

Regularization of Deep Neural Networks with Spectral Dropout

Salman H Khan, *Member, IEEE*, Munawar Hayat, and Fatih Porikli, *Fellow, IEEE*

Abstract—The big breakthrough on the ImageNet challenge in 2012 was partially due to the ‘dropout’ technique used to avoid overfitting. Here, we introduce a new approach called ‘Spectral Dropout’ to improve the generalization ability of deep neural networks. We cast the proposed approach in the form of regular Convolutional Neural Network (CNN) weight layers using a decorrelation transform with fixed basis functions. Our spectral dropout method prevents overfitting by eliminating weak and ‘noisy’ Fourier domain coefficients of the neural network activations, leading to remarkably better results than the current regularization methods. Furthermore, the proposed is very efficient due to the fixed basis functions used for spectral transformation. In particular, compared to Dropout and Drop-Connect, our method significantly speeds up the network convergence rate during the training process (roughly $\times 2$), with considerably higher neuron pruning rates (an increase of $\sim 30\%$). We demonstrate that the spectral dropout can also be used in conjunction with other regularization approaches resulting in additional performance gains.

Index Terms—Regularization, Spectral Analysis, Image Classification, Deep Learning.

I. INTRODUCTION

DEEP neural networks with a huge number of learnable parameters are prone to overfitting problems when trained on a relatively small training set. This leads to poor performance on the held-out test data because the learned weights are tailored only for the training set, and they lack the generalization ability to unseen data. It has been observed that the overfitting problem is caused due to complex co-adaptations of the neurons which make the neural network dependent on their joint response instead of favoring each neuron to learn a useful feature representation [1]. A number of simple, yet powerful methods have been designed over the recent years to prevent overfitting during the network training. These methods include data augmentation [2], ℓ_1 and ℓ_2 regularization [3], elastic net regularization [4], weight decay [5], early stopping [6], max-norm constraints, and random dropout [7].

A general theme to enhance the generalization ability of neural networks has been to impose stochastic behavior in the network’s forward data propagation phase. Examples of such schemes include Dropout, which randomly shuts down neurons [7], Drop-Connect, which randomly deactivates connections between neurons [8], spatial shuffling, which randomly

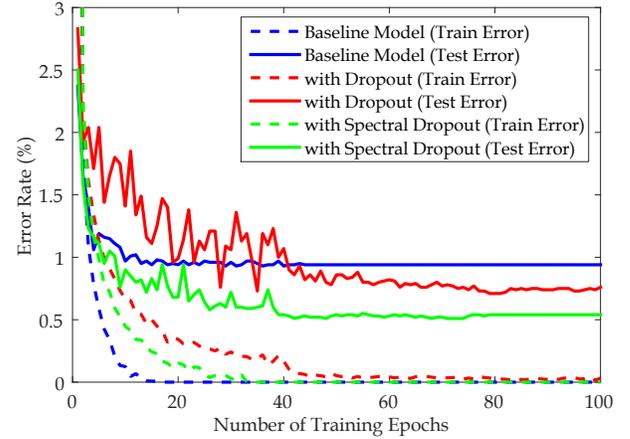


Fig. 1. **Spectral Dropout** improves the network generalization ability and yields better performance on unseen data. Further, the convergence speed is increased approximately by a factor of two compared to random dropout. The error plots are shown for training on the MNIST dataset using LeNet architecture.

performs block-wise image reorganization [9], and fractional max-pooling, which randomly changes the pooling region [10]. Generally, these approaches also perform a marginalization step during the prediction phase to compute the expected output. Data augmentation (e.g., with color jittering [11]), stochastic pooling [12] and model averaging [13] to create an ensemble effect can also be interpreted in a similar way. All these regularization approaches force the network to learn generic feature detectors, instead of merely memorizing the training samples.

In this paper, we propose a different approach for network regularization that does not adopt a fully randomized procedure, yet achieves improved generalization by dropping the noisy spectral components (see Fig. 1). To avoid co-adaptations of the feature detectors and to identify noisy spectral components, we propose to use a decorrelation transform, such as the discrete cosine transform (DCT). In contrast to the Dropout approach that randomly shuts down feature detectors during the training phase, our approach drops out the less significant spectral components to preserve the discriminative ability of network activations. This enables the network to become invariant to the ‘noisy’ spectral components by randomly selecting only the most important basis vectors for signal reconstruction during the spectral dropout regularization process. Furthermore, while Dropout slows down the network convergence speed roughly by a factor of two or more [11], our spectral dropout approach does not impede the network

S. H. Khan is with the Data61, Commonwealth Scientific and Industrial Research Organisation (CSIRO), Canberra ACT 2601 and Australian National University, Canberra ACT 0200, e-mail: salman.khan@csiro.au

M. Hayat is with the University of Canberra, Bruce ACT 2617.

F. Porikli is with the Australian National University, Canberra ACT 0200.

convergence rate and therefore provides gain in both the network performance and training efficiency.

Spectral dropout complements other regularization techniques (e.g., random dropout, ℓ_1/ℓ_2 regularization) and can easily work in conjunction with them to achieve a better performance and superior generalization capability. We demonstrate that the proposed approach generates a very compact and sparse intermediate feature representation that can significantly reduce the storage requirements for applications which perform retrieval, comparisons or matching in the feature space. Our experiments extensively test the proposed approach with different popular network architectures such as the LeNet, Network in Network (NiN) and the recent Residual Network (ResNet). The experimental analysis shows significant improvements in classification performance on the MNIST, CIFAR and SVHN datasets achieving low error rates of 0.38%, 5.76% and 2.12% respectively.

