On the Relationship between Dropout and Equiangular Tight Frames

Dor Bank · Raja Giryes

Abstract Dropout is a popular regularization technique in neural networks. Yet, the reason for its success is still not fully understood. This paper provides a new interpretation of Dropout from a frame theory perspective. By drawing a connection to recent developments in analog channel coding, we suggest that for a certain family of autoencoders with a linear encoder, the minimizer of an optimization with dropout regularization on the encoder is an equiangular tight frame (ETF). Since this optimization is non-convex, we add another regularization that promotes such structures by minimizing the cross-correlation between filters in the network. We demonstrate its applicability in convolutional and fully connected layers in both feed-forward and recurrent networks. All these results suggest that there is indeed a relationship between dropout and ETF structure of the regularized linear operations.

Keywords: Deep Learning, Dropout Regularization, Autoencoders, Equiangular tight frames, MANOVA distribution

1 Introduction

Deep neural networks are powerful computational models that have been used extensively for solving problems in computer vision, speech recognition, natural language processing, and many other areas [30,34,61,70]. The parameters of these architectures are learned from a given training set. Thus, regularization techniques for preventing overfitting of the data are very much required [24,38].

One of the most popular strategies is Dropout, which randomly drops hidden nodes along with their connections at training time [31,58]. During training, in each batch, nodes are kept with a probability \( p \), which causes them to be eliminated with probability \( q = 1 - p \) (with their corresponding input and output weights). The weights of the remaining nodes are trained by back-propagation regularly. At inference time, the outputs of the layer(s) on which Dropout was applied are multiplied by \( p \).

Though very useful, Dropouts explicit regularization is not fully understood yet. Such an understanding is required to exploit the full potential of Dropout, and to deepen our knowledge in neural networks.

In this work, we approach Dropout from a signal processing and information theory perspective. We draw a relationship between Dropout and the problem of signal recovery from erasures in the analog domain (see Fig. 1). In this “analog coding” problem, a signal passes through an encoder \( A \) and then disrupted by an additive noise and part of its values are nullified. Once received, it is recovered by passing through a decoder \( B \). The goal is to find the pair \((A,B)\), which recovers the input signal with a minimal \( \ell_2 \) error.

To draw a connection to Dropout, we make the following steps. First, we examine a specific case, where the encoding \( A \) is performed by a (linear) matrix multiplication \( F \), and the recovery is done by solving a least squares problem with the given measurements and \( F_s \), the subset of columns from the matrix \( F \) corresponding to the kept measurements. It has been suggested in a recent work that frames with MANOVA distribution [19], minimize the expected \( \ell_2 \) error in this setting [25]. Though not proven formally, various empirical measurements lead to the conjecture that Equiangular Tight Frames (ETF), matrices whose columns have unit norm and the smallest possible correlation in absolute value between
and thus minimize the \( \ell_2 \) error in the above setup [26, 27].

Next, we draw a relationship to autoencoders (briefly illustrated in Fig. 1). An autoencoder is a type of a neural network used to learn an efficient data representation in an unsupervised manner. It is decomposed of two parts, the first encodes the data and the second decodes it from the learned representation. Considering an autoencoder with a linear encoder and a Dropout regularization applied on it, we get a very similar structure to the analog coding problem. Thus, if the decoder solves the least squares problem, then an ETF is likely to be a global minimum in the optimization of the encoder.

Last, we notice that the representation learned by autoencoders may be used for classification, e.g., in a semi-supervised learning setup, where the learned encoder serves as a feature extractor. This leads to the conjecture that promoting structure of an ETF in some layers of the network might turn useful for classification tasks as well.

We support our claim by experiments done on various data-sets for image classification and word level prediction. We measure the effect of the ETF regularization when used as a sole regularizer, and when combined with Dropout. For fully connected (FC) layers, we promote an ETF structure for the weight matrix directly by reducing the correlation between its rows. We demonstrate this regularization for both feed-forward and recurrent (LSTM) networks. For convolutional layers, we do not use their corresponding Toeplitz matrix. Instead, for simplicity, the coherence between the convolution kernels is minimized.

2 Background

2.1 Regularization techniques for neural networks

Neural networks usually have many parameters. This increases their likelihood to overfit the training data. To improve their generalization, several regularization techniques have been proposed [24, 38]. We briefly mention some of them.

Batch Normalization [32] aims at normalizing the variance of the features at a given layer of the network. A layer regularized by it, outputs each minibatch with its features having zero mean and unit variance (separately). In addition, the total mean and variance is calculated via moving average, and then used in the normalization of the test set. To let the network reverse the normalization effect, two learnable parameters are added for each feature - one for its mean and one for its variance.

Another example is DropConnect [64], which is a variation of Dropout, where the weights are nullified with probability \( q \) instead of the features. Thus, all the features participate in each batch while only \( p \) of the weights (chosen randomly at each forward pass) are used.

