An Orthogonal Classification Layer with Kasami Sequences for Discriminative Feature Learning in Neural Networks

Mohamed Saadeldin
School of Computer Science
University College Dublin
Belfield D8, Dublin, Ireland
Email: mohamed.saadeldin@ucd.ie

Brian Macnamee
School of Computer Science
University College Dublin
Belfield D8, Dublin, Ireland
Email: brian.macnamee@ucd.ie

***Abstract*—This paper proposes a novel *Orthogonal Classification Layer* (OCL) utilizing *Kasami sequences* for neural networks trained for classification problems. OCL consists of a fully connected layer with fixed orthogonal weights and zero biases for all of its output neurons. Attaching the OCL to the end of any neural network encourages the network to generate a unique latent representation (*orthogonal code*) at its last hidden layer for each data class. This is achieved by associating and fixing the weights for each of the output neurons to a unique code from a set of orthogonal sequences, such as Kasami. When networks use OCL the latent representations they learn for each data class at the end of the network converge to values equivalent to the fixed weights of the OCL. Auto-correlation and cross-correlation properties of orthogonal codes maximize the output of the correct class and increase its separation from the outputs of all other classes. Therefore, networks trained with OCL benefit from a wider classification decision margin than networks without OCL. Moreover, the feature sets extracted by the network for different data classes are well separated and more amenable to human interpretation. The implementation of OCL is simple and OCL can easily be integrated into any neural network architecture without a need for network architecture modifications or changes to the training scheme (for example optimization, normalization, or regularization) adopted in the original network architecture. The computational and memory cost required for the OCL is lower than conventional classification layers since the weights are fixed and no gradient is required. Though simple and practical the proposed OCL enhances learning of discriminative latent representations, generates explainable features, and provides high classification accuracy. We demonstrate this through a set of evaluation experiments comparing the performance of equivalent networks with and without OCL.**

I. Introduction

Advances and developments in neural networks and the tremendous solutions and applications they promise, has attracted a lot of research and industry interest. In particular, efforts to improve optimization, regularization, and explainability for deep neural networks has received a lot of attention to advance the performance and improve the accuracy of networks. Since the introduction of Back Propagation (BP) [1], [2], and its wide adoption by the community, a number of variations and modifications to the original algorithm have been proposed. Similarly, regularization techniques have been applied to reduce the tendency of neural networks to over-fit training data, and to improve the ability of networks to generalize to unseen test data. Early stopping[3], weight decay [4], dropout [5], and data augmentation [6] are among the most common regularization techniques used in modern neural networks.

Recently, attention has been given to *orthogonality* in neural networks. Several studies suggest that applying orthogonality constraints to Multi-Layer Perceptrons (MLPs), Convolutional Neural Networks (CNNs), and Recurrent Neural Networks (RNNs) helps to optimize learning and improve generalization, while also avoiding the gradient vanishing or exploding phenomena. In general the aim is to establish and maintain orthogonal weights within a network during training. Approaches proposed in the literature include orthogonal weight initialization [7], orthogonal weight normalization [8], [9], [10], [11], and orthogonal regularization [12], [13], [14], [15], [16]. In addition to preserving gradient norms, orthogonality helps classification networks to reduce correlation between feature representations learned and enhance the learning of discriminative features.

The ability of a network to generate discriminative features is of most importance for classification problems, and orthogonality methods have been developed explicitly for this objective. The majority of these methods enhance learning of discriminative features in neural networks by modifying the loss function [17], [18], [19], [20], [21] and/or altering the output layer [22], [23], [24] to increase separation between output representations of different classes.

This paper proposes the *Orthogonal Classification Layer* (OCL) as an effective and simple method to enhance the learning of discriminative features in neural networks trained for classification problems. Including an OCL at the output layer of a network increases the separation and uniqueness of latent representations of different classes learned by the network, resulting in higher classification accuracy and more explainable features. The OCL described in this paper utilizes the orthogonality properties of *Kasami sequences* and uses them as fixed weights in all of the output neurons of the fully connected classification layer. Hence a unique orthogonal code from the Kasami set is assigned as a fixed weight vector to

each output neuron of the OCL, and each data class is linked to one of these codes. Benefiting from the auto-correlation and cross-correlation properties of orthogonal sequences, the OCL encourages the neural network to learn and generate latent representations at its last hidden layer equivalent to the orthogonal codes associated with each data class. This maximizes the activation of the correct class output neuron, and minimizes activations of all other output neurons related to incorrect data classes. The implementation of OCL and its integration with neural networks is very simple, since OCL is basically a dummy fully connected layer with pre-calculated fixed orthogonal weights.