The main highlights of this paper include:

- A novel approach to reduce network overfitting using spectral dropout.
- Despite of much higher neuron pruning rates, our approach achieves better convergence performance compared to Dropout and Drop-Connect.
- Spectral dropout can be used to obtain uncertainty estimates during the test phase which can signify the confidence of network predictions.
- Our experiments show a consistent and remarkable performance boost on a diverse set of networks.

The rest of the paper is organized as follows. A brief overview of the closely related approaches is given in Sec. III. The proposed approach is described in Sec. IV followed by a detailed literature review in Sec. II. The specific implementation details are given in Sec. V and our experimental results are given in Sec. VI. We provide a thorough analysis on various aspects of our approach in Sec. VII.

II. RELATED WORK

Network Regularization has been an active research area recently. We review in detail two very popular approaches namely Dropout [7] and Drop-Connect [8] in Sec. III. Batch normalization is another related approach that rather indirectly improves network generalization by reducing internal covariance shift of the feature representations [14]. In this aspect, batch normalization is close to data decorrelation and whitening based approaches which have been traditionally used for automatic feature learning [15]. Other remarkable regularization techniques include normalization [3], weight decay [5], model averaging [13], early stopping [6], Gaussian dropout [16], and sparse constraints [17]. Huang *et al.* recently proposed a stochastic approach to vary network depth during training, which performs surprisingly well in practice [18]. Different to above mentioned approaches, our work proposes a new regularization framework based on spectral representations.

Spectral Representations have been proposed in the literature to achieve computational advantages in the learning and inference of deep neural networks [19]. Mathieu *et al.* [20]

used the Fourier transform to speed-up the expensive convolution operation in the spatial domain. Similar approaches have been reported in [21], [22] to enhance the network efficiency. In addition to enabling fast computations, frequency domain representations have been used to reduce the storage and memory requirements in deep networks [23]. This is made possible due to the fact that network parameters can be compactly represented in the spectral domain removing any redundancy [24], [25]. More recently, Rippel *et al.* [26] proposed spectral representations for activation pooling and parametrization to achieve dimensionality reduction and better network optimization. In contrast to these works, we use spectral representations to enhance the generalization ability of deep networks without compromising on the convergence performance. Note that [26] also achieves faster convergence rates by optimizing filters in the spectral domain. However, they need to switch many times back and forth between spatial and spectral domains to apply spatial-domain non-linearities. This repeated forward and inverse transformations are very expensive and therefore their faster convergence rate does not imply overall reduction in learning time. In contrast, our approach enhances convergence speed while working with spatial domain filters and avoids such additional overhead.

III. BACKGROUND

One of the most popular approaches for neural network regularization is the Dropout technique [7]. During network training, each neuron is activated with a fixed probability (usually 0.5 or set using a validation set). This random sampling of a sub-network with-in the full-scale network introduces an ensemble effect during the testing phase, where the full network is used to perform prediction. Another similar approach is the Drop-Connect [8], which randomly deactivates the network weights (or connections between neurons) instead of randomly reducing the neuron activations to zero. In contrast to random Dropout and Drop-Connect, our approach regularizes the network output by discarding noisy spectral components during the train and test phase.

Let us consider a CNN that is composed of L weight layers, indexed by $l \in \{1 \dots L\}$. Since Dropout and Drop-Connect have predominantly been applied to Fully Connected (FC) layers in the literature, we consider the simpler case of FC layers first. Given output activations \mathbf{a}_{l-1} from the previous layer, a FC layer performs an affine transformation followed by a element-wise non-linearity, as follows:

$$\mathbf{a}_l = f(\mathbf{W} * \mathbf{a}_{l-1} + \mathbf{b}_l). \quad (1)$$

Here, $\mathbf{a}_{l-1} \in \mathbb{R}^n$ and $\mathbf{b} \in \mathbb{R}^m$ denote the activations and biases respectively, $\mathbf{W} \in \mathbb{R}^{m \times n}$ is the weight matrix and $f(\cdot)$ is the Rectified Linear Unit (ReLU) activation function.

a) Dropout: The random dropout layer generates a mask $\mathbf{m} \in \mathbb{B}^m$, where each element m_i is independently sampled from a Bernoulli distribution with a probability ‘ p ’ of being on:

$$m_i \sim \text{Bernoulli}(p), \quad m_i \in \mathbf{m}. \quad (2)$$

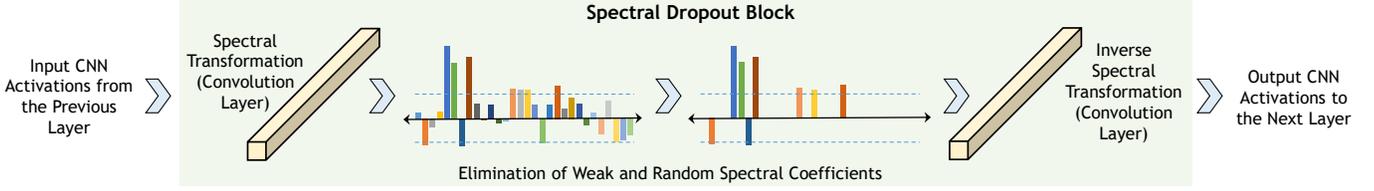


Fig. 2. Spectral Dropout Block with in a CNN enhances network's generalization ability by reducing spectral coefficients.

This mask is used to modify the output activations \mathbf{a}_l :

$$\mathbf{a}_l = \mathbf{m} \circ f(\mathbf{W} * \mathbf{a}_{l-1} + \mathbf{b}_l), \quad (3)$$

where, ' \circ ' denotes the Hadamard product.

b) *Drop-Connect*:: Similar to Dropout, Drop-Connect performs masking out operation on the weight matrix instead of the output activations, therefore:

$$\mathbf{a}_l = f((\mathbf{M} \circ \mathbf{W}) * \mathbf{a}_{l-1} + \mathbf{b}_l), \quad (4)$$

$$M_{i,j} \sim \text{Bernoulli}(p), \quad M_{i,j} \in \mathbf{M}. \quad (5)$$

Next, we describe the proposed spectral dropout approach.

