On the Regularization Properties of Structured Dropout

Ambar Pal Connor Lane René Vidal Benjamin D. Haeffele
Mathematical Institute for Data Science, Johns Hopkins University, Baltimore, MD
{ambar, clane, rvidal, bhaeffele}@jhu.edu

Abstract

*Dropout and its extensions (*e.g. *DropBlock and DropConnect) are popular heuristics for training neural networks, which have been shown to improve generalization performance in practice. However, a theoretical understanding of their optimization and regularization properties remains elusive. Recent work shows that in the case of single hidden-layer linear networks, Dropout is a stochastic gradient descent method for minimizing a regularized loss, and that the regularizer induces solutions that are low-rank and balanced. In this work we show that for single hidden-layer linear networks, DropBlock induces spectral k-support norm regularization, and promotes solutions that are low-rank and have factors with equal norm. We also show that the global minimizer for DropBlock can be computed in closed form, and that DropConnect is equivalent to Dropout. We then show that some of these results can be extended to a general class of Dropout-strategies, and, with some assumptions, to deep non-linear networks when Dropout is applied to the last layer. We verify our theoretical claims and assumptions experimentally with commonly used network architectures.*

1. Introduction

Dropout is a widely-used heuristic for training deep neural networks (NN), which involves setting to zero the output of a random subset of hidden neurons at each training iteration. The improved generalization performance of Dropout in practice has led to many variants of dropout [3, 4, 5, 9], [10, 12, 13, 14]. However, despite the popularity and improved empirical performance of Dropout-style techniques, several theoretical questions remain regarding their optimization and regularization properties, *e.g.*: What objective function is minimized by general Dropout-style techniques? Do these techniques converge to a global minimum? Does Dropout-style regularization induce an explicit regularizer? What is the inductive bias of Dropout-style regularization?

Related Work. Recent work has considered some of these questions in the case of single-layer linear neural networks trained with the squared loss. For example, [2] shows that Dropout is a stochastic gradient descent (SGD) method for minimizing the following objective:

$$\min_{\mathbf{U},\mathbf{V}} \mathbb{E}_{\mathbf{z}} \left\| \mathbf{Y} - \frac{1}{\theta} \mathbf{U} \mathrm{diag}(\mathbf{z}) \mathbf{V}^\top \mathbf{X} \right\|_{\mathrm{F}}^2. \tag{1}$$

Here $\mathbf{X} \in \mathbb{R}^{b \times N}$ and $\mathbf{Y} \in \mathbb{R}^{a \times N}$ denote a training set with N training points, $\mathbf{U} \in \mathbb{R}^{a \times d}$ and $\mathbf{V} \in \mathbb{R}^{b \times d}$ are the output and input weight matrices, respectively, d is number of hidden neurons, and $\mathbf{z}$ is a d-dimensional vector of Dropout variables whose ith entry $z_i \sim \mathrm{Ber}(\theta)$ is i.i.d. Bernoulli with parameter θ. In addition, [2] shows that Dropout induces explicit regularization in the form of a squared nuclear norm, which is known to induce low-rank solutions. Specifically, [2] shows that the optimization problem (1) reduces to

$$\min_{\mathbf{U},\mathbf{V}} \|\mathbf{Y} - \mathbf{U}\mathbf{V}^\top \mathbf{X}\|_F^2 + \frac{1-\theta}{\theta} \sum_{i=1}^{d} \|\mathbf{u}_i\|_2^2 \|\mathbf{X}^\top \mathbf{v}_i\|_2^2, \tag{2}$$

where $\mathbf{u}_i$ and $\mathbf{v}_i$ denote the i^{th} columns of $\mathbf{U}$ and $\mathbf{V}$, resp. Moreover, [2] shows that a global minimum $(\mathbf{U}^*, \mathbf{V}^*)$ of (2) yields a global minimum of $\min_{\mathbf{Z}} \|\mathbf{Y} - \mathbf{Z}\|_F^2 + \lambda \|\mathbf{Z}\|_*^2$, where $\mathbf{Z}^* = \mathbf{U}^* \mathbf{V}^{*\top} \mathbf{X}$ and $\|\mathbf{Z}\|_*$ is the nuclear norm. In addition, [8] shows that the optimal weights $(\mathbf{U}^*, \mathbf{V}^*)$ can be found in polynomial time and are balanced, i.e., the product of the norms of incoming and outgoing weights, $\|\mathbf{u}_i\|_2 \|\mathbf{X}^\top \mathbf{v}_i\|_2$, is the same for all neurons.

Paper Contributions. In this paper, we significantly generalize these results to more general Dropout schemes and more general neural network architectures. We first study DropBlock, an alternative to Dropout for convolutional networks which was recently proposed in [5] and displays improved performance compared to Dropout in practice. Instead of zeroing the output of each neuron independently, DropBlock introduces a structural dropping pattern by zeroing a block of neurons within a local neighborhood together to reflect the strong correlations in responses for neighboring pixels in a CNN. Specifically, for a block-size r, we will look at the following optimization problem:

$$\min_{\mathbf{U},\mathbf{V}} \mathbb{E}_{\mathbf{w}} \left\| \mathbf{Y} - \frac{1}{\theta} \mathbf{U} (\mathrm{diag}(\mathbf{w}) \otimes \mathbf{I}_{\frac{d}{r}}) \mathbf{V}^\top \mathbf{X} \right\|_{\mathrm{F}}^2, \tag{3}$$

where $\otimes$ denotes the Kronecker product and $\mathbf{w}$ are the stochastic Bernoulli variables with one entry $w_i \sim \text{Ber}(\theta)$ getting applied simultaneously to a block of columns in $(\mathbf{U}, \mathbf{V})$ of size r. In this paper, we study the regularization properties of DropBlock, and show that it induces low-rank regularization in the form of a k-support norm on the singular values of the solution, which is known to have some favorable properties compared to the ℓ_1-norm [1]. This provides a step towards explaining the superior performance of DropBlock in practice, as compared to Dropout, which induces an ℓ_1-norm on the singular values. In this paper we also study the properties of the optimal solutions induced by DropBlock. Specifically, we prove that the solutions to (3) are such that the norms of the *factors* are balanced, *i.e.* products of corresponding blocks of r columns of $\mathbf{U}$ and $\mathbf{V}$ have equal Frobenius norms. Combining these results will allow us to get a closed form solution to (3).

We then extend our analysis to more general dropping strategies that allow for arbitrary sampling distributions for the Dropout variables and obtain the explicit regularizer for this general case. We also extend our analysis to Dropout applied to the last layer of a deep neural network and show that this as well as many existing results in the literature can be readily extended to this scenario. We end with a short result on an equivalence between Dropout and DropConnect, which is a different way of performing Dropout.

Finally, various experiments are used to validate the theoretical results and assumptions whenever necessary.

