GraphConnect: A Regularization Framework for Neural Networks

Article · December 2015

Source: arXiv

CITATIONS
3

READS
579

4 authors, including:

Jiaji Huang
Baidu Research
26 PUBLICATIONS **181** CITATIONS
SEE PROFILE

Qiang Qiu
Purdue University
93 PUBLICATIONS **1,393** CITATIONS
SEE PROFILE

Robert Calderbank
Duke University
600 PUBLICATIONS **44,868** CITATIONS
SEE PROFILE

GraphConnect: A Regularization Framework for Neural Networks

Jiaji Huang Qiang Qiu Robert Calderbank Guillermo Sapiro
Department of Electrical and Computer Engineering
Duke University
Durham, NC, 27708
{jiaji.huang, qiang.qiu, robert.calderbank, guillermo.sapiro}@duke.edu

Abstract

Deep neural networks have proved very successful in domains where large training sets are available, but when the number of training samples is small, their performance suffers from overfitting. Prior methods of reducing overfitting such as weight decay, Dropout and DropConnect are data-independent. This paper proposes a new method, GraphConnect, that is data-dependent, and is motivated by the observation that data of interest lie close to a manifold. The new method encourages the relationships between the learned decisions to resemble a graph representing the manifold structure. Essentially GraphConnect is designed to learn attributes that are present in data samples in contrast to weight decay, Dropout and DropConnect which are simply designed to make it more difficult to fit to random error or noise. Empirical Rademacher complexity is used to connect the generalization error of the neural network to spectral properties of the graph learned from the input data. This framework is used to show that GraphConnect is superior to weight decay. Experimental results on several benchmark datasets validate the theoretical analysis, and show that when the number of training samples is small, GraphConnect is able to significantly improve performance over weight decay.

1. Introduction

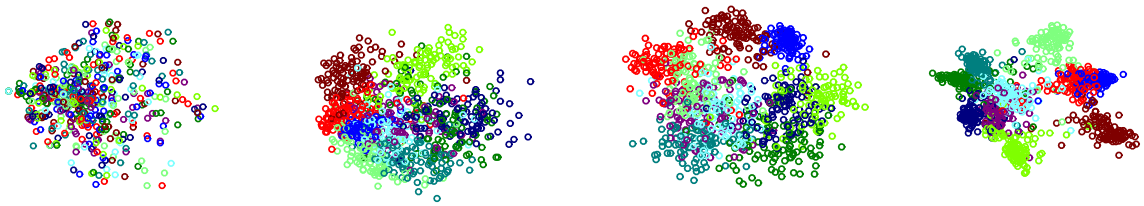
Neural networks have proved very successful in domains where large training sets are available, since their capacity can be increased by adding layers or by increasing the number of units in a layer. When the number of training samples is small their performance suffers from overfitting. The degree of overfitting is measured by the generalization error, which is the difference between the loss on the training set and the loss on the test set. The best known bounds on the generalization error arise from the Vapnik-Chervonenkis (VC) dimension [17], which captures the inherent complexity of a family of classifiers. However it is distribution agnostic, leading to a loose and pessimistic upper bound [4].

In the last decade, distribution dependent measures of complexity have been developed, such as the Rademacher complexity [2], which can lead to tighter bounds on the generalization error. Rademacher complexity has been used to show that the generalization error of a neural network depends more on the size of the weights than on the size of the network [1]. This theoretical result supports a form of regularization called weight decay that simply encourages the ℓ-2 norm of all parameters to be small. A unified theoretical treatment of norm-based control of deep neural networks is given in [12], and this theory is used to provide insight into max-out networks in [6].

Dropout is a new form of regularization, where for each training example, forward propagation involves randomly deleting half the activations in each layer [7]. *Dropconnect* is a recent generalization of *Dropout*, where a randomly selected subset of weights within the network is set to zero [18]. Both methods introduce randomness into training so that the learned network is in some sense a statistical average of an ensemble of realizations. Adding a *Dropconnect* layer to a neural network can reduce the Rademacher complexity by a factor of the dropconnect rate [18].

In this paper we propose a fundamentally different approach to control the complexity of a neural network, thereby preventing overfitting. Our approach differs from the aforementioned methods in that it is data dependent. It is motivated by the empirical observation that data of interest typically lie close to a manifold, an assumption that has previously assisted machine learning tasks such as nonlinear embedding [10], semi-supervised labeling [11], and multi-task classification [5]. The underlying idea in these works is to encourage the relationships between the learned decisions to resemble a graph representing the manifold structure.

In this work we propose to use graph structures, derived from training data, to regularize deep neural networks. We present theoretical and experimental results that



(a) 1000 test samples from MNIST

(b) Transformed test samples when the learned network is regularized by weight decay

(c) Transformed test samples when the learned network is regularized by *GraphConnect-One*

(d) Transformed test samples when the learned network is regularized by *GraphConnect-All*

Figure 1: Embedding of initial and transformed test samples with different colors representing different classes. All networks are learned from the same set of 500 training samples.

demonstrate the importance of this new data dependent approach to prevention of overfitting. Previous experimental studies [19], applying graph regularization to the semi-supervised embedding problem, did not explain how the approach might control capacity. In particular, it is not clear from [19] whether graph-based regularization has better generalization ability than standard approaches such as weight decay. In contrast we demonstrate, both experimentally and theoretically that graph regularization outperforms weight decay, and that when the number of training samples is extremely limited, it is able to significantly improve performance of a deep neural network. An illustrative example is given in Fig. 1, where we transform 1,000 test samples (Fig. 1a) via learned networks regularized by weight decay (Fig. 1b) and our proposed *GraphConnect* (Fig. 1c, 1d). *GraphConnect* significantly improves the discriminability than does weight decay.