IV. SPECTRAL DROPOUT

The spectral dropout approach ignores the noisy and weak frequency domain coefficients corresponding to activations from a CNN layer (Fig. 2). Consequently, three key benefits are achieved: *First*, it provides an effective way to perform regularization by discouraging co-adaptations of feature detectors. *Second*, the network convergence rate during training is increased, roughly by a factor of two, compared to the regular dropout. This is because a significant portion of strong frequency coefficients are retained for signal reconstruction. *Third*, frequency dropout can be applied after any neural network layer to enhance the network generalization. This in contrast to Dropout and Drop-Connect, which are usually applied to the final FC layers (see Sec. III).

Let us consider $\mathbf{A}_{l-1} \in \mathbb{R}^{h \times w \times n}$ to be a tensor representing output CNN activations from the previous layer. We can represent convolutional filters as $\mathbf{F}_l \in \mathbb{R}^{h' \times w' \times n \times m}$ which operate on \mathbf{A}_{l-1} to give output activations $\mathbf{A}'_l \in \mathbb{R}^{h'' \times w'' \times m}$, as follows:

$$\mathbf{A}'_l = f(\mathbf{F}_l \otimes \mathbf{A}_{l-1} + \mathbf{b}_l). \quad (6)$$

The above expression denotes the normal operation of a convolutional or a FC layer. For the spectral dropout, we need to perform frequency domain transformation followed by the truncation of noisy coefficients and inverse transformation to reconstruct the original CNN activations from the last layer indexed as $(l-1)$. This can be represented as:

$$\mathbf{A}_l = \mathcal{T}^{-1}(\mathbf{M} \circ \mathcal{T}(f(\mathbf{F}_l \otimes \mathbf{A}_{l-1} + \mathbf{b}_l))) \quad (7)$$

Here, \mathcal{T} and \mathcal{T}^{-1} denote the frequency transform and its inverse respectively and $\mathbf{M} \in \mathbb{R}^{h'' \times w'' \times m}$ represents the spectral dropout mask. We next describe the transformation and masking operations in detail.

a) *Spectral Masking*: The mask \mathbf{M} involved in the spectral dropout comprises of both deterministic and stochastic components as follows:

$$M_{i,j,k} \sim \text{Bernoulli}(p') \quad (8)$$

$$p' = \begin{cases} p & \forall \{i, j, k\} : T_{i,j,k} > \tau \\ 0 & \forall \{i, j, k\} : T_{i,j,k} < \tau \end{cases} \quad (9)$$

where, $T_{i,j,k} \in \mathbf{T} = \mathcal{T}(\mathbf{A}'_l)$.

τ is a threshold on the magnitude of the coefficients for frequency dropout. The percentage of activations (η) above a fixed threshold τ can be variable for different input batches. Similarly, the threshold can be adaptive if we want to keep η fixed for different inputs. In other words, either τ or η can be kept unchanged in the following equation:

$$\eta = \frac{\sum_{i,j,k} \mathbb{I}[T_{i,j,k} > \tau]}{h''w''m}. \quad (10)$$

b) *Spectral Transformation*: We use a discrete sinusoidal unitary transform with fixed basis functions, known as the DCT-II [27]. The main motivation of using DCT is that it is a real-valued transform and very fast algorithms are available for its computation. Furthermore, DCT is an orthogonal and separable frequency domain transform which avoids redundancy by performing signal decorrelation. If $\mathbf{a} = \{a_i, : i \in [1, n]\}$ denotes the CNN activations, we can perform 1D forward and inverse DCT as follows:

$$x_k = \alpha(k) \sum_{i=1}^n a_i \beta(i, k), \quad k \in [1, n] \quad (11)$$

$$a_i = \sum_{k=1}^n \alpha(k) x_k \beta(i, k), \quad i \in [1, n] \quad (12)$$

where, α and β are defined as:

$$\alpha(k) = \sqrt{1/n} \mathbb{I}[k=1] + \sqrt{2/n} \mathbb{I}[k \neq 1] \quad (13)$$

$$\beta(i, k) = \cos \left[\frac{\pi(2i-1)(k-1)}{2n} \right]. \quad (14)$$

Here, β denotes the cosine basis functions which are mutually orthogonal. The DCT performs energy compaction and retains most of the signal energy in only a few dominant coefficients.

Similarly, for the convolutional layers, a 2D forward and inverse DCT can be defined as:

$$x_{k,\ell} = \alpha(k)\alpha(\ell) \sum_{i=1}^n \sum_{j=1}^n a_{i,j} \beta(i, k) \beta(j, \ell), \quad (15)$$

$$a_{i,j} = \sum_{k=1}^n \sum_{\ell=1}^n \alpha(k)\alpha(\ell) x_{k,\ell} \beta(i, k) \beta(j, \ell), \quad (16)$$

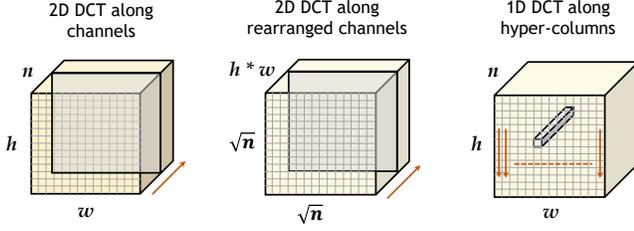


Fig. 3. Spectral transform variants investigated in this work.

where, $\{k, \ell\} \in [1, n]$ and $\{i, j\} \in [1, n]$ in Eq. 15 and Eq. 16, respectively. The 1D and 2D variants are illustrated in Fig. 3.

c) *Frequency Transform as a CNN Layer*: A main bottleneck while introducing frequency domain transformation in the neural network forward and backward signal flow is the computational efficiency. To overcome this, we use a DCT-II transform with fixed basis functions and show that it can be formulated as a convolution operation. This effectively integrates the frequency transformation with-in a regular CNN and requires no modification in the forward and backward signal proration mechanisms during the learning and inference process.