2. DropBlock Analysis

In this section, we study the optimization and regularization properties of DropBlock, a variant of Dropout where blocks of neurons are dropped together. In this setting, we let d be the final hidden layer dimension and let r be the size of the block that is dropped. We make the simplifying assumption that the blocks form a partition of the neurons in the final hidden layer, which requires the hidden dimension d to be a multiple of r. This is a minor assumption when $d \gg r$, which is typically satisfied.[1] Then at each iteration, we sample a binary vector of $k = \frac{d}{r}$ i.i.d. $\text{Ber}(\theta)$ random variables $\mathbf{w} \in \{0,1\}^k$ and set the corresponding block of variables in $\mathbf{z} \in \{0,1\}^d$ to the value of w_i, i.e., $z_j = w_i$ for $(i-1)r < j \leq ir$. This sampling scheme, which we refer to as $\texttt{DropBlockSample}(\theta, r)$, captures the key principle behind DropBlock by dropping a block of neighboring neurons at a time and is a very close approximation of DropBlock (which does not assume the blocks need to be non-overlapping) when $d \gg r$. The resulting DropBlock algorithm that we will study is specified in Algorithm 1. Note that Dropout can be obtained as a particular case of DropBlock when the block size is set to $r = 1$.

[1] We show experimentally in Section 5.3 that a scaling of θ suffices to make this approximation behave largely identical to the original DropBlock strategy.

Algorithm 1: Dropblock Algorithm

1: **Input:** Training Data $\mathcal{D} = \{\mathbf{x}_t, \mathbf{y}_t\}$, Learning Rate η, Retain Probability θ, Block Size r
2: **Output:** Final Iterates $\mathbf{U}_T, \mathbf{V}_T$
3: $\mathbf{U}_0 \leftarrow \mathbf{U}_{\text{init}}$, $\mathbf{V}_0 \leftarrow \mathbf{V}_{\text{init}}$
4: **for** $t = 1, \ldots, T$ **do**
5: $\quad \mathbf{z}_{t-1} \leftarrow \texttt{DropBlockSample}(\theta, r)$
6: $\quad \mathbf{D}_{\mathbf{z}} \leftarrow \text{diag}(\mathbf{z}_{t-1})$
7: $\quad$ Error, $\epsilon_t \leftarrow \left(\left(\frac{1}{\theta}\mathbf{U}_{t-1}\mathbf{D}_{\mathbf{z}}\mathbf{V}_{t-1}^\top \mathbf{x}_t\right) - \mathbf{y}_t\right)$
8: $\quad \mathbf{U}_t \leftarrow \mathbf{U}_{t-1} - \frac{\eta}{\theta}\epsilon_t \mathbf{x}_t^\top \mathbf{V}_{t-1}\mathbf{D}_{\mathbf{z}}$
9: $\quad \mathbf{V}_t \leftarrow \mathbf{V}_{t-1} - \frac{\eta}{\theta}\mathbf{x}_t \epsilon_t^\top \mathbf{U}_{t-1}\mathbf{D}_{\mathbf{z}}$
10: **end for**

Analysis Technique. Before presenting the details of our analysis in the subsequent subsections, we pause and comment on the analysis approach at a high level. Our goal is to understand the regularization induced by DropBlock training, i.e. Algorithm 1. We begin in Section 2.1 by observing that DropBlock training is equivalent to training the original un-regularized network with an additional regularization term. We continue in Section 2.2 by analyzing what happens to this regularizer when the network width is allowed to grow arbitrarily, and observe that the regularizer value goes to 0, hence providing no regularization to the problem. To address this issue, in Sec 2.3 we adaptively scale the dropout rate as a function of the network width and show that this induces a modified optimization problem whose optimal weights are balanced. These results are used in Sec 2.4 to obtain a convex lower bound to the optimization objective, which is shown to be tight, hence allowing us to relate the solutions of the convex lower bound to the solutions of the original objective of interest. Finally, we obtain a closed form solution to the convex lower bound, which additionally allows us to characterize solutions to the non-convex DropBlock optimization problem in closed form.

2.1. Regularizer Induced by DropBlock

We first show that the DropBlock Algorithm 1 can be interpreted as applying SGD to the objective in (3). To that end, recall that the gradient of the expected value is equal to the expected value of the gradient. Thus, the gradient of $\|\mathbf{Y} - \frac{1}{\theta}\mathbf{U}(\text{diag}(\mathbf{w}) \otimes \mathbf{I}_{\frac{d}{r}})\mathbf{V}^\top \mathbf{X}\|_{\text{F}}^2$ with respect to $\mathbf{U}$ and $\mathbf{V}$ for a random sample of $\mathbf{w}$ provides a stochastic gradient for the objective in (3). Steps 8 and 9 of Algorithm 1 compute such gradients. Therefore, we conclude that the DropBlock Algorithm 1 is a SGD method for minimizing (3).

The next step is to understand the regularization properties of DropBlock. The following Lemma[2] shows that the Dropblock optimization problem is equivalent to a deterministic formulation with a regularization term, which we

[2] Proofs of all our results are given in the Supplementary Material.

denote by $\Omega_{\text{DropBlock}}$. That is, DropBlock induces explicit regularization.

Lemma 1. *The stochastic DropBlock objective (3) is equivalent to a regularized deterministic objective:*

$$\mathbb{E}_{\mathbf{w}} \left\| \mathbf{Y} - \frac{1}{\theta}\mathbf{U}(\text{diag}(\mathbf{w}) \otimes \mathbf{I}_{\frac{d}{r}})\mathbf{V}^\top \mathbf{X} \right\|_{\text{F}}^2 = \|\mathbf{Y} - \mathbf{U}\mathbf{V}^\top \mathbf{X}\|_{\text{F}}^2 + \Omega_{\text{DropBlock}}(\mathbf{U}, \mathbf{X}^\top \mathbf{V}), \quad (4)$$

where $\Omega_{\text{DropBlock}}$ is given by

$$\Omega_{\text{DropBlock}}(\mathbf{U}, \mathbf{X}^\top \mathbf{V}) = \frac{1-\theta}{\theta} \sum_{i=1}^{k} \|\mathbf{U}_i \mathbf{V}_i^\top \mathbf{X}\|_{\text{F}}^2 \quad (5)$$

with $\mathbf{U}_i \in \mathbb{R}^{a \times r}$ and $\mathbf{V}_i \in \mathbb{R}^{b \times r}$ denoting the i^{th} blocks of r consecutive columns in $\mathbf{U}$ and $\mathbf{V}$ respectively and $k = \frac{d}{r}$ denoting the number of blocks.

As expected, when we set $r = 1$, *i.e.* when we drop blocks of 1 neuron independently, $\Omega_{\text{DropBlock}}$ reduces to Dropout regularization in (2). Therefore, DropBlock regularization generalizes Dropout regularization in (2) by taking the sum over the squared Frobenius norms of rank-r submatrices. But what is the effect of this modification? Specifically, can we characterize the regularization properties of $\Omega_{\text{DropBlock}}$, and how it controls the capacity of the network?

2.2. Capacity Control Property of DropBlock

In this subsection we first study whether DropBlock is capable of constraining the capacity of the network alone. That is, if the network were allowed to be made arbitrarily large, is DropBlock regularization sufficient to constrain the capacity of the network?

It is clear from the definition of $\Omega_{\text{DropBlock}}$ that for any non-zero $(\mathbf{U}, \mathbf{V})$ the regularizer will be strictly positive. However, it is not clear if the regularizer increases with d. The following Lemma shows that when the Dropout probability, $1-\theta$, is constant with respect to d, DropBlock alone cannot constrain the capacity of the network, because for any output $\mathbf{A}$ one can find a factorization into $\mathbf{U}\mathbf{V}^\top\mathbf{X}$ that makes $\Omega_{\text{DropBlock}}$ arbitrarily small (approaching 0 in the limit) by making the width d of the final layer large enough.

