SC2Net: Sparse LSTMs for Sparse Coding*

Joey Tianyi Zhou[1], Kai Di[1], Jiawei Du[1], Xi Peng[2], Hao Yang[3], Sinno Jialin Pan[4], Ivor W. Tsang[5], Yong Liu[1], Zheng Qin[1], and Rick Siow Mong Goh[1]

[1]Institute of High Performance Computing, A*STAR, Singapore,
[2]College of Computer Science, Sichuan University, China, [3]Amazon, Seattle, USA,
[4]Nanyang Technological University, Singapore, [5]University of Technology Sydney, Australia,
[1]{zhouty,dikai,dujw, liuyong, qinz,gohsm}@ihpc.a-star.edu.sg, [2]pengx.gm@gmail.com,
[3] lancelot365@gmail.com, [4]sinnopan@ntu.edu.sg, [5]ivor.tsang@uts.edu.au

Abstract

The iterative hard-thresholding algorithm (ISTA) is one of the most popular optimization solvers to achieve sparse codes. However, ISTA suffers from following problems: 1) ISTA employs non-adaptive updating strategy to learn the parameters on each dimension with a fixed learning rate. Such a strategy may lead to inferior performance due to the scarcity of diversity; 2) ISTA does not incorporate the historical information into the updating rules, and the historical information has been proven helpful to speed up the convergence. To address these challenging issues, we propose a novel formulation of ISTA (named as adaptive ISTA) by introducing a novel *adaptive momentum vector*. To efficiently solve the proposed adaptive ISTA, we recast it as a recurrent neural network unit and show its connection with the well-known long short term memory (LSTM) model. With a new proposed unit, we present a neural network (termed SC2Net) to achieve sparse codes in an end-to-end manner. To the best of our knowledge, this is one of the first works to bridge the ℓ_1-solver and LSTM, and may provide novel insights in understanding model-based optimization and LSTM. Extensive experiments show the effectiveness of our method on both unsupervised and supervised tasks.

Introduction

Sparse coding (SC) has demonstrated effectiveness in uncovering semantic information from noisy and high dimensional data (Peng et al. 2016), including image super-resolution (Zhong, Lu, and Yang 2012), subspace learning (Peng et al. 2017), background modeling (Cevher et al. 2008), image classification (Mairal et al. 2008), users likes prediction (Guntuku et al. 2016), hashing (Zhou et al. 2016), etc.

To achieve sparse codes, it generally requires to solve an ℓ_1-regularization optimization problem, which is usually computationally expensive. Although a number of ℓ_1-solvers have been proposed, they may suffer from several limitations. First, these methods optimize one variable by fixing others and thus may obtain suboptimal solutions. Second, it is very computationally expensive to the inference of sparse codes. For each data point, the time complexity for sparse coding is proportional to the size and the dimension of the dictionary used, as well as the input dimension. These makes most existing ℓ_1-solvers less impractical in real-world applications.

To address these issues, Gregor and LeCun (2010) recently proposed the Learned ISTA (LISTA) algorithm which unfolds the iterative hard-thresholding (ISTA) algorithm (Blumensath and Davies 2008) – one of the most popular ℓ_1-solvers, into a recurrent neural network (RNN). Benefiting from the new formulation, LISTA simultaneously optimizes the dictionary and sparse codes, which correspond to the weights and outputs of the RNN, respectively. Unlike the traditional ℓ_1-solvers, such as ISTA, LISTA is very computationally efficient in inference since it just passes the input through a neural network instead of solving a convex optimization problem. Although LISTA has received much attention, it suffers from the following limitations as ISTA.

1. LISTA (or ISTA) is a non-adaptive updating approach, which updates the parameters on each dimension with a fixed learning rate. Such a strategy may not be optimal in some cases, and lead to inferior performance. For example, sparse/big data usually requires the per-dimension updating scheme for saving cost and memory.
2. LISTA (or ISTA) does not consider the historical information to design the updating rules. A lot of studies in the optimization community (Qian 1999; Zeiler 2012; Duchi, Hazan, and Singer 2011) have proved that incorporating historical information is helpful to improve the convergence performance of algorithms.

