Deeply-Supervised Nets

Chen-Yu Lee *
Dept. of ECE, UCSD
chl260@ucsd.edu

Saining Xie *
Dept. of CSE and CogSci, UCSD
s9xie@ucsd.edu

Patrick W. Gallagher
Dept. of CogSci, UCSD
patrick.w.gallagher@gmail.com

Zhengyou Zhang
Microsoft Research
zhang@microsoft.com

Zhuowen Tu
Dept. of CogSci, UCSD
ztu@ucsd.edu

Abstract

We propose deeply-supervised nets (DSN), a method that simultaneously minimizes classification error and improves the directness and transparency of the hidden layer learning process. We focus our attention on three aspects of traditional convolutional-neural-network-type (CNN-type) architectures: (1) transparency in the effect intermediate layers have on overall classification; (2) discriminativeness and robustness of learned features, especially in early layers; (3) training effectiveness in the face of "vanishing" gradients. To combat these issues, we introduce "companion" objective functions at each hidden layer, in addition to the overall objective function at the output layer (an integrated strategy distinct from layer-wise pre-training). We also analyze our algorithm using techniques extended from stochastic gradient methods. The advantages provided by our method are evident in our experimental results, showing state-of-the-art performance on MNIST, CIFAR-10, CIFAR-100, and SVHN.

1 Introduction

Neural networks have recently resurged, most prominently in deep learning (DL). The application of DL techniques to large training sets has yielded significant performance gains in image classification [12, 15] and speech recognition [5]. However, even though hierarchical neural networks [8, 11, 16] have shown great promise in automatically learning thousands or even millions of features for pattern recognition, there nonetheless remain many fundamental open questions about DL.

These open questions arise with various aspects of current DL frameworks: the features learned at hidden layers (early hidden layers in particular) are not always "transparent" in their meaning and at times display reduced discriminativeness [31]; "vanishing" gradients can sometimes lead to training difficulty [9, 20]; despite some theoretical work [7], mathematical understanding of DL is at an early stage. Notwithstanding such issues, DL has proven capable of automatically learning rich hierarchical features combined within an integrated network. Recent techniques such as dropout [12], dropconnect [17], pre-training [5], and data augmentation [22] bring enhanced performance from various perspectives. Beyond this, recent code- and experience-sharing [12, 6, 2] has greatly sped the adoption and advance of DL in and out of machine learning.

In this paper, we present a new DL approach we call deeply-supervised nets (DSN). The central idea of DSN is to provide integrated direct supervision to the hidden layers, rather than the standard approach of providing supervision only at the output layer and propagating this supervision back to earlier layers. We provide this integrated direct hidden layer supervision by introducing *companion objective functions* for each hidden layer; these companion objective functions can be seen as an additional (soft) constraint (or as a new regularization) within the learning process. Our experiments show significant performance gains relative to existing methods. In addition, we use analysis techniques from stochastic gradient methods to investigate a restricted setting in which incorporating companion objective functions directly leads to improved convergence rate. The advantage of such integrated deep

Appearing in Proceedings of the 18^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2015, San Diego, CA, USA. JMLR: W&CP volume 38.
 [*equal contribution]

supervision is evident: (1) **for small training data and relatively shallower networks**, deep supervision functions as a strong "regularization" for classification accuracy and learned features; (2) **for large training data and deeper networks** deep supervision makes it convenient to exploit the significant performance gains that extremely deep networks can bring by improving otherwise problematic convergence behavior.

The major characteristic of DSN is its integrated formulation of deep supervision via companion objectives, which in the simplest case are of a form nearly identical to the objective function at the output layer, as shown in Eq. (4). This integrated formulation of deep supervision by means of companion objectives immediately sets us apart from all previous approaches. In effect, we perform network optimization incorporating companion objectives as a proxy for hidden layer feature quality. The combined learning in our integrated formulation stands in contrast to previous research in which greedy layer-wise pre-training was performed separately as either initialization or fine-tuning, a strategy that resulted in overfitting [1].

There are a few other relevant examples: in [26], they introduce an additional term only at the output layer (to use the discriminative label information to assist with learning the generative model), in strong distinction to our approach of deep supervision in a supervised learning setting; in [30] semi-supervised learning is carried out by way of an embedding and no supervised labels are involved, in contrast to our supervised approach; the emphasis of [23] is on providing the output classifier with features learned from the hidden layers — crucially, their technique does not target the training and regularization benefits of our DSN. We also note a parallel development bearing some similarity to DSN: GoogLeNet [28] uses supervision on 2 hidden layers in a 22-layer CNN, but the two approaches differ in focus, formulation, emphasis, and analysis. On the theoretical side, analysis of deep learning is still at an early stage of development; we mention [7] as a recent example. Finally, we mention [4]: this work re-expresses the original output layer classification problem in an equivalent decouple-able form. Their purpose is to apply a standard optimization technique to this decoupled equation; this approach is thus quite distinct from ours. In particular, this approach does not leverage any label information to achieve benefits such as regularization. Our experiments are primarily based on L2SVM objectives, previously used in DL by [29] for the output layer only; we also show that our approach of introducing companion objectives is not dependent on the use of L2SVM objectives — when using softmax we find similar performance improvements. Our experiments show consistent improvement of DSN-L2SVM and DSN-Softmax over CNN-L2SVM and CNN-Softmax, respectively.

We use MNIST as the primary dataset in our experiments; in these MNIST experiments we build DSN on top of Theano [2]. For experiments on non-MNIST datasets we build DSN on top of NIN [18]. Our DSN framework achieves state-of-the-art results on each of the datasets investigated: MNIST, CIFAR-10, CIFAR-100, and SVHN. Finally, we note that our approach is independent of techniques such as averaging [22], dropconnect [17], and Maxout [10]; thus, we expect that combining DSN with these techniques might result in even greater classification error reduction.

2 Deeply-Supervised Nets

Our approach is built using the infrastructure provided by existing supervised CNN-type frameworks [16, 6, 2]. We extend them by introducing a classifier, either SVM or Softmax, at hidden layers.

