Deeply-Supervised Nets

Chen-Yu Lee [*]
Dept. of EECS, UCSD
chl260@ucsd.edu

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

Patrick Gallagher
Dept. of CogSci, UCSD
rexaran@gmail.com

Zhengyou Zhang
Microsoft Research
zhang@microsoft.com

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

Abstract

Our proposed deeply-supervised nets (DSN) method simultaneously minimizes classification error while making the learning process of hidden layers direct and transparent. We make an attempt to boost the classification performance by studying a new formulation in deep networks. Three aspects in convolutional neural networks (CNN) style architectures are being looked at: (1) transparency of the intermediate layers to the overall classification; (2) discriminativeness and robustness of learned features, especially in the early layers; (3) effectiveness in training due to the presence of the exploding and vanishing gradients. We introduce "companion objective" to the individual hidden layers, in addition to the overall objective at the output layer (a different strategy to layer-wise pre-training). We extend techniques from stochastic gradient methods to analyze our algorithm. The advantage of our method is evident and our experimental result on benchmark datasets shows significant performance gain over existing methods (e.g. all state-of-the-art results on MNIST, CIFAR-10, CIFAR-100, and SVHN).

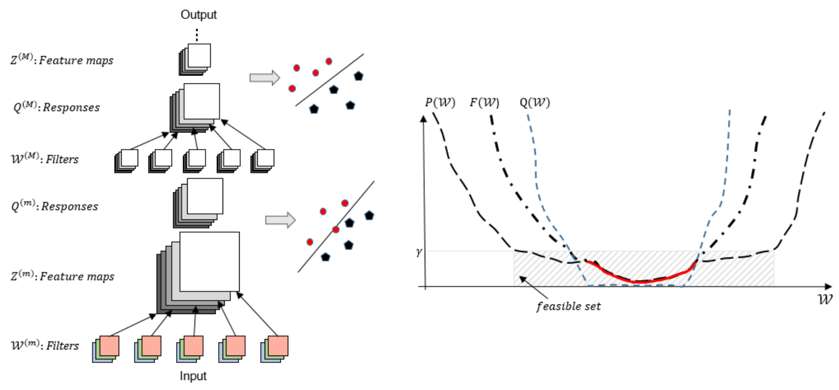
1 Introduction

Much attention has been given to a resurgence of neural networks, deep learning (DL) in particular, which can be of unsupervised [10], supervised [12], or a hybrid form [18]. Significant performance gain has been observed, especially in the presence of large amount of training data, when deep learning techniques are used for image classification [11, 16] and speech recognition [4]. On the one hand, hierarchical and recursive networks [7, 10, 12] have demonstrated great promise in automatically learning thousands or even millions of features for pattern recognition; on the other hand concerns about deep learning have been raised and many fundamental questions remain open.

Some potential problems with the current DL frameworks include: reduced transparency and discriminativeness of the features learned at hidden layers [31]; training difficulty due to exploding and vanishing gradients [8, 22]; lack of a thorough mathematical understanding about the algorithmic behavior, despite of some attempts made on the theoretical side [6]; dependence on the availability of large amount of training data [11]; complexity of manual tuning during training [15]. Nevertheless, DL is capable of automatically learning and fusing rich hierarchical features in an integrated framework. Recent activities in open-sourcing and experience sharing [11, 5, 2] have also greatly helped the adopting and advancing of DL in the machine learning community and beyond. Several techniques, such as dropout [11], dropconnect [19], pre-training [4], and data augmentation [24], have been proposed to enhance the performance of DL from various angles, in addition to a variety of engineering tricks used to fine-tune feature scale, step size, and convergence rate. Features

[*]equal contribution

[†]Corresponding author. Patent disclosure, UCSD Docket No. SD2014-313, filed on May 22, 2014.



