by various factors such as the nature of nonlinear activation functions, weight initialization, neural network architectures, and optimization methods like stochastic gradient descent (SGD). In the past few years, different techniques have been proposed to improve the training process from different perspectives. Considering selection of the activation function, the rectified linear unit (ReLU) was found to be much more effective than the binary unit in feed-forward neural networks (FNNs) and convolutional neural networks (CNNs) [1]. Careful weight initialization based on the properties of the activation function and layerwise neuron-number has also been found to be essential for effective training (e.g., [2]). Nowadays, neural networks with shortcuts (e.g., ResNet [3] and Unet [4]) become increasingly popular as introduction of the shortcuts greatly alleviates the issue of gradient vanishing or explosion, which become severe issues when training extremely deep neural networks. From the optimization point of view, SGD with momentum is empirically found to produce DNNs with good generalization capacity over other gradient based methods (e.g., Adam [5], AdaGrad [6], RMSProp [7]).

Author emails: guoqiang.zhang@uts.edu.au, niwa.kenta@lab.ntt.co.jp, bastiaan.kleijn@ecs.vuw.ac.nz

In recent years, a family of normalization techniques have been proposed to accelerate the training process and produce high quality DNN models. The motivation behind these techniques is to make proper adjustment at each individual layer so that either the input or output statistics of the activation functions of the layer are unified in terms of the first and/or second moments. By doing so, the problem of internal covariance shift can be largely alleviated, thus significantly improving the efficiency of the back-propagation optimization methods. Those techniques can be roughly classified as (a): data-driven normalization, (b): activation-function normalization, and (c): weight-driven normalization.

We now briefly review the above three normalisation techniques. Data-driven normalization operates directly on the layer-wise internal features of training data, which includes for example batch normalization [8], layer normalization [9], and iterative normalization [10]. This type of normalisations was shown to be remarkably effective but one often has to carefully handle the inconsistency between training and inference, as the input statistics at the inference stage might be changed due to a reduced number of input samples. Activation-function normalization intends to design proper activation functions that are able to keep certain statistics unchanged between its input and output [11]. Weight-driven normalization indirectly regulate the statistics of the layer-wise internal features by building and implictly imposing contraints on the weight matrices of the neural layers, which include weight normalization (WN) [12], centered-weight normalization (CWN) [13], and spectral normalization (SN) [14]. It is reported in [13, 15] that WN (or CWN) combined with batch normalization often provides better performance. SN is shown to be effective when training generative adversarial networks (GANs) [14].

Besides weight-driven normalization, different weight regularisation techniques have also been proposed in the literature. The basic idea is to add specific penalty functions of the weight matrices to the original objective function when training the DNN model to influence the behaviours of the weight matrices. The weight decay is one popular technique, which poses a quadratic weight penalty function. In [16], orthonormal regularisation is proposed for pushing the vectors in each weight matrix to be mutually orthonormal with their norms being pushed close to one. We will briefly review orthonormal regularisation in Subsection 2.1 later on to motivate our new regularisation technique.

In this paper, we develop a new weight regularisation method, termed as projected weight regularization (PWR), to improve the generalization capacity of DNNs. Suppose W is a weight matrix extracted from a neural layer. PWR attempts to implicitly shape the eigenvalues of the matrix product WW^T such that it has comparable or roughly uniform eigenvalues. By doing so, it prevents the weight

matrix from having low rank and from overlearning the training data, which leads to better generalization of the resulting DNN model. To start with, the W matrix is projected onto a number of vectors along different randomly generated directions. A penalty function of the projected vectors is then built to regularize their behaviours for shaping the eigenvalues of WW^T. Differently from orthonormal regularisation, PWR provides more freedom to regularize the W matrix in that it can construct different forms of the penalty functions if necessary.

In this work, a general framework for PWR will be defined first. We will show that the functional form of orthonormal regularisation can be taken as a special case of PWR. We then construct a specific functional form for CWN in training a CWN-based DNN model. Our motivation for investigating CWN is that the technique is effective to produce DNN models with promising generalization capacity [13]. Experimental results on training VGG16 for CIFAR10 show that PWR combined with CWN performs better than orthonormal regularisation combined with CWN.