2.1 Motivation

Our motivation for introducing classifiers at hidden layers comes from the following observation: in general, a discriminative classifier trained on highly discriminative features will display better performance than a discriminative classifier trained on less discriminative features. If the features in question are the hidden layer feature maps of a deep network, this observation means that the performance of a discriminative classifier trained using these hidden layer feature maps can serve as a proxy for the quality/discriminativeness of those hidden layer feature maps. We also expect this deep supervision to alleviate the common problem of "vanishing" gradients. One concern with direct pursuit of feature discriminativeness at all hidden layers is that this might interfere with the overall network performance; our experimental results indicate that this is not the case. Our additional deep feedback is brought in by associating a "companion" classification output with each hidden layer. We may think of this companion output as analogous to the final output that a truncated network would have produced. Backpropagation of error proceeds as usual, with the crucial difference that we backpropagate not only from the final layer but also from our local companion output. The empirical results suggest the following main properties of the companion objective: (1) it is a type of feature regularization (albeit an unusual one) — it leads to reduction in testing error, while not necessarily reducing the training error; (2) it results in improved convergence behavior, requiring much less manual tweaking (particularly for very deep networks).

2.2 Formulation

Here we focus on the supervised learning case. We denote our input training data set by $S = \{(\mathbf{X}_i, y_i), i = 1 \dots N\}$, where sample $\mathbf{X}_i \in R^n$ denotes the raw input data and $y_i \in \{1, \dots, K\}$ denotes the corresponding ground truth label for sample $\mathbf{X}_i$. We subsequently drop the subscript i for notational simplicity, since we consider each sample independently. The goal of deep neural networks, specifically convolutional neural networks (CNN) [16], is to learn layers of filters and weights for minimization of output layer classification error. In our present discussion, we absorb the bias term into the weight parameters and do not differentiate weights from filters. Furthermore, we denote a recursive function for each layer $m = 1 \dots M$ as

$$\mathbf{Z}^{(m)} = f(\mathbf{Q}^{(m)}), \quad \text{and} \quad \mathbf{Z}^{(0)} \equiv \mathbf{X}, \tag{1}$$

$$\mathbf{Q}^{(m)} = \mathsf{W}^{(m)} * \mathbf{Z}^{(m-1)}, \tag{2}$$

where M denotes the total number of layers, $\mathsf{W}^{(m)}$, $m = 1 \dots M$ are the filters/weights to be learned, $\mathbf{Z}^{(m-1)}$ is the feature map produced at layer $m-1$, $\mathbf{Q}^{(m)}$ refers to the convolved/filtered responses on the previous feature map, and $f(\cdot)$ is a pooling function on $\mathbf{Q}$. Combining all layers of weights gives

$$\mathsf{W} = (\mathsf{W}^{(1)}, \dots, \mathsf{W}^{(M)}).$$

We next associate one classifier with each hidden layer[1]. We denote the corresponding weights by

$$\mathbf{w} = (\mathbf{w}^{(1)}, \dots, \mathbf{w}^{(M-1)}),$$

in addition to the W from the standard CNN framework. For ease of reference, we write the total objective function as

$$F(\mathsf{W}) \equiv \mathcal{P}(\mathsf{W}) + \mathcal{Q}(\mathsf{W}), \tag{3}$$

in which we have the output objective $\mathcal{P}(\mathsf{W}) \equiv \|\mathbf{w}^{(out)}\|^2 + \mathcal{L}(\mathsf{W}, \mathbf{w}^{(out)})$ and the summed companion objectives $\mathcal{Q}(\mathsf{W}) \equiv \sum_{m=1}^{M-1} \alpha_m [\|\mathbf{w}^{(m)}\|^2 + \ell(\mathsf{W}, \mathbf{w}^{(m)}) - \gamma]_+$. We denote the classifier weights for the output layer by $\mathbf{w}^{(out)}$. Our total combined objective function is then

$$\|\mathbf{w}^{(out)}\|^2 + \mathcal{L}(\mathsf{W}, \mathbf{w}^{(out)}) + \sum_{m=1}^{M-1} \alpha_m [\|\mathbf{w}^{(m)}\|^2 + \ell(\mathsf{W}, \mathbf{w}^{(m)}) - \gamma]_+, \tag{4}$$

where

$$\mathcal{L}(\mathsf{W}, \mathbf{w}^{(out)}) = \sum_{y_k \neq y} [1 - < \mathbf{w}^{(out)}, \phi(\mathbf{Z}^{(M)}, y) - \phi(\mathbf{Z}^{(M)}, y_k) >]_+^2 \tag{5}$$

and

$$\ell(\mathsf{W}, \mathbf{w}^{(m)}) = \sum_{y_k \neq y} [1 - < \mathbf{w}^{(m)}, \phi(\mathbf{Z}^{(m)}, y) - \phi(\mathbf{Z}^{(m)}, y_k) >]_+^2 \tag{6}$$

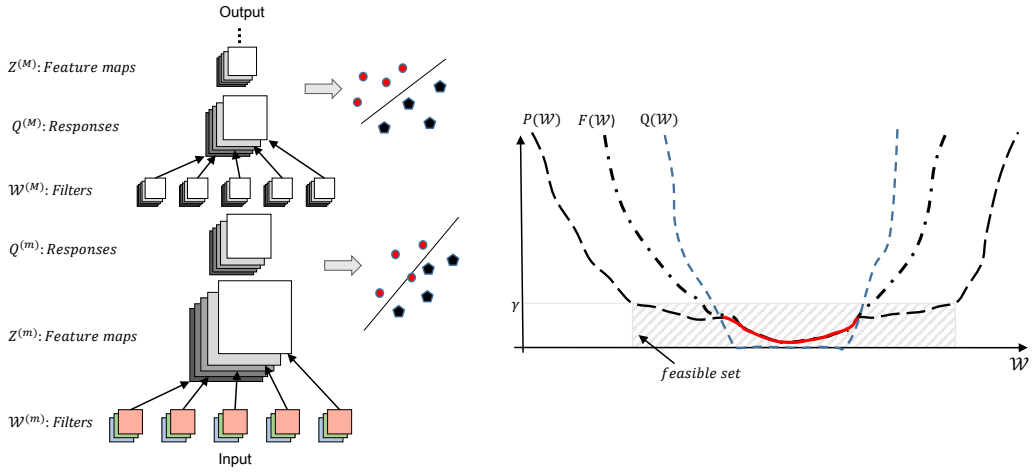
[1]Our derivation is stated for L2SVM but we later show experimental results for both L2SVM and softmax. We also tested L1SVM, finding results that differed negligibly from those of L2SVM.