(a) DSN illustration (b) functions

Figure 1: Network architecture for the proposed deeply-supervised nets (DSN).

learned automatically by the CNN algorithm [12] are intuitive [31]. Some portion of features, especially for those in the early layers, also demonstrate certain degree of opacity [31]. This finding is also consistent with an observation that different initializations of the feature learning at the early layers make negligible difference to the final classification [4]. In addition, the presence of vanishing gradients also makes the DL training slow and ineffective [8]. In this paper, we address the feature learning problem in DL by presenting a new algorithm, deeply-supervised nets (DSN), which enforces direct and early supervision for both the hidden layers and the output layer. We introduce *companion objective* to the individual hidden layers, which is used as an additional constraint (or a new regularization) to the learning process. Our new formulation significantly enhances the performance of existing supervised DL methods. We also make an attempt to provide justification for our formulation using stochastic gradient techniques. We show an improvement of the convergence rate of the proposed method over standard ones, assuming local strong convexity of the optimization function (a very loose assumption but pointing to a promising direction).

Several existing approaches are particularly worth mentioning and comparing with. In [1], layer-wise supervised pre-training is performed. Our proposed method does not perform pre-training and it emphasizes the importance of minimizing the output classification error while reducing the prediction error of each individual layer. This is important as the backpropagation is performed altogether in an integrated framework. In [26], label information is used for unsupervised learning. Semi-supervised learning is carried in deep learning [30]. In [28], an SVM classifier is used for the output layer, instead of the standard softmax function in the CNN [12]. Our framework (DSN), with the choice of using SVM, softmax or other classifiers, emphasizes the direct supervision of each intermediate layer. In the experiments, we show consistent improvement of DSN-SVM and DSN-Softmax over CNN-SVM and CNN-Softmax respectively. We observe all state-of-the-art results on MNIST, CIFAR-10, CIFAR-100, and SVHN. It is also worth mentioning that our formulation is inclusive to various techniques proposed recently such as averaging [24], dropconnect [19], and Maxout [9]. We expect to see more classification error reduction with careful engineering for DSN.

2 Deeply-Supervised Nets

In this section, we give the main formulation of the proposed deeply-supervised nets (DSN). We focus on building our infrastructure around supervised CNN style frameworks [12, 5, 2] by introducing classifier, e.g. SVM model [29], to each layer. An early attempt to combine SVM with DL was made in [28], which however has a different motivation with ours and only studies the output layer with some preliminary experimental results.

2.1 Motivation

We are motivated by the following simple 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, and further to the quality of the upper layer feature maps. By making appropriate use of this feature quality feedback at each hidden layer of the network, we are able to directly influence the hidden layer weight/filter update process to favor highly discriminative feature maps. This is a source of supervision that acts deep within the network at each layer; when our proxy for feature quality is good, we expect to much more rapidly approach the region of good features than would be the case if we had to rely on the gradual backpropagation from the output layer alone. We also expect to alleviate the common problem of having gradients that "explode" or "vanish". One concern with a direct pursuit of feature discriminativeness at all hidden layers is that this might interfere with the overall network performance, since it is ultimately the feature maps at the output layer which are used for the final classification; our experimental results indicate that this is not the case.

Our basic network architecture will be similar to the standard one used in the CNN framework. Our additional deep feedback is brought in by associating a companion local output with each hidden layer. We may think of this companion local output as analogous to the final output that a truncated network would have produced. Backpropagation of error now proceeds as usual, with the crucial difference that we now backpropagate not only from the final layer but also simultaneously from our local companion output. The empirical result suggests the following main properties of the companion objective: (1) it acts as a kind of feature regularization (although an unusual one), which leads to significant reduction to the testing error but not necessarily to the train error; (2) it results in faster convergence, especially in presence of small training data (see Figure (2) for an illustration on a running example).

2.2 Formulation

We focus on the supervised learning case and let $S = \{(\mathbf{X}_i, y_i), i = 1..N\}$ be our set of input training data where sample $\mathbf{X}_i \in R^n$ denotes the raw input data and $y_i \in \{1, .., K\}$ is the corresponding groundtruth label for sample X_i. We drop i for notational simplicity, since each sample is considered independently. The goal of deep nets, specifically convolutional neural networks (CNN) [12], is to learn layers of filters and weights for the minimization of classification error at the output layer. Here, we absorb the bias term into the weight parameters and do not differentiate weights from filters and denote a recursive function for each layer $m = 1..M$ as:

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

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

M denotes the total number of layers; $\mathsf{W}^{(m)}, m = 1..M$ 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; $f()$ is a pooling function on $\mathbf{Q}$; Combining all layers of weights gives

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

Now we introduce a set of classifiers, e.g. SVM (other classifiers like Softmax can be applied and we will show results using both SVM and Softmax in the experiments), one for each hidden layer,

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

in addition to the W in the standard CNN framework. We denote the $\mathbf{w}^{(out)}$ as the SVM weights for the output layer. Thus, we build our overall combined objective function as:

$$\|\mathbf{w}^{(out)}\|^2 + \mathcal{L}(\mathsf{W}, \mathbf{w}^{(out)}) + \sum_{m=1}^{M-1} \alpha_m [\|\mathbf{w}^{(m)}\|^2 + \ell(\mathsf{W}, \mathbf{w}^{(m)}) - \gamma]_+, \tag{3}$$

where

$$\mathcal{L}(\mathsf{W}, \mathbf{w}^{(out)}) = \sum_{y_k \neq y} [1 - < \mathbf{w}^{(out)}, \phi(\mathbf{Z}^{(M)}, y) - \phi(\mathbf{Z}^{(M)}, y_k) >]_+^2 \tag{4}$$

and

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

We name $\mathcal{L}(\mathsf{W}, \mathbf{w}^{(M)})$ as the *overall loss* (output layer) and $\ell(\mathsf{W}, \mathbf{w}^{(m)})$ as the *companion loss* (hidden layers), which are both squared hinge losses of the prediction errors. The above formulation can be understood intuitively: in addition to learning convolution kernels and weights, $\mathsf{W}^\star$, as in the standard CNN model [12], enforcing a constraint at each hidden layer for directly making a good label prediction gives a strong push for having discriminative and sensible features at each individual layer. In eqn. (3), $\|\mathbf{w}^{(out)}\|^2$ and $\mathcal{L}(\mathsf{W}, \mathbf{w}^{(out)})$ are respectively the margin and squared hinge loss of the SVM classifier (L2SVM [1]) at the output layer (we omit the balance term C in front of the hinge for notational simplicity); in eqn. (4), $\|\mathbf{w}^{(m)}\|^2$ and $\ell(\mathsf{W}, \mathbf{w}^{(m)})$ are respectively the margin and squared hinge loss of the SVM classifier at each hidden layer. Note that for each $\ell(\mathsf{W}, \mathbf{w}^{(m)})$, the $\mathbf{w}^{(m)}$ directly depends on $\mathbf{Z}^{(m)}$, which is dependent on $\mathsf{W}^1, .., \mathsf{W}^m$ up to the mth layer. $\mathcal{L}(\mathsf{W}, \mathbf{w}^{(out)})$ depends on $\mathbf{w}^{(out)}$, which is decided by the entire W. The second term in eqn. (3) often goes to zero during the course of training; this way, the overall goal of producing good classification of the output layer is not altered and the companion objective just acts as a proxy or regularization. This is achieved by having γ as a threshold (a hyper parameter) in the second term of eqn. (3) with a hinge loss: once the overall value of the hidden layer reaches or is below γ, it vanishes and no longer plays role in the learning process. α_m balances the importance of the error in the output objective and the companion objective. In addition, we could use a simple decay function as $\alpha_m \times 0.1 \times (1 - t/N) \rightarrow \alpha_m$ to enforce the second term to vanish after certain number of iterations, where t is the epoch step and N is the total number of epochs (wheather or not to have the decay on α_m might vary in different experiments although the differences may not be very big).

To summarize, we describe this optimization problem as follows: we want to learn filters/weights W for the entire network such that an SVM classifier $\mathbf{w}^{(out)}$ trained on the output feature maps (that depend on those filters/features) will display good performance. We seek this output performance while also requiring some "satisfactory" level of performance on the part of the hidden layer classifiers. We are saying: restrict attention to the parts of feature space that, when considered at the internal layers, lead to highly discriminative hidden layer feature maps (as measured via our proxy of hidden-layer classifier performance). The main difference between eqn. (3) and previous attempts in layer-wise supervised training is that we perform the optimization altogether with a robust measure (or regularization) of the hidden layer. For example, greedy layer-wise pretraining was performed as either initialization or fine-tuning which results in some overfitting [1]. The state-of-the-art benchmark results demonstrate the particular advantage of our formulation. As shown in Figure 2(c), indeed both CNN and DSN reach training error near zero but DSN demonstrates a clear advantage of having a better generalization capability.

