Cascaded Compressed Sensing Networks: A Reversible Architecture for Layerwise Learning

Weizhi Lu, Mingrui Chen, Kai Guo and Weiyu Li

Abstract—Recently, the method that learns networks layer by layer has attracted increasing interest for its ease of analysis. For the method, the main challenge lies in deriving an optimization target for each layer by inversely propagating the global target of the network. The propagation problem is ill posed, due to involving the inversion of nonlinear activations from low-dimensional to high-dimensional spaces. To address the problem, the existing solution is to learn an auxiliary network to specially propagate the target. However, the network lacks stability, and moreover, it results in higher complexity for network learning. In the letter, we show that target propagation could be achieved by modeling the network’s each layer with compressed sensing, without the need of auxiliary networks. Experiments show that the proposed method could achieve better performance than the auxiliary network-based method.

Index Terms—Deep neural networks, reversible networks, layerwise learning, compressed sensing

I. INTRODUCTION

The error backpropagation (BP) method has achieved great success in the supervised learning of deep neural networks [1]. Due to taking the entire network as a whole to optimize, the method can hardly disclose the contribution of each layer, causing difficulty to network analysis. As an alternative to BP, the emerging layerwise learning method [2], [3] seems more suitable for network analysis. The method aims to learn the network layer by layer and thus reduces the network analysis to the layer level.

For layerwise learning, the key is to derive an optimization target for each layer, simply called local target hereafter. Currently, the most simple method is to directly employ the global target [4], namely the optimization target of the entire network, as the target of each layer. However, the global target cannot guarantee to be optimal for each layer to minimize the global loss. To achieve BP level performance, the methods in [5], [6] propose to append a classifier to the layer of interest, and then optimize them together with the global target. This forms a shallow network with at least one hidden layer, in which the layer of interest is still hidden and thus hard to analyze [7], [8], [9]. To directly optimize and analyze each layer, we resort to another emerging layerwise learning method known as target propagation (TP) [2], [3], [10], which derives the local target of each layer by inversely propagating the global target, and optimizes the layer by matching its output to the target. In the method, local targets are required to be of two properties. First, their feedforward outputs should be identical or close to the global target. This ensures that the optimization of each layer could reduce the global loss to the maximum extent. Second, the local target of each layer should be close to the forward output of the layer. This will help to reduce the variation of each layer over optimization, while the low variations incline to improve the network’s generalization [11], [12], [13]. However, the two properties are hard to achieve because the inverse propagation of global target involves the inversions of nonlinear activations from low-dimensional to high-dimensional spaces. This is an ill-posed problem. For the problem, the existing solution is to learn an auxiliary network to specially propagate the target. Obviously, the method will increase the complexity of network learning, and moreover, the auxiliary network has no guarantee for stable and accurate target propagation. In the letter, we show that auxiliary networks are not necessary for layerwise learning, and target propagation could be achieved with guaranteed accuracy by modeling the network’s each layer with compressed sensing.

In compressed sensing [14], [15], it has been proved that a high-dimensional signal \( x \in \mathbb{R}^n \) with sparse transformations \( s = Dx \) over an orthonormal dictionary \( D \in \mathbb{R}^{n \times m} \), could be recovered from its low-dimensional projection \( y = WDx = Ws \), where \( W \in \mathbb{R}^{m \times n} \), \( m \ll n \). The recovery is implemented by first recovering the sparse vector \( s \) from \( y \) via solving an \( \ell_0 \) or \( \ell_1 \)-regularized least square problem [16], and then simply deriving \( x = D^\top s \). Inspired by this reverse process, we propose to model each network layer to be a compressed sensing process. Specifically, we formulate each layer as a cascade of \( W \) and \( D \), such that the input \( s \) of \( W \) could be recovered from its output \( y \) via sparse recovery, and the input \( x \) of \( D \) (namely the output \( y \) of \( W \)) could be simply recovered from its output \( s \) by multiplying with \( D^\top \). Repeating this process layer by layer, target propagation will be finally accomplished. To obtain better performance, as usual, we suggest to further deploy a nonlinear function \( f(\cdot) \) following each dictionary \( D \), in order to remove small activations in magnitude. This operation brings two benefits. First, it further improves the sparseness of the output activations of \( D \), which is beneficial to the sparse recovery on the following \( W \). Second, it tends to reduce within-class scattering and thus improve the classification accuracy [17]. Overall, with the perfect combination of \( W, D \) and \( f(\cdot) \), we establish a compressed sensing framework for each layer, and then obtain a cascaded compressed sensing architecture with all layers. To the best of our knowledge, this is the first time that compressed sensing is introduced for layerwise learning to handle the target propagation problem.
Figure 1: The proposed depth-$L$ network structure, with feedforward activations marked on the upper part and feedback targets marked on the lower part.