2. PROJECTED WEIGHT REGULARIZATION

2.1. Preliminary

Suppose we have a sequence of L pairs of training samples $\{(x_i, y_i)|i = 1, \ldots, L\}$, where x_i and y_i represent the input and output, respectively. For simplicity, we consider training a fully connected neural network with the weights $\{W_i|i = 1, \ldots, N\}$ of N layers.[1] With the considered DNN model, each sample x_i undergoes a sequence of matrix multiplications and nonlinear functional operations to yield prediction of y_i. The objective is to find the proper weights $\{W_i\}$ so that the network maps the input $\{x_i\}$ to the output $\{y_i\}$ accurately. Mathematically, the training procedure intends to solve a highly nonlinear and nonconvex optimization problem of the form

$$\min_{\{W_i\}} \sum_{i=1}^{L} \mathrm{dis}(f_N(\ldots f_2(W_2 f_1(W_1 x_i))), y_i) + \lambda \sum_{i}^{N} p(W_i), \quad (1)$$

where $\mathrm{dis}(\cdot, \cdot)$ denotes the distance measure between the network prediction for the sample x_i and its ground truth y_i, f_i denotes nonlinear activation function at layer i, and λ is a scalar coefficient. The 2nd term in (1) represents a regularization penalty function of the weight matrices. For the well-known weight decay technique [17], $p(W_i)$ becomes a quadratic penalty function of W_i, which prevents the weight matrices from growing out of control.

Next we briefly review the orthonormal regularisation proposed in [16], of which the penalty function for a weight matrix W of a neural layer takes the form of

$$p_{orth}(W) = \frac{1}{m^2} \|WW^T - I\|_2^2, \quad (2)$$

where m denotes the number of row vectors of W and I represents the identity matrix. Basically, the penalty function p_{orth} intends to make all the row vectors of W matrix to be orthogonal to each other while having unit norm when a large scalar coefficient λ is selected. For the ideal case that $WW^T = I$, it is immediate that all the eigenvalues of WW^T becomes 1, leading to flat eigenvalue distributions.

[1] One can extend the work to include the bias vectors and CNN neural layers.

We note that orthonormal regularisation is just one way to make an impact on the behaviour of the weight matrix W. In next subsection, we will introduce the projected weight regularisation (PWR). Conceptually speaking, orthonormal regularisation can be taken as a special case of PWR, which we will explain in the following.

2.2. Definition of PWR

Without loss of generality, we consider the input-output relationship under a weight matrix W at a particular neural layer. We drop the layer index for simplicity. The output z can be expressed as

$$z = Wv, \quad (3)$$

where W is of size $m \times n$, and $v \in \mathbb{R}^n$ represents the output from layer below right after a nonlinear activation function. Our objective is to generalize orthonormal regularisation to have more freedom in shaping the eigenvalues of the matrix product WW^T.

Intuitively speaking, suppose there exist extremely low-rank weight matrices $\{W_i\}$ for (1) that fits the training data well. It suggests that the DNN model does not fully make use of its parameter space and is highly redundant in the number of parameters. In other words, information of the training data is concentrated only on a small manifold of the parameter space. It is likely that information of unseen data may fall outside of the small manifold, leading to unsatisfactory performance. The above analysis suggests that it is preferable to search for a DNN model of which the weight matrices have high rank and where most of eigenvalue magnitudes are comparable w.r.t. the largest eigenvalue magnitude. Therefore, we propose PWR to serve the above purpose.

Next we introduce the basic framework of PWR. The first step is to generate k random m-dimensional vectors $\{q_j|j = 1, \ldots, k\}$ from a certain probability distribution. The k vectors are then normalized by their respective norms to produce k unified directions, denoted as

$$\hat{q}_j = q_j / \|q_j\| \quad j = 1, \ldots, k. \quad (4)$$

After that, we compute and investigate the projection of W matrix over the obtained k random directions. In principle, as k increases, the projected vectors $\{\hat{q}_i^T W|j = 1, \ldots, k\}$ would carry sufficient information of the W matrix to be able to shape the eigenvalues of WW^T.