To overcome these limitations, we propose a novel ℓ_1-solver, termed adaptive ISTA, by introducing *adaptive momentum vectors* to enable per-parameter updates and incorporate historical information. Accompanying with advantages of adaptive ISTA, the disadvantage is the difficulty in optimization due to overmany parameters. To be specific, our adaptive ISTA needs automatically learning d_s parameters, where d_s is the dimension of sparse codes, while ISTA only involves one parameter. To make optimization of adaptive ISTA tractable, we recast our adaptive ISTA as a novel RNN unit, named as sparse LSTM (SLSTM), which could be regarded as a variant of long short term memory (LSTM) (Gers, Schraudolph, and Schmidhuber 2002; Wang, Gao, and Yuan 2017). Specifically, we show that the

*Corresponding Author: Xi Peng.

adaptive momentum vectors act as the input and the forget gates in the proposed SLSTM unit. Besides the structure difference in the computational unit, another difference between SLSTM and the vanilla LSTM is that the former could generate sparse outputs thanks to its specifically designed activation function. With the proposed SLSTM unit, we develop a neural network (termed SC2Net) to achieve sparse codes in an end-to-end manner.

Although LISTA and our method are RNN-based optimizers, they are different in following aspects. First, our method achieves sparse codes using a LSTM unit instead of a simple RNN unit. In consequence, it is able to capture historic information which is helpful to speeding up the convergence and improving the performance of our model. Second, the proposed SC2Net does not depend on other sparsity optimizers. In contrast, LISTA requires using the sparse codes given by other ℓ_1-solvers such as ISTA as supervisor. Such differences make our method working in an end-to-end manner possible, thus boosting the performance for un-/supervised tasks. The main contributions of this paper are summarized as follows,

1. We develop a novel variant of ℓ_1-solver by introducing the *adaptive momentum vectors* into ISTA to enable per-parameter updates and encapsulate the historical information in optimization.
2. We propose to solve the proposed adaptive ISTA by recasting it as a new RNN which is a variant of the well-known LSTM. To the best of our knowledge, *this is the first work to bridge the classic sparse optimization methods and LSTM* and may provide novel insights and understandings in model-based optimization and LSTM.
3. Different from some existing simple RNN based solvers such as LISTA, the proposed SC2Net is not an approximation to existing sparse coding approaches. Instead, it is a data driven sparse coding approach and does not require the precomputed sparse codes.

Preliminaries

Given a data matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n] \in \mathbb{R}^{d_x \times n}$, sparse coding aims to learn a dictionary $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_{d_s}] \in \mathbb{R}^{d_x \times d_s}$ that is used to generate sparse codes $\mathbf{S} = [\mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_n] \in \mathbb{R}^{d_s \times n}$ for the input data $\mathbf{X}$. The optimization problem can be formulated as follows,

$$\begin{aligned} \min_{\mathbf{S},\mathbf{B}} \quad & \sum_i \|\mathbf{x}_i - \mathbf{B}\mathbf{s}_i\|_2^2, \\ \text{s.t.} \quad & \|\mathbf{s}_i\|_0 \leq k, \text{ and } \|\mathbf{b}_j\|^2 \leq 1,\ j = 1, \cdots d_s. \end{aligned} \tag{1}$$

The above optimization is hard to solve due to the non-convexity of the ℓ_0 norm. Therefore, it is often relaxed to the following problem with the ℓ_1 norm,

$$\begin{aligned} \min_{\mathbf{S},\mathbf{B}} \quad & \sum_i \|\mathbf{x}_i - \mathbf{B}\mathbf{s}_i\|_2^2 + \lambda\|\mathbf{s}_i\|_1^2, \\ \text{s.t.} \quad & \|\mathbf{b}_j\|^2 \leq 1, \text{ and } j = 1, \cdots d_s. \end{aligned} \tag{2}$$

To solve (2), a conventional way is to alternatingly optimize $\mathbf{B}$ and $\mathbf{S}$, which correspond to two optimization procedures: dictionary learning and sparse approximation.