2. *GraphConnect*

A note on the notation: Upper and lower case bold letters denote matrices and vectors respectively. Plain letters denote scalars. We consider L-way classification using a deep neural network. Given a datum $\mathbf{x}$, the network first learns a multidimensional feature $\mathbf{g}(\mathbf{x})$, and then applies a softmax classifier to generate a probability distribution over the L classes. We use cross entropy loss to measure the effectiveness of the learning machine, and view the deep network as a function from a datum $\mathbf{x}$ to the corresponding loss $\ell(\mathbf{x})$.

The average loss achieved on the training set $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ is the empirical loss ℓ_{emp}, given by

$$\ell_{emp} = \frac{1}{N} \sum_{i=1}^{N} \ell_i.$$

The expected loss ℓ is estimated from a large test set and is given by

$$\ell = \mathbb{E}_{\mathbf{x}}[\ell(\mathbf{x})].$$

The difference $\ell - \ell_{emp}$ between the expected loss on the test data and the empirical loss on the training data is the *generalization error*. When training samples are scarce, statistical learning theory predicts overfitting to the training data [17]. The larger the generalization error, the more severe is the problem of overfitting. We recall from [8] that the generalization error is (almost surely) bounded by the empirical Rademacher complexity [2], of the loss function. Hence we can reduce the degree of overfitting by controlling the empirical Rademacher complexity.

Definition 1 (Empirical Rademacher Complexity)**.** *Let $\mathcal{D}$ be a probability distribution on a set $\mathcal{X}$ and assume that $\mathbf{x}_1, \ldots, \mathbf{x}_N$ are independent samples from $\mathcal{D}$. Let $\mathcal{U}$ be a class of functions mapping from $\mathcal{X}$ to $\mathbb{R}$. The empirical Rademacher complexity of $\mathcal{U}$ is*

$$\hat{\mathcal{R}}_N(\mathcal{U}) = \mathbb{E}_{\sigma_i}\left[\frac{2}{N}\sup_{u\in\mathcal{U}}\left|\sum_{i=1}^{N}\sigma_i u(\mathbf{x}_i)\right| : \mathbf{x}_1, \ldots, \mathbf{x}_N\right],$$

where the σ_i's are independent uniform $\{\pm 1\}$-valued random variables.

Remark 1. *Over-fitting occurs when a statistical model describes random error or noise instead of the underlying signal. We seek to minimize the empirical Rademacher complexity because it measures correlation with random errors. However we need to keep the objective of classification in mind, since we can reduce the empirical Rademacher complexity to zero by simply mapping every datum $\mathbf{x}$ to 0. Therefore minimization of Rademacher complexity needs to be performed over a class of functions $\mathcal{U}$ that is able to discriminate between classes.*

Dropconnect, [18], is a recent method for regularizing large fully connected layers within neural networks. A layer consists of a linear feature extraction, followed by activation, followed by a softmax classifier. The Rademacher complexity of the composite layer differs from

the Rademacher complexity of the linear feature extraction component by a multiplicative factor that is determined by the classifier [18]. Hence we follow [18] and focus on the Rademacher complexity of the linear feature extraction.

2.1. Analysis: Regularizing a Linear Layer

The functions $\mathcal{U}$ appearing in Definition 1 map multidimensional inputs to scalars, therefore in order to make use of Rademacher complexity we need to work with individual coordinate entries of multidimensional features. Given an input $\mathbf{z}$ to a linear layer, we consider the linear map $f(\mathbf{z}) = \mathbf{v}^\top \mathbf{z}$, where the linear weights $\mathbf{v}$ are from a set $\mathcal{V}$ specified by graph-based regularization. More formally we consider the function family,

$$\mathcal{F} = \left\{ f(\mathbf{z}) | f(\mathbf{z}) = \mathbf{v}^\top \mathbf{z}, \mathbf{v} \in \mathcal{V} \right\}. \tag{1}$$

Suppose now that we have learned a graph where symmetric edge weights $\mathbf{W}$ encode the relationships between N input samples $\mathbf{Z} \stackrel{\text{def}}{=} [\mathbf{z}_1, \ldots, \mathbf{z}_N]$. The set $\mathcal{V}$ that defines the function family $\mathcal{F}$ is given by

$$\left\{ \mathbf{v} : \frac{1}{N} \left((1-\eta) \sum_{i,j=1}^{N} W_{i,j} [f(\mathbf{z}_i) - f(\mathbf{z}_j)]^2 + \eta \|\mathbf{v}\|^2 \right) \leq \frac{B^2}{4} \right\}, \tag{2}$$

for some positive constant B and for some $\eta \in (0, 1]$. When $\eta = 1$, this condition enforces conventional weight decay, and as η approaches 0, it enforces graph regularization.