Compared to auxiliary networks-based TP methods, the proposed compressed sensing-based TP method has two major advantages. First, it avoids the usage of auxiliary networks and thus reduces the complexity of network learning. Second, it could achieve target propagation with guaranteed accuracy. In layerwise learning, the amount of local targets we need to compute and store is equal to the number of activations in the layer. Considering convolution networks contain huge amounts of activations and pose great computational challenges to layerwise learning, as in [3], [10], we only test the proposed layerwise learning method on fully-connected networks for tractable computation. Our performance advantage is verified by the classification experiments on MNIST and CIFAR10.

II. Method

Let us consider a deep fully connected network with $L$ layers as shown in Fig. 1. Each layer consists of two sublayers $\{W_l, D_l\}$, where $W_l \in \mathbb{R}^{m_l \times n_l}$ with $m_l \ll n_l$ denotes a projection matrix and $D_l \in \mathbb{R}^{m_l \times m_l}$ means an orthonormal dictionary, $D_l^T D_l = I$. Following $D_l$ is a pointwise thresholding function $f_l(\cdot)$ which zeroes out all but the $k_l$ ($\ll m_l$) largest positive elements, acting similarly to the ReLU function. Note that the output layer is a classifier which has dense input and output activations marked on the upper part and feedback targets marked on the lower part.

As detailed later, the reduced dimensionality of dictionaries can be learned with a procrustean approach [20], which is introduced in Algorithm 2 for completeness. In the letter, we decide to put the input activation $h^{0}_W$ and the global target $\tilde{h}^{L}_W$ as output. During the feedforward propagation, we initialize the projection matrix $W_l$ by drawing its elements from standard normal distribution. The distribution has been proved nearly optimal for compressed sensing [18]. To keep the input activation $h^{0}_W$ unchanged in $\ell_2$ norm after projections, $W_l$ needs to be scaled by a factor $1/\sqrt{m_l}$ [19]. For the output layer $W_L$, besides random initialization, we further update it by

$$W_L = W_L - \alpha \frac{\partial \mathcal{L}(h^{L}_W, \tilde{h}^{L}_W)}{\partial W_L}, \quad (3)$$

in order to reduce the difference between the final output $h^{L}_W$ and the global target $\tilde{h}^{L}_W$. As detailed later, the reduced difference is beneficial to the final layerwise optimization. Empirically, the parameter $\alpha$ in (3) should be small in case of overfitting.

Algorithm 1: The proposed layerwise learning

1. **Phase 1**: Activation forward propagation.
2. **for** $l = 1$ to $L$ **do**
3. Generate $h^{l}_W$ and $h^{l}_D$ by (1) and (2).
4. Initialize $W_l$ with Gaussian distributions.
5. Learn $D_l$ with its input $h^{l}_W$ by Algorithm 2.
6. **end for**
7. Further update the final $W_L$ by (3).
8. **Phase 2**: Target backward propagation.
9. Generate $\tilde{h}^{l-1}_D$ by (6) and $\tilde{h}^{l-1}_W$ by (5).
10. **for** $l = L - 2$ to 2 **do**
11. Generate $h^{l-1}_D$ by (4) and $h^{l-1}_W$ by (5).
12. **end for**
13. **Phase 3**: Forward layerwise learning.
14. **for** $l = 1$ to $L - 1$ **do**
15. Update $W_l$ by (7) and $D_l$ by (9) and (10).
16. **end for**
17. Update the final $W_L$ by (12).

A. Activation forward propagation

By (1) and (2), we know the network takes $h^{0}_D$ as input and $h^{l}_W$ as output. During the feedforward propagation, we initialize the projection matrix $W_l$ by drawing its elements from standard normal distribution. The distribution has been proved nearly optimal for compressed sensing [18]. To keep the input activation $h^{0}_W$ unchanged in $\ell_2$ norm after projections, $W_l$ needs to be scaled by a factor $1/\sqrt{m_l}$ [19]. For the output layer $W_L$, besides random initialization, we further update it by

$$h^{l}_W = W_l h^{l-1}_D, \quad l = 1, \ldots, L - 1 \quad (1)$$

$$h^{L}_W = g(W_L h^{L-1}_D), \quad l = L \quad (2)$$

where $g(\cdot)$ is a softmax function, and the notations $h^{0}_D$ and $h^{L}_W$ denote the network’s input and output, respectively.