The frequency domain transformation can be applied to the CNN feature maps in three possible ways. (a) Given the feature map \mathbf{A}'_l , a 2D transformation can be applied individually to each channel of the feature map. (b & c) A frequency domain transformation can also be applied to hyper-columns¹ corresponding to each individual spatial location in the feature map. In this case, either a 2D transform can be applied by rearranging each hyper-column $\mathbf{a}'_l \in \mathbb{R}^m$ to form $\sqrt{m} \times \sqrt{m}$ dimensional maps ($h'' \times w''$ in total), or only a 1D transform can be applied to each hyper-column by treating it as a 1D signal.

Specifically, the frequency transform \mathcal{T} can be defined as a CNN convolution layer with filter weights $\mathbf{F}_{\mathcal{T}} \in \mathbb{R}^{h' \times w' \times n \times m}$, such that $h' = w' = 1$ and we assume an equi-dimensional feature representation in the spectral domain i.e., $n = m$. The filter weights are set as follows:

$$\mathbf{F}_{\mathcal{T}} = [\mathbf{v}_1, \dots, \mathbf{v}_m] : \mathbf{v}_i \in \mathbb{R}^m \quad (17)$$

$$\mathbf{v}_i = [\alpha(1)\beta(i, 1), \dots, \alpha(m)\beta(i, m)] \quad (18)$$

for a 1D DCT-II transform and

$$\mathbf{v}_i = \text{vec}(\mathbf{v}'_p \mathbf{v}'_q) : \mathbf{v}'_i \in \mathbb{R}^{\sqrt{m}},$$

$$\mathbf{v}'_i = [\alpha(1)\beta(i, 1), \dots, \alpha(\sqrt{m})\beta(i, \sqrt{m})], \text{ where}$$

$$p = \lceil \frac{i}{\sqrt{m}} \rceil, q' = i - \sqrt{m} \lfloor \frac{i}{\sqrt{m}} \rfloor, \text{ and} \quad (19)$$

$$q = q' \llbracket q' \neq 0 \rrbracket + \sqrt{m} \llbracket q' = 0 \rrbracket, \quad (20)$$

for a 2D DCT-II transform. Similarly for an inverse DCT, we have:

$$\mathbf{F}_{\mathcal{T}^{-1}} = [\hat{\mathbf{v}}_1, \dots, \hat{\mathbf{v}}_m] : \hat{\mathbf{v}}_m \in \mathbb{R}^m, \text{ where} \quad (21)$$

$$\hat{\mathbf{v}}_i = [\alpha(i)\beta(1, i), \dots, \alpha(i)\beta(m, i)] \quad (22)$$

¹We use the term *hypercolumns* to denote tube-vectors (mode-3 vectors) in the tensor \mathbf{A}_{l-1} of CNN activations.

TABLE I
CNN ARCHITECTURES USED FOR EVALUATION AND ANALYSIS OF SPECTRAL DROPOUT APPROACH. IN THE FIRST COLUMN, THE FIRST TERM INSIDE THE BRACKETS SHOW NUMBER OF FILTERS FOR MNIST WHILE THE SECOND TERM SHOWS NUMBER OF FILTERS FOR CIFAR AND SVHN.

LeNet	NiN	ResNet
cnv-5 × 5 (20/32)	cnv-5 × 5 (192)	cnv-3 × 3 (16)
maxpool (↓2)	cnv-1 × 1 (160)	cnv-1 × 1 (16)
cnv-5 × 5 (20/32)	cnv-1 × 1 (96)	cnv-3 × 3 (16)
maxpool (↓2)	maxpool (↓2)	cnv-1 × 1 (64)
cnv-5 × 5 (50/64)	dropout	cnv-1 × 1 (32)
maxpool (↓2)	cnv-5 × 5 (192)	cnv-3 × 3 (32)
cnv-4 × 4 (500/64)	cnv-1 × 1 (192)	cnv-1 × 1 (128)
cnv-1 × 1 (C)	cnv-1 × 1 (192)	cnv-1 × 1 (64)
softmax loss	maxpool (↓2)	cnv-3 × 3 (64)
	dropout	cnv-1 × 1 (256)
	cnv-3 × 3 (192)	avgpool
	cnv-1 × 1 (192)	cnv-1 × 1 (C)
	cnv-1 × 1 (C)	softmax loss
	maxpool (↓7)	
	softmax loss	

for a 1D DCT-II and

$$\hat{\mathbf{v}}_i = \text{vec}(\hat{\mathbf{v}}'_p \hat{\mathbf{v}}'_q) : \hat{\mathbf{v}}'_i \in \mathbb{R}^{\sqrt{m}},$$

$$\hat{\mathbf{v}}'_i = [\alpha(i)\beta(1, i), \dots, \alpha(i)\beta(\sqrt{m}, i)],$$

for a 2D DCT-II transform, where p, q are defined similar to Eq. 19 and Eq. 20 respectively.

In our experiments, a DCT transform of the same dimensions as input is applied for simplicity. Note that the only limitation for a 2D transform along feature channels is that the number of channels should satisfy: $\text{mod}(\sqrt{n}, 1) = 0$.

V. CNN ARCHITECTURES

We experiment with a number of popular CNN architectures to explore the efficacy of the spectral dropout approach. Rather than pushing state of the art, our main goal is to study the learning behaviors and performance trends of the networks when spectral dropout is applied during the train and test phases. Therefore, we focus on only simpler and standard network architectures proposed in the literature ‘as is’, and plug-in our proposed regularization module to clearly demonstrate the performance improvements.