We train the DSN model using SGD. The gradient w.r.t W follows the conventional CNN-based model, plus the gradient that comes from the hidden layer direct supervision; we also use companion objective zero-ing, described in the following.

We refer to $\mathcal{L}(\mathsf{W}, \mathbf{w}^{(out)})$ as the *overall loss* (associated with the output layer) and to $\ell(\mathsf{W}, \mathbf{w}^{(m)})$ as the *companion loss* (associated with the hidden layers); in the L2SVM setting these are both (squared) hinge losses of prediction errors. Intuitively, in addition to learning convolution kernels and weights, W (as in the standard CNN model [16]) we incorporate an additional objective at each hidden layer associated with good label prediction for that layer; this additional objective strongly favors features that are discriminative/sensible at each hidden layer. In Eq. (4), $\|\mathbf{w}^{(out)}\|^2$ and $\mathcal{L}(\mathsf{W}, \mathbf{w}^{(out)})$ are, respectively, the margin and the (squared) hinge loss of the classifier output layer; we omit the balance parameter C in front of the (squared) hinge loss for notational simplicity. In Eq. (5), $\|\mathbf{w}^{(m)}\|^2$ and $\ell(\mathsf{W}, \mathbf{w}^{(m)})$ are, respectively, the margin and the (squared) hinge loss of the hidden layer classifier.

Note that for each $\ell(\mathsf{W}, \mathbf{w}^{(m)})$, the $\mathbf{w}^{(m)}$ directly depends on $\mathbf{Z}^{(m)}$, which is in turn dependent on $\mathsf{W}^1, .., \mathsf{W}^m$ up to the mth layer. Analogously, $\mathcal{L}(\mathsf{W}, \mathbf{w}^{(out)})$ depends on $\mathbf{w}^{(out)}$, which is decided by the entire W. The second term in Eq. (4) often is sent to zero in the course of training; this means that the overall goal of producing good classification of the output layer is not fundamentally altered and the companion objective acts as a type of regularization or as a proxy for discriminative features. One method by which we pursue this companion objective zero-ing is by having γ as a threshold (a hyper parameter) in the second term of Eq. (4): once the companion objective of each hidden layer falls to γ (or below), it vanishes and no longer contributes to the gradient update in the learning process. The mth balance parameter α_m represents a trade-off between the output objective and the corresponding companion objective. An alternative approach to companion objective zero-ing is to use of a simple decay function such as $\alpha_m \times 0.1 \times (1 - t/N) \rightarrow \alpha_m$ to enforce that the second term vanish after a certain number of iterations, where t is the epoch step and N is the total number of epochs. We have obtained promising preliminary results from each of these two approaches; this issue remains to be more fully explored.

The main difference between Eq. (4) and previous research using layer-wise supervised training is that we perform the optimization together with a term serving as a proxy for the hidden layer feature quality. This integrated learning is in contrast to previous



(a) DSN illustration (b) functions

Figure 1: Network architecture for the proposed deeply-supervised nets (DSN). The total objective function $F(\mathrm{W}) \equiv \mathcal{P}(\mathrm{W}) + \mathcal{Q}(\mathrm{W})$, in which we have the output objective $\mathcal{P}(\mathrm{W}) \equiv \|\mathbf{w}^{(out)}\|^2 + \mathcal{L}(\mathrm{W}, \mathbf{w}^{(out)})$ and the summed companion objectives $\mathcal{Q}(\mathrm{W}) \equiv \sum_{m=1}^{M-1} \alpha_m [\|\mathbf{w}^{(m)}\|^2 + \ell(\mathrm{W}, \mathbf{w}^{(m)}) - \gamma]_+$.

research in which greedy layer-wise pre-training was performed separately as either initialization or fine-tuning, a strategy that resulted in overfitting [1]. The state-of-the-art benchmark results show that the deep supervision approach does not display harmful levels of overfitting: as seen in Figure (2.c), while the training error for both CNN and DSN is eventually near zero, DSN shows lower test error and so demonstrates its advantage in generalization over standard CNN. This is a matter for further investigation, but the present results are very promising.

2.3 Stochastic Gradient Descent View

In addition to the present observation that learned features in CNN are not always intuitive or discriminative [31], broader discussion of deep neural network training difficulties has appeared in [9, 20]. As seen in Eq. (1) and (2), the change of the bottom layer weights gets propagated through many layers; this can lead to "vanishing" gradients [20]. Various techniques and parameter tuning tricks have been proposed to better train deep neural networks, such as pre-training. This difficulty in training partially raises some concerns about the DL frameworks. Here we provide analysis of our proposed formulation, in a hope to understand some aspects of its experimentally-observed advantage in effectiveness and efficiency over existing CNN-style approaches.

The objective function in deep neural networks is highly non-convex, a characteristic contributing to the current lack of a clear mathematical/statistical analysis of DL frameworks. A common approach is to sacrifice some realism for increased tractability, typically by restricting attention only to settings in which local convexity holds. Here we follow this example by supposing that our objective functions are locally λ-strongly convex and following analysis techniques from stochastic gradient descent [3].