Weight decay [37], is a regularization that minimizes the \( \ell_2 \) norm of the layer weights, intentionally reducing their flexibility. Replacing the \( \ell_2 \) norm with \( \ell_1 \) has been also explored [55, 69]. While the latter induces sparse weight matrices, it is harder to optimize, yielding weight decay a more commonly used method. Another alternative to weight decay is Jacobian regularization: The first derivative of the output features with respect to the input features is added to the loss, to make the output features more robust and less dependent on the input. It has been proposed for autoencoders in [51], and suggested in [57] for general neural networks showing that it increases the input margin in them, and thus improves their performance.

2.2 Dropout regularization

The Dropout regularization introduced by Srivastava et al. [58], is one of the most popular regularization strategies. When applied on a given layer, it randomly drops hidden nodes along with their connections [31]. 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 its 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 [33].

Gal and Ghahramani use Dropout to measure the uncertainty of a network. They approximate the likelihood functions with Monte Carlo sampling, which is done via Dropout [21]. 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 [22].

2.3 Understanding Dropout

Given the empirical success of Dropout, an extensive research was performed to reveal its theoretical foundation.

One of the disadvantages of Dropout is, that it slows down the training time. Wang and Manning, have implied that Dropout makes a Monte Carlo assessment of the layers output. Drawing from the central limit theorem, they assume that for each sample, the output of a layer with Dropout has a Gaussian distribution, with its own mean and variance. Using that, for each batch during training, they approximate the sample-wise mean and variance, and train the model to reduce the variance and adjust the mean by backpropagation. This leads to a reduction in the training time [65].

Frazier-Logue and Hanson, propose a similar concept but from a different perspective. They suggest that Dropout is just a special case of a stochastic delta rule (SDR), where each weight is parameterized as a random variable with a mean and variance of its own. At each forward pass, the value of each weight is drawn randomly. In the case of Dropout, it is just randomly selecting between 0 and the weight value. In their work, they draw the weight from a Gaussian distribution, where the mean represents the weight value. Both the mean and the variance are updated by backpropagation. In addition, the variance diminishes such that at the end of the training phase, we get a deterministic value. Their method leads to faster convergence than using Dropout [20].

Other works that analyzed Dropout regarded its effect as a generation of an ensemble of sub-networks with the active nodes. Hara et al. compared training with Dropout to ensemble learning, where several sub-networks are learned independently, and then the final result is an aggregation of all of them. This type of ensemble learning and dropout have great similarity, except that Dropout uses a different set of hidden units in each learning iteration, and thus is mostly superior.

Baldi et al. introduced a general formalism for studying Dropout in neural networks with the sigmoid activation function. They study Dropout in a shallow network with a single output unit which produces the outputs $O_i, ..., O_m$ with probabilities $P_i, ..., P_m$ (depending on the Dropout keep probability). After defining the output expectation $E = \sum_i O_i$, the weighted geometric mean (WGM) of the outputs $G = \prod_i O_i^{P_i}$, and the WGM of the output complements $G' = \prod_i (1 - O_i)^{P_i}$, they show that the normalized WGM $\frac{G'}{G}$ is a good approximation for $E$, and explain how it could be estimated via a single forward pass [5,46].

Wager et al. analyzed Dropout applied to the logistic loss for generalized linear models (GLM) [63]. They consider a GLM $A(x_i; \beta)$ that for a sample $x_i$ outputs a response $y_i$, whose distribution is a function of the multiplication of $x_i$ with the weight vector $\beta$. They show that for GLM, the penalty of Dropout takes the form of $c \cdot \beta^T \text{diag}(X^TV(\beta)X)\beta$, where $X$ is the data matrix, and $V(\beta)$ is a diagonal matrix whose elements $V(\beta)_{ii}$ are equal to the variance of $y_i$. Given infinite data, if $\beta$ is calculated using the maximum likelihood estimator, then $\frac{1}{n}X^TV(\beta)X = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{n} X_{ij}^2 \beta_{ij}$ is an estimation for the Fisher Information Matrix $I_i$. This leads the authors to the conclusion that Dropout is similar to applying $\ell_2$ regularization, where each squared weight is normalized by its appropriate part in $\text{diag}(I_i)^{-1/2}$.

In a following work, Helmond and Lond discussed the mathematical properties of Dropout and derived a sufficient condition to guarantee a unique minimizer for a loss function in which Dropout is used [29]. In order to better differentiate between the bias induced by Dropout and the one induced by $\ell_2$ regularization, they provide examples for input data distributions for which the error achieved by Dropout is lower than the one of $\ell_2$, and examples for the opposite case (where $\ell_2$ is better).

In another work, Wager et al. have shown that under a generative Poisson topic model with long documents, Dropout training improves the exponent in the generalization bound for empirical risk minimization [62]. Cavazza et al. have discussed the equivalence between Dropout and a fully deterministic model for Matrix Factorization in which the factors...
are regularized by the sum of the product of the squared Euclidean norms of the columns of the matrix $\mathbf{I}$. Furthermore, under certain conditions, they showed that Dropout can be used as a low-rank regularizer with data dependent singular-value thresholding.