We define the penalty function for PWR to be of form

$$p_{pwr}(W|\{q_j\}) = p_{pwr}(\hat{q}_1^T W, \hat{q}_2^T W, \ldots, \hat{q}_k^T W). \quad (5)$$

(5) is quite general and provides high degrees of freedom for functional construction if necessary. In fact, it can be shown that (5) includes (2) of orthonormal regularisation as a special case by choosing both the k directions $\{\hat{q}_j^T\}$ and the penalty p_{pwr} carefully. One can simply take $\{\hat{q}_j^T = e_j^T\}$ where the vector e_j has entry 1 at the jth position and zeros at other positions and then work out the functional form of p_{pwr} to produce (2).

From a perspective of linear system, W and $\hat{q}_j^T$ can be treated as a linear filter and the normalised input to the filter, respectively. Conceptually speaking, orthonormal regularisation only penalises the output response of a limited normalised input space. It is not clear how the linear system responds for the whole normalised input space when applying orthonormal regularisation when the parameter λ in (1) is bounded from above. On the contrary, PWR automatically considers the output response of the whole normalised input space by generating random directions per training iteration, which allows

PWR to gain more controllability of the behaviour of the linear filter W than orthonormal regularisation.

Next, as an example, we design a particular functional form for $p_{pwr}(\cdot)$, which is given by

$$p_{pwr}^{var}(W|\{q_j\}) = \frac{1}{k}\sum_{j=1}^{k}\left(\|\hat{q}_j^T W\|^2 - \frac{1}{k}\sum_r \|\hat{q}_r^T W\|^2\right)^2. \quad (6)$$

Equ. (6) penalizes the variance of the squared norms of the projected vectors. Small variance implies that the output responses of the linear filter W over the whole normalised input space have roughly the same norms. In other words, the eigenvalues of the matrix product WW^T would have roughly the same magnitudes, which is in line with our research goal.

Remark 1. *We note that in certain aspect, PWR is similar to sliced Wasserstein distance (SWD) [18]. Basically, SWD approximates the distance of two high-dimensional probability distributions by first projecting the two distributions to one-dimensional space along a number of directions and then measures the distance of the projected distributions accordingly. PWR is designed to first project the weight matrix along a number of directions and then operate on the projected vectors to influence the behavior of the weight matrix.*

3. DESIGN OF PWR FOR CWN

In this section, we first briefly introduce CWN. After that, we propose a specific PWR form for CWN.

3.1. Centered-Weight Normalization(CWN)

CWN is proposed in [13] as a slight modification of WN. It conducts operation on each row vector of W, which is associated with weight incoming connections of the neurons from a layer below. We use w_l^T to denote the lth row vector of W in (3). CWN computes a function of w_l^T as

$$\hat{w}_l^T = \frac{w_l^T - \frac{1}{n}w_l^T\mathbf{1}_n}{\|w_l^T - \frac{1}{n}w_l^T\mathbf{1}_n\|}, \quad (7)$$

where $\mathbf{1}_n$ denotes an n-dimensional vector of all ones. As the name *centered-weight normalization* suggests, the new row vector $\hat{w}_l^T$ is obtained by first subtracting the mean value from the row vector w_l^T, which is then normalized to have a unit norm. To compensate for the effect of normalization, an additional parameter ν is introduced in [13] to perform scale transformation for $\hat{w}_l^T$, which can be expressed as

$$\hat{w}_{s,l}^T = \nu\hat{w}_l^T. \quad (8)$$

When performing forward propagation over the neural network, w_l^T is replaced by $\hat{w}_{s,l}^T$ in (1) so that the weight vector $\hat{w}_l^T$ always has zero mean and unit norm. In this case, the weight decay technique is not necessary anymore as $\hat{w}_l^T$ would never grow out of control.

The authors' motivation for subtraction mean value from each weight vector is that the vector is often initialized using a zero-mean Gaussian distribution before training the DNN model. Thus, it is natural to keep the zero mean property of the weight vector during the training process. It is found empirically in [13] that the mean-subtraction operation indeed helps with accelerating the training speed and making the DNN model more general.