Specifically, by fixing $\mathbf{S}$, (2) reduces to the following ℓ_2-constrained optimization problem,

$$\begin{aligned} \min_{\mathbf{B}} \quad & \|\mathbf{X} - \mathbf{B}\mathbf{S}\|_F^2, \\ \text{s.t.} \quad & \|\mathbf{b}_i\|^2 \leq 1, \text{ and } i = 1, \cdots d_s. \end{aligned} \tag{3}$$

The above problem is the well-known ridge regression problem, which has a closed-form solution.

By fixing $\mathbf{B}$, (2) reduces to the sparse approximation problem which aims to represent the input $\mathbf{x}$ by a linear combination of $\mathbf{B}$ as follows,

$$\min_{\mathbf{s}} \sum_i \|\mathbf{x}_i - \mathbf{B}\mathbf{s}_i\|_F^2 + \lambda\|\mathbf{s}_i\|_1. \tag{4}$$

The iterative hard-thresholding (ISTA) algorithm is one of the most popular solvers to solve (4). Basically, ISTA decomposes the objective of (4) into two parts: the differentiable part $g(\mathbf{s}) = \|\mathbf{x} - \mathbf{B}\mathbf{s}\|_F^2$, which is updated by the gradient descent algorithm, and the ℓ_1 regularization part, which is updated by the hard thresholding operator. The updating formula can be mathematically expressed as

$$\mathbf{s}^{(t)} = sh_{(\lambda\tau)}\left(\mathbf{s}^{(t-1)} - \tau\nabla g\left(\mathbf{s}^{(t-1)}\right)\right), \tag{5}$$

where the shrinkage function is defined as $sh_{(\lambda\tau)}(\mathbf{s}) = \text{sign}(\mathbf{s})(|\mathbf{s}| - \lambda\tau)_+$. The solution of (5) can be achieved through the following update rule,

$$\begin{aligned} \mathbf{s}^{(t)} &= sh_{(\lambda\tau)}\left(\mathbf{s}^{(t-1)} - \tau\left(\mathbf{B}^T\left(\mathbf{B}\mathbf{s}^{(t-1)} - \mathbf{X}\right)\right)\right) \\ &= sh_{(\lambda\tau)}\left(\mathbf{W}_e\mathbf{s}^{(t-1)} + \mathbf{W}_d\mathbf{x}\right), \end{aligned} \tag{6}$$

where $\mathbf{W}_e = \mathbf{I} - \tau\mathbf{B}^T\mathbf{B}$ and $\mathbf{W}_d = \tau\mathbf{B}^T$. Note that (6) could be treated as a simple RNN. In other words, the ℓ_1-oriented optimization algorithm can be reformulated as a model-based optimization algorithm. The major advantages of LISTA are twofold. First, it could perform fast inference once the network achieves convergence. Second, different from ISTA, LISTA learns a dictionary for sparse coding by updating $\mathbf{W}_e$ and $\mathbf{W}_d$.

Despite of the success of ISTA and LISTA, they suffer from the two limitations discussed in the previous Section: 1) updating the parameters on each dimension with a fixed learning rate, and 2) historical information is discarded during optimization.

Learning Sparse Coding with Sparse LSTM

To overcome aforementioned limitations, in this section, we offer a novel ℓ_1-solver by introducing an *adaptive momentum vector* into ISTA. To efficiently optimize the proposed adaptive ISTA, we recast it as a variant of LSTM with newly designed unit for sparsity, termed SLSTM.