The two methods most related to our work are the one by Mianjy et al. [44] and Decov [13]. The first studies the implicit bias of Dropout [44]. 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:

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

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

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

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

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

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 [45], 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 if $\forall i, \|a_i\| \|b_i\| = \|a_i\| \|b_i\|$. 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.

The second related work by Cogswell et al. [13], use the fact that Dropout leads to less correlated features. Denote $h^o \in \mathbb{R}^d$ 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 $C$:

$$C_{i,j} = \frac{1}{N} \sum_n (h^o_i - \mu_i) (h^o_j - \mu_j), \quad (4)$$

where $\mu_i = \frac{1}{N} \sum_n h^o_i$ is the sample mean of activation $i$ over the batch. The matrix $C$ is used in the DeCov [13] regularization, which explicitly regularizes the covariance of the features with respect to the training data by adding the following loss:

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

where $\|\|$ is the frobenius norm, and $\text{diag}()$ 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.

Hereafter, we compare our theory to the one of Mianjy et al. [44] and show that our proposed regularization method enforces jointly equalized matrices when performed in a linear autoencoder, and mention the connection between our work and Decov. More details are given in Section 4.2.

2.4 Autoencoders

Autoencoders have been first introduced in [52] 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 $A : \mathbb{R}^n \rightarrow \mathbb{R}^p$ (encoder) and $B : \mathbb{R}^p \rightarrow \mathbb{R}^n$ (decoder) that satisfy

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

where $\Delta$ is an arbitrary distortion function, which is set to be the $\ell_2$-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 [50]. In the special case that $A$ and $B$ are linear operations, we get a linear autoencoder [4].

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 option is using denoising autoencoders [59]. 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 [55]. 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 [43], this is done with convolutional autoencoders, and in [60] with denoising ones.
2.5 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 [53]. Notice that different types of autoencoders may be mixed to form new ones, as in [49], which uses them for classification, captioning, and unsupervised learning.

3 Signal reconstruction from a frame representation with erasures

We now address a notorious problem in information theory: Signal reconstruction from a frame representation with erasures, as illustrated in Fig. 1. Later on, we shall use its resemblance to autoencoders. Consider the signal vector \( x \in \mathbb{R}^m \) and a frame \( F \). First, the vector is encoded by \( F \), i.e., yielding \( xF \), which is then transmitted in an analog channel. In the channel, part of the values are nullified with probability \( p \), and then the remaining values are disrupted by an additive white Gaussian noise (AWGN).

Notice that nullifying the values in \( xF \) with probability \( p \) is equivalent to removing columns from \( F \) with probability \( p \) and then multiplying it with \( x \). Denote by \( S(p) \) the pattern that defines which vectors of \( F \) are used, with respect to the probability \( p \), and by \( F_i \) the sub-matrix of \( F \) with the vectors corresponding to \( S(p) \). Then the resulted vector after the addition of the AWGN \( w \) is defined as

\[
y = xF_i + w.
\]

In order to recover the input from \( y \), one may use the least square solution

\[
\hat{x} = \arg\min_{\hat{x}} \| y - \hat{x}F_i \|_2^2 = yF_i^T,
\]

where \( F_i^T \) is the pseudo-inverse of \( F_i \). Thus, if one wishes to optimize \( F \) for minimizing the reconstruction error in the \( \ell_2 \) sense, the target objective is:

\[
\arg\min_{F} \mathbb{E} \| x - \hat{x} \|_2^2 = \arg\min_{F} \mathbb{E} \| x - yF_i \|_2^2,
\]

where the expectation is with respect to the noise variable \( w \), the distribution of the input variable \( x \), and the sampling vector \( S(p) \).

3.2 Equiangular tight frames

The Gram matrix of a frame \( F \) is defined by \( G_F = F^TF \) and contains outside its diagonal the cross-correlation values between the columns of the frame \( F \), i.e., \( G_{i,j} \) contains the cross-correlation value between the \( i \)th and \( j \)th columns of \( F \). The Welch bound [66] provides a universal lower bound on the mean and maximal absolute value of the cross-correlations between the frame vectors. A frame that achieves the Welch lower bound on the maximal absolute cross-correlation value is known as an equiangular tight frame (ETF). Notice that ETFs are a sub-group of tight frames. The Gram matrix \( G_{ETF} \) of a \( m \times n \) ETF satisfies:

\[
|G_{ETF_{ij}}| = \begin{cases} 1 & i=j \\ \frac{n-m}{(n-1)m} & \text{else.} \end{cases}
\]

Intuitively, its \( n \) vectors are spread uniformly across an \( m \) dimensional space with an angle of \( \theta = \arccos \sqrt{\frac{n-m}{(n-1)m}} \) between them as illustrated at Fig. 2. The maximal off-diagonal value in the Gram matrix is denoted the mutual coherence [15] or simply the coherence value.