The remainder of this paper proceeds as follows. Section II describes related work including existing approaches that introduce orthogonality into neural network training. In Section III, details of the OCL implementation and how it works are explained. The experimental setup and evaluation results of a set of experiments that validate the performance of OCL are presented and discussed in Section IV. Finally, conclusions and directions for future work are outlined in Section V.

II. Related Work

This section describes existing efforts to use orthogonality to improve neural network work training. we group these efforts into those that focus on *weight initialisation*, those that develop *regularisation* approaches, those that use *normalisation*, and those that focus on finding *discriminative features*.

A. Orthogonal weight initialization.

Initializing neural networks with orthogonal weights has been studied and discussed in several papers [7], [25], [26], [27]. Le et al [25] proposed the use of the identity matrix and its scaled version to initialize the weight matrix of an RNN with rectified linear units. They claimed that this orthogonal initialization of weights helped the network to train more easily, provided a better model of long-range dependencies, and dealt with the vanishing and exploding gradient problems. In [7] Sokol & Park tried to understand how orthogonal initialization benefits neural networks, and the relation between more isometric networks and faster training. They utilized the Fisher information matrix to unveil the connection between gradient smoothness and the spectral radius of the input-output Jacobian. Mishkin & Matas [27] proposed layer-sequential unit-variance (LSUV) initialization and evaluated it on GoogLeNet, CaffeNet, FitNets and Residual nets, achieving near state-of-the-art results. LSUV initializes weights with orthonormal matrices, then normalizes the variance of each layer output to 1.

B. Orthogonal regularization.

Encouraged by the benefits that orthogonal initialization offered to neural networks a lot of work in literature tried to enforce orthogonality throughout training. Yoshida & Miyato [10] proposed spectral norm regularization, where the network is penalized for having a high spectral norm of its weight matrices. Their experimental results confirm that training with this constraint on the weight spectral norm leads to better generalization and less sensitivity to perturbations of test data. Harandi & Fernando [12] proposed an extension to the back propagation of error algorithm that preserves structural forms on network weights. Utilizing this extension they introduced and evaluated Stiefel layers [28] to be used in designing orthogonal filters for neural networks. When Stiefel layers were applied to different network architectures, results showed improvement in the classification accuracy. The work in [13] studied effective and easy ways for orthogonal regularization, and proposed Mutual Coherence Regularization and Spectral Restricted Isometry Property Regularization. They experimentally evaluated these method and compared its performance to both Soft Orthogonality Regularization and Double Soft Orthogonality Regularization. Wang et al [14] proposed orthogonal convolutional based on doubly block-Toeplitz (DBT). Their proposed Orthogonal Convolutional Neural Network (OCNN), when trained and regularized with the orthogonality loss, produces more uniform spectrum and more diverse features.

C. Orthogonal normalization

Since optimization of neural networks with strict orthogonality constraints is complex and can be computationally expensive, many proposals tried to go around enforcing orthogonality by applying matrix transformation and normalization. The work in [29] established parameterization optimization utilizing unitary group matrices from the fields of Lie group theory [30] and Riemannian geometry [31]. Singular Value Bounding (SVB) was presented in [8] where all singular values of the weight matrix are bounded around the value of 1. Inspired by unitary Recurrent Neural Networks (uRNNs), Helfrich et al [32] proposed the scaled Cayley orthogonal Recurrent Neural Network (scoRNN) to maintain orthogonality in recurrent weight matrices by using the Cayley transform for parameterization. A norm-preserving orthogonal permutation linear unit activation function (oplu) was proposed in [33] where the derivative of oplu is an orthogonal operator allowing norm preservation over all layers that were initialized with orthogonal weight matrices.