Let $\mathbf{1}$ be the all-one vector, let $\mathbf{D}$ be the diagonal matrix with entries $\mathbf{W1}$, and let $\mathbf{L} = \mathbf{D} - \mathbf{W}$ be the graph Laplacian. The Laplacian is symmetric, hence

$$\sum_{i,j=1}^{N} W_{i,j} [f(\mathbf{z}_i) - f(\mathbf{z}_j)]^2 = \mathbf{v}^\top (\mathbf{Z}\mathbf{L}\mathbf{Z}^\top) \mathbf{v}, \tag{3}$$

and since the Laplacian is also diagonally dominant, it is positive semidefinite. Therefore by introducing identity matrix $\mathbf{I}$, we can describe the set $\mathcal{V}$ very simply as

$$\mathcal{V} = \left\{ \mathbf{v} \left| \mathbf{v}^\top \left((1-\eta) \mathbf{Z}\mathbf{L}\mathbf{Z}^\top + \eta \mathbf{I} \right) \mathbf{v} \leq \frac{NB^2}{4} \right. \right\}. \tag{4}$$

Note that since $\eta > 0$, the matrix $(1-\eta)\mathbf{Z}\mathbf{L}\mathbf{Z}^\top + \eta \mathbf{I}$ is positive definite. We now bound $\hat{\mathcal{R}}_N(\mathcal{F})$.

Theorem 1 (Empirical Rademacher Complexity of Graph Regularization)**.** *Let $\mathcal{F}$ be the class of linear functions defined in Eq. (1), where $\mathcal{V}$ is the set defined in Eq. (4), and let $\mathbf{Z} \stackrel{\text{def}}{=} [\mathbf{z}_1, \ldots, \mathbf{z}_N] \in \mathbb{R}^{n \times N}$ be the sample set on which the Rademacher complexity $\hat{\mathcal{R}}_N(\mathcal{F})$ is evaluated. Then*

$$\hat{\mathcal{R}}_N(\mathcal{F}) \leq B \sqrt{\frac{\operatorname{tr}\left[\mathbf{Z}^\top \left((1-\eta)\mathbf{Z}\mathbf{L}\mathbf{Z}^\top + \eta \mathbf{I} \right)^{-1} \mathbf{Z} \right]}{N}}$$

Proof. See the supplementary material. □

Graph regularization includes weight decay regularization as the special case $\eta = 1$. In the definition of $\mathcal{V}$ and Theorem 1, we have excluded the value $\eta = 0$ so that the inverse $[(1-\eta)\mathbf{Z}\mathbf{L}\mathbf{Z}^\top + \eta \mathbf{I}]^{-1}$ exists. However, in our experiments with only a graph regularizer ($\eta = 0$), we have also observed strong generalization performance.

Reducing the Rademacher complexity of a single linear layer is an important step in reducing the Rademacher complexity of a multi-layer network. The bound in Theorem 1 depends on the eigenvalues of the Laplacian $\mathbf{L}$, which in turn depend on the edge weights $\mathbf{W}$ of the graph. Whatever graph we choose, the Rademacher complexity of graph regularization will be at least as good as that of weight decay. We now provide an example to show that by choosing an appropriate graph we can achieve significant improvements.

Example. Consider the classical MNIST data. There are 10 classes in the dataset, and samples are 784-dimensional (28x28 images). We randomly select N samples (N/10 samples per class), remove the sample mean, and form the $784 \times N$ matrix $\mathbf{Z}$. The edge weights $\mathbf{W}$ are given by $W_{i,j} = \exp\left(-\frac{\|\mathbf{z}_i - \mathbf{z}_j\|^2}{\gamma^2} \right)$, where γ is the average of all pairwise distances. We form the Laplacian $\mathbf{L}$ and evaluate the bound given in Theorem 1 for several values of N. We observe in Fig. 2 that the upper bound decreases significantly as η moves away from 1 (weight decay), showing the added value of graph regularization. As the sample size N increases, the bound decreases steadily, consistent with our intuition about the generalization error.

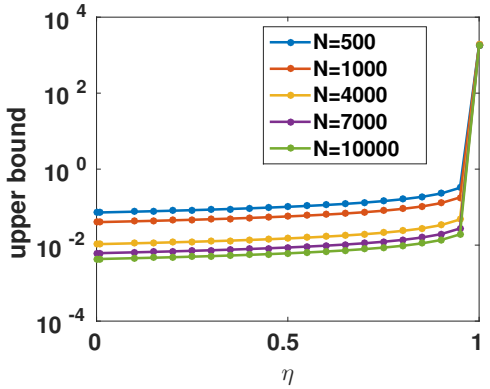


Figure 2: Evaluation of the upper bound (Theorem 1) on the Rademacher complexity for the MNIST benchmark.