3.3 MANOVA distribution

The MANOVA distribution (also known as Jacobi ensemble) is the eigenvalue distribution of the ensemble of random Jacobi matrices. In [19], it is shown that a \( m \times k \) MANOVA distributed matrix, has an eigenvalue distribution of the shape:

\[
f(x) = \frac{\sqrt{(x-r_-)(r_+ - x)}}{2\beta \pi x(1-\gamma^2)} + \left(1 + \frac{1}{\beta} - \frac{1}{\beta \gamma}\right)^+ \delta\left(x - \frac{1}{\gamma}\right),
\]

where

\[
r_{\pm} = \sqrt{\beta(1-\gamma)} \pm \sqrt{(1-\beta \gamma)^2},
\]
Fig. 2 Illustration of the four vectors in a three dimensional space, which represents a $3 \times 4$ ETF. Between each two vectors there is an angle of $\arccos \sqrt{\frac{4}{11}} = 70.53^\circ$.

Fig. 3 MANOVA distribution for $\beta = 0.8$ and $\gamma = 0.5$.

$\beta = \frac{k}{m}$ and $\gamma = \frac{m}{n}$. The distribution is illustrated in Fig. 3 when the decoder is the pseudo inverse and the inputs are i.i.d Gaussian distributed, it is conjectured in [25] that frames whose random subsets resemble the classical MANOVA distribution, minimize the squared loss in the high SNR regime.

For the setup of Fig. 1, when the decoder is the pseudo inverse and the inputs are i.i.d Gaussian distributed, it is conjectured in [25] that frames whose random subsets resemble the classical MANOVA distribution, minimize the squared loss in the high SNR regime.

Notice that minimizing the estimation error at the output of the decoder is equivalent to minimizing $E[Tr(F_s^TF_i)^{-1}]$ because

$$\arg \min_m E\|x - \hat{x}\|_2 = \arg \min_m E\|x - F_s^TF_i F_s^T F_i w\|_2$$

(13)

$$= \arg \min_m E\|F_i^T w\|_2$$

$$= \arg \min_m E(\|w^T F_i^T F_i w\|)$$

$$= \arg \min_m E(Tr(w^T F_i^T F_i w))$$

$$= \arg \min_m E(Tr(F_i^T F_i)^{-1})$$

$$= \arg \min_m E(Tr(F_i^T F_i)^{-1}).$$

Assuming the frame columns are normalized, notice that for each subframe with $k$ columns, the value of $E[Tr(F_s^TF_i)] = \sum_{i=1}^k \lambda_i$ is fixed, where $\lambda_i$ denotes the $k$ eigenvalue. In addition, notice that the expression to be minimized can by written as $E \left[ \sum_{i=1}^k \frac{1}{\lambda_i} \right]$. Following these two observations, it is clear that the best possible distribution is $p(\lambda) = \delta(\lambda)$, i.e., each sub-matrix is unitary. Yet, clearly, this is impossible to maintain as the frame is an over complete base and thus all its sub-matrices cannot be unitary.

To assess that Manova is the optimal choice, various popular random matrices with known distributions were tested [26]. Those matrices include Low pass frames with Van dermonde distribution [54], Gaussian frames that obey the Marchenko-Pastur distribution [26], and frames whose distribution resembles the MANOVA distribution such as ETF, Random Fourier, and Haar [26]. The conjecture is supported by the fact that MANOVA is shown to be the distribution closest to $\delta(\lambda)$. Moreover, this distribution is likely to be generated by an ETF as we shall see in the next subsection.

3.4 The connection between ETF and MANOVA

It has been demonstrated in [26] 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. 4 as an example).

In a following work [27], 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

$$m_d = \frac{1}{n} E[Tr((FF^T)^d)].$$

(14)
where $\text{Tr}(\cdot)$ 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, and by thus it is conjectured that ETFs are indeed the global minimum for the settings of Eq. (9).

4 An ETF perspective of Dropout

4.1 The relationship between Dropout and ETF

We start with the relationship to autoencoders. Notice the great resemblance between a denoising autoencoder (DAE) with linear encoder and Dropout applied on it and the analog coding problem, as illustrated in Fig. 1. Though in standard DAE the noise is added at the input, in the DAE we present here, we put the noise at the output of the encoder, as suggested in [2]. There is a similarity between the two models in the linear case as adding noise at the output of the decoder is equivalent to adding noise at its input with a covariance matrix equal to the pseudo-inverse of the decoder.

Given the above information, in the case that the encoder is linear and the decoder calculates the least squares solution, we conjecture that the global minimum of training with Gaussian distributed data and noise, and Dropout on the encoder should be an ETF for the encoder (or very close to it if the setting slightly changes).

Notice that for a given Dropout/erasure pattern, the decoder is a linear operation. Since the encoder is also linear, the autoencoder with the fixed pattern becomes a shallow linear autoencoder as used in [44] (See Section 2.3). In that work it is claimed that Dropout induces the matrices of such a shallow linear autoencoder to be jointly equalized. In our case, the optimal encoder is claimed to be an ETF and thus the linear encoder and decoder in the linear autoencoder induced by the Dropout are a sub-matrix of an ETF and its pseudo-inverse, respectively. Interestingly, it turns out that this pair is indeed jointly equalized, which corresponds with the theory derived in [44]. Notice that this is not exactly the result derived in that work, since unlike their assumption that the decoder is linear, here it is non-linear (it is linear only given a specific erasure pattern). Yet, we believe that this relationship between the two works requires further study.