To train the DSN model using SGD, the gradients of the objective function w.r.t the parameters in the model are:

$$\frac{\partial F}{\partial \mathbf{w}^{(out)}} = 2\mathbf{w}^{(out)} - 2\sum_{y_k \neq y}[\phi(\mathbf{Z}^{(M)}, y) - \phi(\mathbf{Z}^{(M)}, y_k)][1 - <\mathbf{w}^{(out)}, \phi(\mathbf{Z}^{(M)}, y) - \phi(\mathbf{Z}^{(M)}, y_k)>]_+$$

$$\frac{\partial F}{\partial \mathbf{w}^{(m)}} = \begin{cases} \alpha_m \left\{ 2\mathbf{w}^{(m)} - 2\sum_{y_k \neq y}[\phi(\mathbf{Z}^{(m)}, y) - \phi(\mathbf{Z}^{(m)}, y_k)][1 - <\mathbf{w}^{(m)}, \phi(\mathbf{Z}^{(m)}, y) - \phi(\mathbf{Z}^{(m)}, y_k)>]_+ \right\}, & \text{otherwise} \\ 0, & \text{if } \|\mathbf{w}^{(m)}\|^2 + \ell(\mathsf{W}, \mathbf{w}^{(m)}) \leq \gamma \end{cases} \tag{6}$$

The gradient w.r.t W just follows the conventional CNN based model plus the gradient that directly comes from the hidden layer supervision.

Next, we provide more discussions to and try to understand intuitively about our formulation, eqn. (3). For ease of reference, we write this objective function as

$$F(\mathsf{W}) \equiv \mathcal{P}(\mathsf{W}) + \mathcal{Q}(\mathsf{W}), \tag{7}$$

where $\mathcal{P}(\mathsf{W}) \equiv \|\mathbf{w}^{(out)}\|^2 + \mathcal{L}(\mathsf{W}, \mathbf{w}^{(out)})$ and $\mathcal{Q}(\mathsf{W}) \equiv \sum_{m=1}^{M-1} \alpha_m [\|\mathbf{w}^{(m)}\|^2 + \ell(\mathsf{W}, \mathbf{w}^{(m)}) - \gamma]_+$.

2.3 Stochastic Gradient Descent View

We focus on the convergence advantage of DSN, instead of the regularization to the generalization aspect. In addition to the present problem in CNN where learned features are not always intuitive and discriminative [31], the difficulty of training deep neural networks has been discussed [8, 22].

[1]It makes negligible difference between L1SVM and L2SVM.

As we can observe from eqn. (1) and (2), the change of the bottom layer weights get propagated through layers of functions, leading to exploding or vanishing gradients [22]. Various techniques and parameter tuning tricks have been proposed to better train deep neural networks, such as pre-training and dropout [11]. Here we provide a somewhat loose analysis to our proposed formulation, in a hope to understand its advantage in effectiveness.

The objective function in deep neural networks is highly non-convex. Here we make the following assumptions/observations: (1) the objective/energy function of DL observes a large "flat" area around the "optimal" solution where any result has a similar performance; locally we still assume a convex (or even λ-strongly convex) function whose optimization is often performed with stochastic gradient descent algorithm [3].

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

$$F(\mathsf{W}') \geq F(\mathsf{W}) + < \mathbf{g}, \mathsf{W}' - \mathsf{W} > + \frac{\lambda}{2} \|\mathsf{W}' - \mathsf{W}\|^2, \quad (8)$$