Adaptive ISTA

Recently, a number of algorithms have been proposed for optimizing neural networks by incorporating the "momentum" into dynamics of stochastic gradient descent (SGD). These methods have shown promising performance in improving the robustness and convergence of SGD since the

momentum incorporates the historically updating information (Qian 1999). To further improve the performance of the momentum-based SGD, Adagrad (Duchi, Hazan, and Singer 2011) and Adadelta (Zeiler 2012) introduce adaptation into SGD so that the learning rates are different for different parameters. These methods basically perform larger updates for infrequent parameters and smaller updates for frequent parameters. Extensive numerical studies have demonstrated that the adaptation improves convergence performance drastically over the non-adaptive SGD methods.

Borrowing the high-level idea from these methods, we introduce the *adaptive momentum vectors* $\mathbf{i}^{(t)}, \mathbf{f}^{(t)}$ into ISTA at the time stamp t, which can be formulated as follows,

$$\begin{aligned} \tilde{\mathbf{c}}^{(t)} &= \mathbf{W}_e \mathbf{s}^{(t-1)} + \mathbf{W}_d \mathbf{x}, \\ \mathbf{c}^{(t)} &= \mathbf{f}^{(t)} \odot \mathbf{c}^{(t-1)} + \mathbf{i}^{(t)} \odot \tilde{\mathbf{c}}^{(t)}, \\ \mathbf{s}^{(t)} &= sh_{(\lambda\tau)}(\mathbf{c}^{(t)}), \end{aligned} \tag{7}$$

where $\odot$ is the element-wise product of the vectors. Note that following the above notations, the updating rule in ISTA can be equivalently expressed as $\mathbf{s}^{(t)} = sh_{(\lambda\tau)}(\tilde{\mathbf{c}}^{(t)})$.

Different from ISTA, our method not only takes the current information, but also the previous information into consideration. To be exact, our method formulates the linear combination of $\mathbf{c}^{(t-1)}$ at the previous iteration and $\tilde{\mathbf{c}}^{(t)}$ at the current iteration, which are weighted by adaptive momentum vectors $\mathbf{f}^{(t)}$ and $\mathbf{i}^{(t)}$, respectively. The adaptive momentum vectors allow combination of two outputs on the level of parameters, which is different from directly applying momentum methods into ISTA. Although scuh a linear combination may be helpful to give sparse codes, we pass $\tilde{\mathbf{c}}^{(t)}$ into the shrinkage function again to ensure sparsity. We name this method as "adaptive ISTA" in the sequel. In adaptive ISTA, $\mathbf{c}^{(t)}$ accumulates all the historical information with different weights $\mathbf{f}^{(t)}$ and $\mathbf{i}^{(t)}$ for different iteration t, which is spiritually similar to the diagonal matrix containing the sum of the squares of the past gradients in Adagrad.

Sparse Long Short Term Memory Unit

One issue of our proposed adaptive ISTA is that it makes the parameters learning difficulty, i.e., how to adaptively determine the values of momentum vectors $\mathbf{f}^{(t)}$ and $\mathbf{i}^{(t)}$. Existing SGD methods, such as Adadelta, solve a similar problem by empirically reducing the value of the momentum after several training epochs. Clearly, such a strategy is not practical in our case since the momentum in our adaptive ISTA is a vector instead of a scalar. Thus, it is preferable to learn $\mathbf{f}^{(t)}$ and $\mathbf{i}^{(t)}$ from data.

To this end, we propose to parameterize the adaptive momentum vectors with the output of the sparse codes at the previous layer as well as the input data, such that $\mathbf{f}^{(t)}$ and $\mathbf{i}^{(t)}$ are learned from the data without tediously hand-craft tuning. More interestingly, such an idea could be implemented by recasting adaptive ISTA as a novel LSTM unit, termed sparse LSTM (SLSTM), as shown in Figure 1. In the figure, the "input gate" and the "forget gate" correspond to $\mathbf{i}^{(t)}$ and $\mathbf{f}^{(t)}$, respectively. Note that, SLSTM does not have the

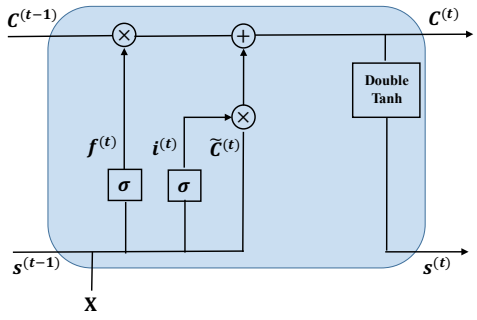


Figure 1: The proposed Sparse LSTM (SLSTM) Unit.