2.2. Analysis: Regularizing Multiple Layers

We now extend our analysis to include the effects in intermediate layers of pooling and of activation functions such as rectifiers. We consider a K-layer network that maps

an input $\mathbf{x}$ onto a multidimensional feature $\mathbf{g}(\mathbf{x})$, then restrict to a single dimension to obtain a scalar $g(\mathbf{x})$ given by

$$g(\mathbf{x}) = \mathbf{v}_K^\top s_{K-1}(\cdots s_2(\mathbf{V}_2 s_1(\mathbf{V}_1 \mathbf{x}))), \text{ where } \mathbf{v}_K, \mathbf{V}_i \in \mathscr{V}, \tag{5}$$

where the nonlinear mapping $s_k(\cdot)$ represents activation and pooling. $\mathbf{V}_1, \ldots, \mathbf{V}_{K-1}$ are matrices representing linear layers, and $\mathbf{v}_k$ is a vector that maps input to a single coordinate in the final output feature. The $\mathbf{V}_1, \ldots, \mathbf{V}_{K-1}$ and $\mathbf{v}_K$ are taken from a set $\mathscr{V}$ that is defined by the property

$$\frac{1}{N} \sum_{i,j=1}^{N} W_{i,j}[g(\mathbf{x}_i) - g(\mathbf{x}_j)]^2 \leq \frac{B^2}{4}. \tag{6}$$

The symmetric edge weights $\mathbf{W}$ encode the relationships between the N input samples samples $[\mathbf{x}_1, \ldots, \mathbf{x}_N]$.

Set $g(\mathbf{X}) = [g(\mathbf{x}_1), \ldots, g(\mathbf{x}_N)]^\top$ and recall that

$$\sum_{i,j=1}^{N} W_{i,j}[g(\mathbf{x}_i) - g(\mathbf{x}_j)]^2 = g(\mathbf{X}) \mathbf{L} g(\mathbf{X})^\top,$$

where $\mathbf{L}$ is as before the Lapacian of $\mathbf{W}$. As above, we want to work with a positive definite matrix so we add a small multiple of the identity matrix $\mathbf{I}_n$. We now derive an upper bound on the empirical Rademacher complexity for the function class $\mathcal{G}$ defined by

$$\mathcal{G} = \Big\{ g(\mathbf{x}) | \quad g(\mathbf{x}) = \mathbf{v}_K^\top s(\cdots s(\mathbf{V}_2 s(\mathbf{V}_1 \mathbf{x}))) \}, \text{where } g(\mathbf{X})(\mathbf{L} + \epsilon \mathbf{I}) g(\mathbf{X})^\top \leq \tfrac{NB^2}{4} \Big\}. \tag{7}$$

Theorem 2. $\hat{\mathcal{R}}_N(\mathcal{G}) \leq B \sqrt{\frac{\mathrm{tr}[(\mathbf{L}+\epsilon \mathbf{I}_n)^{-1}]}{N}}$

Proof. See supplementary material. □

Remark 2. *Theorem 2 provides an upper bound on complexity that is not very sensitive to the number of layers in the network, in contrast to weight decay where complexity is exponential in the number of layers [20].*

2.3. The *GraphConnect* Algorithm

The theory presented in Section 2.1 and 2.2 applies to scalar valued functions of vector inputs, but the generalization to vector valued functions is straightforward. Eq. (2) becomes

$$\frac{1}{N} \sum_{i,j=1}^{N} W_{i,j} \|\mathbf{f}(\mathbf{z}_i) - \mathbf{f}(\mathbf{z}_j)\|^2 \leq \frac{B^2}{4}, \tag{8}$$

where $\mathbf{f}(\mathbf{z})$ is the multidimensional output of a linear layer. Similarly, when we consider a K-layer network, Eq. (6) becomes

$$\frac{1}{N} \sum_{i,j=1}^{N} W_{i,j} \|\mathbf{g}(\mathbf{x}_i) - \mathbf{g}(\mathbf{x}_j)\|^2 \leq \frac{B^2}{4}, \tag{9}$$

where $\mathbf{g}(\mathbf{x})$ is the multidimensional feature output at layer K.

Fig. 3 describes two flavors of *GraphConnect*, that are two different ways of using a graph to regularize a neural network. The first *GraphConnect-One* uses the constraint given by Eq. (8) to regularize individual layers, and this method can be applied to all layers or to some layers but not others. The second *GraphConnect-All* uses the constraint given by Eq. (9) to regularize the final learned features. Implementation requires multiplying the *GraphConnect* regularizer by some $\lambda > 0$ then adding this quantity to the original objective function used to train the neural network. We conclude this section by showing that both *GraphConnect* regularization schemes require only minor changes to the standard back-propagation algorithm.

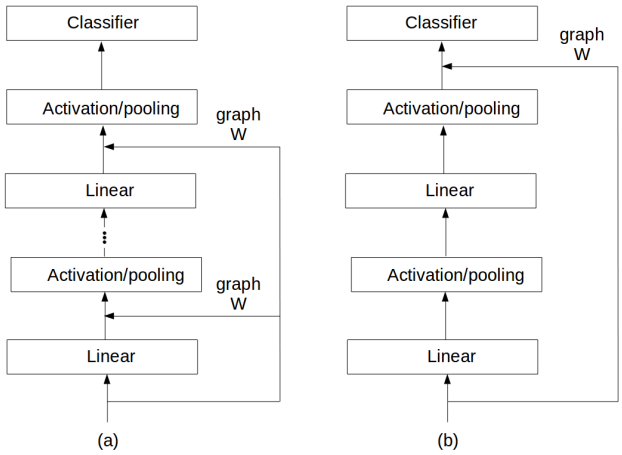


Figure 3: (a) *GraphConnect-One* regularizes individual linear layers so that individual outputs align with a graph $\mathbf{W}$; (b) *GraphConnect-All* regularizes final output features to align with a graph $\mathbf{W}$.