D. Discriminative feature learning

There are also a number of approaches that use orthogonality to directly influence networks to learn more discriminative features. Entropy–Orthogonality Loss (EOL) was proposed in [17], where an explicit modification was introduced to the conventional entropy loss function to make the learned feature vectors from different classes orthogonal. In [23] an Orthogonal AutoEncoder (OAE), which encourages orthogonality among the learned embedding was proposed with the intention of enhancing clustering by increasing separation between different clusters. Virtual softmax [24] is a technique that encourages learned features to be more compact and separable by insertion of a virtual dynamic negative class into the softmax layer. Weiyang et al [18], [20] proposed a generalized Large-margin softmax (L-Softmax) and Angle Softmax (A-Softmax) loss which explicitly encourages learned features to have intra-class compactness and inter-class separability by allowing

adjustment of the angle margin between classes in the softmax layer. Similarly, the center loss introduced in [19] learns a center for deep features of each class and penalizes the network according to the distance between features and their corresponding class centers. A large-margin Gaussian Mixture (L-GM) loss [21] assumes learned features follow a Gaussian Mixture distribution. Accordingly high classification performance and better modeling of the feature distribution are achieved by introducing a classification margin and a likelihood regularization. Finally, the Orthogonal softmax Layer (OSL) [22] was introduced with the main objective of helping networks learn discriminative features by constructing a fixed orthogonal classification layer. Orthogonality in OSL was obtained as a result of removing some connections in the softmax layer. When OSL is used each neuron in the last hidden layer is assigned and connected to only one specific class output neuron.

The OCL proposed herein focuses on orthogonality of the learned latent representations at the end of the network and achieve this goal by exploiting correlation properties of Kasami orthogonal codes. Unlike other discriminative feature learning approaches previously introduced, that only apply modifications to Softmax layer and loss function, OCL establishes orthogonal weights beforehand in all neurons of the output layer and fix it throughout training. This allow OCL to force the network to produce specific and known latent representations that are known in-advance and orthogonal by design. Compared to OSL, the proposed OCL makes use of all neurons in the last hidden layer to calculate its final classification prediction for each data class. This makes OCL predictions more robust and allows the network to utilize the full capacity of the classification layer to generate richer and more diverse features sets.

III. The Orthogonal Classification Layer

This section describes the Orthogonal Classification Layer (OCL), which is the main contribution of this paper. First we describe the Kasami sequences that our approach is based on, before describing how orthogonal weights are used in OCL,and the implications this has on the loss function used to train the network.

A. Kasami sequences

Orthogonal spreading codes are widely used for code-division multiple access (CDMA) in wireless communication [34]. Due to their correlation properties orthogonal codes significantly reduce mutual interference between multiple communication channels. Among the most common short codes used for CDMA are Barker, Walsh, Zadoff–Chu, Hadamard, M-sequence, Gold, and Kasami [35]. In this work we use Kasami codes as fixed weights for the OCL. Kasami codes are binary sequences with values equal to 1 and -1, where each code is orthogonal to all other codes in the set. Kasami codes therefore have very good auto-correlation and cross-correlation properties.

Kasami codes can be generated using a periodic maximum length sequence, S_{max}, and a secondary sequence, S_{sec}, that is obtained by cyclic decimation sampling of S_{max}. A set of N unique Kasami codes $K_0, K_1, K_i, ..., K_{N-1}$ is obtained by Modulo 2 addition (xor) of S_{max} to N cyclically time shifted versions of S_{sec} [36]. The generated set of codes are orthogonal, meaning that each code has high auto-correlation value with itself $K_i * K_i$, and low cross-correlation value with all other codes in the set $K_i * K_j$ for all $i \neq j$ [37]. Table I shows an example of a family set of 4 orthogonal Kasami codes with 15 bits length.

TABLE I: Set of 4 orthogonal Kasami codes

Code A	1	1	1	1	-1	1	-1	1	1	-1	-1	1	-1	-1	-1
Code B	-1	-1	1	-1	1	1	1	-1	1	1	1	1	1	1	-1
Code C	1	-1	-1	1	1	-1	-1	-1	-1	-1	1	-1	-1	1	1
Code D	-1	1	-1	-1	-1	-1	1	1	-1	1	-1	-1	1	-1	1

B. Fixed orthogonal weights

The output layer or classification layer of a neural network for a classification problem should indicate the correct class that an input example belongs to with high probability. In a typical classification problem the network is structured to have a number of output activations equivalent to the number of data classes. The network is trained to maximize the output associated with the correct class of the input data while minimizing all other outputs. Conventional classification layers consist of a fully connected layer that takes as an input latent representations generated at the last hidden layer of the network and outputs the final classification results.