We use three standard CNN architectures namely, (a) LeNet architecture (b) Network in Network (NiN) and the recent (c) Residual Network (ResNet) with pre-activation units. The architectures of these networks are shown in Table I. The LeNet architecture is slightly different from the one proposed in [28], which leads to a lower training error. Each convolution layer in LeNet and NiN is followed by a ReLU layer. For the residual network, each residual unit has a ‘bottleneck’ architecture, as shown inside the braces in Table I. The residual unit is repeated ‘ t ’ times depending on the total weight layers (L) in the network, $t = (L - 2)/9$. For our experiments, we keep $t = 18$ ($L = 164$). With in each residual block, there exist identity short-cut connections from the input to the output of the plain layers, combined together with a sum layer at the end of each residual block. Each weight layer (except the initial

convolution layer) has a pre-activation mechanism consisting of a weight layer preceded by a batch normalization (BN) and a ReLU layer. The pre-activation mechanism leads to a better performance than the original ResNet [29].

As a baseline NiN architecture, we used the one proposed in [30] without two intermediate dropout layers. For experiments involving NiN architecture with Dropout and Drop-Connect regularization, we use these regularizer layers at the same locations as proposed in [30]. Since other network architectures do not have an ideal predefined location for the regularizer, we place these layers at the same location where spectral dropout has been found to give an optimal performance on a validation set. Note that the spectral dropout is plugged in different architectures which sometimes already have a regularization mechanism e.g., Dropout in NiN and Batch Normalization in ResNet. Based on the improvements described in Sec. VI, the spectral dropout contributes to better performance both with and without other regularizers (e.g., in LeNet and NiN respectively).

VI. EXPERIMENTAL ANALYSIS

A. Overview

In all our experiments, we report performances of single networks (LeNet, NiN and ResNet). Note that previous literature reports on the use of different augmentation arrangements for different datasets to obtain an optimal performance. Our goal here is not to establish new state of the art, but to make a fair comparison with other related approaches and demonstrate the performance gain with spectral dropout. Therefore, we do not use any form of data augmentation in our experiments. In principle, the use of data augmentation with spectral dropout should result in better performance.

In Sec. IV, we described two possible ways to perform spectral masking. An almost comparable performance was noted by keeping either of the two variables (τ and η) fixed. However, an adaptive τ required more computational resources. We therefore keep it fixed during our experiments. For each dataset, we used a held-out validation set (20% of the training set) to tune the position of the spectral dropout block, the threshold parameter τ and the learning rates. Similar to [30], we retrain the networks from scratch on the complete training data once these parameters are fixed. Other hyper-parameters were kept same during all experiments e.g., batch size (200), number of epochs (200) and weight decay (5×10^{-4}). A mean image was subtracted from the input images in all cases.

B. MNIST

This dataset contains 70,000 gray-scale images of handwritten digits (0 – 9) with size 28×28 . There are 60,000 images for training and another 10,000 for testing. For consistency of architectures used in our experiments, we resized 28×28 MNIST images to 32×32 to match with the image size available in other datasets.

The LeNet architecture for MNIST is slightly different compared to the one used for CIFAR and SVHN datasets (see Table I). More precisely, the number of convolutional filters in the initial three weight layers is slightly less than the

TABLE II
RESULTS ON MNIST DATASET: ALL PERFORMANCES ARE REPORTED FOR A SINGLE NETWORK WITH NO DATA AUGMENTATION.

Method	Error (%)		
Drop-Connect [8]	0.63		
Stochastic Pooling [12]	0.47		
Maxout Networks [31]	0.45		
Deeply-Supervised Net [32]	0.39		
Model (\rightarrow)	LeNet	NiN	ResNet
Baseline	0.93	0.64	0.55
with Dropout	0.71	0.45	0.44
with Drop-Connect	0.77	0.46	0.42
with Spectral Dropout	0.51	0.40	0.38

TABLE III
COMPARISON OF RESULTS WITH DIFFERENT VARIANTS OF SPECTRAL TRANSFORMATION ON THE MNIST DIGITS DATASET.

Variant	1D-DCT(H)	2D-DCT(H)	2D-DCT(C)
Error (%)	0.52	0.51	0.55

number used for experiments on CIFAR and SVHN datasets with LeNet architecture. This is because the MNIST being a relatively smaller dataset needs less parameters in the initial weight layers.

We report our results and comparisons in Table II. For each baseline architecture, we report performances with Dropout, Drop-Connect, our proposed spectral dropout, and without any of these regularization modules (baseline). We achieve a significant reduction in error on MNIST dataset using the spectral dropout technique, which sets a new state of the art for single model performance using LeNet, NiN and ResNet architectures without any data augmentation.

We also experiment with different variants of spectral transformation on the MNIST dataset (see Table III). We note that although there is minimal difference in performance for different DCT variants, however, 2D DCT along hyper-columns achieve lowest error rate of 0.51% with the LeNet model. Therefore, in our experiments on CIFAR-10 and SVHN datasets, we use a 2D DCT transformation along the output activation hyper-columns.

C. CIFAR-10

The CIFAR-10 dataset contains 60,000 color images of 32×32 for ten object classes. There are 50,000 images for training and another 10,000 for testing.

The results and comparisons are summarized in Table IV. Data whitening and contrast normalization were applied as a preprocessing step. Our approach achieves a consistent boost in performance for all the three network architectures. We note that random dropout was mostly outperformed by the Drop-Connect approach by a small margin on the CIFAR-10 dataset. We do not perform hyperparameter tuning for local receptive field size and weight decay. This is the reason why we achieved slightly lower performance numbers for NiN compared to those reported in [30].