Table 1. Procedure for building p_{pwr}^1 for CWN

Input: a weigh matrix W, the number k of projections
1: Compute $\hat{W}$ by applying (7) on row vectors of W.
2. Randomly generate k vectors $\{q_j|j = 1,\ldots,k\}$
3. Normalize the k vectors to obtain $\{\hat{q}_j = q_j/\|q_j\|\}$
4. Compute $p_{pwr}^1(\hat{W}|\{q_j\})$ by following (9)

3.2. PWR form

We use $\hat{W}$ to denote the weight matrix obtained by stacking all the newly obtained row vectors $\{\hat{w}_l^T\}$ from (7) by CWN. We would like to design p_{pwr} for $\hat{W}$ to further shape the eigenvalues of $\hat{W}\hat{W}^T$. Our motivation behind it is that CWN itself does not pose any condition on the eigenvalues of W or $\hat{W}\hat{W}^T$. It may happen that the matrix $\hat{W}$ after training has extremely low rank, which is undesirable from the analysis in Subsection 2.2. Therefore, an additional regularization function is needed to shape the eigenvalues of $\hat{W}\hat{W}^T$.

Given $\hat{W}$ computed from (7) by CWN, we define p_{pwr} to be

$$p_{pwr}^1(\hat{W}|\{q_j\}) = \frac{1}{k}\sum_{j=1}^{k}(\|\hat{q}_j^T\hat{W}\|^2 - 1)^2, \quad (9)$$

where the superscript 1 in p_{pwr}^1 indicates that the norm of each projected vector is pushed towards 1 in the function. Correspondingly, the eigenvalues of $\hat{W}\hat{W}^T$ would also be pushed to 1. The procedure for computing p_{pwr}^1 in (9) is summarized in Table (1). We note that one can also design other functions of $\|\hat{q}_j^T\hat{W}\|^2 - 1$ instead of the quadratic function for better performance, which we leave for future investigation.

The main difference between $p_{pwr}^{var}(\cdot)$ in (6) and $p_{pwr}^1(\cdot)$ is that the latter function uses 1 to replace the mean value $\frac{1}{k}\sum_r \|\hat{q}_r^T\hat{W}\|^2$ of the squared norms of the projected vectors in $p_{pwr}^{var}(\cdot)$. Our motivation for the design of $p_{pwr}^1(\cdot)$ is that as $\hat{W}$ is computed from (7), each of its row vectors $\{\hat{w}_l^T\}$ has a unit norm already. Therefore, it is reasonable to also push the norm of each projected vector $\hat{q}_j^T\hat{W}$ to 1 for consistency.

4. EXPERIMENTS

4.1. Experimental setup

In the experiments, we consider training the VGG16 network on CIFAR10. Five configurations have been tested, which are VGG16 with BN, BN+CWN, BN+CWN+orthonormal, BN+CWN+PWR$_1$, and BN+CWN+PWR$_{\text{var}}$, where *orthonormal* refers to orthonormal regularisation defined by $p_{orth}(\hat{W})$. PWR$_1$ and PWR$_{\text{var}}$ refer to the two penalty functions $p_{pwr}^1(\hat{W})$ and $p_{pwr}^{\text{var}}(\hat{W})$ of PWR, respectively. Our primary interest is the validation performance gain due to the introduction of PWR.

The implementation of the training and testing procedure was conducted on the pytorch platform. SGD with momentum was employed for training each network configuration, where the momentum was set to be 0.9. The maximum number of epochs was 160. The initial learning rate was 0.1, and scheduled to be divided by 2 at 60 and 120 epochs sequentially. The scalar coefficient λ in (1) for orthonormal regularization and PWR was set to be 5.0. For PWR, the number k of projections was $k = 128$. To alleviate the effect of the randomness in the training process, five experimental repetitions