As sketched in Algorithm 1, the proposed layerwise learning is implemented within three phases. First, we initialize the network via feeding forward the network input $h^{0}_D$. During this process, the projection matrices $W_l$ are randomly generated, and the orthonormal dictionaries $D_l$ are learned with their input activations $h^{l}_W$.

Second, as shown in the lower part of Fig. 1 by compressed sensing the global targets $\tilde{h}^{L}_W$ are inversely propagated layer by layer, producing local targets $\tilde{h}^{l}_W$ and $\tilde{h}^{l}_D$ respectively for $W_l$ and $D_l$, $1 \leq l < L$. Finally, we feed forward the input $h^{0}_D$ again, in order to successively minimize the local losses $\mathcal{L}_l(h^{l}_W, \tilde{h}^{l}_W)$ and $\mathcal{L}_l(h^{l}_D, \tilde{h}^{l}_D)$ of each layer by updating $W_l$ and $D_l$, $1 \leq l < L$. This finally minimizes the global loss $\mathcal{L}(h^{L}_W, \tilde{h}^{L}_W)$. In the letter, we simply define the local losses with Euclidean distances and the global loss with cross entropy. In the following, we detail the three phases.
of the output of $D_l$. Note that $f_l(\cdot)$ only keeps a portion of positive elements and discards the negative ones, because empirically both of them tend to share the similar information for classification. The choice of $k_l$ depends on both the sparse degree of the output of $D_l$ and the compression ratio of the following $W_{l+1}$. Empirically, as discussed later, a large $k_l$ that allows for containing most of positive elements tends to provide good classification performance, when the dictionary output is not very sparse.

### B. Target backward propagation

The global target $\tilde{h}_D^l$ is propagated backward by repeating the following compressed sensing recovery process:

$$\tilde{h}_D^{l-1} = \arg \min_{h_D^{l-1}} \| h_D^{l-1} - W_l h_D^l \|^2 \text{ s.t. } \|h_D^{l-1}\|_0 \leq k_{l-1} \quad (4)$$

and

$$\tilde{h}_W^{l-1} = D_{l-1}^\top \tilde{h}_D^{l-1} \quad (5)$$

where the layer index $l$ decreases from $L$ to 2; $\tilde{h}_D^{l-1}$ and $\tilde{h}_W^{l-1}$ denote the local targets derived respectively for $D_{L-1}$ and $W_{L-1}$. The derivations of the two targets are analyzed as follows. Let us first see the deriving of $\tilde{h}_D^{l-1}$ by (4). By compressed sensing, the row size $m_l$ of $W_l$ in (3) should be at least twice its input sparsity $k_{l-1}$, namely $m_l > 2k_{l-1}$, in order to approximately recover $\tilde{h}_D^{l-1}$ from $\tilde{h}_W$. Empirically, the condition requires not to be strictly met, and a good classification is usually obtained as $m_l \approx 2k_{l-1}$. This implies that for layerwise learning, the feedbackforward of sufficient amounts of features may be more important than the accuracy of target propagation. We could solve (4) with many algorithms [21]. In our experiments, we will employ the known OMP [22] algorithm, which has a parallel implementation in the software SPAMS [23]. As for the local target $\tilde{h}_W^{l-1}$, it is seen in (5) that the target could be simply derived from $h_D^{l-1}$ by multiplication, benefiting from the orthonormality of $D_1$.

Usually, we prefer to simply label each class with a one-hot vector $h_W^l$, such that the instances of the same class will share the same local target at each layer. Note that the label sharing may cause difficulty for layerwise optimization, as the instances of the same class have high variations, as often encountered at early layers [3]. To avoid the problem, an effective solution is to adopt diverse labels/tags to represent the instances of the same class [3]. To achieve this, instead of (4), we propose to derive the local target $\tilde{h}_D^{L-1}$ of the penultimate layer by

$$\tilde{h}_D^{L-1} = h_D^{L-1} - \beta \frac{\partial \mathcal{L}(h_W^{L-1}, \tilde{h}_W^{L-1})}{\partial h_D^{L-1}} \quad (6)$$

and then derive the targets for the rest layers by (4) and (5).