Lemma 2. *Given any matrix $\mathbf{A}$, if the number of columns, d, in $(\mathbf{U}, \mathbf{V})$ is allowed to vary, with θ held constant, then*

$$\inf_{d} \inf_{\substack{\mathbf{U} \in \mathbb{R}^{m \times d}, \mathbf{V} \in \mathbb{R}^{n \times d} \\ \mathbf{A} = \mathbf{U}\mathbf{V}^\top \mathbf{X}}} \Omega_{\text{DropBlock}}(\mathbf{U}, \mathbf{X}^\top \mathbf{V}) = 0. \quad (6)$$

Note that this result is also true for regular Dropout (a special case of DropBlock) with a fixed Dropout probability.

In what follows, we show that if the Dropout probability, $1-\theta$, increases with d, then DropBlock is capable of constraining the network capacity. Specifically, let us denote the retain probability for dimension d as:

$$\theta(d) = \frac{\bar{\theta} r}{\bar{\theta} r + (1-\bar{\theta}) d}, \quad (7)$$

where $\bar{\theta} = \theta(r)$ denotes the value of the DropBlock parameter when there is only one block, and $d = r$. With $\theta = \theta(d)$, Lemma 1 gives us the following deterministic equivalent of the DropBlock objective:

$$f(\mathbf{U}, \mathbf{V}, d) = \|\mathbf{Y} - \mathbf{U}\mathbf{V}^\top \mathbf{X}\|_{\text{F}}^2 + \frac{d}{r}\frac{1-\bar{\theta}}{\bar{\theta}} \sum_{i=1}^{k} \|\mathbf{U}_i \mathbf{V}_i^\top \mathbf{X}\|_{\text{F}}^2. \quad (8)$$

In order to study the minimizers of $f(\mathbf{U}, \mathbf{V}, d)$, note that at any minimizer $(\mathbf{U}^*, \mathbf{V}^*, d^*)$ we would have the following:

$$f(\mathbf{U}^*, \mathbf{V}^*, d^*) = \|\mathbf{Y} - \mathbf{U}^*\mathbf{V}^{*\top}\mathbf{X}\|_{\text{F}}^2 + \inf_{d} \inf_{\substack{\mathbf{U} \in \mathbb{R}^{a \times d}, \mathbf{V} \in \mathbb{R}^{b \times d} \\ \mathbf{U}\mathbf{V}^\top \mathbf{X} = \mathbf{U}^*\mathbf{V}^{*\top}\mathbf{X}}} \frac{d}{r}\frac{1-\bar{\theta}}{\bar{\theta}} \sum_{i=1}^{k} \|\mathbf{U}_i \mathbf{V}_i^\top \mathbf{X}\|_{\text{F}}^2 \quad (9)$$

where the last term denotes the fact that given the global minimizer of the matrix product $(\mathbf{U}^*\mathbf{V}^{*\top}\mathbf{X})$ the regularization induced by DropBlock will induce factors $(\mathbf{U}^*, \mathbf{V}^*)$ which minimize the induced regularization term $\Omega_{\text{DropBlock}}$.

This motivates a study of the regularization induced by DropBlock in the product-space, which we denote as $\Lambda(\mathbf{A})$:

$$\Lambda(\mathbf{A}) = \frac{1-\bar{\theta}}{\bar{\theta}} \inf_{d} \inf_{\substack{\mathbf{U} \in \mathbb{R}^{a \times d}, \mathbf{V} \in \mathbb{R}^{b \times d} \\ \mathbf{A} = \mathbf{U}\mathbf{V}^\top \mathbf{X}}} \frac{d}{r} \sum_{i=1}^{k} \|\mathbf{U}_i \mathbf{V}_i^\top \mathbf{X}\|_{\text{F}}^2. \quad (10)$$

By the definition of $\Lambda(\mathbf{A})$ in (10), one can define a function

$$\bar{F}(\mathbf{A}) = \|\mathbf{Y} - \mathbf{A}\|_F^2 + \Lambda(\mathbf{A}) \quad (11)$$

that globally lower bounds $f(\mathbf{U}, \mathbf{V}, d)$, i.e.,

$$\bar{F}(\mathbf{A}) \leq f(\mathbf{U}, \mathbf{V}, d), \forall (\mathbf{U}, \mathbf{V}, \mathbf{A}) \text{ s.t. } \mathbf{U}\mathbf{V}^\top \mathbf{X} = \mathbf{A} \quad (12)$$

with equality for $(\mathbf{U}, \mathbf{V}, d)$ that achieve the infimum in (10). As a result, $\bar{F}(\mathbf{A})$ provides a useful analysis tool to study the properties of solutions to the problem of interest $f(\mathbf{U}, \mathbf{V}, d)$ as it provides a lower bound to our problem of interest in the output space (i.e., $\mathbf{U}\mathbf{V}^\top\mathbf{X}$).

While it is simple to see that $\bar{F}(\mathbf{A})$ is a lower bound of our problem of interest, it is not clear whether $\bar{F}(\mathbf{A})$ is a useful lower bound or whether the minimizers to $\bar{F}$ can characterize minimizers of f. In the following analysis, we will prove that the answer to both questions is positive. That is, we will show that $\bar{F}(\mathbf{A})$ is a tight convex lower bound of f, generalizing existing results in the literature [2, 8], and that minimizers of $\bar{F}(\mathbf{A})$ can be computed in closed form.

2.3. DropBlock Induces Balanced Weights

In order to characterize the minimizers of $f(\mathbf{U}, \mathbf{V}, d)$, we first need to define the notion of *balanced factors*.

Definition 3. *A matrix pair $(\mathbf{U}, \mathbf{V})$ is called **balanced** if the norms of the products of the corresponding blocks of $\mathbf{U}$ and $\mathbf{V}$ are equal, i.e., $\|\mathbf{U}_1\mathbf{V}_1^\top\mathbf{X}\|_\mathrm{F} = \|\mathbf{U}_2\mathbf{V}_2^\top\mathbf{X}\|_\mathrm{F} = \ldots = \|\mathbf{U}_k\mathbf{V}_k^\top\mathbf{X}\|_\mathrm{F}$, where $\mathbf{U}_i$ and $\mathbf{V}_i$ denote the i^{th} blocks of r consecutive columns in $\mathbf{U}$ and $\mathbf{V}$ respectively.*

The following result shows that all minimizers of $f(\mathbf{U}, \mathbf{V}, d)$ are balanced.

Theorem 4. *If $(\mathbf{U}^*, \mathbf{V}^*, d^*)$ is a minimizer of (8), then $(\mathbf{U}^*, \mathbf{V}^*)$ is balanced.*

Theorem 4 provides a characterisation of the minimizers of the DropBlock objective (8), saying that all the summands in the regulariser are equal at optimality. With this result, we will be able to link the minimizers of f and $\bar{F}$, and hence find the regularization induced by DropBlock.