Gradient Descent Solver for *GraphConnect-One* We seek to minimize

$$\ell_{emp} + J,$$

where J is the *GraphConnect* regularization term given by

$$J = \lambda \sum_{i,j=1}^{N} W_{i,j} \|\mathbf{f}(\mathbf{z}_i) - \mathbf{f}(\mathbf{z}_j)\|^2 = \lambda \mathrm{tr}[\mathbf{f}(\mathbf{Z}) \mathbf{L} \mathbf{f}(\mathbf{Z})^\top].$$

The gradient of J w.r.t. $\mathbf{f}(\mathbf{Z})$ is

$$\frac{\partial J}{\partial \mathbf{f}(\mathbf{Z})} = 2\lambda \mathbf{f}(\mathbf{Z}) \mathbf{L}.$$

So we just need to add an extra term $2\lambda \mathbf{f}(\mathbf{Z}) \mathbf{L}$ to the original gradient with respect to $\mathbf{f}(\mathbf{Z})$.

Gradient Descent Solver for *GraphConnect-All* The

analysis is very similar and we just need to add an extra term $2\lambda \mathbf{g}(\mathbf{X})\mathbf{L}$ to the original gradient with respect to $\mathbf{g}(\mathbf{X})$. *GraphConnect* regularization requires only minor modifications to standard back propagation algorithms and is very efficient in practice. In the next section we demonstrate that when the training set is small, it can lead to significant improvements in classification performance.

2.4. Discussion

Generalization error differs from empirical Rademacher complexity by a multiplicative factor that is determined by the overall softmax classifier [18]. The method of graph regularization can be applied to a single layer or to multiple layers. It is designed to learn attributes that are present in data samples, in contrast to weight decay, *Dropout* [7], and *Dropconnect* [18] which are designed to prevent learning non-attributes (overfitting to random error or noise). The approach taken in both *Dropout* and *Dropconnect* is to introduce randomness into training so that learned network is in some sense a statistical average of an ensemble of realizations. They complement our approach of regularizing output using a graph and we plan to explore a combination of these approaches in future work.

3. Experiments

We are particularly interested in how the graph regularization methods in Section 2.3 compare with the conventional weight decay. The experiments section are organized as follows. In section 3.1, we use the MNIST dataset to show that the generalization error of *GraphConnect-One* and *GraphConnect-All* are both signifiantly smaller than weight decay, and they achieve superior classification accuracy especially when the training set is small. Section 3.2 presents extensive comparisons between the proposed *GraphConnect* and weight decay on CIFAR-10 and SVHN. Section 3.3 demonstrates the improved performance of *GraphConnect* on the face verification task.

3.1. MNIST Proof of Concept

The MNIST dataset contains approximately 60,000 training images (28×28) and 10,000 test images. While state-of-the-art methods often use the entire training set, we are interested in quantifying what is possible with much smaller training sets. Table 1 describes the network architecture that we use to compare graph regularization with the standard weight decay.

We begin by using 500 training samples (50 per class) to train two neural networks. Image mean is estimated from the training set and subtracted as a preprocessing step. The first experiment uses *GraphConnect-One* to regularize the outputs of layers 3 and 5. The second uses *GraphConnect-All* to regularize the output of layer 6. The graph edge

Table 1: Network architecture common to the MNIST experiments.

Layer	Type	Parameters
1	conv	size: $5 \times 5 \times 1 \times 20$ stride: 1, pad: 0
2	maxPool	size: 2×2, stride: 2, pad: 0
3	conv	size: $5 \times 5 \times 20 \times 50$ stride: 1, pad: 0
4	maxPool	size: 2×2, stride: 2, pad: 0
5	conv	size: $4 \times 4 \times 50 \times 500$ stride: 1, pad: 0
6	ReLu	N/A

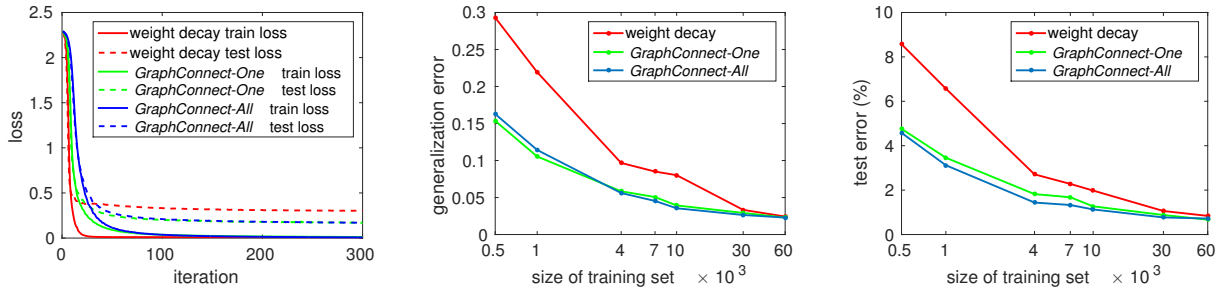
weights $W_{i,j}$ are given by

$$W_{i,j} = \begin{cases} \exp\left(\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma_c^2}\right) & \text{if } \mathbf{x}_i, \mathbf{x}_j \in \text{class } c \\ 0 & \text{if } \mathbf{x}_i, \mathbf{x}_j \in \text{different classes} \end{cases},$$