Considering the global loss $\mathcal{L}(h_W^L, \tilde{h}_W^L)$ has been previously decreased in the initialization phase by updating $W_L$ in (3), we use a small $\beta$ in (6) to further reduce the global loss to achieve a good classification performance. It is seen that a small $\beta$ will result in a small difference $h_D^{L-1} - \tilde{h}_D^{L-1}$, and then a sequence of small $h_D^l - \tilde{h}_D^l$ (with $l$ decreasing from $L - 2$ to 1) by the stable recovery of compressed sensing [25]. The small differences between forward activations and backward targets will reduce the variations of $D_l$ and $W_l$ in optimization, while the low variations over optimization incline to improve the network’s generalization [11, 12, 13]. To achieve such property, the TP method in [3] constrains the difference between targets and activations, as learning the auxiliary network. However, the network-based method lacks stability, thus inferior to our compressed sensing-based method.

### C. Forward layerwise learning

Feeding forward the network input $h_0^0$, we successively update the parameters $\{W_l, D_l\}$ of each layer by matching their outputs $h_W^l$ and $h_D^l$ to their local targets $\tilde{h}_W^l$ and $\tilde{h}_D^l$. Formally, this process is realized by iterating

$$\tilde{W}_l = \arg \min_{W_l} \| h_W^l - W_l h_D^l \|^2 + \lambda W_l\|W_l\|^2 \quad (7)$$

$$h_W^l = \tilde{W}_l h_D^l \quad (8)$$

$$h_W^l h_D^l = P \Sigma Q^\top \quad (9)$$

$$\tilde{D}_l = Q P^\top \quad (10)$$

$$h_D^l = f_l(\tilde{D}_l h_W^l) \quad (11)$$

where the index $l$ increases from 1 to $L - 1$. As a classifier, the final output layer is updated by

$$\tilde{W}_L = W_L - \eta \frac{\partial \mathcal{L}(h_W^L, \tilde{h}_W^L)}{\partial W_L}. \quad (12)$$

Then the resulting $\{\tilde{W}_l, \tilde{D}_l\}_{l=1}^L$ constitute the final network. Note that the dictionary updating rules (9) and (10) are adapted from Algorithm 2. For tractable computation, we simply update $W_l$ with the ridge regression model (7), although other more sophisticated models, such as the Lasso [26] and elastic-net [27], probably yield better updates.

For large-scale data, such as ImageNet [28], we suggest to process the data in batches and perform layerwise learning iteratively. But for small or middle-scaled data, such as MNIST [29] and CIFAR10 [30], the learning could be accomplished in a single pass. In this case, compressed sensing-based target propagation needs to be performed only once, implying that

---

**Algorithm 2** Online orthonormal dictionary learning [24]

1: Input: Dataset $X \in \mathbb{R}^{m \times n}$ of $n$ samples of length $m$ and initialized dictionary $D^{(0)}$.
2: for $j = 1$ to $J$ do
3: $S^{(j)} = T(D^{(j-1)} X)$, $T(\cdot)$ is a threshold function to zero out all but $d$ largest elements (in magnitudes) in each column of its matrix input.
4: Run SVD for $X S^{(j)} \top = P \Sigma Q^\top$.
5: $D^{(j+1)} = PQ^\top$.
6: end for

7: Output: Learned dictionary $D = D^{(J)} \top$.
the dictionaries are not required to be orthonormal in the final forward optimization and they could be simply updated by

\[
\hat{D}_l = \arg \min_{D_l} \| \hat{h}_D^l - D_l \hat{h}_W^l \|^2_2 + \lambda D_l \| D_l \|^2_2, \tag{13}
\]

instead of the SVD method used in \([2]\) and \([10]\).

III. EXPERIMENTS

In this section, we aim to prove that the proposed compressed sensing-based target propagation (CSTP) method could achieve better performance than the auxiliary network-based target propagation (ANTP) method \([3]\), on the layerwise learning of deep networks. Moreover, we analyze the proposed CSTP method by ablation study. For comparison, as in \([3]\), we test the proposed CSTP method on two typical datasets, MNIST \([29]\) and CIFAR10 \([30]\). The MNIST dataset consists of 28 \times 28 gray-scale images of handwritten digits. It has a training set of 60,000 samples, and a test set of 10,000 samples. The CIFAR10 dataset consists of 32 \times 32 natural color images in 10 classes. There are 50,000 samples in the training set and 10,000 samples in the testing set.