and the update rule in Stochastic Gradient Descent (SGD) at step t is $\mathsf{W}_{t+1} = \Pi_{\mathcal{W}}(\mathsf{W}_t - \eta_t \hat{\mathbf{g}})$, where $\eta_t = \Theta(1/t)$ refers to the step rate and $\Pi_{\mathcal{W}}$ helps to project onto the space of $\mathcal{W}$. Let W^* be the optimum solution, upper bounds for $\mathbb{E}[\|\mathsf{W}_T - \mathsf{W}^*\|^2]$ and $\mathbb{E}[(F(\mathsf{W}_T) - F(\mathsf{W}^*)^2]$ in [23] for the strongly convex function, and $\mathbb{E}[(F(\mathsf{W}_T) - F(\mathsf{W}^*)^2]$ for convex function in [25]. Here we make an attempt to understand the convergence of eqn. (8) w.r.t. $\mathbb{E}[\|\mathsf{W}_T - \mathsf{W}^*\|^2]$, due to the presence of large area of flat function shown in Figure (1.b). In [21], a convergence rate is given for the M-estimators with locally convex function with compositional loss and regularization terms. Both terms in eqn. (8) here refer to the same class label prediction error, a reason for calling the second term as *companion objective*. Our motivation is two-fold: **(1)** encourage the features learned at each layer to be directly discriminative for class label prediction, while keeping the ultimate goal of minimizing class label prediction at the output layer; **(2)** alleviate the exploding and vanishing gradients problem as each layer now has a direct supervision from the ground truth labels. One might raise a concern that learning highly discriminative intermediate stage filters may not necessarily lead to the best prediction at the output layer. An illustration can been seen in Figure (1.b). Next, we give a loose theoretical analysis to our framework, which is also validated by comprehensive experimental studies with overwhelming advantages over the existing methods.

Definition We name $\mathcal{S}_\gamma(F) = \{\mathsf{W} : F(\mathsf{W}) \leq \gamma\}$ as the γ-feasible set for a function $F(\mathsf{W}) \equiv \mathcal{P}(\mathsf{W}) + \mathcal{Q}(\mathsf{W})$.

First we show that a feasible solution for $\mathcal{Q}(\mathsf{W})$ leads to a feasible one to $\mathcal{P}(\mathsf{W})$. That is:

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

Proof As we can see from an illustration of our network architecture shown in fig. (1.a), for $\forall$ $(\hat{\mathsf{W}}^{(1)}, .., \hat{\mathsf{W}}^{(m)})$ such that $\ell((\hat{\mathsf{W}}^{(1)}, .., \hat{\mathsf{W}}^{(m)}), \mathbf{w}^{(m)}) \leq \gamma$. Then there is a trivial solution for the network for every 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 $\mathcal{P}(\mathsf{W})$, but it may not be the case the other way around. That is: a W that makes $\mathcal{P}(\mathsf{W})$ small may not necessarily produce discriminative features for the hidden layers to have a small $\mathcal{Q}(\mathsf{W})$. However, $\mathcal{Q}(\mathsf{W})$ can be viewed as a regularization term. Since $\mathcal{P}(\mathsf{W})$ observes a very flat area near even zero on the training data and it is ultimately the test error that we really care about, we thus only focus on the $\mathsf{W}, \mathsf{W}^\star$, which makes 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 $\mathcal{P}(\mathsf{W}))$ be 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 $\mathcal{Q}(\mathsf{W}') \geq \mathcal{Q}(\mathsf{W}) + < \mathbf{gq}, \mathsf{W}' - \mathsf{W} >$

[2]Note that we drop the $\mathsf{W}^{(j)}, j > m$ since the filters above layer m do not participate in the computation for the objective function of this layer.

$+\frac{\lambda_1}{2}\|\mathsf{W}' - \mathsf{W}\|^2$, where $\mathbf{gp}$ and $\mathbf{gq}$ are the subgradients for $\mathcal{P}$ and $\mathcal{Q}$ at W respectively. It can be directly seen that $F(\mathsf{W})$ is also strongly convex and for subgradient $\mathbf{gf}$ of $F(\mathsf{W})$ at W, $\mathbf{gf} = \mathbf{gp}+\mathbf{gq}$.