To examine further the relationship between Dropout and ETFs, we set an experiment with an autoencoder that has a similar structure to the analog coding problem setup described in Section 3. The encoder $A$ in this network is a linear one, represented by a randomly initialized matrix. Specifically, we use a matrix $A$ of size $75 \times 150$.

For the decoder we do not use $A^T_s$ since it is hard to calculate its derivative with respect to $A$ in the network training. Instead, we use the fact that the pseudo inverse is the least squares solution and perform ten iterations of gradient descent $\hat{x}^{i+1} = \hat{x}^i - \mu A^T_s(A_s\hat{x}^i - y)$, where $\hat{x} = \hat{x}^0$ and $x = \hat{x}^0$. The learning rate $\mu$ is the inverse of the largest eigenvalue of the Gram matrix $A^T_sA_s$ as in [44]. For getting the sample pattern $S(p)$ we simply apply Dropout on the encoder.

The input signals are generated as i.i.d. Gaussian vectors with a standard deviation of 1 and the noise is generated with the same distribution but with a standard deviation of 0.001.

The experiment is performed in two different settings: the first includes an infinite amount of data, and the second deals with a finite and limited one. In the case of infinite data, we seek to find correspondence between the encoder
coherence and the squared loss. We use an “online” learning setup with 100 signals per batch. First, we optimize over the squared loss and measure the coherence. Second, we optimize over the following “coherence loss”:

\[ CL = \|A^T A - |G_{ETF}| \|_\infty, \]  

(15)

where \(|\cdot|\) is an element-wise absolute value, and \(\|\cdot\|_\infty\) returns the maximum absolute value in the matrix. As can be seen in Fig. 5, the coherence and the reconstruction loss are closely related. Notice that coherence minimization induces a MSE reduction and vice versa. This validates our claim on the relationship between the two. Indeed, the coherence does not reach the Welch bound, and the error is much higher than the one of an ETF. We conjecture that this is mainly due to the non-convexity of the problem.

For the case of limited data, regularization of the coherence is considered such that the new loss is

\[ L = MSE + \beta \cdot CL, \]  

(16)

where \(\beta\) is the regularization coefficient. Notice that this term encourages getting an ETF-like structure. We train the autoencoder with this new regularization over several values of \(\beta\). We use a training set of 100 signals, where the training phase includes randomizing the noise vector and the sampling pattern in each batch. The test set size is chosen as 5000 to accurately measure the test error. We train the model for 300 epoch, in which the minimal and final test error are taken (and are different when over fitting occurs), along with the final coherence.

It is known that regularization techniques increase the bias of a model. If successful, they reduce the models variance such that the total error is reduced, and thus prevent overfitting. Therefore, we expect low regularization coefficients to have little effect on the performance, large ones to perform poorly due to high bias, and for a specific range to increase the training error while decreasing the test error. Fig. 6 shows that until a certain value, both the error and the coherence diminish as \(\beta\) increases. High \(\beta\) values (in this case - higher than 0.1), result with optimization difficulties. This demonstrates that adding this term indeed helps in improving the convergence of the encoding frame closer to the desired "global minimum".

4.2 Promoting an ETF structure in general neural networks

Recalling the setting of Section 3, notice that the encoding part is exactly equivalent to a FC layer in a neural network, where the frame \(F\) plays the role of the weight matrix, and the nullification with probability \(p\) acts as Dropout. Though the specific setup discussed here is more relevant to autoencoders, we believe that the new understandings about Dropout may be carried also to more general neural networks. Inspired by the usage of autoencoders for classification, we conjecture that although ETFs are introduced here for signal recovery, they can be also used to enhance the Dropout regularization also in other networks training, e.g. for classification.

Since there are infinitely many ETFs, we do not want to regularize a layer towards a specific one. Moreover, we do not always have an ETF construction for every combination of \(m\) and \(n\). Yet, the structure of the Gram matrix is easily accessible and is the same for all ETFs that have the same value of \(m\) and \(n\).

For these reasons, and the ones specified in Section 4.1, we adopt the "ETF similarity" term presented in Eq. 16 also for general neural networks and in particular for ones performing classification tasks. Notice that in the case where \(m > n\), all vectors can be independent, and we penalize the distance from \(I_n\), which is the same as reducing the magnitude of the off-diagonal entries of \(A^T A\).

To apply our regularization on convolutional layers, we may use their corresponding convolution Toeplitz matrix as illustrated in Fig. 7. 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. 7), which is the central part of the stride matrix. In the multi-dimensional case, each kernel is column-stacked and treated as a column vector in the regularized matrix.