The operation performed by this fully connected layer can be viewed as correlation between the latent representation vector $H = f(x)$ associated with input data x, and each of the weight vectors, W_i, of every output neuron in the classification layer. Considering an input x_j that belongs to data class j, for which the network generates a latent representation vector $H_j = f(x_j)$, the network is trained with the objective of generating a final output vector Y where the values of $y_i = H_j * W_i$ is maximized for $i = j$ and minimized for all $i \neq j$. The proposed OCL benefits from correlation properties of orthogonal sequences to achieve this classification layer objective, and performs its task without the need to learn the weights W_i of this layer. This is basically due to the fact that Kasami codes used as fixed weights for OCL are orthogonal by design, meaning that correlation between any pair of these codes $c_i \& c_j$ is maximum for $i = j$ and minimal for all $i \neq j$.

The OCL uses a set of Kasami codes, $[K_0, K_1, ...K_{N-1}]$ as fixed weights, W_i, for the classification layer such that $W_i = K_i$ for $i = 0$ to $N-1$, where N is the number of output data classes. The objective of maximizing the output value y_i when $i = j$ while minimizing it for all $i \neq j$ considering $y_i = H_j * W_i$ can be well satisfied if $H_j \approx K_j$. Therefore during training the network associates each data class i with a unique Kasami code K_i and tries to output it as its latent representation. The implication of this is that the network is encouraged to learn and generate latent representations, H_i, at its last hidden layer with values equivalent to, or very close

to, the orthogonal code, K_i, for all values of i. Though no learning happens within the OCL itself, it makes the network learn and produce a unique orthogonal latent representation for each of the input data classes. These latent representations then propagate through the OCL benefiting from the correlation properties of orthogonal codes to provide the final network outputs, where the correct class output is maximized, and all other outputs of incorrect classes are minimized.

C. Loss function

Cross entropy is the main loss function used to train neural networks for classification problems, typically together with a softmax layer at the output of the network. The final output from softmax can be viewed as predicted probabilities, where each neuron output value indicates the likelihood of the input data point belonging to the class associated with that neuron. In addition to the use of the conventional cross entropy loss function, OCL allows construction of other loss functions based on the predicted latent representations at the last hidden layer rather than the final softmax prediction. The network converges to output correct classifications when successfully learns to generate latent representations very close or equal to the fixed orthogonal weights of the OCL, this can be utilized to assess and guide training process through an additional loss function. The root mean square error (rmse) between the predicted latent representations and the fixed weights is one option of a valid loss functions. More over this advantage of OCL can be even further used to divide the network into a group of independent sub networks (ensemble of networks) that can be trained in parallel and their outputs then concatenated to generate the final required latent representation for input to the OCL. This idea of utilizing OCL for parallel training of ensemble of networks seems interesting and we plan to address it in future work.

In our current implementation we combine cross entropy loss ($Loss_{ce}$) with the root mean square error ($Loss_{rmse}$) between the predicted latent representations and the fixed weight vector of the output neuron associated with the correct data class. The ($Loss_{rmse}$) means for any input data class, i, the loss function aims to minimize $(H_i - W_i)^2$, leading the network to converge towards correct classification output results as $H_i \approx W_i$.

A hyper parameter β was introduced that controls the contribution of the two loss functions towards the overall loss used to train the network, where β can hold any value between 0 and 1:

$$Loss_{final} = (1 - \beta) * Loss_{ce} + \beta * Loss_{rmse} \qquad (1)$$

The overall loss, ($Loss_{final}$), is used for back-propagation through the network during training. Setting β to 0 means using only cross entropy loss ($Loss_{ce}$), while setting β to 1 results on training with only root mean square error loss ($Loss_{rmse}$).

IV. Experimental Evaluation

To evaluate the effectiveness of the proposed OCL we evaluate the performance of a network trained with conventional classification layer and with OCL on a set of well known image classification problems. This section describes the models and datsets used in this experiment and the results of the experiment. We also explore the interpretability of the latent representations learned with OCL compared to those learned without it.

A. Datasets & Models

As they are well understood and widely use din the community, in our experiments we use four popular image classification datasets: MNIST [38], Fashion MNIST [39], CIFAR10 [40], and SVHN [41]. Each of these datasets was divided into an 80% training set and a 20% test set.