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 time stamp* T

$$\mathbb{E}[\|\mathsf{W}_T - \mathsf{W}^\star\|^2] \leq \frac{12G^2}{(\lambda_1 + \lambda_2)^2 T} \tag{9}$$

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 [23], this upper bound directly holds. □

Lemma 3 *Following the assumptions in lemma 2, but now we assume* $\eta_t = 1/t$ *since* λ_1 *and* λ_2 *are not always readily available, then started from* $\|\mathsf{W}_1 - \mathsf{W}^\star\|^2 \leq D$ *the convergence rate is bounded by*

$$\mathbb{E}[\|\mathsf{W}_T - \mathsf{W}^\star\|^2] \leq e^{-2\lambda(\ln T + 0.578)}D + (T-1)e^{-2\lambda\ln(T-1)}G^2 \tag{10}$$

Proof Let $\lambda = \lambda_1 + \lambda_2$, we have

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

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

Thus,

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

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

$$\begin{aligned}
\mathbb{E}[\|\mathsf{W}_{t+1} - \mathsf{W}^\star\|^2] &= \mathbb{E}[\|\Pi_{\mathcal{W}}(\mathsf{W}_t - \eta_t\hat{\mathbf{gf}}_t) - \mathsf{W}^\star\|^2] \\
&\leq \mathbb{E}[\|\mathsf{W}_t - \eta_t\hat{\mathbf{gf}}_t - \mathsf{W}^\star\|^2] \\
&= \mathbb{E}[\|\mathsf{W}_t - \mathsf{W}^\star\|^2] - 2\eta_t\mathbb{E}[< \mathbf{gf}_t, \mathsf{W}_t - \mathsf{W}^\star >] + \eta_t\mathbb{E}[\|\hat{\mathbf{gf}}_t\|^2] \\
&\leq (1 - 2\lambda/t)\mathbb{E}[\|\mathsf{W}_t - \mathsf{W}^\star\|^2] + G^2/t^2
\end{aligned} \tag{11}$$

With $2\lambda/t$ being small, we have $1 - 2\lambda/t \approx e^{-2\lambda/t}$.

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

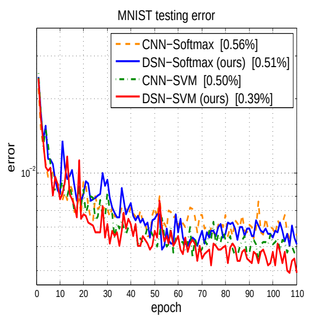
Theorem 1 *Let* $\mathcal{P}(\mathsf{W})$ *be* λ_1*-strongly convex and* $\mathcal{Q}(\mathsf{W})$ *be* λ_2*-strongly convex near optimal* $\mathsf{W}^\star$ *and denote* $\mathsf{W}_T^{(F)}$ *and* $\mathsf{W}_T^{(\mathcal{P})}$ *as the solution after* T *iterations when applying SGD on* $F(\mathsf{W})$ *and* $\mathcal{P}(\mathsf{W})$ *respectively. Then our deeply supervised framework in eqn. (3) improves the the speed over using top layer only by* $\frac{\mathbb{E}[\|\mathsf{W}_T^{(\mathcal{P})} - \mathsf{W}^\star\|^2]}{\mathbb{E}[\|\mathsf{W}_T^{(F)} - \mathsf{W}^\star\|^2]} = \Theta(1 + \frac{\lambda_2^2}{\lambda_1^2}),\ when\ \eta_t = 1/\lambda t, \quad and,\ \frac{\mathbb{E}[\|\mathsf{W}_T^{(\mathcal{P})} - \mathsf{W}^\star\|^2]}{\mathbb{E}[\|\mathsf{W}_T^{(F)} - \mathsf{W}^\star\|^2]} = \Theta(e^{\ln(T)\lambda_2}),\ when\ \eta_t = 1/t.$

Proof Lemma 1 shows the compatibility of the companion objective of $\mathcal{Q}$ w.r.t the output objective $\mathcal{P}$. The first equation can be directly derived from lemma 2 and the second equation can be seen from lemma 3. In general $\lambda_2 \gg \lambda_1$ which leads to a great improvement in convergence speed and the constraints in each hidden layer also helps to learning filters which are directly discriminative. □