The definition of λ-strong convexity is standard: A function $F(\mathrm{W})$ is λ-strongly convex on a set $\mathcal{W}$ if $\forall\ \mathrm{W}, \mathrm{W}' \in \mathcal{W}$ and any subgradient $\mathbf{g}$ at W,

$$F(\mathrm{W}') \geq F(\mathrm{W}) + < \mathbf{g}, \mathrm{W}' - \mathrm{W} > + \frac{\lambda}{2}\|\mathrm{W}' - \mathrm{W}\|^2. \quad (7)$$

The Stochastic Gradient Descent (SGD) update rule at step t is $\mathrm{W}_{t+1} = \Pi_{\mathcal{W}}(\mathrm{W}_t - \eta_t \hat{\mathbf{g}})$, where $\eta_t = \Theta(1/t)$ refers to the step factor and $\Pi_{\mathcal{W}}$ denotes projection onto the set $\mathcal{W}$. Let $\mathrm{W}^\star$ be the optimal solution, suppose that there are upper bounds for $\mathbb{E}[\|\mathrm{W}_T - \mathrm{W}^\star\|^2]$ and $\mathbb{E}[F(\mathrm{W}_T) - F(\mathrm{W}^\star)]$ in the strongly convex function setting [21], or for $\mathbb{E}[F(\mathrm{W}_T) - F(\mathrm{W}^\star)]$ in the convex function setting [24]. Here we make an attempt to understand the convergence of Eq. (3) w.r.t. $\mathbb{E}[\|\mathrm{W}_T - \mathrm{W}^\star\|^2]$ keeping in mind the assumed characteristics roughly illustrated in Figure (1.b). In [19], a convergence rate is given for M-estimators with locally convex function with compositional loss and regularization terms.

Definition We denote by $\mathcal{S}_\gamma(F) = \{\mathrm{W} \mid F(\mathrm{W}) \leq \gamma\}$ the γ-sublevel set, stated here for the function $F(\mathrm{W}) \equiv \mathcal{P}(\mathrm{W}) + \mathcal{Q}(\mathrm{W})$.

First we show that $\mathrm{W} \in \mathcal{S}_\gamma(\mathcal{Q})$ implies that $\mathrm{W} \in \mathcal{S}_\gamma(\mathcal{P})$. That is:

Lemma 1 *$\forall m, m' = 1 \ldots M-1$, and $m' > m$ if $\|\mathbf{w}^{(m)}\|^2 + \ell((\hat{\mathrm{W}}^{(1)}, .., \hat{\mathrm{W}}^{(m)}), \mathbf{w}^{(m)}) \leq \gamma$ then there exists $(\hat{\mathrm{W}}^{(1)}, .., \hat{\mathrm{W}}^{(m)}, .., \hat{\mathrm{W}}^{(m')})$ such that $\|\mathbf{w}^{(m')}\|^2 + \ell((\hat{\mathrm{W}}^{(1)}, .., \hat{\mathrm{W}}^{(m)} .., \hat{\mathrm{W}}^{(m')}), \mathbf{w}^{(m')}) \leq \gamma$.* [2]

[2] Note that we drop the $\mathrm{W}^{(j)}, j > m$ since the filters

Proof As we can see from an illustration of our network architecture shown in fig. (1.a), for all $(\hat{\mathsf{W}}^{(1)}, .., \hat{\mathsf{W}}^{(m)})$ such that $\ell((\hat{\mathsf{W}}^{(1)}, .., \hat{\mathsf{W}}^{(m)}), \mathbf{w}^{(m)}) \leq \gamma$ there is a "trivial" way to achieve this performance at the output layer: for each layer $j > m$ up to m', we let $\hat{\mathsf{W}}^{(j)} = \mathbf{I}$ and $\mathbf{w}^{(j)} = \mathbf{w}^{(m)}$, meaning that the filters will be identity matrices. This results in $\ell((\hat{\mathsf{W}}^{(1)}, .., \hat{\mathsf{W}}^{(m)}.., \hat{\mathsf{W}}^{(m')}), \mathbf{w}^{(m')}) \leq \gamma$. □

Remark Lemma 1 shows that a good solution for $\mathcal{Q}(\mathsf{W})$ is also a good one for $\mathcal{P}(\mathsf{W})$; of course, the converse needs not be the case. That is: a W that makes $\mathcal{P}(\mathsf{W})$ small may not necessarily produce features for which the hidden layers will have a small $\mathcal{Q}(\mathsf{W})$. As mentioned previously, $\mathcal{Q}(\mathsf{W})$ can be viewed as a regularization term. If it happens that $\mathcal{P}(\mathsf{W})$ possesses an essentially flat area (relative to the training data) in the vicinity of some local optimum of interest and it is ultimately the test error that we really care about, we would be able to focus on the setting of W, $\mathsf{W}^\star$, that will make both $\mathcal{Q}(\mathsf{W})$ and $\mathcal{P}(\mathsf{W})$ small. Therefore, it is not unreasonable to assume that $F(\mathsf{W}) \equiv \mathcal{P}(\mathsf{W}) + \mathcal{Q}(\mathsf{W})$ and $\mathcal{P}(\mathsf{W})$ share the same optimal $\mathsf{W}^\star$.

Let $\mathcal{P}(\mathsf{W})$ and $\mathcal{Q}(\mathsf{W})$ be locally strongly convex around $\mathsf{W}^\star$, $\|\mathsf{W}' - \mathsf{W}^\star\|^2 \leq D$ and $\|\mathsf{W} - \mathsf{W}^\star\|^2 \leq D$, with $\mathcal{P}(\mathsf{W}') \geq \mathcal{P}(\mathsf{W}) + < \mathbf{gp}, \mathsf{W}' - \mathsf{W} > + \frac{\lambda_1}{2}\|\mathsf{W}' - \mathsf{W}\|^2$ and $\mathcal{Q}(\mathsf{W}') \geq \mathcal{Q}(\mathsf{W}) + < \mathbf{gq}, \mathsf{W}' - \mathsf{W} > + \frac{\lambda_2}{2}\|\mathsf{W}' - \mathsf{W}\|^2$, where $\mathbf{gp}$ and $\mathbf{gq}$ are subgradients for W for $\mathcal{P}$ and $\mathcal{Q}$, respectively. It can be directly seen that $F(\mathsf{W})$ is also strongly convex and that $\mathbf{gf} = \mathbf{gp} + \mathbf{gq}$ is a subgradient of $F(\mathsf{W})$ at W.