For these problems a relatively small CNN with 7 convolutional layers was constructed, using 3 VGG like blocks [42] and a single point-wise convolutional layer at the end. Each of the 3 blocks consists of two convolutional layers with kernel size of 3X3, padding of 1, and a stride of 1. Batch normalization is applied after each convolutional layer, followed by ReLU non-linearity. The spatial dimension is reduced by half at the end of each block using a 2X2 max pooling layer. The 3 VGG blocks are then preceded by a final layer of point-wise convolution where a kernel of size 1X1 is used to reduce the number of output feature maps. Finally a Leaky ReLU non-linearity is applied followed by global average pooling to produce the output latent representation vector. This latent representation vector is then passed to either the OCL or a conventional classification layer depending on which setup is being tested.

The adam optimizer [43] with learning rate equal to 0.001 and zero weight decay is used throughout training. A batch size of 200 examples was used and each network was allows to train for 200 epochs. These values were selected after a limited hyper-parameter tuning applied to the network described above when using a conventional classification layer at the end of the network. The same hyper-parameters were used when training the network with OCL without further tuning or adjustments.

We evaluate three different architectures—*Plain*, *Dropout*, and *OCL*—in which only the final output layer is different. In *Plain* a conventional classification layer consisting of a fully connected layer is used directly after the last hidden layer of the network. In *Dropout* a dropout layer with probability of 0.25 is placed just before the fully connected layer at the end of the network. In *OCL* the proposed orthogonal layer with fixed Kasami weights is used as described in section III.

B. Optimising β

Using the network architecture and training approach described above the performance of the network with OCL was evaluated for different values of the loss function parameter β. Table II summarizes the classification accuracy achieved by the network using the 4 datasets. The classification accuracy over the validation set was evaluated and reported when training the network with values of β varying between 0 (only $Loss_{ce}$) and 1 (only $Loss_{rmse}$). Results does not indicate significant difference in performance favouring any of the two loss functions, though since maximum classification accuracy for all the 4 evaluated data-sets were achieved when training the network with $\beta = 0.9$, This value was used when training networks with OCL.

TABLE II: Classification accuracy for different values of β

	0.0	0.1	0.3	0.5	0.7	0.9	1.0
MNIST	99.51	99.48	99.57	99.6	99.6	**99.63**	99.58
FashionMNIST	94.53	94.07	94.33	94.09	94.45	**94.83**	94.7
CIFAR10	89.86	88.44	88.31	88.42	88.36	**89.92**	88.69
SVHN	94.16	94.05	94.04	94.03	94.58	**94.75**	94.39

TABLE III: Summary of classification accuracy on test-set

		MNIST	MNISTFashion	CIFAR10	SVHN
Plain	mean	99.55	93.96	87.59	94.10
	std	±0.04	±0.10	±0.22	±0.24
Dropout	mean	99.49	93.56	87.96	95.18
	std	±0.08	±0.31	±0.22	±0.26
OCL	mean	**99.65**	**94.03**	**89.70**	**95.46**
	std	±0.03	±0.17	±0.22	±0.17

C. Experimental Results

The performance of the 3 network types—Plain, Dropout, and OCL—was evaluated over the four datasets used—MNIST, FashionMNIST, CIFAR10, and SVHN. For each of the 4 evaluated data-sets the trained model providing best accuracy on the validation set was saved, and later used to obtain classification results on the test set. Training and testing was repeated for 6 independent runs and the mean and standard deviation (std) of classification accuracy obtained on the test set of each data-set were calculated. Table III summarizes and compares obtained classification accuracy results (mean & std) for the three different network types. These results indicate that the network trained with OCL consistently outperforms the other network types, providing higher classification accuracy on the test set of all 4 evaluated data-sets.

Figure 1 shows classification accuracy for the training set and validation set over 200 training epochs using each of the three network types (Plain in red, Dropout in blue , and OCL in orange) on each of the four datsets.

Looking at the learning and validation curves in Figure 1 OCL has a slower learning curve compared to Plain, and Dropout. we believe this is related to the fact that network trained with OCL is challenged with a sophisticated and more restricted learning task since it has to generate specific latent representations that are equivalent to the fixed weights of the OCL layer. But this additional complexity and constraint enforced upon the network is contributing to the observed better generalization of the network leading to higher classification accuracy on the validation set with OCL.