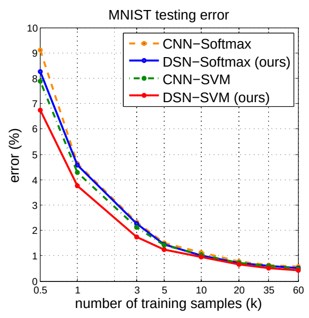
3 Experiments

We evaluate the proposed DSN method on four standard benchmark datasets: MNIST, CIFAR-10, CIFAR-100 and SVHN. We follow a common training protocol used by Krizhevsky et al. [15] in all experiments. We use SGD solver with mini-batch size of 128 at a fixed constant momentum value of 0.9. Initial value for learning rate and weight decay factor is determined based on the validation set. For a fair comparison and clear illustration of the effectiveness of DSN, we match the complexity of our model with that in network architectures used in [20] and [9] to have a comparable number of parameters. We also incorporate two dropout layers with dropout rate at 0.5. Companion objective at the convolutional layers is imposed to backpropagate the classification error guidance to the underlying convolutional layers. Learning rates are annealed during training by a factor of 20 according to an epoch schedule determined on the validation set. The proposed DSN framework is not difficult to train and there are no particular engineering tricks adopted. Our system is built on top of widely used Caffe infrastructure [14]. For the network architecture setup, we adopted the mlpconv layer and global averaged pooling scheme introduced in [20]. DSN can be equipped with different types of loss functions, such as Softmax and SVM. We show performance boost of DSN-SVM and DSN-Softmax over CNN-SVM and CNN-Softmax respectively (see Figure (2.a)). The performance gain is more evident in presence of small training data (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 in all four datasets (without data augmentation), 0.39% for MINIST, 9.78% for CIFAR-10, 34.57% for CIFAR-100, and 1.92% for SVHN (8.22% for CIFAR-10 with data augmentation). All results are achieved without using averaging [24], which is not exclusive to our method. Figure (3) gives an illustration of some learned features.

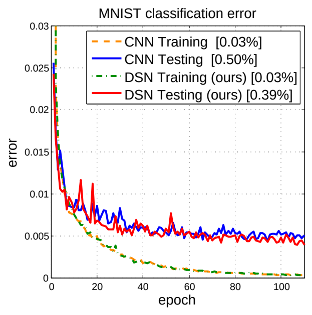
3.1 MNIST



(a)



(b)



(c)

Figure 2: Classification error on MNIST test. (a) shows test error of competing methods; (b) shows test error w.r.t. the training sample size. (c) training and testing error comparison.

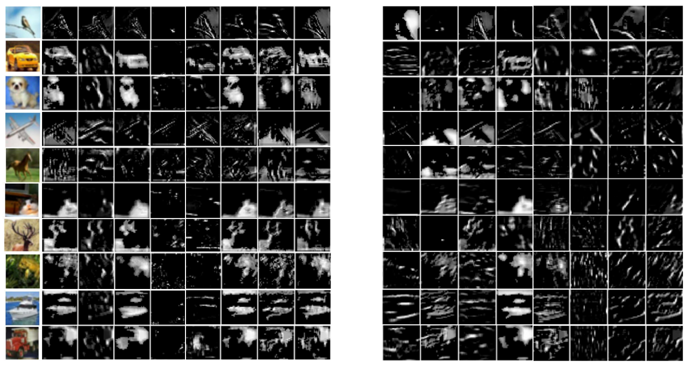
We first validate the effectiveness of the proposed DSN on the MNIST handwritten digits classification task [17], a widely and extensively adopted benchmark in machine learning. MNIST dataset consists of images of 10 different classes (0 to 9) of size 28×28 with 60,000 training samples and 10,000 test samples. 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 max-margin objective (CNN-SVM) , and (4) the proposed DSN with max-margin objective (DSN-SVM). DSN-Softmax and DSN-SVM outperform both their competing CNN algorithms (DSN-SVM shows classification error of 0.39% under a single model without data whitening and augmentation). Figure 2(b) shows classification error of the competing methods when trained w.r.t. varying sizes of training samples (26% gain of DSN-SVM over CNN-Softmax at 500 samples. Figure 2(c) shows a comparison of generalization error between CNN and DSN.