Lemma 2 *Suppose* $\mathbb{E}[\|\hat{\mathbf{gp}}_t\|^2] \leq G^2$ *and* $\mathbb{E}[\|\hat{\mathbf{gq}}_t\|^2] \leq G^2$*, and we use the update rule of* $\mathsf{W}_{t+1} = \Pi_{\mathcal{W}}(\mathsf{W}_t - \eta_t(\hat{\mathbf{gp}}_t + \hat{\mathbf{gq}}_t))$ *where* $\mathbb{E}[\hat{\mathbf{gp}}_t] = \mathbf{gp}_t$ *and* $\mathbb{E}[\hat{\mathbf{gq}}_t] = \mathbf{gq}_t$*. If we use* $\eta_t = 1/(\lambda_1 + \lambda_2)t$*, then at iteration* T

$$\mathbb{E}[\|\mathsf{W}_T - \mathsf{W}^\star\|^2] \leq \frac{4G^2}{(\lambda_1 + \lambda_2)^2 T} \tag{8}$$

Proof Since $F(\mathsf{W}) = \mathcal{P}(\mathsf{W}) + \mathcal{Q}(\mathsf{W})$, it can be directly seen that

$$F(\mathsf{W}') \geq F(\mathsf{W}) + <\mathbf{gp}+\mathbf{gq}, \mathsf{W}' - \mathsf{W}> + \frac{\lambda_1+\lambda_2}{2}\|\mathsf{W}' - \mathsf{W}\|^2.$$

Based on lemma 1 in [21], this upper bound directly holds. □

Lemma 3 *We follow the assumptions in lemma 2, with the exception that we assume* $\eta_t = 1/t$ *since* λ_1 *and* λ_2 *are not always readily available; as we discuss in the appendix, we also expect the combined* λ *to be small. When we begin in the region* $\|\mathsf{W}_1 - \mathsf{W}^\star\|^2 \leq D$ *and the assumptions used in the appendix hold, the convergence rate is bounded by*

$$\mathbb{E}[\|\mathsf{W}_T - \mathsf{W}^\star\|^2] \leq e^{-2\lambda(\ln(T-1)+0.578)} D + 4G^2 \sum_{t=1}^{T-1} \frac{1}{t^2}\left(\frac{t}{T-1}\right)^{2\lambda} \tag{9}$$

Proof See appendix.

Theorem 1 *Let* $\mathcal{P}(\mathsf{W})$ *be* λ_1*-strongly convex and* $\mathcal{Q}(\mathsf{W})$ *be* λ_2*-strongly convex near optimal* $\mathsf{W}^\star$ *and denote by* $\mathsf{W}_T^{(F)}$ *and* $\mathsf{W}_T^{(\mathcal{P})}$ *the solution after* T *iterations when following SGD on* $F(\mathsf{W})$ *and* $\mathcal{P}(\mathsf{W})$*, respectively. Then DSN framework in Eq. (4) improves the relative convergence speed* $\frac{\mathbb{E}[\|\mathsf{W}_T^{(\mathcal{P})} - \mathsf{W}^\star\|^2]}{\mathbb{E}[\|\mathsf{W}_T^{(F)} - \mathsf{W}^\star\|^2]}$*, viewed from the ratio of their upper bounds as* $\Theta(\frac{(\lambda_1+\lambda_2)^2}{\lambda_1^2})$*, when* $\eta_t = 1/\lambda t$*.*

Proof Lemma 1 shows the compatibility of the companion objective of $\mathcal{Q}$ w.r.t the output objective $\mathcal{P}$. This equation can be directly derived from lemma 2.

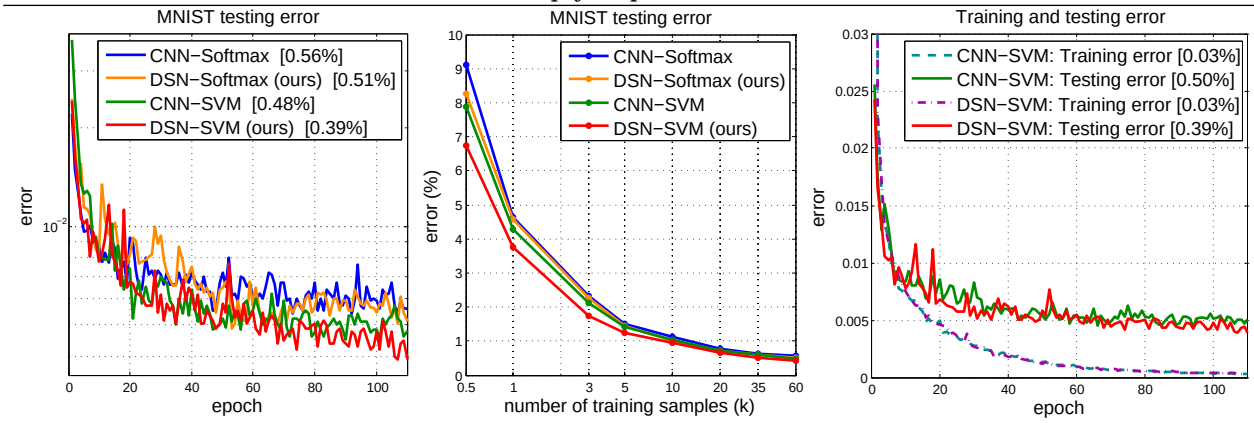
Remark When $\eta_t = 1/t$, T is large, and the first term of Eq. (9) dominates, the upper bound ratio for $\frac{\mathbb{E}[\|\mathsf{W}_T^{(\mathcal{P})} - \mathsf{W}^\star\|^2]}{\mathbb{E}[\|\mathsf{W}_T^{(F)} - \mathsf{W}^\star\|^2]}$ is roughly at the order of $\Theta(e^{2\ln(T)\lambda_2})$. If the second term of Eq. (9) dominates, the convergence of $F(\mathsf{W})$ over $\mathcal{P}(\mathsf{W})$ is also advantageous with the ratio at an order of $\Theta(e^{2\ln((T-1)/(T-2))\lambda_2})$ loosely.