A. Settings

In the following, we briefly introduce the parameters set for the proposed CSTP-based layerwise learning algorithm. As shown in Algorithm 1, the algorithm consists of three phases. In the initial feedforward phase, the dictionaries are learned with Algorithm 2 for which we set the iteration number to be 20 and the sparsity parameter \(d \approx m_l/3\). The thresholding function \(f_l(\cdot)\) keeps about \(k_l \approx m_l/2\) positive elements. By \([3]\), we initialize the output layer \(W_L\), namely the classifier. To obtain better performance, we suggest to handle \([3]\) with stochastic gradient descent (SGD), for which we set the momentum as 0.9, weight decay as 0.0005, batch size as 64, and epoch number as 30. The learning rate is initialized as 0.01, which is decayed by multiplying a factor of 0.9 every 4 epochs. In the target backpropagation phase, we first diversify the targets by \([6]\) with \(\beta = 0.2\). Then the targets are backpropagated by \([3]\), which is implemented with OMP \([22]\), with \(k_l \approx m_l/3\). In the final layerwise learning phase, we update \(W_l\) and \(D_l\) by directly solving (7) and (13). The output layer \(W_L\) is updated in (12) by SGD, with the same parameters as set in (3). The above parameter settings are adopted both for MNIST and CIFAR10.

B. Results

1) Performance comparison on CIFAR10: For the layerwise learning with CSTP, we derive a test error of 47.76% on a three-layer network of size 3072-2500-1500-10. In contrast, in \([3]\) ANTP reports an error of 49.29% on a four-layer network of size 3072-1000-1000-1000-10. This means that CSTP could achieve higher accuracy than ANTP, when given appropriate network sizes. Recall that the network size of CSTP is constrained by compressed sensing. Similarly to ANTP \([3]\), the CSTP-based layerwise learning performs worse than the prevailing BP method, with an accuracy gap about 4% on the network we test. As discussed in \([31]\), the inferior performance is due to the mismatching between the poorly-separable features and the strong supervision constraint imposed on each layer.

2) Performance comparison on MNIST: In \([3]\), ANTP achieves a test error of 1.94% on a network with 7 hidden layers each consisting of 240 units. The error is reduced about 0.15%, as we test CSTP on a wide but shallow network of size 784-3500-2500-10. From Table 1, it is seen that wider layers tend to yield better performance for CSTP. The reason is as follows. The performance of CSTP depends on the dictionary learning-based feature selection and compressed sensing-based target propagation. For dictionary learning, higher dimensions (namely wider layers) usually could lead to sparser transforms, which are beneficial not only to feature selection but also to compressed sensing.

3) Ablation study of \(D_l\) and \(f_l(\cdot)\): To see the importance of the dictionary \(D_l\) and nonlinear function \(f_l(\cdot)\) in CSTP, we remove them in Algorithm 1 and after fine tuning, we still witness an accuracy reduction about 1.2% for CIFAR10 and 0.7% for MNIST. If only removing \(f_l(\cdot)\), the accuracy decreases about 0.3% for CIFAR10 and 0.1% for MNIST.

Table I: The layerwise learning performance of CSTP on different network architectures.

| Datasets  | Network sizes     | Error (%) |
|----------|-------------------|-----------|
| MNIST    | 784-4000-3000-10  | 1.72      |
|          | 784-3500-2500-10  | 1.79      |
|          | 784-2000-1000-10  | 2.86      |
| CIFAR10  | 3072-4000-3000-10 | 47.48     |
|          | 3072-2500-1500-10 | 47.76     |
|          | 3072-2000-1000-10 | 48.87     |

IV. CONCLUSION

In the letter, we have shown that the cascade of compressed sensing can be established as a reversible network, which allows for layerwise learning and analysis. The optimization target of each layer is derived by inversely propagating the target of the network via compressed sensing. The method avoids learning an auxiliary network to specially propagate the target, reducing the complexity of network learning. Also, the method could achieve favorable performance owing to the stableness of compressed sensing. The proposed target propagation involves two major computations: dictionary learning and sparse recovery, which both have polynomial complexity. To reduce the complexity, it is interesting to sparsify the structures of the dictionaries and projection matrices, such that some combinatorial algorithms with linear complexity could be adopted \([32], [33]\). This will be left for our future work. Note that in the letter we focus our interest on the layerwise learning method that optimizes exactly one layer at a time and thus reduces the network analysis to the layer level. Currently, the method seems difficult to achieve BP level performance, especially on large-scale data \([31]\). The reason is that the finite number of parameters in one layer can hardly handle the matching between poorly-separable features and strong supervision constraints \([31]\). The problem could be addressed if more than one layer is learned at a time \([5], [6]\).