We now note some connections to recent literature. Our result generalizes the balancing result obtained in [8], which corresponds to the particular case $k = 1$. However, our proof technique is radically different. The proof in [8] exploits the rank-1 structure to show the existence of an orthonormal matrix $\mathbf{Q}$ such that a given factorization $(\mathbf{U}, \mathbf{V}, d)$ can be transformed to a balanced transformation $(\mathbf{UQ}, \mathbf{VQ}, d)$. In contrast, the intuition behind our proof is that when $(\mathbf{U}, \mathbf{V})$ are not balanced, we can add additional, duplicate blocks of neurons in a particular way to make the block-product-norms $\|\mathbf{U}_i\mathbf{V}_i^\top\mathbf{X}\|_\mathrm{F}$ more balanced, reducing the objective.

Having shown this necessary condition for solutions to $f(\mathbf{U}, \mathbf{V}, d)$, we now use this result to show that $\bar{F}$ is a tight lower bound of (8).

Theorem 5. *If $(\mathbf{U}^*, \mathbf{V}^*, d^*)$ is a global minimizer of the factorized problem f, then $\mathbf{A}^* = \mathbf{U}^*\mathbf{V}^{*\top}\mathbf{X}$ is a global minimizer of the lower bound $\bar{F}$. Furthermore, the lower bound is tight, i.e. we have $f(\mathbf{U}^*, \mathbf{V}^*, d^*) = \bar{F}(\mathbf{A}^*)$.*

Theorem 5 provides a link between the hard non-convex problem of interest, f, and the lower bound, $\bar{F}$, and gives us a guarantee that we can verify solutions to f by showing they are solutions to $\bar{F}$. Hence, we now focus our attention on characterizing solutions of $\bar{F}(\mathbf{A})$.

2.4. DropBlock Induces k-support Norm Regularization

Based on the above discussion, we now analyze the global minimizers of $\bar{F}(\mathbf{A})$. Unfortunately, it is not clear yet whether $\Lambda(\mathbf{A})$ is convex w.r.t. $\mathbf{A}$, which complicates the analysis of the global minimizers of $\bar{F}(\mathbf{A})$. Therefore, we will consider instead the lower convex envelope of $\Lambda(\mathbf{A})$, $\Lambda^{**}(\mathbf{A})$, and show that it gives a tight bound to the problem in (8). Furthermore, we will show that $\Lambda(\mathbf{A})$ (and by extension $\bar{F}(\mathbf{A})$) is indeed convex by showing that $\Lambda(\mathbf{A}) = \Lambda^{**}(\mathbf{A})$, $\forall \mathbf{A}$.

First, recall that the *lower convex envelope* [11] of a function $h(x)$ is the largest convex function $g(x)$ such that $\forall x\ g(x) \leq h(x)$, and is given by the Fenchel double dual (i.e., the Fenchel dual of the Fenchel dual). For $\Lambda(\mathbf{A})$, the following result provides the lower convex envelope. Note that in this sub-section, we will assume that $\mathbf{X}$ has full column rank. This is typically a minor assumption since if $\mathbf{X}$ is not full rank adding a very small amount of noise will make $\mathbf{X}$ full rank.

Theorem 6. *When $\mathbf{X}$ has full column rank in (10), the lower convex envelope of the DropBlock regularizer $\Lambda(\mathbf{A})$ in (10) is given by*

$$\Lambda^{**}(\mathbf{A}) = \frac{1-\bar{\theta}}{\bar{\theta}} \left(\sum_{i=1}^{\rho^*-1} a_i^2 + \frac{(\sum_{i=\rho^*}^{d} a_i)^2}{r - \rho^* + 1} \right), \qquad (13)$$

where ρ^* is the integer in $\{1, 2, \ldots, r\}$ that maximizes (13), and $a_1 \geq a_2 \ldots \geq a_d$ are the singular values of $\mathbf{A}$.

Note that the quantity ρ^* mentioned in (13) is purely a property of the matrix $\mathbf{A}$, the hidden dimension d and the block size r, and is determined completely in time $d \log d$, given an SVD of $\mathbf{A}$.

We again note some connections to recent literature. The form of the solution (13) is particularly interesting because it is a matrix norm that has recently been discovered in the sparse prediction literature by [1], where it is called the k-Support Norm and provides the tightest convex relaxation of sparsity combined with an ℓ_2 penalty. When applied to the singular values of a matrix (as is the case here), it is called the Spectral k-Support Norm, as studied recently in [7].

Properties of the k-Support Norm. We are often interested in obtaining sparse or low-rank solutions to problems, as they have been shown to generalize well and are useful in discarding irrelevant features. Specifically, if we are learning a vector w, we can get sparse solutions by constraining the ℓ_0 *norm* of w, that is the number of non-zero entries in w. However, $\|\cdot\|_0$ is not a convex function (and hence not a norm), and it is hard to solve an optimization problem with the constraint set $S_0 = \{w : \|w\|_0 \leq k\}$. Hence, typically we relax the regularizer to be the ℓ_1 norm, which has nicer properties. Constraining the ℓ_1 norm does not yield a convex relaxation of S_0, in the sense that $\|w\|_0$ might be small while $\|w\|_1$ is large. However, additionally constraining the ℓ_2 norm fixes this problem, as the convex hull of the set $S_{0,2} = \{w : \|w\|_0 \leq k, \|w\|_2 \leq 1\}$ is a subset of $S_{1,2} = \{w : \|w\|_1 \leq \sqrt{k}, \|w\|_2 \leq 1\}$, i.e. $\mathrm{conv}(S_{0,2}) \subseteq S_{1,2}$. This motivates the use of the elastic-net

regularizer in literature. Recently, researchers have looked at whether $S_{1,2}$ is the *tightest* convex relaxation of $S_{0,1}$, and found that it is not. Specifically, [1] show that this tightest convex envelope can be obtained in closed form as a norm, which they call the k-Support norm of w.

The k-Support Norm is essentially a trade-off between an ℓ_2 penalty on the largest components, and an ℓ_1 penalty on the remaining smaller components. In our case, when $\rho^* = 1$ in (13), $\Lambda^{**}(\mathbf{A})$ reduces to $\frac{c_0}{r}(\sum_{i=1}^{d} a_i)^2 = \frac{c_0}{r}\|\mathbf{A}\|_*^2$, which is (a scaling of) the nuclear norm (squared) of $\mathbf{A}$. On the other hand, when the block size r is larger, ρ^* will take higher values, implying the regularizer $\Lambda^{**}(\mathbf{A})$ will move closer to $c_0 \sum_{i=1}^{d} a_i^2 = c_0\|\mathbf{A}\|_{\mathrm{F}}^2$, which is (a scaling of) the squared Frobenius norm of $\mathbf{A}$. Therefore, the DropBlock regularizer acts as an interpolation between (squared) nuclear norm regularization when the block size is small to (squared) Frobenius norm regularization when the block size becomes very large. Further, [1, 7] observe that regularization using the $k-$Support norm achieves better performance than other forms of regularization on some real-world datasets and this might be a step towards theoretically explaining the superior performance of DropBlock compared to Dropout, as was observed experimentally in [5].