In general we might expect $\lambda_2 \gg \lambda_1$ which would lead to a great improvement in convergence speed. □

3 Experiments

We evaluate our proposed DSN on four datasets: MNIST, CIFAR-10, CIFAR-100, and SVHN. We also use ImageNet to assess the behavior of DSN on a large dataset. We use MNIST as our primary running example. For our MNIST experiments we used an implementation of DSN built on top of the Theano platform [2]. The rest of the experiments are performed using DSN built on top of [18], except for ImageNet where we use [14] due to its parallel computing infrastructure. We use the SGD solver with mini-batch size of 128 and fixed momentum of 0.9. Initial values for learning rate and weight decay factor are determined from the validation set. For fair comparison and clear illustration of the effectiveness of DSN, we match the complexity of our model to the complexity of the network architectures used in [18] and [10] so that there are a comparable number of parameters. We also incorporate two dropout layers with dropout rate 0.5. Companion objective at the convolutional layers is imposed to back-propagate the classification error guidance at the associated layer. Our proposed DSN framework is not difficult to train and no particular engineering tricks are adopted.

above layer m do not participate in the computation for the loss function of this layer.



(a) (b) (c)

Figure 2: Classification error on MNIST test. (a) shows test error of competing methods; (b) shows test error versus training sample size. (c) train and test error comparison.

DSN can be equipped with different types of classifier objective functions; we consider L2SVM and softmax and show DSN-L2SVM and DSN-Softmax yield gains over corresponding CNN-L2SVM and CNN-Softmax approaches (see Figure (2.a)). The performance gain is more evident in the small training data regime (see Figure (2.b)); this might partially ease the burden of requiring large training data for DL. Overall, we observe state-of-the-art classification error on all four datasets. All results are achieved without using averaging [22], which could lead to further improvement in classification accuracy. Figure (4) gives an illustration of some learned features. The ImageNet classification result is also very encouraging (recent winning systems [28, 25] used many specific engineering steps and a cluster for training). Again, our DSN method can be combined with many existing CNN-type methods. Overall, as stated before: for small training data and relatively shallow networks, DSN functions as a strong "regularization"; for large training data and very deep networks, DSN makes the training process convenient, which might be otherwise problematic in standard CNN.

MNIST

The MNIST dataset consists of 28×28 images from 10 different classes (0 to 9) with 60,000 training and 10,000 test samples. We use a fairly standard network containing 2 convolutional layers followed by a fully-connected layer. Both convolutional layers use 5×5 filter size with 32 and 64 channels, respectively. Relu hidden units and 2×2 max pooling with stride 2 are used after each convolutional layer. The fully-connected layer has 500 hidden nodes that uses Relu activation with dropout rate 0.5.

Figure (2.a) and (b) show results from four methods, namely: (1) conventional CNN with softmax loss (CNN-Softmax), (2) the proposed DSN with softmax loss (DSN-Softmax), (3) CNN with L2SVM objective (CNN-L2SVM) , and (4) the proposed DSN with L2SVM objective (DSN-L2SVM). DSN-Softmax and DSN-L2SVM outperform their respective competing CNN algorithms (DSN-L2SVM shows classification error of 0.39% under a single model without data whitening and augmentation). Figure (2.b) shows the classification error of the competing methods when trained with varying sizes of training samples (26% gain of DSN-L2SVM over CNN-Softmax at 500 samples. Figure (2.c) shows a comparison of generalization error between CNN and DSN.

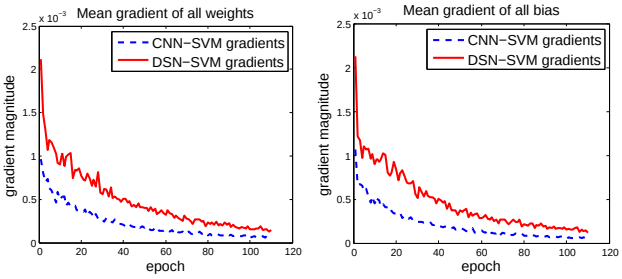
We also plot averaged absolute values of gradient matrices during training on MNIST for both CNN and DSN, shown in Figure (3). DSN backpropagates stronger feedback than CNN for both weights and biases, thanks to deep supervision. Table 1 compares performance. To investigate the difference between DSN and layer-wise pre-training (shown as CNN with pre-training in Table 1), we train a CNN model using greedy layer-wise pre-training as in [1]. Under the same network architecture, layer-wise pre-training performs better than CNN without pre-training but worse than DSN.

Table 1: MNIST classification result (without using data augmentation and model averaging).

Method	Error(%)
CNN [13]	0.53
Stochastic Pooling [32]	0.47
Network in Network [18]	0.47
Maxout Networks [10]	0.45
CNN (layer-wise pre-training)	0.43
DSN (ours)	**0.39**

CIFAR-10 and CIFAR-100

The CIFAR-10 dataset consists of 32×32 color images. A collection of 60,000 images is split into 50,000 train-

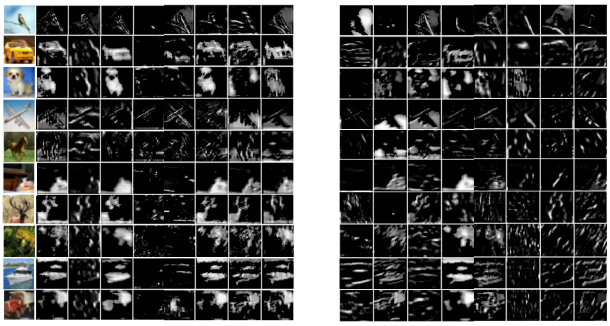


(a) (b)

Figure 3: Average absolute value of gradient matrices during the training on MNIST for (a) weights; (b) biases.

ing and 10,000 testing images. The dataset is preprocessed by global contrast normalization. To compare our results to previous state-of-the-art, for this dataset we also augmented the dataset by zero-padding 4 pixels on each side; we also perform corner cropping and random flipping on the fly during training. For a fair comparison, we adopt a network architecture similar to [18] that contains 3 convolutional layers with 192 filter channels. We also use the mlpconv layer after each convolutional layer and a global averaged pooling scheme with kernel size 8 for the output prediction. Relu neuron with 0.5 dropout rate and 3×3 max pooling with stride 2 are used after the first two convolutional layers.

No model averaging is done at test phase. Table (2) shows our results. Our DSN model achieves an error rate of 9.69% without data augmentation and of 7.97% with data augmentation (the best result to our knowledge). Note that DSN still outperforms greedy layer-wise pre-training (shown as CNN with pre-training in Table 2) under the same network architecture.