In an LSTM cell, we have four different FC gates: One to create a new state vector; one to create a \(forget\) vector,
which decides how much to keep from the old state; One for an input vector, which decides how much to keep from the new state; and one for the cell’s output. We promote ETF-like matrices on each one of them separately, since we do not want to impose low coherence between the vectors of the different FC layers (We may still want that the same filter will be used in the different gates.).

Interestingly, our proposed coherence based regularization technique may also be motivated by the sparse coding theory, where it is well known that it is easier to recover the sparse representation of a vector from a matrix that has a low coherence [15][17]. In a recent work, it has been shown that the layers of a convolutional neural network may be viewed as stages for reconstructing the sparse representation of the input [48]. Moreover, recovery guarantees have been developed based on the coherence showing that a smaller coherence leads to better reconstruction of the sparse representation of the input by the network [47]. While that work focuses mainly on convolutional layers, it definitely provides another motivation for our new regularization technique. This is especially true since in classical sparse coding the coherence is also used with regular matrices (equivalent to the weights in the FC layers).

5 Experiments

We evaluate our method apart of and on top of Dropout. We emphasize that we do not try to compete with Dropout, nor we try to reach state-of-the-art results, but simply measure the relationship between Dropout and our ETF similarity regularization. The exact value of the ETF regularization is chosen by cross validation. To isolate the effect of the two methods, no other regularization techniques are used. We demonstrate our proposed strategy on FC layers, convolutional layers, and LSTM based recurrent neural networks (RNNs). Four known datasets are used with their appropriate architectures.
**Fashion MNIST.** The Fashion MNIST [67] 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 → 120 → 84 → 10 to 400 → 800 → 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 5×5×32 and 5×5×64 convolution layers with 2×2 max pooling, followed by two FC layers of 1600 → 1024 → 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 50,000 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 [56] 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 → 4096 → 2048 → 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.

5.1 Fully connected layers

We start by applying our ETF regularization on the FC layers on the three image classification datasets: Fashion MNIST (Table 2), CIFAR-10 (Table 3) and Tiny ImageNet (Table 4).

As can be seen in tables 2, 3 and 4, The ETF regularization improves the test accuracy, with and without Dropout.
Table 5  Fashion MNIST - conv layer regularization

| Regularization | Test Accuracy |
|----------------|---------------|
| None           | 88.36%        |
| Dropout        | 91.14%        |
| ETF            | 90.30%        |
| Dropout+ETF    | 91.58%        |

Table 6  CIFAR-10 - conv layer regularization

| Regularization | Test Accuracy |
|----------------|---------------|
| None           | 84.41%        |
| Dropout        | 85.75%        |
| ETF            | 85.15%        |
| Dropout + ETF  | 86.36%        |

Table 7  Tiny ImageNet - conv layer regularization

| Regularization | Test top-1 | Test top-5 |
|----------------|------------|------------|
| None           | 39.92%     | 65.29%     |
| Dropout        | 43.44%     | 69.03%     |
| ETF            | 42.13%     | 67.05%     |
| Dropout + ETF  | 45.55%     | 69.80%     |

Notice that it always improves the results of Dropout when combined together with it and that on the Fashion MNIST data it gets better performance also when it is used alone.

5.2 Convolutional layers

Next, we apply our ETF regularization on the convolutional layers. For Fashion MNIST (Table 5) and CIFAR 10 (Table 6), we apply it on the second convolutional layer - right before the FC ones. For Tiny ImageNet (Table 7), 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).

It can be observed that the ETF regularization has less effect on the convolutional layers compared to the FC ones, both when applied with and without Dropout. We conjecture that for classification tasks, the kernels of the different channels have already lower coherence than the columns of a FC weight matrix. It might be also that a regularization of the coherence of the stride matrix may lead to better results.

5.3 Recurrent neural networks

Lastly, we apply our ETF regularization on LSTM cells. We test it for both the small sized model (Table 8), and the medium sized one (Table 9).

Table 8  Penn Tree Bank - small model

| Regularization | Val Perp. | Test Perp. |
|----------------|-----------|------------|
| None           | 121.39    | 115.91     |
| Dropout        | 98.260    | 93.927     |
| ETF            | 104.425   | 99.398     |
| Dropout + ETF  | 93.998    | 90.139     |

Table 9  Penn Tree Bank - medium model

| Regularization | Val Perp. | Test Perp. |
|----------------|-----------|------------|
| None           | 123.012   | 122.853    |
| Dropout        | 87.059    | 83.059     |
| ETF            | 115.868   | 111.956    |
| Dropout + ETF  | 85.267    | 81.646     |

Notice that in this case, we also see the positive effect of the ETF regularization mainly when combined with the Dropout regularization. When applied alone, the effect of the ETF regularization is weaker in the medium model compared to the small one though it always leads to improvement. We believe that this difference should be further investigated.