3.2 CIFAR-10 and CIFAR-100

CIFAR-10 dataset consists of 32×32 color images. A total number of 60,000 images are split into 50,000 training and 10,000 testing images. The dataset is preprocessed by global contrast normalization. To compare our results with the previous state-of-the-art, in this case, we also augmented the dataset by zero padding 4 pixels on each side, then do corner cropping and random flipping on the fly during training. No model averaging is done at the test phase and we only crop the center of

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

Method	**Error(%)**
CNN [13]	0.53
Stochastic Pooling [32]	0.47
Network in Network [20]	0.47
Maxout Networks[9]	0.45
DSN (ours)	**0.39**



(a) by DSN (b) by CNN

Figure 3: Visualization of the feature maps learned in the convolutional layer.

a test sample. Table (2) shows our result. Our DSN model achieved an error rates of 9.78% without data augmentation and 8.22% with data agumentation (the best known result to our knowledge).

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 relieved burden on heavy hyperparameter tuning. We also compared the gradients in DSN and 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 [9], and the assumptions and motivations we make in this work. To see what the features have been learned in DSN vs. CNN, we select one example image from each of the ten categories of CIFAR-10 dataset, run one forward pass, and show the feature maps learned from the first (bottom) convolutional layer in Figure (3). Only the top 30% activations are shown in each of the feature maps. Feature maps learned by DSN show to be more intuitive than those by CNN.

CIFAR-10 classification error

Method	**Error(%)**
No Data Augmentation	
Stochastic Pooling [32]	15.13
Maxout Networks [9]	11.68
Network in Network [20]	10.41
DSN (ours)	**9.78**
With Data Augmentation	
Maxout Networks [9]	9.38
DropConnect [19]	9.32
Network in Network [20]	8.81
DSN (ours)	**8.22**

CIFAR-100 classification error

Method	**Error(%)**
Stochastic Pooling [32]	42.51
Maxout Networks [9]	38.57
Tree based Priors [27]	36.85
Network in Network [20]	35.68
DSN (ours)	**34.57**

Table 2: Method comparison on CIFAR-10 and CIFAR-100 test data.

CIFAR-100 dataset is similar to CIFAR-10 dataset, except that it has 100 classes. 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. Table (2) shows previous best results and 34.57% is reported by DSN. The performance boost consistently shown on both CIFAR-10 and CIFAR-100 again demonstrates the advantage of the DSN method.

Method	Error(%)
Stochastic Pooling [32]	2.80
Maxout Networks [9]	2.47
Network in Network [20]	2.35
Dropconnect [19]	1.94
DSN (ours)	**1.92**

Table 3: SVHN classification error.

3.3 Street View House Numbers

Street View House Numbers (SVHN) dataset consists of $73,257$ digits for training, $26,032$ digits for testing, and $53,1131$ extra training samples on 32×32 color images. We followed the previous works for data preparation, 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 [9] 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 3 shows recent comparable results. Note that Dropconnect [19] uses data augmentation and multiple model voting.

4 Conclusions

In this paper, we have presented a new formulation, deeply-supervised nets (DSN), attempting to make a more transparent learning process for deep learning. Evident performance enhancement over existing approaches has been obtained. A stochastic gradient view also sheds light to the understanding of our formulation.

5 Acknowledgments

This work is supported by NSF award IIS-1216528 (IIS-1360566) and NSF award IIS-0844566 (IIS-1360568). We thank Min Lin, Naiyan Wang, Baoyuan Wang, Jingdong Wang, Liwei Wang, and David Wipf for help discussions. We are greatful for the generous donation of the GPUs by NVIDIA.

References

[1] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, U. D. Montral, and M. Qubec. 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] 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.

[5] 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.

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

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

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

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

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

[11] 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.

[12] F. J. Huang and Y. LeCun. Large-scale learning with svm and convolutional for generic object categorization. In *CVPR*, 2006.

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

[14] Y. Jia. Caffe: An open source convolutional architecture for fast feature embedding. `http://caffe.berkeleyvision.org/`, 2013.

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

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

[17] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, 1998.

[18] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *ICML*, 2009.

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

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

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

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

[23] A. Rakhlin, O. Shamir, and K. Sridharan. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *ICML*, 2012.

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

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

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

[29] V. N. Vapnik. The nature of statistical learning theory. In *Springer, New York*, 1995.

[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.