TABLE IV
RESULTS ON THE CIFAR-10 DATASET (USING A SINGLE NETWORK AND NO DATA AUGMENTATION).

Method	Error (%)		
Stochastic Pooling [12]	15.1		
Maxout Networks [31]	11.7		
Drop-Connect (with aug) [8]	11.1		
Deeply-Supervised Net [32]	9.69		
Model (→)	LeNet	NiN	ResNet
Baseline	19.2	15.8	6.02
with Dropout	17.3	11.4	5.97
with Drop-Connect	17.0	10.9	5.99
with Spectral Dropout	16.3	9.14	5.76

TABLE V
RESULTS ON THE SVHN DATASET (USING A SINGLE NETWORK AND NO DATA AUGMENTATION).

Method	Error (%)		
Stochastic Pooling [12]	2.80		
Maxout Networks [31]	2.47		
Drop-Connect [8]	2.23		
Deeply-Supervised Net [32]	1.92		
Model (→)	LeNet	NiN	ResNet
Baseline	4.29	2.74	2.22
with Dropout	4.17	2.72	2.17
with Drop-Connect	4.18	2.72	2.19
with Spectral Dropout	4.05	2.67	2.12

D. SVHN

We experiment on the Street View House Numbers (SVHN) dataset which contains 604,388 training and 26,032 testing images. The images are centered on the digits 0 – 9 (ten classes) in the MNIST-like 32×32 format. But different from MNIST, the dataset contains color images with size and appearance variations along-with presence of distortions especially near the image periphery.

Similar to previous works [12], [30], we perform contrast normalization as a preprocessing step. Note that we obtain a slightly larger error rate on SVHN using NiN architecture because the validation set was used only to tune the learning rates (keeping other hyper parameters same as before). Still, our approach consistently performed better than Dropout and Drop-Connect.

E. Test-time Spectral Dropout

At test time, the deep CNN model only gives point estimates without any uncertainty information. This lack of information about uncertainty makes it difficult to estimate the confidence level of a prediction. Therefore, similar to recent works which use dropout as a tool to obtain uncertainty estimates [33], [34], we apply multiple runs of spectral dropout at test time to study the resulting uncertainty estimates and performance gains. Since spectral dropout incorporates stochastic dropout in the spectral domain, the uncertainty bounds are approximation of the estimates from a Gaussian process [34].

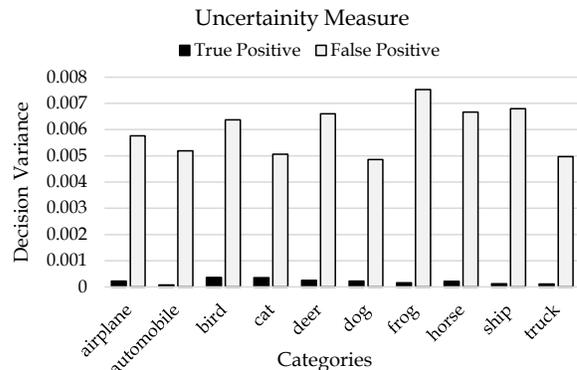


Fig. 4. Plot of uncertainty measures for all categories in the CIFAR-10 dataset.

TABLE VI
EFFECT OF TEST-TIME SPECTRAL DROPOUT ON THE ERROR RATE FOR CIFAR-10 DATASET USING NiN ARCHITECTURE.

Runs	10^0	10^1	10^2	10^3	10^4
Error (%)	9.14	9.11	9.06	9.04	9.03

Let us suppose that we obtain a set of predictions by running T iterations of spectral dropout network with the same input \mathbf{x}_i to obtain $\{\hat{\mathbf{y}}_i^t\} : t \in [1, T]$. The moments of the empirical posterior distribution are useful for our purpose. The final class decisions are established using predictive mean (Eq. 23) and the uncertainty is accounted by variance (Eq. 24), as follows:

$$\mathbf{y}_i = \mathbb{E}[\hat{\mathbf{y}}_i] = \frac{1}{T} \sum_{t=1}^T \hat{\mathbf{y}}_i^t \quad (23)$$

$$\text{var}(\hat{\mathbf{y}}_i) = \frac{1}{T} \sum_{t=1}^T (\hat{\mathbf{y}}_i^t)^T \hat{\mathbf{y}}_i^t - \mathbb{E}[\hat{\mathbf{y}}_i]^T \mathbb{E}[\hat{\mathbf{y}}_i] \quad (24)$$

Figure 4 shows the averaged uncertainty measures of both correct and incorrect predictions for all classes in the CIFAR-10 dataset. The correct predictions are highly confident compared to the incorrect predictions, which are largely uncertain ($10\times$ to $40\times$). Table VI shows the performance improvement with the increasing number of spectral dropout runs for each sample. We found that although the uncertainty estimates are reliable, we get slightly better performance using the majority voting: $\text{mode}(\{\hat{z}_i^1 \dots \hat{z}_i^T\})$ compared with of $z_i = \arg\max_c y_i^c$.