(a) by DSN (b) by CNN

Figure 4: Visualization of the feature maps learned in the convolutional layer.

DSN also provides added robustness to hyperparameter choice, in that the early layers are guided with direct classification loss, leading to a faster convergence rate and reduced dependence on heavy hyperparameter tuning. We also compared the gradients in DSN with those in CNN, observing 4.55 times greater gradient variance of DSN over CNN in the first convolutional layer. This is consistent with an observation in [10] and with the assumptions and motivations we make in this work. To see what features have been learned in DSN vs. CNN, we select one example image from each of the ten categories of CIFAR-10, run one forward pass, and show the feature maps learned from the bottom convolutional layer in Figure (4). Only the top 30% of activations are shown in each of the feature maps. Feature maps learned by DSN appear to be more intuitive than those learned by CNN.

Table 2: CIFAR-10 classification error.

Method	Error(%)
No Data Augmentation	
Stochastic Pooling [32]	15.13
Maxout Networks [10]	11.68
Network in Network [18]	10.41
NIN (layer-wise pre-training)	9.92
DSN (ours)	**9.69**
With Data Augmentation	
Maxout Networks [10]	9.38
Dropconnect [17]	9.32
Network in Network [18]	8.81
DSN (ours)	**7.97**

CIFAR-100 is similar to CIFAR-10, but with 100 classes rather than 10. The number of images for each class is then 500 instead of 5,000 as in CIFAR-10, which makes the classification task more challenging. We use the same network settings as in CIFAR-10. The consistent performance boost (shown on both CIFAR-10 and CIFAR-100) again demonstrates the advantage of the DSN method.

Table 3: CIFAR-100 classification error.

Method	Error(%)
Stochastic Pooling [32]	42.51
Maxout Networks [10]	38.57
Tree based Priors [27]	36.85
Network in Network [18]	35.68
DSN (ours)	**34.57**

Street View House Numbers

The Street View House Numbers (SVHN) dataset consists of 32×32 color images: 73,257 digits for training, 26,032 digits for testing, and 531,131 extra training samples. We prepared the data in keeping with previous work, namely: we select 400 samples per class from the training set and 200 samples per class from the extra set. The remaining 598,388 images are used for training. We followed [10] to preprocess the dataset by Local Contrast Normalization (LCN). We do not do data augmentation in training and use only a single model in testing. Table 4 shows recent comparable results. Note that Dropconnect [17] uses data augmentation and multiple model voting.

Table 4: SVHN classification error.

Method	Error(%)
Stochastic Pooling [32]	2.80
Maxout Networks [10]	2.47
Network in Network [18]	2.35
Dropconnect [17]	1.94
DSN (ours)	**1.92**

3.1 ImageNet Challenge

To further illustrate the effectiveness of DSN on large training datasets and with deeper networks, we test both CNN and DSN (8-layer and 11-layer networks) on the ImageNet 2012 dataset, which consists of 1.2 million training images, $50,000$ validation, and $100,000$ testing. We emphasize that our intention here is not to directly compete with the best performing result in the challenge (since the winning methods [28] require many additional aspects), but to provide an illustrative comparison of the relative benefits of DSN versus CNN on this data set. That said, it is worth noting that the supervision applied to two hidden layers in [28] is a suggestive indication of additional support for our method. The 8-layer network's configuration is based on the AlexNet [14] with 5 convolutional layers and 3 fully-connected layers, while the 11-layer network's infrastructure contains 8 convolutional layers and 3 fully-connected layers. To decouple the effect of parameter tuning, we use the same network and parameter setup as in [14] for both CNN and DSN. We simply add three more convolutional layers for the 11-layer networks. We start with learning rate 0.01 and lower by a factor of 10 when validation error stops decreasing. This procedure is repeated until learning rate reaches 10^{-5}. All results are tested under a single net without multi-scale training/testing in order to reduce the effect of model/prediction averaging. Table 5 shows a direct comparison between CNN and DSN.

When the depth of the network increases to 11 layers, standard backpropagation for training CNN faces a big challenge. We observe that the average absolute gradient of the weight matrix of conv1 layer to be at magnitude of 10^{-10}, even during the first few epochs of the training process, making the networks significantly difficult to converge. Therefore, we have to adopt a pre-training strategy suggested in [25]: we first pre-train an 8-layer network and then restart the training process with another three convolutional layers for the 11-layer CNN configuration. In contrast, our DSN of 11 layers behaves regularly during training and the objective function moves down directly starting at the first few epochs without pre-training; CNN converges for 95 epochs with pre-training while DSN converges for 58 epochs. Table 5 shows the performance advantage of DSN over CNN, in addition to a significant gain in training convenience and saved manual adjustments.

Table 5: ImageNet 2012 classification error.

Method	top-1 val. error(%)	top-5 val. error(%)
CNN 8-layer [14]	40.7	18.2
DSN 8-layer (ours)	39.6	17.8
CNN 11-layer	34.5	13.9
DSN 11-layer (ours)	33.7	13.1

Acknowledgments

This work is supported by NSF awards IIS-1216528 (IIS-1360566) and IIS-0844566(IIS-1360568). We thank Min Lin, Naiyan Wang, Baoyuan Wang, Jingdong Wang, Liwei Wang, Ying Nian Wu, and David Wipf for helpful discussions. We gratefully acknowledge the support of NVIDIA Corporation with their donation of the GPUs used for this research.

