An ETF view of Dropout regularization - supplementary material

Dor Bank
dorbank@gmail.com
Raja Giryes
raja@tauex.tau.ac.il

School of Electrical Engineering
The Iby and Aladar Fleischman Faculty of Engineering
Tel Aviv University
Tel Aviv, Israel

A Dropout regularization

Dropout regularization, introduced by Srivastava et al. [19], is one of the most popular regularization strategies. When applied on a given layer, it randomly drops hidden nodes along with their connections [8]. During training, in each batch, neurons are multiplied by a Bernoulli(p) variable, which causes them to nullify with a probability $q = 1 - p$. Each weight connected to a nullified neuron does not influence the output, thus, it is not updated by backpropagation at the given training step. Clearly, the remaining weights are trained regularly. At test time, all outputs are multiplied by p to sustain the overall weight norm.

Besides its great success in improving the generalization of neural networks, Dropout is also beneficial to avoid saddle points during training due to it stochasticity. Moreover, it incurs only a small additional complexity since it is linear in the features dimension.

Jindal et al. have used Dropout to deal with noisy labels, by reducing the certainty of a trained model and making it more robust. After the usual softmax layer, which is located at the end of the network, they have added a fully connected layer with Dropout, followed by another softmax layer. This led to a smoother label distribution for each sample, which is less affected by the noisy labels [9].

Note that each Dropout operation leads to a different approximation of the output of the network and therefore to a different optimization step. Thus, it is suggested that LSTM and GRU cells should have the same units dropped at each time step so the prediction would be consistent with respect to time [5].

Implicit bias. Mianjy et al. [13] study the implicit bias of Dropout [13]. It focuses on the case of a shallow network with a single hidden layer. Denote its matrices as $A \in \mathbb{R}^{m_1 \times n}$ and $B \in \mathbb{R}^{m_2 \times n}$. By applying Dropout with probability p and using the squared loss, their optimization objective reads as:

$$f(A,B) = \mathbb{E}_{b_i \sim Ber(p), x \sim \mathcal{D}} \left[\left\| y - \frac{1}{p} B \operatorname{diag}(b) A^T x \right\|^2 \right], \tag{1}$$

where $\mathcal{D}$ is the distribution of x. Notice that when $p = 1$, we simply get the case without dropout denoted as:

$$\ell(A,B) = \mathbb{E}_{x \sim \mathcal{D}} \left[\left\| y - BA^T x \right\|^2 \right]. \tag{2}$$

First they show that the Dropout objective in (1) can be rewritten as:

$$f(A,B) = \ell(A,B) + \lambda \sum_{i=1}^{n} \|a_i\| \|b_i\|, \tag{3}$$

where $\lambda = \frac{1-p}{p}$, and a_i and b_i represent the i^{th} columns of A and B respectively. They note that this is equivalent to the square of the convex Path-Regularization [14], which is the square-root summation over all paths in the network, where in each path the squared weights product is calculated. In addition, they argue that the additional term of (3) is an explicit instantiation of the implicit bias of dropout. They then define the term of jointly equalized matrices, where matrices A and B are considered jointly equalized iff $\forall i, \|a_i\| \|b_i\| = \|a_1\| \|b_1\|$. This notation is used where it is proven that if (A,B) is a global minimum of (3), then A and B are jointly equalized. A clear but important observation, is that when $y = x$ and $m_1 = m_2$ (the dimensions of A and B are equal) we get an objective of an autoencoder.

DeCov. Cogswell et al. [4] use the fact that Dropout leads to less correlated features. Denote $h^n \in \mathbb{R}^d$ as the activations at a given hidden layer, where $n \in 1,...,N$ refers to an index of one example from a batch of size N. The covariances between all pairs of activations i and j form the matrix C:

$$C_{i,j} = \frac{1}{N} \sum_n (h_i^n - \mu_i)(h_j^n - \mu_j), \tag{4}$$

where $\mu_i = \frac{1}{N} \sum_n h_i^n$ is the sample mean of activation i over the batch. The matrix C is used in the DeCov [4] regularization, which explicitly regularizes the covariance of the features with respect to the training data by adding the following loss:

$$\mathcal{L}_{DeCov} = \frac{1}{2}(\|C\|_F^2 - \|\text{diag}(C)\|_2^2), \tag{5}$$

where $\|\cdot\|$ is the frobenius norm, and $\text{diag}(\cdot)$ returns the diagonal of a matrix. Though useful, the connection between it and Dropout is not fully established. Moreover, DeCov is even shown to be adversarial to Dropout on some occasions.