where the diameter σ_c is an estimate of the average distance between pairs of samples in class c. We tune the regularizer for each network, and select the value that maximizes classification accuracy. We then calculate the empirical loss ℓ_{emp} on the training set, the loss ℓ on the test data, and the generalization error $\ell_{emp} - \ell$. The results presented in Fig. 4 show that the two variants of *GraphConnect* have approximately the same generalization error and that both are superior to weight decay, in particular with small training sets. Fig. 1 illustrates why *GraphConnect* outperforms weight decay. We take 1000 test samples, transform them using the learned networks, then embed the features into a two dimensional coordinate via PCA and represent each class by a different color. All learned features (Fig. 1b to 1d) are more discriminative than the initial data (Fig. 1a). Moreover, it is evident that both *GraphConnect-One* and *GraphConnect-All* better distinguishes the different classes than does weight decay.

Next we vary the size of the training set from 500 to 6,000, and repeat the above experiment. We tune the regularizer for each method and select the value to maximize classification accuracy. When the number of training samples is small, Fig. 4b shows *GraphConnect* yields a generalization error that is significantly smaller than that yielded by weight decay. Performance becomes broadly similar as the size of the training set increases. The same trend is evident in Fig. 4c which compares classification accuracy of the three methods.

Since performance of the two variants of *GraphConnect* is broadly similar, and since *GraphConnect-One* involves tuning multiple regularizers, we focus on *GraphConnect-All* in the sequel.



(a) Evolution of training and test loss with the number of training iterations (500 training samples).

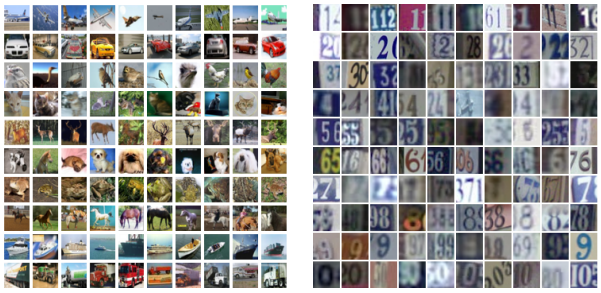
(b) Dependence of generalization error on the size of the training set (after 100 iterations).

(c) Dependence of classification error on the size of the training set (after 100 iterations).

Figure 4: Comparing *GraphConnect* against weight decay on the MNIST dataset.

3.2. Comparison on CIFAR-10 and SVHN

CIFAR-10 and SVHN are benchmark RGB image datasets, each containing 10 classes, that are more challenging than the MNIST benchmark because of more significant intra-class variation (see Fig. 5). We compare regularization using *GraphConnect-All* with regularization using weight decay on these two datasets. Table 2 specifies the network architecture (similar to [7]), all images are mean-subtracted in a preprocessing step, and the graph weights $\mathbf{W}$ used in *GraphConnect-All* are computed in the same fashion as for the MNIST experiment. The network transforms sample images into 2048-dimensional features that are input to a softmax classifier, and the cross entropy loss is evaluated on the classifier output.



(a) (b)

Figure 5: Representative images from (a) CIFAR-10 and (b) SVHN[1], where there are 10 classes and images in the same row are taken from the same class. Data variation within each class is much more significant than in the MNIST benchmark.

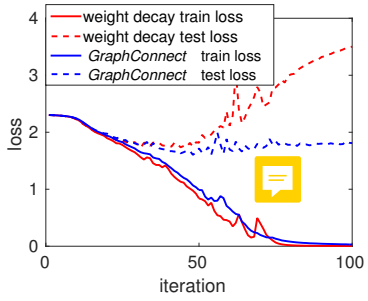
We first train the network on a very small training set (subset of the whole training ensemble), and evaluate the empirical loss ℓ_{emp} on the training set and the expected loss ℓ on test set. The regularizer for each method is chosen such that the best classification accuracy is achieved. Fig 6a and 7a show how ℓ_{emp} and ℓ vary throughout iterations on these two datasets. Weight-decay overfits on the training set and its test loss increases after some iterations. In contrast, *GraphConnect* has a smaller gap between ℓ and ℓ_{emp}, indicating smaller generalization error.

[1]Arts by courtesy of https://www.kaggle.com/c/cifar-10 and http://ufldl.stanford.edu/housenumbers/

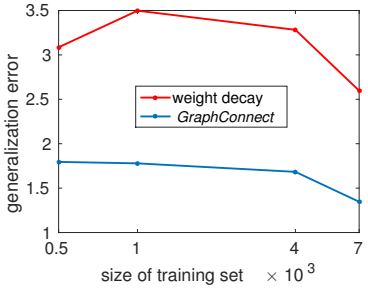
Table 2: Network architecture common to CIFAR-10 and SVHN experiments.