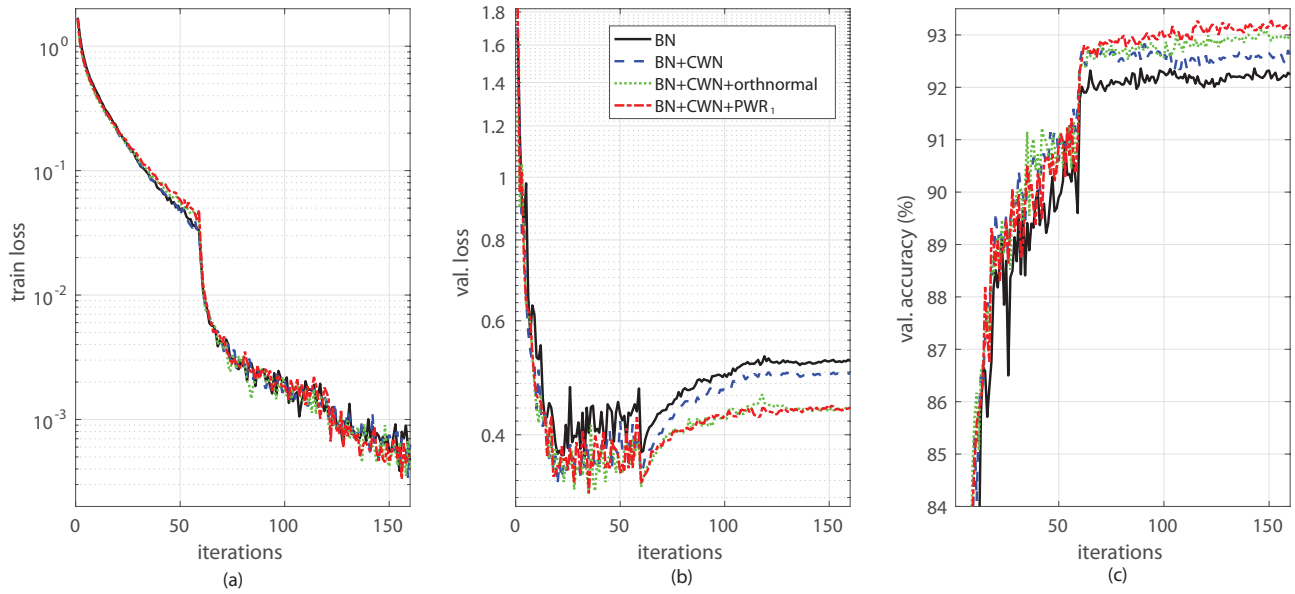

Fig. 1. Performance comparison of four configurations for training a VGG16 on CIFAR10, where *orthonormal* refers to orthonormal regulirazation. The curve for each configuration is selected from five experimental repetitions, which gives the highest validation accuracy.

Table 2. Validation accuracy (in percentage) of five experimental repetitions per configuration for training VGG16 over CIFAR10. PWR_1 and PWR_{var} refer to the penalty function $p^1_{pwr}(\hat{W}|q_j)$ and $p^{var}_{pwr}(\hat{W}|q_j)$, respectively.

BN	BN+CWN	BN+CWN +orthonormal
92.22±0.09	92.68±0.08	92.87±0.12
BN+CWN+PWR_1	$\text{BN+CWN+PWR}_{\text{var}}$	
93.04±0.13	93.0±0.09	

were conducted for each configuration.

4.2. Analysis of experimental results

Table 2 shows the validation accuracy of five experimental repetitions per configuration for the five configurations. It is clear that introduction of PWR indeed helps CWN to improve generalisation of the obtained VGG16 model. Furthermore, PWR performs slightly better than orthonormal regularisation. This might be due to the fact that PWR exploits full normalized input space of each weight matrix by generating random projection directions while orthonormal regularisation only considers a limited normalized input space. Our experiments also confirm that CWN helps BN to obtain better validation performance. In brief, BN+CWN+PWR yields the best validation performance among the five tested configurations.

As is illustrated in Table 2, the validation accuracy of PWR_{var} has a smaller confidence interval than PWR_1. This suggests that the performance of PWR_{var} tends to be less sensitive to the randomness introduced by DNN initialisation and the training procedure. Therefore, PWR_{var} might be a good candidate to impliment PWR in practice.