6 Conclusions

This work provides a novel interpretation of the role of Dropout by bringing together two, similar but "unacquainted", research fields, namely, deep learning and frame theory. This combination provides the understanding that Dropout promotes an ETF structure when applied on a linear encoder in an autoencoder model. We have shown that adding a regularization that encourages an ETF structure improves the performance in these networks. The fact that in semi-supervised learning, the encoder also serves many times as a feature extractor for classification tasks, has led us to the usage of this ETF regularization also in standard neural networks, e.g., for classification, along with Dropout. This combination has shown improvement in different tasks and network types.

It appears that the study of frames can help to gain a better understanding of the Dropout regularization. We believe that this paper makes the first steps in this direction by studying the optimal frame created by Dropout in an autoencoder architecture that has a linear encoder. The improvement demonstrated in this work by the ETF regularization together with Dropout, for various tasks such as classification, suggests that the role of ETF in neural network optimization should be more deeply analyzed in these contexts.

As stated in Section 2.3, Mianjy et al. [44] suggest that there is a strong connection between Dropout, which is applied on a single layer, and path-regularization [45], which explicitly regularizes more than one layer. Thus, we believe that the ETF structure should be used to express the effects...
of more complex network architectures, extending the results shown here for a shallow autoencoder.

Acknowledgments

We thank Prof. Ram Zamir and Marina Haikin for fruitful discussion and for introducing us to the analog coding setup. This work is supported by the ERC-STG SPADE grant.

References

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

2. A. Achille and S. Soatto. Information dropout: Learning optimal representations through noisy computation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2018.

3. P. Baldi. Autoencoders, unsupervised learning, and deep architectures. In L. Guyon, G. Dror, V. Lemaire, G. Taylor, and D. Silver, editors, Proceedings of ICML Workshop on Unsupervised and Transfer Learning, volume 27 of Proceedings of Machine Learning Research, pages 37–49, Bellevue, USA, 02 Jul 2012. PMLR.

4. P. Baldi and K. Hornik. Neural networks and principal component analysis: Learning from examples without local minima. Neural Netw., 2(1):53–58, Jan. 1989.

5. P. Baldi and P. J. Sadowski. Understanding dropout. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 26, pages 2814–2822. Curran Associates, Inc., 2013.

6. B. G. Bodmann, P. Casazza, and R. Balan. Frames for linear reconstruction without phase. The 42nd Annual Conference on Information Sciences and Systems, pages 721–726, 2008.

7. B. G. Bodmann and V. I. Paulsen. Frames, graphs and erasures. Linear Algebra and Its Applications, 404:118 – 146, 2005.

8. P. G. Casazza and J. Kovačević. Equal-norm tight frames with erasures. Advances in Computational Mathematics, 18(2):387–430, Feb 2003.

9. P. G. Casazza and G. Kutyniok. Robustness of fusion frames under erasures of subspaces and of local frame vectors. Contemporary Mathematics, 25:114–132, 2008.

10. P. G. Casazza, G. Kutyniok, and S. Li. Fusion frames and distributed processing. Applied and Computational Harmonic Analysis, 25:114–132, 2008.

11. J. Cavazza, P. Morerio, B. Haefele, C. Lane, V. Murino, and R. Vidal. Dropout as a low-rank regularizer for matrix factorization. Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics, 84:435–444, 09–11 Apr 2018.

12. O. Christensen. An introduction to frames and riesz bases. Birkhauser, 2003.

13. M. Cogswell, F. Ahmed, R. B. Girshick, L. Zitnick, and D. Batra. Reducing overfitting in deep networks by decorrelating representations. CoRR, 2015.

14. N. Coifman and J. P. Gazeau. Finite tight frames and some applications. Journal of Physics A: Mathematical and Theoretical, 43(19):193001, 2010.

15. D. Donoho and M. Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via l1 minimization. Proc. Nat. Acad. Sci., 100(5):2197–2202, Mar. 2003.

16. J. M. Duarte-Carvaljino and G. Sapiro. Learning to sense sparse signals: Simultaneous sensing matrix and sparsifying dictionary optimization. IEEE Transactions on Image Processing, 18(7):1395–1408, July 2009.

17. M. Elad. Optimized projections for compressed sensing. IEEE Transactions on Signal Processing, 55(12):5695–5702, Dec 2007.

18. Y. C. Eldar. Sampling Theory: Beyond Bandlimited Systems. Cambridge University Press, 2015.

19. L. Erdős and B. Farrell. Local Eigenvalue Density for General MANOVA Matrices. Journal of Statistical Physics, 152:1003–1032, Sept. 2013.

20. N. Frazier-Logue and S. J. Hanson. Dropout is a special case of the stochastic delta rule: faster and more accurate deep learning. CoRR, abs/1808.03578, 2018.

21. Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48, ICML’16, pages 1050–1059, 2016.

22. Y. Gal and Z. Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In NIPS, 2016.

23. W. Ge, W. Huang, D. Dong, and M. R. Scott. Deep metric learning with hierarchical triplet loss. In V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, editors, Computer Vision – ECCV 2018, pages 272–288, Cham, 2018. Springer International Publishing.