Layer	Type	Parameters
1	conv	size: $5 \times 5 \times 3 \times 96$ stride: 1, pad: 2
2	ReLu	N/A
3	maxPool	size: 3×3, stride: 2, pad: 0
4	conv	size: $5 \times 5 \times 96 \times 128$ stride: 1, pad: 2
5	ReLu	N/A
6	maxPool	size: 3×3, stride: 2, pad: 0
7	conv	size: $4 \times 4 \times 50 \times 500$ stride: 1, pad: 0
8	ReLu	N/A
9	maxPool	size: 3×3, stride: 2, pad: 0
10	fully connected	#output: 2048
11	ReLu	N/A
12	fully connected	#output: 2048
13	ReLu	N/A

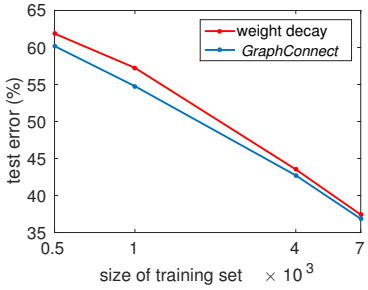
Next we vary the size of the training set and evaluate the generalization error (Figs. 6b and 7b) and classification accuracy (Fig. 6c and 7c). *GraphConnect* exhibits smaller generalization error than weight decay. The classification error is also smaller but less significant than the generalization error since cross entropy loss is a nonlinear func-



(a) Evolution of training and test loss with the number of training iterations (1,000 training samples)

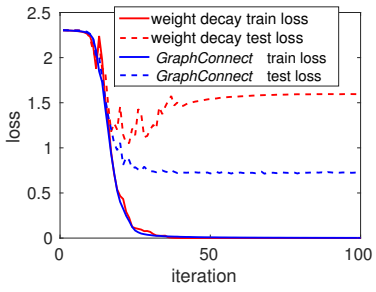


(b) Dependence of generalization error on the size of the training set (after 100 iterations).

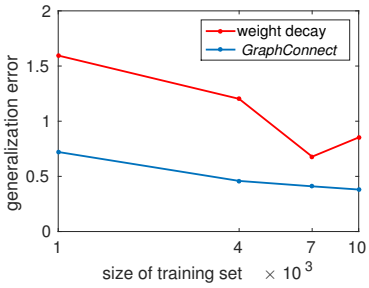


(c) Dependence of classification error on the size of the training set (after 100 iterations).

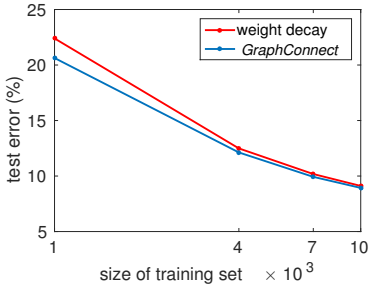
Figure 6: Comparing *GraphConnect* against weight decay on the CIFAR-10 dataset.



(a) Evolution of training and test loss with the number of training iterations (1,000 training samples)



(b) Dependence of generalization error on the size of the training set (after 100 iterations).



(c) Dependence of classification error on the size of the training set (after 100 iterations).

Figure 7: Comparing *GraphConnect* against weight decay on the SVHN dataset.

tion w.r.t. the probability produced by softmax classifier. Compared with the MNIST example (Section3.1), the improvement in classification accuracy is modest because in contrast to the MNIST benchmark, the intra-class variation here is substantial. More sophisticated preprocessing methods such as contrast normalization [13] and ZCA whitening [9] will reduce intra-class variation, and we would expect them to improve performance further. We leave this direction for future research.

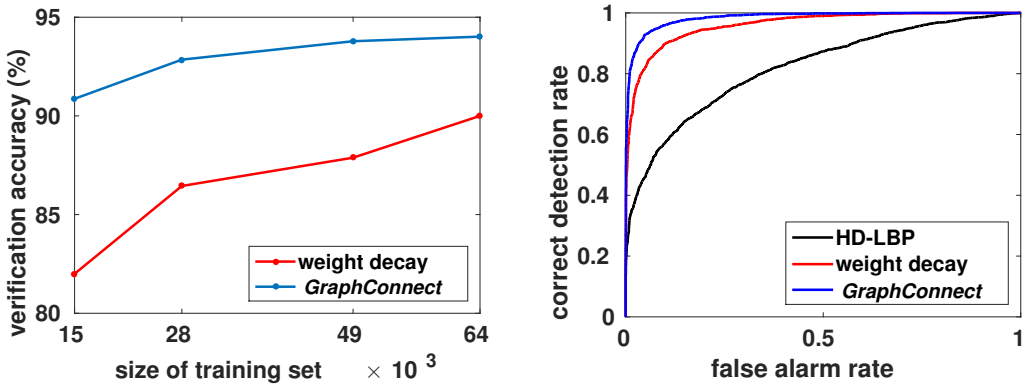
3.3. Face Verification on LFW

We now evaluate GraphConnect on face verification, using the Labeled Faces in the Wild (LFW) benchmark dataset. The face verification task is to decide, when presented with a pair of facial images, whether the two images represent the same subject. Impressive verification accuracies are possible when deep neural networks are able to train on extremely large labeled training sets [14, 16]. The training sets are often proprietary, making it difficult to reproduce these successes, but that is not our aim in this work. Given the same network architecture, we seek to compare the performance of *GraphConnect* with that of weight decay.

We adopt the experimental framework used in [3], and train a deep network on the WDRef dataset, where each face is described using a high dimensional LBP feature (available at [2]) that is reduced to a 5,000-dimensional feature using PCA. The WDRef dataset is significantly smaller than the proprietary datasets in [14, 15, 16]. For example, [16] uses 4.4 million labeled faces from 4,030 individuals. [14] and [15] use 202,599 labeled faces from 10,177 individuals, while WDRef contains 2,995 subjects with only about 30 samples per subject, clearly a much more challenging task.