VII. ANALYSIS AND DISCUSSION

A. Positional Analysis

The effect of the position of spectral dropout block within the deep network has been analyzed in Fig. 5. For each of the three CNN architectures, namely LeNet, NiN and ResNet, we report the performance trend on the validation set versus the position of spectral dropout module. We note that the best results are obtained when the spectral dropout is applied at the intermediate levels of abstraction. To emphasize this trend, we fit a second degree polynomial on the achieved performances. This is in contrast to the regular Dropout approach, which

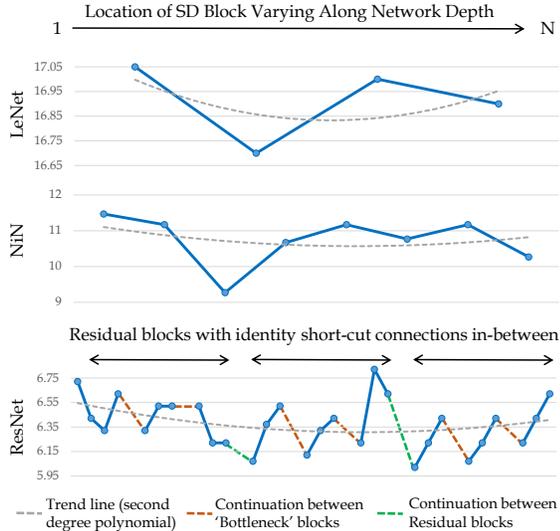


Fig. 5. Performance trend with different locations of SD layer on the CIFAR-10 dataset.

is usually applied at the end of a deep network (adjacent to fully connected layers). Specifically, for the case of ResNet, we observed a decrease in performance when the position of spectral dropout module was shifted from the first layer in the bottleneck block towards the last layer. Note that for each residual block, we analyzed error rate by plugging in the spectral dropout module after first, middle and the last bottleneck blocks (separated by dotted line). The best performance was observed when the spectral dropout was applied after the first convolution layer in the last residual block. Note that all the module positions in Fig. 5 are on the highway signal path. We also experimented with spectral dropout block in the short-cut connections within a residual network. However, this resulted in a decrease in the overall performance. This observation is consistent with the literature [29], where it has been found that modifying the identity connections results in performance degradation.

Furthermore, we also experimented with *multiple* spectral dropout blocks within each of the LeNet, NiN and ResNet architecture, which did not provide much performance gain while increasing the network convergence time and computational requirements.

B. Computational Complexity

The spectral dropout block comprises of two convolutional layers, which adds computational overhead of $\mathcal{O}(2h''w''nm)$. However, it is important to note that vector multiplication is implemented as an efficient block matrix multiplication routine in BLAS library, which is faster than directly computing FFT and IFFT on a CPU. On the CIFAR-10 dataset with a batch size of 200, using a core i7 machine with a NVIDIA Titan-X 12GB card and 16GB RAM, it takes $127.9\mu s$ and $44.7\mu s$ respectively to process one image with a spectral dropout CNN during the train and test phases. In comparison, a baseline CNN takes $110.3\mu s$ and $41.0\mu s$, while a dropout CNN takes $125.9\mu s$ and $41.2\mu s$ during the train and test phase, respectively.

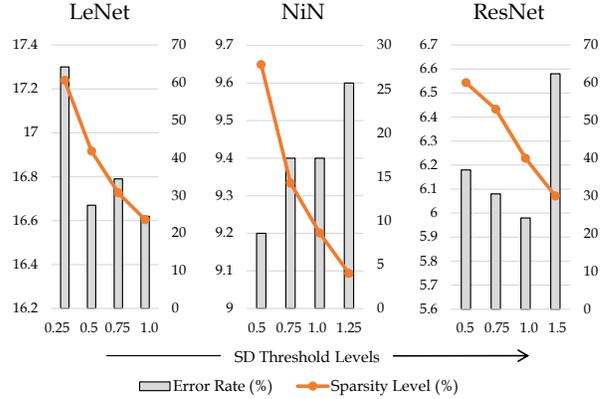


Fig. 6. Performance and sparsity trend for different thresholds of SD on the CIFAR-10 dataset. The axis scale on left denotes error rate while the right scale represents the percentage of retained neural activations.

C. Sparsity Analysis

We analyze the impact of spectral dropout threshold on the percentage of pruned neural activations and the corresponding performance levels. Figure 6 summarizes the trend on CIFAR-10 dataset with LeNet, NiN and ResNet architectures. The analysis is performed on a validation set similar to the positional analysis. The comparison is shown for the best SD position in each network as identified in Sec. VII-A. We note that while the performance is sensitive to the percentage of pruned activations, there exists a consistency in the activation levels for different inputs and network architectures. For the LeNet and NiN architectures, the best performance was achieved when $\sim 70\%$ of the activations were pruned. For the ResNet architecture, we noted the best performance at a slightly lower pruning rate i.e., $\sim 60\%$. Due to this consistency, we found that setting a fixed activation threshold level performs identical to a threshold on the sparsity level, but with considerable gains in computational efficiency. It is also interesting to note that we apply random dropout with a relatively low pruning probability (20%) on top of the threshold based pruning on spectral activations. Therefore, with much high sparsity levels ($\sim 70 - 80\%$) compared to random dropout (which normally deactivates 50% of the neurons), we are able to achieve lower error rates with much faster convergence during the network training.

D. Effect on Convergence Time

The spectral dropout speeds up the convergence rate during the training process. In Fig. 7, top-1 and top-5 error rates for the CIFAR-10 dataset are shown. Note that the baseline model converges rapidly, but does not generalize well to the unseen data. Compared to the Dropout, spectral dropout not only achieves better performance but also enhances the convergence speed. Similar convergence behavior can be seen for the MNIST dataset in Fig. 1.

E. Power Analysis

We study the effect of different spectral dropout thresholds on the retained power of the feature vectors propagating

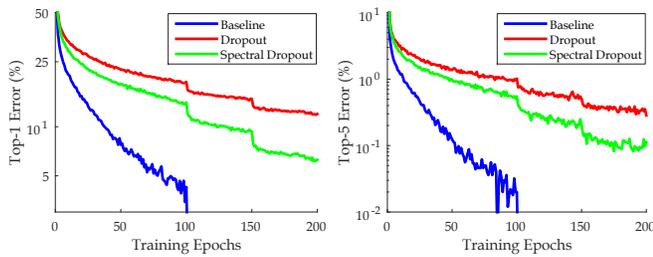


Fig. 7. Network convergence rate during training.