Appendix

Proof for Lemma 3. Letting $\lambda = \lambda_1 + \lambda_2$, we have

$$F(\mathsf{W}^\star) - F(\mathsf{W}_t) \geq \ < \mathbf{gf}_t, \mathsf{W}^\star - \mathsf{W}_t > + \frac{\lambda}{2}\|\mathsf{W}^\star - \mathsf{W}_t\|^2,$$

$$\text{and} \qquad F(\mathsf{W}_t) - F(\mathsf{W}^\star) \geq \frac{\lambda}{2}\|\mathsf{W}_t - \mathsf{W}^\star\|^2.$$

Thus,

$$< \mathbf{gf}_t, \mathsf{W}_t - \mathsf{W}^\star > \ \geq \lambda \|\mathsf{W}_t - \mathsf{W}^\star\|^2.$$

Therefore, with $\eta_t = 1/t$,

$$\begin{aligned} & \mathbb{E}[\|\mathsf{W}_{t+1} - \mathsf{W}^\star\|^2] = \mathbb{E}[\|\Pi_{\mathcal{W}}(\mathsf{W}_t - \eta_t \hat{\mathbf{gf}}_t) - \mathsf{W}^\star\|^2] \\ \leq \ & \mathbb{E}[\|\mathsf{W}_t - \eta_t \hat{\mathbf{gf}}_t - \mathsf{W}^\star\|^2] \\ = \ & {\scriptstyle \mathbb{E}[\|\mathsf{W}_t - \mathsf{W}^\star\|^2] - 2\eta_t \mathbb{E}[< \mathbf{gf}_t, \mathsf{W}_t - \mathsf{W}^\star >] + \eta_t^2 \mathbb{E}[\|\hat{\mathbf{gf}}_t\|^2]} \\ \leq \ & (1 - 2\lambda/t)\mathbb{E}[\|\mathsf{W}_t - \mathsf{W}^\star\|^2] + 4G^2/t^2 \end{aligned} \tag{10}$$

Since our focus is on the setting of "nearly flat" objective behavior, we expect $2\lambda/t$ to be small; in such a case $1 - 2\lambda/t \approx e^{-2\lambda/t}$.

$$\begin{aligned} & \mathbb{E}[\|\mathsf{W}_T - \mathsf{W}^\star\|^2] \\ & \leq e^{-2\lambda(\frac{1}{1} + \frac{1}{2} + \cdots \frac{1}{T-1})} D + \sum_{t=1}^{T-1} \frac{4G^2}{t^2} e^{-2\lambda(\frac{1}{t} + \frac{1}{t+1} + \cdots \frac{1}{T-1})} \\ & = e^{-2\lambda(\ln(T-1) + 0.578)} D + 4G^2 \sum_{t=1}^{T-1} \frac{1}{t^2} e^{-2\lambda \ln(T-1) + 2\lambda \ln(t)} \\ & \leq e^{-2\lambda(\ln(T-1) + 0.578)} D + 4G^2 \sum_{t=1}^{T-1} \frac{1}{t^2} \left(\frac{t}{T-1}\right)^{2\lambda} \end{aligned} \quad \square$$

Further, we can approximate

$$\mathbb{E}[\|\mathsf{W}_{t+1} - \mathsf{W}^\star\|^2] < e^{-2\lambda(\ln(T-1) + 0.578)} D + e^{-2\lambda(\ln((T-1)/(T-2)))} 4G^2 \zeta(2), \tag{11}$$

where $\zeta(2)$ is the Riemann zeta function.

References

[1] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In *NIPS*, 2007.

[2] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010.

[3] L. Bottou. Online algorithms and stochastic approximations. *Cambridge University Press*, 1998.

[4] M. Carreira-Perpinan and W. Wang. Distributed optimization of deeply nested systems. In *AISTATS*, 2014.

[5] G. E. Dahl, D. Yu, L. Deng, and A. Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Tran. on Audio, Speech, and Lang. Proc.*, 20(1):30–42, 2012.

[6] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *arXiv*, 2013.

[7] D. Eigen, J. Rolfe, R. Fergus, and Y. LeCun. Understanding deep architectures using a recursive convolutional network. In *arXiv:1312.1847v2*, 2014.

[8] J. L. Elman. Distributed representations, simple recurrent networks, and grammatical. *Machine Learning*, 7:195–225, 1991.

[9] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTAT*, 2010.

[10] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. C. Courville, and Y. Bengio. Maxout networks. In *ICML*, 2013.

[11] G. E. Hinton, S. Osindero, and Y. W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18:1527–1554, 2006.

[12] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. In *CoRR, abs/1207.0580*, 2012.

[13] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *ICCV*, 2009.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[15] Q. Le, J. Ngiam, Z. Chen, D. Chia, P. W. Koh, and A. Ng. Tiled convolutional neural networks. In *NIPS*, 2010.

[16] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. Backpropagation applied to handwritten zip code recognition. In *Neural Computation*, 1989.

[17] W. Li, M. Zeiler, S. Zhang, Y. LeCun, and R. Fergus. Regularization of neural networks using dropconnect. In *ICML*, 2013.

[18] M. Lin, Q. Chen, and S. Yan. Network in network. In *ICLR*, 2014.

[19] P.-L. Loh and M. J. Wainwright. Regularized m-estimators with nonconvexity : statistical and algorithmic theory for local optima. In *arXiv:1305.2436v1*, 2013.

[20] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *arXiv:1211.5063v2*, 2014.

[21] A. Rakhlin, O. Shamir, and K. Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. In *ICML*, 2012.

[22] J. Schmidhuber. Multi-column deep neural networks for image classification. In *CVPR*, 2012.

[23] P. Sermanet and Y. LeCun. Traffic sign recognition with multi-scale convolutional networks. In *International Joint Conference on Neural Networks (IJCNN)*, 2011.

[24] O. Shamir and T. Zhang. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *ICML*, 2013.

[25] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *arXiv:1409.1556*, Sept. 4, 2014.

[26] J. Snoek, R. P. Adams, and H. Larochelle. Nonparametric guidance of autoencoder representations using label information. *J. of Machine Learning Research*, 13:2567–2588, 2012.

[27] N. Srivastava and R. Salakhutdinov. Discriminative transfer learning with tree-based priors. In *NIPS*, 2013.

[28] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *arXiv:1409.4842*, Sept. 17, 2014.

[29] Y. Tang. Deep learning using linear support vector machines. In *Workshop on Representational Learning, ICML*, 2013.

[30] J. Weston and F. Ratle. Deep learning via semi-supervised embedding. In *ICML*, 2008.

[31] M. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *arXiv 1311.2901*, 2013.

[32] M. D. Zeiler and R. Fergus. Stochastic pooling for regularization of deep convolutional neural networks. In *ICLR*, 2013.