24. I. Goodfellow, Y. Bengio, and A. Courville. Deep Learning. MIT Press, 2016.

25. M. Haikin and R. Zamir. Analog coding of a source with erasures. In 2016 IEEE International Symposium on Information Theory (ISIT), pages 2074–2078, July 2016.

26. M. Haikin, R. Zamir, and M. Gavish. Random subsets of structured deterministic frames have MANOVA spectra. Proceedings of the National Academy of Sciences, 114(26):E5024–E5033, 2017.

27. M. Haikin, R. Zamir, and M. Gavish. Frame moments and Welch bound with erasures. CoRR, abs/1801.04548, 2018.

28. D. Han and D. R. Larson. Frames, bases and group representations. American Mathematical Society, 697, 2000.

29. D. P. Helmbold and P. M. Long. On the inductive bias of dropout. Journal of Machine Learning Research, 16:3403–3454, 2015.

30. G. Hinton, L. Deng, D. Yu, G. Dahl, A.-R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition. IEEE Signal Processing Magazine, 29(6):82–97, 2012.

31. G. E. Hinton, N. Srivatava, A. Krozhvsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing coadaptation of feature detectors. NIPS, 2012.

32. S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37, ICML’15, pages 448–456. JMLR.org, 2015.

33. I. Jindal, M. S. Nokleby, and X. Chen. Learning deep networks with hierarchical triplet loss. In V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, editors, Computer Vision – ECCV 2018, pages 1–9, 2018.

34. D. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems, pages 1–9, 2012.

35. D. P. Kingma and M. Welling. Auto-encoding variational bayes. CoRR, abs/1312.6114, 2013.

36. A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In Advances In Neural Information Processing Systems, pages 1–9, 2012.

37. A. Krogh and J. A. Hertz. A simple weight decay can improve generalization. In Proceedings of the 7th Annual Conference on Computational Learning Theory, pages 95–100, 1994.

38. J. Kukaka, V. Golkov, and D. Cremers. Regularization for deep learning: A taxonomy. In ArXiv preprint, 2017. hrfc://https://arxiv.org/abs/1710.10686.

39. D. Larson and S. Scholze. Signal reconstruction from frame and sampling erasures. Journal of Fourier Analysis and Applications, 21(5):1146–1167, Oct 2015.
40. Y. LeCun, P. Y. Simard, and B. Pearlmutter. Automatic learning rate maximization by on-line estimation of the hessian's eigenvectors. In S. J. Hanson, J. D. Cowan, and C. L. Giles, editors, Advances in Neural Information Processing Systems 5, pages 156–163. Morgan-Kaufmann, 1993.

41. T.-Y. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. 2017 IEEE International Conference on Computer Vision (ICCV), pages 2999–3007, 2017.

42. M. P. Marcus, M. A. Marcinkiewicz, and S. B. Building a large annotated corpus of english: The penn treebank. Computational Linguistics, 19(2):313–330, 1993.

43. J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In T. Honkela, W. Duch, M. Girolami, and S. Kaski, editors, Artificial Neural Networks and Machine Learning – ICANN 2011, pages 52–59, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

44. Ø. Ryan and M. Debbah. Asymptotic behaviour of random vandermonde matrices with entries on the unit circle. volume 55, pages 156–163, 2016.

45. B. Neyshabur, R. Tomioka, and N. Srebro. Norm-based capacity control in neural networks. In P. Grnwald, E. Hazan, and S. 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.

46. P. S. P. Baldi. The dropout learning algorithm. Artificial Intelligence, 210:78–122, 2014.

47. V. Papyan, Y. Romano, and M. Elad. Convolutional neural networks analyzed via convolutional sparse coding. Journal of Machine Learning Research (JMLR), (18):1–52, 2017.

48. V. Papyan, Y. Romano, J. Sulam, and M. Elad. Theoretical foundations of deep learning via sparse representations: A multilayer sparse model and its connection to convolutional neural networks. IEEE Signal Processing Magazine, 35(4):72–89, July 2018.

49. Y. Pu, Z. Gan, R. Henao, X. Yuan, C. Li, A. Stevens, and L. 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.

50. 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.

51. S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In International Conference on International Conference on Machine Learning (ICML), pages 833–840, 2011.

52. 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.

53. M. Rupf and J. L. Massey. Optimum sequence multisets for synchronous code-division multiple-access channels. IEEE Trans. Inf. Theor., 40(4):1261–1266, Sept. 2006.

54. Ø. Ryan and M. Debbah. Asymptotic behaviour of random vandermonde matrices with entries on the unit circle. volume 55, pages 3115–3148, 2009.

55. S. Scardapane, D. Commiiniello, A. Hussain, and A. Uncini. Group sparse regularization for deep neural networks. Neurocomputing, 241:81 – 89, 2017.

56. K. Simonian and A. Zisserman. Very deep convolutional networks for large-scale image recognition. International Conference on Learning Representations (ICLR), 2015.

57. J. Sokolic, R. Giryes, G. Sapiro, and M. R. D. Rodrigues. Robust large margin deep neural networks. IEEE Transactions on Signal Processing, 65(16):4265–4280, 2017.