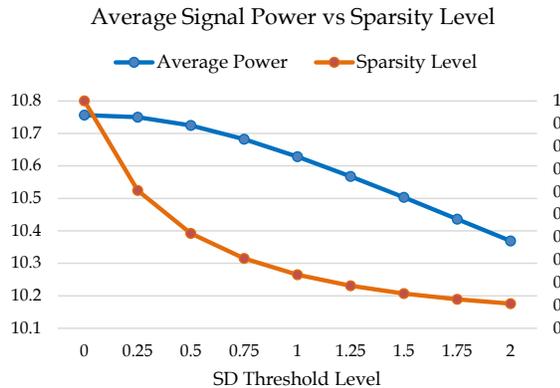


Fig. 8. Analyzing the effect of sparsity level on the signal power. The axis scale on left shows average power, while the scale on right indicates percentage of retained neural activations.

through the network (see Fig. 8). Since the signal power and its energy are directly proportional, this analysis holds true for energy as well. With the increase in the percentage of pruned neurons, we notice a consistent decline in signal power. However, this decline is not significant when compared to original power, i.e., only a 3.6% drop in signal power as a result of 90% pruning. This explains why the network convergence rate is relatively higher compared to the Dropout approach.

VIII. CONCLUSION

We introduced a new approach to perform regularization in the deep neural networks in an efficient manner. The proposed approach uses spectral domain transformation to identify and ignore weak frequency coefficients such that the co-adaptations of the feature detectors are avoided. Furthermore, we show that the spectral domain transformation can be formulated as a convolution operation, thus enabling computational efficiency and an end-to-end trainable network. Our results demonstrate a superior performance compared to other regularization methods and baseline approaches on a range of popular CNN architectures and image classification datasets.

REFERENCES

- [1] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.
- [2] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis," in *ICDAR*, vol. 3, 2003, pp. 958–962.
- [3] G. B. Orr and K.-R. Müller, *Neural networks: tricks of the trade*. Springer, 2003.
- [4] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, 2005.
- [5] J. Moody, S. Hanson, A. Krogh, and J. A. Hertz, "A simple weight decay can improve generalization," *Advances in neural information processing systems*, vol. 4, pp. 950–957, 1995.
- [6] N. Morgan and H. Bourlard, *Generalization and parameter estimation in feedforward nets: Some experiments*. International Computer Science Institute, 1989.
- [7] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.
- [8] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, 2013, pp. 1058–1066.
- [9] M. Hayat, S. H. Khan, M. Bennamoun, and S. An, "A spatial layout and scale invariant feature representation for indoor scene classification," *IEEE Transactions on Image Processing*, vol. 25, no. 10, pp. 4829–4841, 2016.
- [10] B. Graham, "Fractional max-pooling," *arXiv preprint arXiv:1412.6071*, 2014.
- [11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.
- [12] M. Zeiler and R. Fergus, "Stochastic pooling for regularization of deep convolutional neural networks," in *Proceedings of the International Conference on Learning Representation (ICLR)*, 2013.
- [13] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1026–1034.
- [14] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, 2015, pp. 448–456.
- [15] A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009.
- [16] S. Wang and C. Manning, "Fast dropout training," in *Proceedings of The 30th International Conference on Machine Learning*, 2013, pp. 118–126.
- [17] H. Zhou, J. Alvarez, and F. Porikli, "Less is more: Towards compact cnns," in *Proceedings of the European Conference on Computer Vision*, 2016.
- [18] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *European Conference on Computer Vision*. Springer, 2016, pp. 646–661.
- [19] S. Ben-Yacoub, B. Fasel, and J. Luetttin, "Fast face detection using mlp and fft," in *Proc. Second International Conference on Audio and Video-based Biometric Person Authentication (AVBPA'99)*, no. EPFL-CONF-82563, 1999, pp. 31–36.
- [20] M. Mathieu, M. Henaff, and Y. LeCun, "Fast training of convolutional networks through ffts," *arXiv preprint arXiv:1312.5851*, 2013.
- [21] A. Lavin, "Fast algorithms for convolutional neural networks," *arXiv preprint arXiv:1509.09308*, 2015.
- [22] N. Vasilache, J. Johnson, M. Mathieu, S. Chintala, S. Piantino, and Y. LeCun, "Fast convolutional nets with fbfft: A gpu performance evaluation," *arXiv preprint arXiv:1412.7580*, 2014.
- [23] V. Sindhwani, T. Sainath, and S. Kumar, "Structured transforms for small-footprint deep learning," in *Advances in Neural Information Processing Systems*, 2015, pp. 3088–3096.
- [24] W. Chen, J. Wilson, S. Tyree, K. Weinberger, and Y. Chen, "Compressing neural networks with the hashing trick," in *International Conference on Machine Learning*, 2015, pp. 2285–2294.
- [25] Y. Cheng, F. X. Yu, R. S. Feris, S. Kumar, A. Choudhary, and S.-F. Chang, "An exploration of parameter redundancy in deep networks with circulant projections," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2857–2865.
- [26] O. Rippel, J. Snoek, and R. P. Adams, "Spectral representations for convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 2449–2457.
- [27] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE transactions on Computers*, vol. 100, no. 1, pp. 90–93, 1974.

- [28] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [29] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," *arXiv preprint arXiv:1603.05027*, 2016.
- [30] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.
- [31] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *Proceedings of The 30th International Conference on Machine Learning*, 2013, pp. 1319–1327.
- [32] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "{Deeply-Supervised Nets}," in *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, 2015, pp. 562–570.
- [33] A. Kendall, V. Badrinarayanan, and R. Cipolla, "Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding," *arXiv preprint arXiv:1511.02680*, 2015.
- [34] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*, 2016, pp. 1050–1059.