B Autoencoders

Autoencoders have been first introduced in [17] as a neural network that is trained to reconstruct its input. Their main purpose is learning in an unsupervised manner an "informative" representation of the data that can be used for clustering. The problem, as formally defined in [3], is to learn the functions $A : \mathbb{R}^n \to \mathbb{R}^p$ (encoder) and $B : \mathbb{R}^p \to \mathbb{R}^n$ (decoder) that satisfy

$$\arg\min_{A,B} \mathbb{E}[\Delta(x, B \circ A(x)], \tag{6}$$

where Δ is an arbitrary distortion function, which is set to be the ℓ_2-norm in our case, and $\mathbb{E}$ is the expectation over the distribution of x.

In the most popular form of autoencoders, A and B are neural networks [16]. In the special case that A and B are linear operations, we get a linear autoencoder [2].

Since in training one may just get the identity operator for A and B, which keeps the achieved representation the same as the input, some additional regularization is required. One option is to make the dimension of the representation smaller than the input. Another

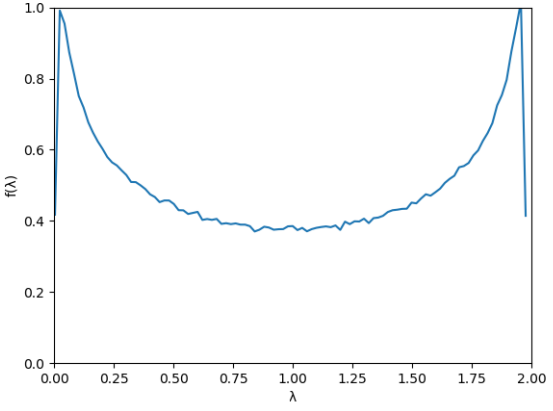

Figure 1: Empirical eigenvalue distribution of submatrices sampled from an ETF with $\beta = 0.8$ and $\gamma = 0.5$.

option is using denoising autoencoders [20]. In these architectures, the input is disrupted by some noise (e.g., additive white Gaussian noise or erasures using Dropout) and the autoencoder is expected to reconstruct the clean version of the input.

Another major improvement in the representation capabilities of autoencoders has been achieved by the variational autoencoders [10]. These encode the input to latent variables, which represent the distribution that the data came from, and decode it by learning the posterior probability of the output from it.

Autoencoders may be trained in an end-to-end manner or gradually layer by layer. In the latter case, they are "stacked" together, which leads to a deeper encoder. In [12], this is done with convolutional autoencoders, and in [21] with denoising ones.

B.1 Use of autoencoders for classification.

Autoencoders are also used for classification by using the encoder as a feature extractor and "plugging" it into a classification network. This is mainly done in the semi-supervised learning setup. First, the autoencoders are trained as described above in an unsupervised way. Then (or in parallel), the encoder is used as the first part of a classification network, and its weights may be fine tuned or not vary during training [10]. Notice that different types of autoencoders may be mixed to form new ones, as in [15], which uses them for classification, captioning, and unsupervised learning.

C The connection between ETF and MANOVA

It has been demonstrated in [6] that frames that reach the Welch bound (also known as Equiangular Tight Frames(ETF)), have MANOVA distribution. The eigenvalue distribution of the submatrices of an ETF is shown empirically to resemble the MANOVA distribution (see Fig. 1 as an example).

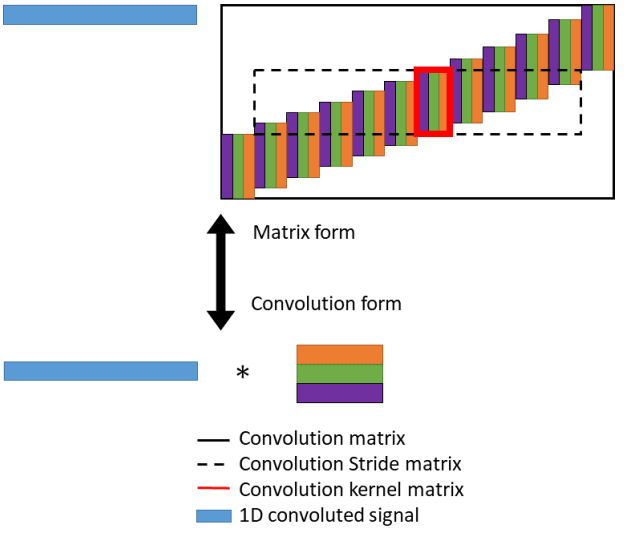

Figure 2: An illustration (1D case) of the equivalence between a convolution with three kernels and a multiplication with the equivalent Toeplitz matrix. Notice that the coherence of the convolution Toeplitz matrix is the same as the coherence of the smaller convolution stride matrix (marked by dashed lines).