Closed Form Solutions for DropBlock. Continuing our analysis, with the convex envelope of $\Lambda(\mathbf{A})$, we can construct a convex lower bound of the DropBlock objective $f(\mathbf{U}, \mathbf{V}, d)$, as follows:

$$\mathrm{F}(\mathbf{A}) = \|\mathbf{Y} - \mathbf{A}\|_{\mathrm{F}}^2 + \Lambda^{**}(\mathbf{A}) \tag{14}$$

with the relationship $\mathrm{F}(\mathbf{A}) \leq \bar{F}(\mathbf{A}) \leq f(\mathbf{U}, \mathbf{V})$ for all $(\mathbf{U}, \mathbf{V}, \mathbf{A})$ such that $\mathbf{U}\mathbf{V}^\top = \mathbf{A}$. As shown earlier in Theorem 5 the lower bound $\bar{F}(\cdot)$ takes the same minimum value as $f(\cdot, \cdot, \cdot)$. By properties of the lower convex envelope, we know that the function $\bar{F}(\cdot)$ takes the same minimum value as its convex lower bound $\mathrm{F}(\cdot)$. Following this reasoning, we now complete the analysis by deriving a closed form solution for the global minimum of $\mathrm{F}(\mathbf{A})$.

Theorem 7. *When* $\mathbf{X}$ *has full column rank in* (10)*, the global minimizer of* $\mathrm{F}(\mathbf{A})$ *is given by* $\mathbf{A}_{\rho,\lambda} = \mathbf{U}_{\mathbf{Y}}\mathrm{diag}(\mathbf{a}_{\rho,\lambda})\mathbf{V}_{\mathbf{Y}}^\top$*, where* $\mathbf{Y} = \mathbf{U}_{\mathbf{Y}}\mathrm{diag}(\mathbf{m})\mathbf{V}_{\mathbf{Y}}^\top$ *is an SVD of* $\mathbf{Y}$*, and* $\mathbf{a}_{\rho,\lambda}$ *is given by*

$$\mathbf{a}_{\rho,\lambda} = \begin{cases} \left(\frac{m_1}{\beta+1}, \frac{m_2}{\beta+1}, \ldots, \frac{m_\lambda}{\beta+1}, 0, \ldots, 0\right) & \text{if } \lambda \leq \rho - 1 \\ \left(\begin{array}{l} \frac{m_1}{\beta+1}, \frac{m_2}{\beta+1}, \ldots, \frac{m_{\rho-1}}{\beta+1}, \\ m_\rho - \frac{\beta}{c}S, m_{\rho+1} - \frac{\beta}{c}S, \ldots, \\ m_\lambda - \frac{\beta}{c}S, 0, \ldots, 0 \end{array}\right) & \text{if } \lambda \geq \rho. \end{cases}$$

The constants are $\beta = \frac{1-\bar{\theta}}{\bar{\theta}}$, $S = \sum_{i=\rho}^{\lambda} m_i$, $c = r + \beta\lambda + (\beta+1)(1-\rho)$*, while* $\rho \in \{1, 2, \ldots, r-1\}$ *and* $\lambda \in \{1, 2, \ldots, d\}$ *are chosen such that they minimize* $\mathrm{F}(\mathbf{A}_{\rho,\lambda})$.

Note that the constants mentioned in Theorem 7 depend purely on the matrix $\mathbf{Y}$, and can be computed in time $O(d^2)$ given the singular values m_i. Finally, the following Corollaries obtained from Theorem 4 complete the picture by showing that the solution computed in Theorem 7 recovers the value of the global minimizer of the DropBlock objective $f(\cdot, \cdot, \cdot)$, and that Λ is convex:

Corollary 8. *If* $\mathbf{A}^*$ *is a global minimizer of the lower convex envelope* F*, and* $(\mathbf{U}^*, \mathbf{V}^*, d^*)$ *is a global minimizer of the non-convex objective* f*, then we have* $\mathrm{F}(\mathbf{A}^*) = \bar{F}(\mathbf{A}^*) = f(\mathbf{U}^*, \mathbf{V}^*, d^*)$ *with* $\mathbf{A}^* = \mathbf{U}(\mathbf{V}^*)^\top\mathbf{X}$.

Corollary 9. $\Lambda(\mathbf{A})$ *is convex and equal to its lower convex envelope* $\Lambda^{**}(\mathbf{A})$ *(i.e.,* $\Lambda(\mathbf{A}) = \Lambda^{**}(\mathbf{A}), \forall \mathbf{A}$*).*

Note that the Corollary 8 also trivially applies if $\mathbf{A}^*$ is a global minimizer of $\bar{F}$ since F and $\bar{F}$ share the same set of global minimizers. Having understood the properties of one particular generalization of dropout for a single hidden-layer linear network, we will now show how our methods can be generalised to other Dropout variants applied to the last layer of an overparameterized neural network.

3. Generalized Dropout Framework

In practice, commonly used neural network architectures typically have a fully-connected linear layer as the final layer in the network, and it is also common to use Dropout-style regularization only on this final fully-connected layer. This leads us to consider the effect of training deep NNs with Dropout-style regularization applied to a final linear layer. Specifically, we will consider a NN training problem with squared-loss of the form

$$\min_{\mathbf{U},\Gamma} \mathbb{E}_{\mathbf{z}} \left\|\mathbf{Y} - \mathbf{U}\mathrm{diag}(\boldsymbol{\mu})^{-1}\mathrm{diag}(\mathbf{z})g_\Gamma(\mathbf{X})\right\|_{\mathrm{F}}^2, \tag{15}$$

where we follow the notation in the introduction, and in addition we let g_Γ denote the output of the second to last layer of a NN with weight parameters Γ (i.e., the j^{th} column of g_Γ is the output of second to last layer of the network given input $\mathbf{x}_j$), $\mathbf{U} \in \mathbb{R}^{a \times d}$ be the weight matrix for the final linear layer, with d being the size of the output of the second to last layer, and $\boldsymbol{\mu} \in \mathbb{R}^d \backslash \mathbf{0}$ be a vector of the means of the Dropout variables, $\mu_i = \mathbb{E}[z_i]$ (note that in expectation, the output of the i^{th} hidden unit of g_Γ is scaled by $\mathbb{E}[z_i]$, so to counter this effect, we rescale the output by $\mathbb{E}[z_i]^{-1}$).

We will assume that the Dropout variables $\mathbf{z}$ are stochastically sampled at each iteration of the algorithm from an arbitrary probability distribution $\mathcal{S}$ with covariance matrix $\mathbf{C} = \mathrm{Cov}(\mathbf{z}, \mathbf{z})$ and mean $\boldsymbol{\mu}$. Assuming that each entry of $\boldsymbol{\mu}$ is non-zero, we define the Characteristic Matrix $\bar{\mathbf{C}}$ from entries of the mean and covariance of $\mathbf{z}$, μ_i and $c_{i,j}$, as

$$\bar{c}_{i,j} = \frac{c_{i,j}}{\mu_i \mu_j} \quad \text{or} \quad \bar{\mathbf{C}} = \mathrm{diag}(\boldsymbol{\mu})^{-1}\mathbf{C}\mathrm{diag}(\boldsymbol{\mu})^{-1}. \tag{16}$$

Recall that one iteration of a typical Dropout algorithm can be interpreted as performing one iteration of stochastic gradient descent on (15), where the gradient of (15) is approximated by a single stochastic sample of the Dropout variables, $\mathbf{z}$. In this setting, we can obtain the deterministic form of (15), which is a generalisation of Lemma 1:

Lemma 10. *The Generalized Dropout objective (15) is equivalent to a regularized deterministic objective:*

$$\mathbb{E}_{\mathbf{z}} \left\| \mathbf{Y} - \mathbf{U}\text{diag}(\boldsymbol{\mu})^{-1}\text{diag}(\mathbf{z})g_\Gamma(\mathbf{X}) \right\|_{\text{F}}^2 = \|\mathbf{Y} - \mathbf{U}g_\Gamma(\mathbf{X})\|_{\text{F}}^2 + \Omega_{\mathbf{C},\boldsymbol{\mu}}(\mathbf{U}, g_\Gamma(\mathbf{X})^\top), \quad (17)$$

where the "generalized Dropout" regularizer is defined as

$$\Omega_{\mathbf{C},\boldsymbol{\mu}}(\mathbf{U},\mathbf{V}) = \sum_{i,j=1}^{d} c_{i,j} \frac{(\mathbf{u}_i^\top \mathbf{u}_j)(\mathbf{v}_i^\top \mathbf{v}_j)}{\mu_i \mu_j} = \langle \bar{\mathbf{C}}, \mathbf{U}^\top\mathbf{U} \odot \mathbf{V}^\top\mathbf{V} \rangle, \quad (18)$$

with $\mathbf{u}_i$ and $\mathbf{v}_i$ denoting the i^{th} columns of $\mathbf{U}$ and $\mathbf{V}$, resp.

Note we have defined $\Omega_{\mathbf{C},\boldsymbol{\mu}}$ for general matrices $(\mathbf{U},\mathbf{V})$ for notational simplicity, but typically we will have $\mathbf{V} = g_\Gamma(\mathbf{X})^\top$). Notice also that $\bar{\mathbf{C}}$ completely determines the regularization properties of any dropout scheme. For example, in classical Dropout, the entries of $\mathbf{z}$ are i.i.d. Bernoulli variables with mean θ, hence $\mathbf{C}$ is diagonal with diagonal entries $c_{i,i} = \theta(1-\theta)$ and $\mu_i = \theta$, hence $\bar{\mathbf{C}}$ is diagonal with diagonal entries $\bar{c}_{i,i} = \frac{1-\theta}{\theta}$. For DropBlock with block-size r, we have $\bar{\mathbf{C}} = \frac{1-\theta}{\theta}\text{BlkDiag}(\mathbf{1}_r\mathbf{1}_r^\top, \ldots, \mathbf{1}_r\mathbf{1}_r^\top)$, where $\text{BlkDiag}(\cdot)$ denotes forming a block diagonal matrix with the function arguments along the diagonal and $\mathbf{1}_r$ denotes an r-dimensional vector of all ones. In the case of Dropout, we recover an immediate simple corollary for the regularization induced by Dropout in the final layer of nonlinear networks:

Corollary 11. *For regular Dropout applied to objective (15) the following equivalence holds:*

$$\mathbb{E}_{\mathbf{z}} \left\| \mathbf{Y} - \mathbf{U}\text{diag}(\boldsymbol{\mu})^{-1}\text{diag}(\mathbf{z})g_\Gamma(\mathbf{X}) \right\|_{\text{F}}^2 = \|\mathbf{Y} - \mathbf{U}g_\Gamma(\mathbf{X})\|_{\text{F}}^2 + \sum_{i=1}^{d} \|\mathbf{u}_i\|_2^2 \|g_\Gamma^i(\mathbf{X})\|_2^2, \quad (19)$$

where $g_\Gamma^i(\mathbf{X}) \in \mathbb{R}^N$ denotes the output of the i^{th} neuron of g_Γ (i.e., the i^{th} row of $g_\Gamma(\mathbf{X})$).

Given this result, a simple interpretation of Dropout in the final layer of the network is that it adds a form of weight-decay both to the weight parameters in the final layer, $\mathbf{U}$, and the output of g_Γ. Additionally, from this result it is relatively simple to show the following characterization of the regularization induced by Dropout applied to the final layer of a network. Note that the following result (Proposition 12) can be shown using similar arguments to those used in [2] along with a sufficient capacity assumption.

Proposition 12. *If the network architecture, g_Γ, has sufficient capacity to span $\mathbb{R}^{d\times N}$ (i.e., for all $\mathbf{Q} \in \mathbb{R}^{d\times N}$ there is a set of network weights $\bar{\Gamma}$ such that $g_{\bar{\Gamma}}(\mathbf{X}) = \mathbf{Q}$) and $d \geq \min\{a, N\}$, then the global optimum of (15) with $\mathbf{z} \overset{i.i.d}{\sim}$ Bernoulli(θ) is given by:*

$$\min_{\mathbf{U},\Gamma} \mathbb{E}_{\mathbf{z}} \left\| \mathbf{Y} - \mathbf{U}\text{diag}(\boldsymbol{\mu})^{-1}\text{diag}(\mathbf{z})g_\Gamma(\mathbf{X}) \right\|_{\text{F}}^2 = \min_{\mathbf{A}} \|\mathbf{Y} - \mathbf{A}\|_F^2 + \tfrac{1-\theta}{\theta}\|\mathbf{A}\|_*^2 \quad (20)$$

where $\|\mathbf{A}\|_*$ denotes the nuclear norm of $\mathbf{A}$.

The above result has interesting implications in the sense that it implies that even in the limit where the g_Γ network has infinite capacity and can represent an arbitrary output perfectly, applying Dropout to the final layer still induces capacity constraints on the output of the overall network in the form of (squared) nuclear norm regularization, where the strength of the regularization depends on the Dropout rate $(1-\theta)$. A result similar to Proposition 12 can be obtained for DropBlock applied to the last layer of a network with sufficient capacity (the sampling strategy for $\mathbf{z}$ changes and the regularizer changes from nuclear norm squared to k-support norm squared). Having analyzed Dropout and its variants, we now consider an alternative but closely related approach, DropConnect in the next section.

4. DropConnect Analysis

DropConnect, proposed in [12], is very similar to Dropout, but instead of setting the outputs of hidden neurons to zero, DropConnect instead sets *elements* of the connection weights to zero independently with probability $1-\theta$. Hence, the DropConnect algorithm samples a random matrix $\mathbf{Z} \in \mathbb{R}^{b\times d}$, with each $z_{i,j}$ drawn independently from the Bernoulli distribution with parameter θ. For Dropconnect applied to the second-last layer weights $\mathbf{V}$ of a deep network parameterized as $\mathbf{U}\mathbf{V}^\top g_\Gamma(\mathbf{X})$, the optimization problem then becomes the following:

$$\min_{\mathbf{U},\mathbf{V},\Gamma} \mathbb{E}_{\mathbf{Z}} \left\| \mathbf{Y} - \frac{1}{\theta}\mathbf{U}(\mathbf{Z} \odot \mathbf{V})^\top g_\Gamma(\mathbf{X}) \right\|_{\text{F}}^2 \quad (21)$$

Note that we apply DropConnect to the second-last layer $\mathbf{V}$ instead of $\mathbf{U}$ in order to match the original authors proposal [12]. We show that DropConnect induces the same regularization as Dropout. Specifically, the regularizer induced in (21) is the same as applying vanilla Dropout on the last layer:

Theorem 13. *For Dropconnect applied to the second-to-last layer weights $\mathbf{V}$ of a deep network parameterized as $\mathbf{U}\mathbf{V}^\top g_\Gamma(\mathbf{X})$, the following equivalence holds:*

$$\mathbb{E}_{\mathbf{Z}} \left\| \mathbf{Y} - \frac{1}{\theta}\mathbf{U}(\mathbf{Z} \odot \mathbf{V})^\top g_\Gamma(\mathbf{X}) \right\|_{\text{F}}^2 = \left\| \mathbf{Y} - \mathbf{U}\mathbf{V}^\top g_\Gamma(\mathbf{X}) \right\|_{\text{F}}^2 + \frac{1-\theta}{\theta} \sum_{i=1}^{d} \|\mathbf{u}_i\|_2^2 \|g_\Gamma(\mathbf{X})^\top \mathbf{v}_i\|_2^2$$

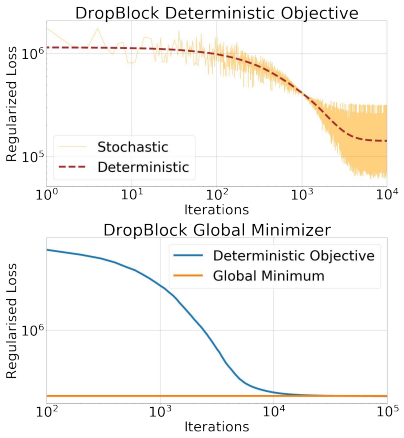

Figure 1. Top: Stochastic DropBlock training with SGD is equivalent to the deterministic objective (5). Bottom: DropBlock converges to the global minimum computed in Theorem 7.

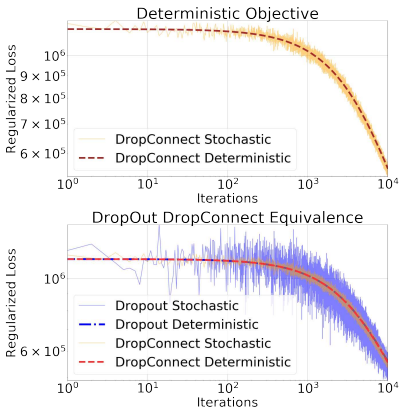

Figure 2. Comparing DropConnect to DropOut. Top: Stochastic DropConnect training with SGD is equivalent to the deterministic Objective (22). Bottom: DropConnect training is equivalent to Dropout training for the squared loss.

where $g^i_\Gamma(\mathbf{X}) \in \mathbb{R}^N$ *denotes the output of the* i^{th} *neuron of* g_Γ *(i.e., the* i^{th} *row of* $g_\Gamma(\mathbf{X})$*).*

Taking $g_\Gamma(\mathbf{X}) = \mathbf{X}$ in Theorem 13 then gives us the following result for a single layer linear network.

Corollary 14. *For single layer linear networks, the stochastic DropConnect objective* (21) *is equivalent to the vanilla Dropout deterministic objective:*

$$\mathbb{E}_{\mathbf{Z}} \left\| \mathbf{Y} - \frac{1}{\theta}\mathbf{U}(\mathbf{Z} \odot \mathbf{V})^\top \mathbf{X} \right\|_2^2$$
$$= \|\mathbf{Y} - \mathbf{U}\mathbf{V}^\top \mathbf{X}\|_2^2 + \frac{1-\theta}{\theta} \sum_{i=1}^{d} \|\mathbf{u}_i\|_2^2 \|\mathbf{X}^\top \mathbf{v}_i\|_2^2 \quad (22)$$

Note that by an identical line of arguments as made in [2, 8] the above result also implies that DropConnect induces low-rank solutions in linear networks.

5. Experiments

In this section, we conduct experiments in training single hidden layer linear networks as well as multilayer nonlinear networks to validate the theory developed so far.

5.1. Shallow Network Experiments

We first create a simple synthetic dataset $\mathcal{D}_{\text{syn}}$ by taking 1000 i.i.d samples of $\mathbf{x}$ from a 100-dimensional standard normal distribution. Then, $\mathbf{y} \in \mathbb{R}^{80}$ is generated as $\mathbf{y} = \mathbf{M}\mathbf{x}$, where $\mathbf{M} = \mathbf{U}_{\text{true}}\mathbf{V}_{\text{true}}^\top$. To ensure a reliable comparison, all the experiments start with the same choice of $\mathbf{U}_0 = \mathbf{U}_{\text{init}} \in \mathbb{R}^{80\times 50}$ and $\mathbf{V}_0 = \mathbf{V}_{\text{init}} \in \mathbb{R}^{100\times 50}$. The entries of all the matrices $\mathbf{U}_{\text{true}}, \mathbf{V}_{\text{true}}, \mathbf{U}_{\text{init}}, \mathbf{V}_{\text{init}}$ are sampled elementwise from $\mathcal{N}(0,1)$.

Verifying Deterministic Formulations. We first verify the correctness of the deterministic formulations for various dropout schemes analyzed in this paper, i.e. (5) and (22), in the top panels of Figure 1 and Figure 2. In Figure 1, the curve labelled *DropBlock Stochastic* is the training objective plot, *i.e.* it plots $\left\| \mathbf{Y} - \frac{1}{\theta}\mathbf{U}_t(\text{diag}(\mathbf{w}_t) \otimes \mathbf{I}_{\frac{d}{r}})\mathbf{V}_t^\top \mathbf{X} \right\|_{\text{F}}^2$ as the training progresses via Algorithm 1. For generating the curve labeled *DropBlock Deterministic*, we take the current iterate, *i.e.* $\mathbf{U}_t, \mathbf{V}_t$, and plot the Deterministic DropBlock objective obtained in Lemma 1 at every iteration. The deterministic equivalent of the DropConnect objective is similarly verified in Figure 2. Both the figures show plots for $\theta = 0.5$, and the plots for more values of θ are deferred to the Appendix. It can be seen that the expected value of DropConnect and DropBlock over iterations matches the values derived in our results. Additionally, the bottom panel of Figure 2 shows that Dropout and DropConnect have the same expected value of the objective at each iteration.

Verifying Convergence to the Global Minimum. We next verify the convergence of DropBlock to the theoretical global minimum computed in Theorem 7. The bottom panel of Figure 1 plots the deterministic DropBlock objective as the training progresses, showing convergence to the computed theoretical global minimum. It can be seen that the training converges to the DropBlock Global minimum computed in Theorem 7.