Fig. 1 displays the convergence results of four configurations. The results for PWR_{var} are omitted due to the fact that PWR_{var} and PWR_1 have similar performance. Each curve in the plot is selected from five experimental repetitions which gives the highest validation accuracy. It is seen that the convergence behaviours of the training loss are quite similar for the four configurations. When it comes validation accuracy, the configuration with PWR produces noticeably better convergence results than others. The above property suggests that PWR is able to successfully shape the eigenvalues of each matrix product $\hat{W}\hat{W}^T$ in VGG16 as expected. It is also clear from the figure that VGG16 with BN alone performs the worst. This might be due to the fact BN operates on the internal features of training samples and lacks controllability of weight matrices in comparison to CWN, orthonormal regularisation, and PWR.

5. CONCLUSIONS

In this paper, we have proposed projected weight regularisation (PWR), a new weight regularisation technique. PWR can be viewed as a generalisation of orthonormal regularisation as both techniques intend to shape the eigenvalues of each weight matrix in a DNN model such that most of the eigenvalues have comparable magnitudes. Instead of imposing constraints directly on the row vectors of a weight matrix as in orthonormal regularisation, PWR projects the weight matrix to a number of vectors along different directions and then builds a penalty function of the projected vectors. Conceptually speaking, PWR treats the weight matrix as a linear filter. It then attempts to regulate the output response of the filter by feeding randomly generated normalised inputs. Therefore, PWR is able to cover the whole normalised inputs while orthonormal regularisation only considers a limited normalized input space. Experimental results show that PWR performs slightly better than orthonormal regularisation which might be due to the difference of normalized input space.

6. REFERENCES

[1] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," in *Proceedings of the 27th*

International Conference on Machine Learning,, 2010.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on Imagenet Classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[4] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," arXiv:1505.04597 [cs.CV], 2015.

[5] D. P. Kingma and J. L. Ba, "Adam: A Method for Stochastic Optimization," arXiv preprint arXiv:1412.6980v9, 2017.

[6] J. Duchi, E. Hazan, and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[7] T. Tieleman and G. Hinton, "Lecture 6.5-RMSProp: Divide the Gradient by A Running Average of Its Recent Magnitude," COURSERA: Neural networks for machine learning, 2012.

[8] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *volume 37 of JMLR Proceedings*, pp. 448–456, 2015.

[9] G. E. Hinton J. L. Ba, J. R. Kiros, "Layer Normalization," arXiv:1607.06450 [stat.ML], 2016.

[10] L. Huang, Y. Zhou, F. Zhu, L. Liu, and L. Shao, "Iterative Normalization: Beyond Standardization towards Efficient Whitening," in *Conference on Computer Vision and Pattern Recognition(CVPR)*, 2019, pp. 4874–4883.

[11] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-Normalizing Neural Networks," in *31st Conference on Nueral Information Processing Systems (NIPS)*, 2017.

[12] D. P. Kingma T. Salimans, "Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks," arXiv:1602.07868 [cs.LG], 2016.

[13] L. Huang, X. Liu, Y. Liu, B. Lang, and D. Tao, "Centered Weight Normalization in Accelerating Training of Deep Neural Networks," in *International Conference on Computer Vision*, 2017, pp. 2803–2811.

[14] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral Normalization for Generative Adversarial Networks," in *ICLR*, 2018.

[15] E. Hoffer, R. Banner, I. Golan, and D. Soudry, "Norm Matters: Efficient and Accurate Normalization Schemes in Deep Networks," arXiv:1803.01814 [stat.ML], 2018.

[16] A. Brock, T. Lim, J. M. Ritchie, and N. Westona, "Neural Photo Editing with Introspective Adversarial Networks," arXiv preprint arXiv:1609.07093, 2016.

[17] A. Krogh and J. A. Hertz, "A Simple Weight Decay Can Improve Generalization," in *Advances in Neural Information Processing*, 1992, pp. 950–957.

[18] S. Kolouri, P. E. Pope, C. E. Martin, and G. K. Rohde, "Sliced-Wasserstein Autoencoder: An Embarrassingly Simple Generative Model," arXiv:1804.01947 [cs.LG], 2018.