In a following work [7], the relationship between the MANOVA distribution and ETFs has been further supported by showing similarity between the moments of the MANOVA distribution and the ones of the ETF. The d-th moment of a random subset in F is defined as

$$m_d \triangleq \frac{1}{n}\mathbb{E}[Tr((FPF^T)^d)], \tag{7}$$

where $Tr(\cdot)$ is the trace operator and P is a diagonal matrix with independent Bernoulli(p) elements on its diagonal.

It has been proven for $d = 2,3,4$ that these moments are lower bounded by the moments of matrices with the Wachter's classical MANOVA distribution, plus a vanishing term (as n goes to infinity with $\frac{m}{n}$ held constant). The bound is proven to hold with equality for ETFs, where in the case of $d = 4$ it is shown that it holds only for ETFs. This leads us to assume that the subsets of ETF matrices indeed have MANOVA distribution.

D Regularizing convolution kernels

To apply our regularization on convolutional layers, we may use their corresponding convolution Toeplitz matrix as illustrated in Fig. 2. Notice that the coherence of the convolution Toeplitz matrix is the same as the coherence of the smaller convolution stride matrix (marked by dashed lines) and thus, we apply the regularization directly on the stride convolution matrix. Yet, for simplicity, we just regularize the coherence between the convolution kernels (the matrix marked in red in Fig. 2), which is the central part of the stride matrix. In the

multi-dimensioanl case, each kernel is column-stacked and treated as a column vector in the regularized matrix.

E Implementation details

Fashion MNIST. The Fashion MNIST [22] is a dataset similar to MNIST but with fashion related classes that are harder to classify compared to the standard MNIST. It is composed of 60,000 examples as the train set, and 10,000 as the test set. Each example is a 28×28 grayscale image, associated with a label from 10 fashion related classes.

The architecture we used is based on LeNet5, and was changed a bit to examine a case where $m > n$. The FC layers were changed from $400 \rightarrow 120 \rightarrow 84 \rightarrow 10$ to $400 \rightarrow 800 \rightarrow 10$. For the FC layers, the ETF parameter was set to 100, and the Dropout to 0.5. For the convolutional layers, when used as a sole regularizer, the ETF parameter was increased to 1000. The batch size was 128 and the score was taken as the best one in 400 epochs. The optimizer used was ADAM with a learning rate that diminished from 10^{-3} to 10^{-5}.

CIFAR-10. The CIFAR-10 dataset is composed of 10 classes of natural images with 50,000 training images, and 10,000 testing images. Each image is an RGB image of size 32×32.

The architecture is based on a variant of Lenet5 for this data set. It involves $5\times5\times32$ and $5\times5\times64$ convolution layers with 2×2 max pooling, followed by two FC layers of $1600 \rightarrow 1024 \rightarrow 10$. For the FC layers, the ETF parameter was set to 10 when it is the sole regularizer, and to 1 when combined with Dropout. For the convolutional layers, it was set to 10. The Dropout parameter was 0.5. The batch size was 128 and the score was taken as the best one in 300 epochs. The optimizer used was Nesterov Momentum with a momentum parameter of 0.9 and a learning rate that diminished from 10^{-2} to 10^{-3}.

Tiny ImageNet. The Tiny Imagenet dataset is composed of 200 classes of natural images with 500 training examples per class, and 10,000 images for validation. Each image is an RGB image of size 64×64. It is tested by top-1 and top-5 accuracy.

The architecture we use is an adaptation of the VGG-16 model [18] to the Tiny Imagenet dataset [1]. It consists of ten 3×3 convolution layers, separated to four parts: two layers with 64 feature maps, two with 128, three with 256, and three with 512. All parts are separated by a 2×2 max pool, and after the last convolutional layer there is no pooling but FC layers of $25088 \rightarrow 4096 \rightarrow 2048 \rightarrow 200$.

For the FC layers, the ETF parameter was set to 10 when it is the sole regularizer, and to 1 when combined with Dropout. For convolutional layers, it was set to 1. The Dropout parameter was 0.5. The batch size was 64 and the score was taken as the best one in 50 epochs. The optimizer used was Nesterov Momentum with a momentum parameter of 0.9 and a learning rate that diminished from 10^{-2} by a factor of 5 when the validation top-1 accuracy ceased increasing.

Convolution regularization details. In the convolution regularization experiments, we apply the dropout and ETF regularization as follows. For Fashion MNIST and CIFAR 10, we apply the regularization on the second convolutional layer - right before the FC ones. For Tiny ImageNet, we apply it on the last three convolution layers - the ones with feature maps of size 512. Dropout in all cases is applied once after the convolutional layers. In the case of

the first two networks, it is applied after the pooling operation that follows the convolutions (since this gives better performance with Dropout).