5.2. Deep Network Experiments

In order to test our predictions on common network architectures, we modify the standard Resnet-50 architecture by removing the last layer and inserting a fully-connected (FC) layer to reduce the hidden layer dimensionality to 80 (to make the experiments consistent with the Synthetic Experiments). Hence, the network architecture now is, $\mathbf{x} \rightarrow$ `Resnet-50 Layers` $\rightarrow$ `FC` $\rightarrow$ `Dropout` $\rightarrow$ `FC` $\rightarrow \mathbf{y}$. We then train the entire network on small datasets $\mathcal{D}_{\text{MNIST}}, \mathcal{D}_{\text{CIFAR10}}$ with DropBlock applied to the last layer with a block size of 5. Figure 3 shows that the so-

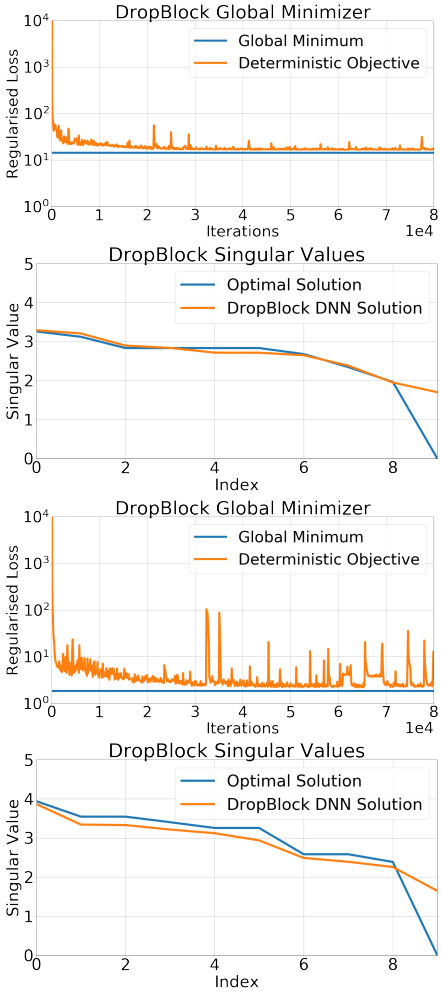

Figure 3. Results for Resnet-50 training on MNIST (first and second panels) and CIFAR10 (third and fourth panels) data. The first and third panels show the deterministic loss during each training iteration as training progresses, and the second and fourth panels show the singular values of the product matrix of the final iterate.

lution found by gradient descent is very close to the lower bound predicted by Theorem 7: The objective value is plotted on the first and third panels, and the singular values of the final predictions matrix $\mathbf{U}g_\Gamma(\mathbf{X})$ are plotted in decreasing order on the second and fourth panels. Note that qualitatively the singular values of the final predictions matrix closely match the theoretical prediction, with the exception of the least significant singular value, which we attribute to the highly non-convex network training problem not converging completely to the true global minimum.

5.3. Effect of DropBlock approximation

The original DropBlock method [5] allows dropping blocks at arbitrary locations, in this paper we made an approximation by constraining the blocks to be non-overlapping, as mentioned in the beginning of Section 2. This approximation is a minor constraint, and the block

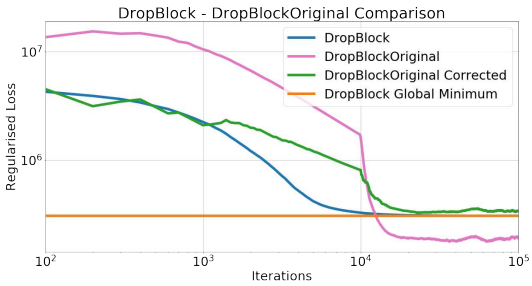

Figure 4. Comparison of DropBlockOriginal and the approximation DropBlock we have made. The training curves correspond to $\theta_{\text{DropBlock}} = 0.5$ of Fig 1. The curves have been smoothed via an exponential moving average.

retaining probability θ can be scaled appropriately to recover the original behavior. DropBlockOriginal with the same θ as DropBlock leads to a higher *effective* dropping rate. This can be corrected by solving for $\theta'_{\text{DBOriginal}}$ such that the probability of dropping any neuron in DropBlock with retain probability $\theta_{\text{DropBlock}}$ is same as the probability of dropping a neuron in DropBlockOriginal with retain probability $\theta'_{\text{DBOriginal}}$. Specifically, referring to the notation in Section 2, under the Original DropBlock scheme, the probability of $z_i = 0$ is same as the probability of (none of the $w_j = 1$) over all j, where $|i-j| \leq k$. This probability is $(1-\theta'_{\text{DBOriginal}})^{2k-1}$. Under our approximation, the probability of $z_i = 0$ is $1-\theta_{\text{DropBlock}}$. Equating these quantities, we can solve for $\theta'_{\text{DBOriginal}}$ as $\theta'_{\text{DBOriginal}} = 1-(1-\theta_{\text{DropBlock}})^{\frac{1}{2k-1}}$. As can be seen in Fig 4, DropBlockOriginal with the appropriate correction is approximately the same as DropBlock, as the green, blue, orange curves are very close in log-scale at iteration 10^5.

6. Conclusion

In this work, we have analysed the regularization properties of structured Dropout training of neural networks, and characterized the global optimum obtained for some classes of networks and structured Dropout strategies. We showed that DropBlock induces spectral k-Support norm regularization on the weight matrices, providing a potential way of theoretically explaining the empirically observed superior performance of DropBlock as compared to Dropout. We also proved that Dropout training is equivalent to DropConnect training for some network classes. Finally, we showed that our techniques can be extended to other generic Dropout strategies, and to Deep Networks with Dropout-style regularization applied to the last layer of the network, significantly generalizing prior results.

Acknowledgements This work was supported by IARPA contract D17PC00345 and NSF Grants 1618485 and 1934979.

References

[1] Andreas Argyriou, Rina Foygel, and Nathan Srebro. Sparse prediction with the k-support norm. In *Advances in Neural Information Processing Systems*, pages 1457–1465, 2012. 2, 4, 5

[2] J. Cavazza, B.D. Haeffele, C. Lane, P. Morerio, V. Murino, and R. Vidal. Dropout as a low-rank regularizer for matrix factorization. In *International Conference on Artificial Intelligence and Statistics*, volume 84, pages 435–444, 2018. 1, 3, 6, 7

[3] Yarin Gal, Jiri Hron, and Alex Kendall. Concrete dropout. In *Advances in Neural Information Processing Systems*, 2017. 1

[4] Xavier Gastaldi. Shake-shake regularization. In *arXiv preprint arXiv:1705.07485*, 2017. 1

[5] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Dropblock: A regularization method for convolutional networks. In *Advances in Neural Information Processing Systems*, pages 10750–10760, 2018. 1, 5, 8

[6] Benjamin David Haeffele and René Vidal. Structured low-rank matrix factorization: Global optimality, algorithms, and applications. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[7] Andrew M McDonald, Massimiliano Pontil, and Dimitris Stamos. Spectral k-support norm regularization. In *Advances in Neural Information Processing Systems*, pages 3644–3652, 2014. 4, 5

[8] Poorya Mianjy, Raman Arora, and Rene Vidal. On the implicit bias of dropout. In *International Conference on Machine Learning*, 2018. 1, 3, 4, 7

[9] P. Morerio, J. Cavazza, R. Volpi, R. Vidal, and V. Murino. Curriculum dropout. In *IEEE International Conference on Computer Vision*, Oct 2017. 1

[10] Steven J Rennie, Vaibhava Goel, and Samuel Thomas. Annealed dropout training of deep networks. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, 2014. 1

[11] Ralph Tyrell Rockafellar. *Convex analysis*. Princeton university press, 2015. 4

[12] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *International Conference on Machine Learning*, pages 1058–1066, 2013. 1, 6

[13] Yoshihiro Yamada, Masakazu Iwamura, Takuya Akiba, and Koichi Kise. Shakedrop regularization for deep residual learning. In *arXiv preprint arXiv:1802.02375*, 2018. 1

[14] Konrad Zolna, Devansh Arpit, Dendi Suhubdy, and Yoshua Bengio. Fraternal dropout. In *arXiv preprint arXiv:1711.00066*, 2017. 1