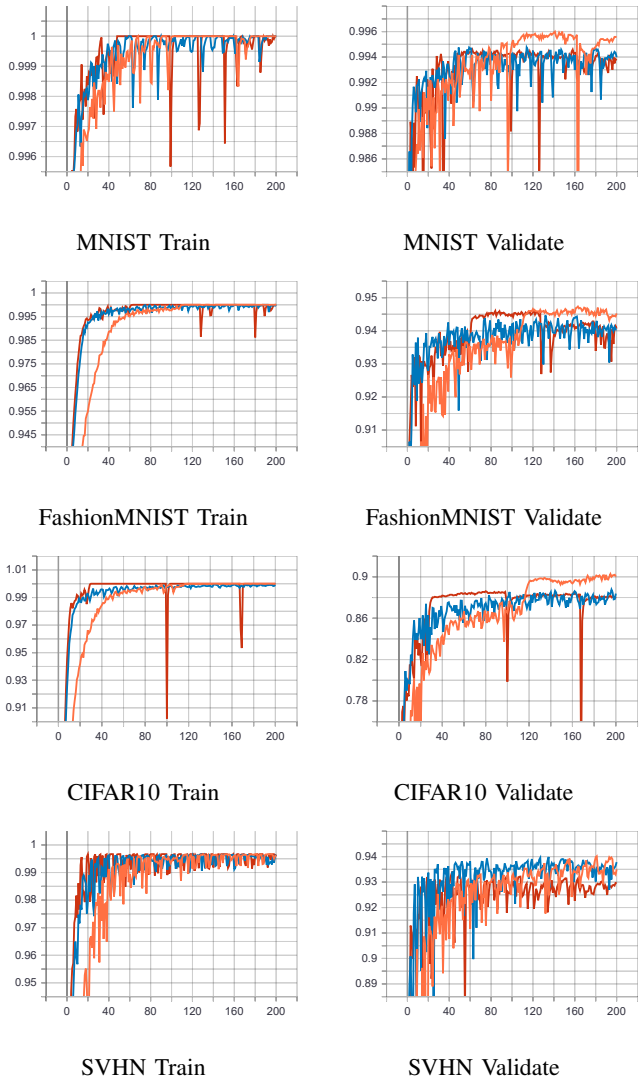

Fig. 1: Training and validation accuracy during training on the four datasets used for the three different network types (Red: Plain, Blue: Drop-out, Orange: OCL)

V. Conclusion

The proposed OCL showed to be a simple and practical output layer that outperforms conventional classification layers. OCL not only provides higher classification accuracy, but also enables the network to generate more distinct feature sets that are interpretable. Results obtained on the synthetic noisy sinusoidal data-set suggest that OCL helps the network to exploit and focus more on the unique features in the data. The binary nature of Kasami codes made it easier to express features as relevant or irrelevant to different data classes, and also guarantees each data class is identified and distinguished by a reasonable number of features and not rely on a single or few features. OCL unlocks the classification output layer and allows latent representations at the end of the network to be known and predictable before training, this opens the door to many future research directions to utilize these known latent representations during supervised training, instead of only using class labels.

References

[1] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in *Neural networks for perception*. Elsevier, 1992, pp. 65–93.

[2] Y. Chauvin and D. E. Rumelhart, *Backpropagation: theory, architectures, and applications*. Psychology press, 1995.

[3] C. D. Doan and S.-y. Liong, "Generalization for multilayer neural network bayesian regularization or early stopping," in *Proceedings of Asia Pacific Association of Hydrology and Water Resources 2nd Conference*, 2004, pp. 5–8.

[4] G. Zhang, C. Wang, B. Xu, and R. Grosse, "Three mechanisms of weight decay regularization," *arXiv preprint arXiv:1810.12281*, 2018.

[5] A. Labach, H. Salehinejad, and S. Valaee, "Survey of dropout methods for deep neural networks," *arXiv preprint arXiv:1904.13310*, 2019.

[6] L. Perez and J. Wang, "The effectiveness of data augmentation in image classification using deep learning," *arXiv preprint arXiv:1712.04621*, 2017.

[7] P. A. Sokol and I. M. Park, "Information geometry of orthogonal initializations and training," *arXiv preprint arXiv:1810.03785*, 2018.

[8] K. Jia, D. Tao, S. Gao, and X. Xu, "Improving training of deep neural networks via singular value bounding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4344–4352.

[9] K. Jia, S. Li, Y. Wen, T. Liu, and D. Tao, "Orthogonal deep neural networks," *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[10] Y. Yoshida and T. Miyato, "Spectral norm regularization for improving the generalizability of deep learning," *arXiv preprint arXiv:1705.10941*, 2017.

[11] 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," *arXiv preprint arXiv:1709.06079*, 2017.