Penn Tree Bank. We perform word level prediction experiments on the Penn Tree Bank data set [11]. It consists of 929,000 training words, 73,000 validation words, and 82,000 test words. The vocabulary has 10,000 words. In this data set, we measure the results by the attained perplexity, which we aim at reducing.

The architecture is as described in [23]. Two models are considered, where all of them involve LSTMs with two-layer, which are unrolled for 35 steps. The *small* model includes 200 hidden units, and the *medium* includes 650.

Small model parameters: When used as a sole regularizer, the ETF parameter was set to 1, and when combined with Dropout, to 0.1. The Dropout was set to 0.75. The score was taken as the best one in 30 epochs on the validation set. The optimizer used was stochastic gradient descent (SGD) and the learning rate diminished from 1 by a factor of 0.7.

Medium model parameters: When used as a sole regularizer, the ETF parameter was set to 50, and when combined with Dropout, to 1. The Dropout was set to 0.5. The score was taken as the best one in 45 epochs on the validation set. The optimizer used was SGD and the learning rate diminished from 1 by a factor of 0.8.

References

[1] VGG code for tiny imagenet, 2017. https://github.com/pat-coady/tiny_imagenet.

[2] P. Baldi and K. Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Netw.*, 2(1):53–58, January 1989. ISSN 0893-6080. doi: 10.1016/0893-6080(89)90014-2. URL http://dx.doi.org/10.1016/0893-6080(89)90014-2.

[3] Pierre Baldi. Autoencoders, unsupervised learning, and deep architectures. In Isabelle Guyon, Gideon Dror, Vincent Lemaire, Graham Taylor, and Daniel Silver, editors, *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, volume 27 of *Proceedings of Machine Learning Research*, pages 37–49, Bellevue, Washington, USA, 02 Jul 2012. PMLR.

[4] Michael Cogswell, Faruk Ahmed, Ross B. Girshick, Larry Zitnick, and Dhruv Batra. Reducing overfitting in deep networks by decorrelating representations. *CoRR*, 2015.

[5] Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *NIPS*, 2016.

[6] Marina Haikin, Ram Zamir, and Matan Gavish. Random subsets of structured deterministic frames have MANOVA spectra. *Proceedings of the National Academy of Sciences*, 114(26):E5024–E5033, 2017.

[7] Marina Haikin, Ram Zamir, and Matan Gavish. Frame moments and welch bound with erasures. *CoRR*, abs/1801.04548, 2018.

[8] G. E. Hinton, N. Srivatava, A. Krizhvsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *NIPS*, 2012.

[9] Ishan Jindal, Matthew S. Nokleby, and Xuewen Chen. Learning deep networks from noisy labels with dropout regularization. *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 967–972, 2016.

[10] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

[11] M. P. Marcus, M. A. Marcinkiewicz, and Santorini B. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313—-330, 1993.

[12] Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In Timo Honkela, Włodzisław Duch, Mark Girolami, and Samuel Kaski, editors, *Artificial Neural Networks and Machine Learning – ICANN 2011*, pages 52–59, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-21735-7.

[13] Poorya Mianjy, Raman Arora, and Rene Vidal. On the implicit bias of dropout. In *ICML*, 2018.

[14] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. Norm-based capacity control in neural networks. In Peter Grünwald, Elad Hazan, and Satyen Kale, editors, *Proceedings of The 28th Conference on Learning Theory*, volume 40 of *Proceedings of Machine Learning Research*, pages 1376–1401, Paris, France, 03–06 Jul 2015. PMLR. URL http://proceedings.mlr.press/v40/Neyshabur15.html.

[15] Yunchen Pu, Zhe Gan, Ricardo Henao, Xin Yuan, Chunyuan Li, Andrew Stevens, and Lawrence Carin. Variational autoencoder for deep learning of images, labels and captions. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2352–2360, 2016.

[16] M. Ranzato, F. J. Huang, Y. Boureau, and Y. LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2007. doi: 10.1109/CVPR.2007.383157.

[17] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Learning Internal Representations by Error Propagation, pages 318–362. MIT Press, Cambridge, MA, USA, 1986. ISBN 0-262-68053-X. URL http://dl.acm.org/citation.cfm?id=104279.104293.

[18] Karen Simonian and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations (ICLR)*, 2015.

[19] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15, 2014. 1929–1958.

[20] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 1096–1103, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4. doi: 10.1145/1390156.1390294. URL http://doi.acm.org/10.1145/1390156.1390294.

[21] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408, December 2010. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1756006.1953039.

[22] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017.

[23] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization, 2014. URL https://arxiv.org/abs/1409.2329.