We consider the two-layer fully connected network described in Tab. 3, where the activation function is a rectifier. The network transforms a 5,000-dimensional input vector to a 2,000-dimensional feature vector, which is then input to a softmax classifier. The network parameters are learned using WDRef and the testing is carried out on the LFW dataset. Our focus is the expressiveness of the learned feature, so we do not employ advanced verification methods such as those used in [3] (those will make the study of

[2] http://home.ustc.edu.cn/chendong/



(a) (b)

Figure 8: (a) Verification accuracy of GraphConnect and weight decay as a function of the size of the training set; (b) ROC curves when using 64,000 training samples.

the network itself very obscure). Instead, we simply compute the Euclidean distance between a pair of (learned) face features and compare it with a threshold to make a decision.

Table 3: Fully connected network for face verification.

Layer	Type	Parameters
1	fully connected	#output: 2000
2	ReLu	N/A
3	fully connected	#output: 2000
4	ReLu	N/A

Table 4: Verification accuracies and AUCs when using a training set of size 64,000

Method	Accuracy (%)	AUC ($\times 10^{-2}$)
HD-LBP	74.73	82.22 ± 1.00
weight decay	90.00	96.14 ± 0.61
GraphConnect	**94.02**	$\mathbf{98.48 \pm 0.21}$

We vary the number of training samples per class and evaluate verification performance. We report results for the value of the regularization parameter that optimizes verification accuracy. Fig. 8a compares verification accuracies for *GraphConnect* and weight decay as a function of the size of the training set. *GraphConnect* consistently outperforms weight decay.

Fig. 8b compares the ROCs curves when a training set of size 64K is used. Corresponding Area Under Curves (AUCs) are reported in Tab. 4. As a baseline, we also evaluate the verification performance on the initial LBP features (without any learning). We observe from Fig. 8b and Tab. 4 that the learned features significantly outperform the initial LBP features, while *GraphConnect* further improves upon weight decay, validating the effectiveness of *GraphConnect* regularization when training set is small.

4. Conlusion

We have proposed *GraphConnect*, a data-dependent framework for regularizing deep neural networks, and we have compared performance against data-independent methods of regularization that are in widespread use. We proved that the empirical Rademacher complexity of *GraphConnect* is smaller than that of weight decay, justifying our claim that it is better at preventing overfitting. We presented experimental results that validate our theoretical claims, showing that when the training set is small the improvements in generalization error are significant. Our proposed framework is complementary to data-independent approaches that prevent overfitting, such as Dropout and DropConnect, and future work will explore the value of combining these methods.

References

[1] P. Bartlett. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*, 44(2):525–536, 1998. 1

[2] P. L. Bartlett and S. Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2002. 1, 2

[3] D. Chen, X. Cao, L. Wang, F. Wen, and J. Sun. Bayesian face revisited: A joint formulation. In *In European Conference on Computer Vision (ECCV)*, 2012. 7

[4] D. Cohn and G. Tesauro. How tight are the vapnik-chervonenkis bounds? *Neural Computation*, 4(2):249–269, 1992. 1

[5] T. Evgeniou, C. Micchelli, and M. Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6:615–637, 2005. 1

[6] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. C. Courville, and Y. Bengio. Max-out networks. In *In Proceedings of the 30th International Conference on Machine Learning*, 2013. 1

[7] G. 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. 1, 5, 6

[8] V. Koltchinskii and D. Panchenko. Empirical margin distributions and bounding the generalization error of combined classifiers. *Annals of Statistics*, 30(1):1–50, 2002. 2

[9] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009. 7

[10] B. Mikhail and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003. 1

[11] B. Mikhail, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006. 1

[12] B. Neyshabur, R. Tomioka, and N. Srebro. Norm-based capacity control in neural networks arxiv preprint arxiv:1503.00036 (2015). In *The 28th Conference on Learning Theory (COLT)*, 2015. 1

[13] P. Sermanet, S. Chintala, and Y. LeCun. Convolutional neural networks applied to house numbers digit classification. In *International Conference on Pattern Recognition (ICPR)*, 2012. 7

[14] Y. Sun, Y. Chen, X. Wang, and X. Tang. Deep learning face representation by joint identification-verification. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1988–1996, 2014. 7

[15] Y. Sun, X. Wang, and X. Tang. Deep learning face representation from predicting 10,000 classes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1891–1898, 2014. 7

[16] Y. Taigman, M. Yang, M. A. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1701–1708, 2014. 7

[17] V. Vapnik. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5):988999, 1999. 1, 2

[18] L. Wan, M. Zeiler, S. Zhang, Y. LeCun, and R. Fergus. Regularization of neural networks using dropconnect. In *In Proceedings of the 30th International Conference on Machine Learning*, 2013. 1, 2, 3, 5

[19] J. Weston, F. Ratle, H. Mobahi, and R. Collobert. Deep learning via semi-supervised embedding. *In Neural Networks: Tricks of the Trade. Springer Berlin Heidelberg*, pages 639–655, 2012. 2

[20] H. Xu and S. Mannor. Robustness and generalization. *Machine learning*, 86(3):391–423, 2012. 4