[12] M. Harandi and B. Fernando, "Generalized backpropagation,\'{E} tude de cas: Orthogonality," *arXiv preprint arXiv:1611.05927*, 2016.

[13] 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*, 2018, pp. 4261–4271.

[14] J. Wang, Y. Chen, R. Chakraborty, and S. X. Yu, "Orthogonal convolutional neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 505–11 515.

[15] D. Xie, J. Xiong, and S. Pu, "All you need is beyond a good init: Exploring better solution for training extremely deep convolutional neural networks with orthonormality and modulation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6176–6185.

[16] A. V. Gayer and A. V. Sheshkus, "Convolutional neural network weights regularization via orthogonalization," in *Twelfth International Conference on Machine Vision (ICMV 2019)*, vol. 11433. International Society for Optics and Photonics, 2020, p. 1143326.

[17] W. Shi, Y. Gong, D. Cheng, X. Tao, and N. Zheng, "Entropy and orthogonality based deep discriminative feature learning for object recognition," *Pattern Recognition*, vol. 81, pp. 71–80, 2018.

[18] W. Liu, Y. Wen, Z. Yu, and M. Yang, "Large-margin softmax loss for convolutional neural networks." in *ICML*, vol. 2, no. 3, 2016, p. 7.

[19] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," in *European conference on computer vision*. Springer, 2016, pp. 499–515.

[20] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "Sphereface: Deep hypersphere embedding for face recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 212–220.

[21] W. Wan, Y. Zhong, T. Li, and J. Chen, "Rethinking feature distribution for loss functions in image classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9117–9126.

[22] X. Li, D. Chang, Z. Ma, Z.-H. Tan, J.-H. Xue, J. Cao, J. Yu, and J. Guo, "Oslnet: Deep small-sample classification with an orthogonal softmax layer," *IEEE Transactions on Image Processing*, 2020.

[23] W. Wang, D. Yang, F. Chen, Y. Pang, S. Huang, and Y. Ge, "Clustering with orthogonal autoencoder," *IEEE Access*, vol. 7, pp. 62 421–62 432, 2019.

[24] B. Chen, W. Deng, and H. Shen, "Virtual class enhanced discriminative embedding learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 1942–1952.

[25] Q. V. Le, N. Jaitly, and G. E. Hinton, "A simple way to initialize recurrent networks of rectified linear units," *arXiv preprint arXiv:1504.00941*, 2015.

[26] A. M. Saxe, J. L. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," *arXiv preprint arXiv:1312.6120*, 2013.

[27] D. Mishkin and J. Matas, "All you need is a good init," *arXiv preprint arXiv:1511.06422*, 2015.

[28] I. M. James, *The topology of Stiefel manifolds*. Cambridge University Press, 1976, vol. 24.

[29] M. Arjovsky, A. Shah, and Y. Bengio, "Unitary evolution recurrent neural networks," in *International Conference on Machine Learning*, 2016, pp. 1120–1128.

[30] A. Iserles, H. Z. Munthe-Kaas, S. P. Nørsett, and A. Zanna, "Lie-group methods," *Acta numerica*, vol. 9, pp. 215–365, 2000.

[31] P. Petersen, S. Axler, and K. Ribet, *Riemannian geometry*. Springer, 2006, vol. 171.

[32] K. Helfrich, D. Willmott, and Q. Ye, "Orthogonal recurrent neural networks with scaled cayley transform," in *International Conference on Machine Learning*, 2018, pp. 1969–1978.

[33] A. Chernodub and D. Nowicki, "Norm-preserving orthogonal permutation linear unit activation functions (oplu)," *arXiv preprint arXiv:1604.02313*, 2016.

[34] K. S. Zigangirov, *Theory of code division multiple access communication*. John Wiley & Sons, 2004, vol. 6.

[35] J. M. Velazquez-Gutierrez and C. Vargas-Rosales, "Sequence sets in wireless communication systems: A survey," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 2, pp. 1225–1248, 2016.

[36] X. Wang, Y. Wu, and B. Caron, "Transmitter identification using embedded pseudo random sequences," *IEEE Transactions on Broadcasting*, vol. 50, no. 3, pp. 244–252, 2004.

[37] J. Lahtonen, "On the odd and the aperiodic correlation properties of the kasami sequences," *IEEE Transactions on information theory*, vol. 41, no. 5, pp. 1506–1508, 1995.

[38] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[39] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[40] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[41] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," 2011.

[42] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[43] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.