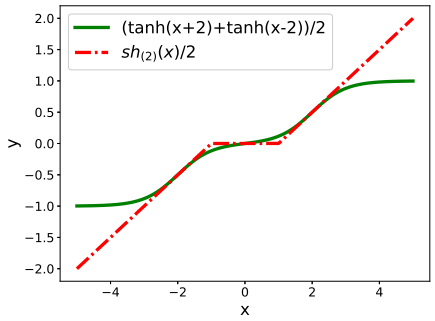
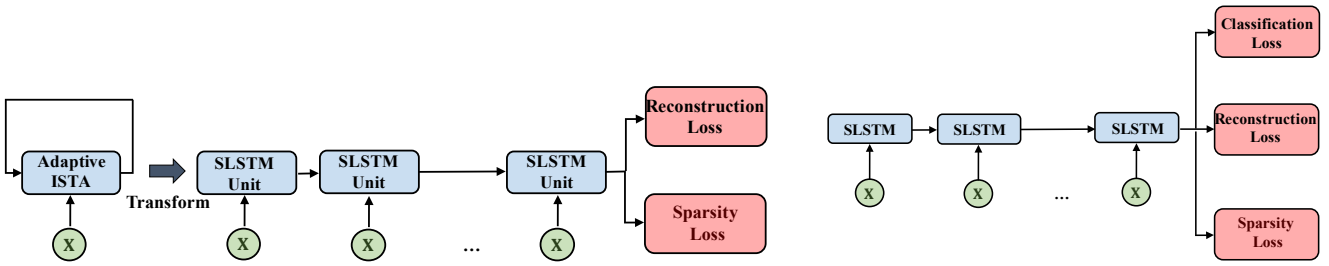


Figure 2: Examples of the Double Tanh (in green color) function and the Shrinkage (in red color) function.

"output gate" like the vanilla LSTM. The SLSTM unit is achieved by rewriting (7) as follows:

$$\begin{aligned} \mathbf{i}^{(t)} &= \sigma(\mathbf{W}_{is}\mathbf{s}^{(t-1)} + \mathbf{W}_{ix}\mathbf{x}), \\ \mathbf{f}^{(t)} &= \sigma(\mathbf{W}_{fs}\mathbf{s}^{(t-1)} + \mathbf{W}_{fx}\mathbf{x}), \\ \tilde{\mathbf{c}}^{(t)} &= \mathbf{W}_e\mathbf{s}^{(t-1)} + \mathbf{W}_d\mathbf{x}, \\ \mathbf{c}^{(t)} &= \mathbf{f}^{(t)} \odot \mathbf{c}^{(t-1)} + \mathbf{i}^{(t)} \odot \tilde{\mathbf{c}}^{(t)}, \\ \mathbf{s}^{(t)} &= h_{(\mathbf{D},\mathbf{u})}(\mathbf{c}^{(t)}), \end{aligned} \tag{8}$$

where $\mathbf{W}_*$ denotes the weight matrices, e.g., $\mathbf{W}_{is}$ is the matrix of weights from the input gate to the outputs, $\sigma(\mathbf{x}) = \frac{1}{1+e^{-\mathbf{x}}}$, and $h_{(\mathbf{D},\mathbf{u})} = \mathbf{D}(tanh(\mathbf{x}+\mathbf{u}) + tanh(\mathbf{x}-\mathbf{u}))$. The variables $\mathbf{u}$ and $\mathbf{D}$ denote a trainable vector and a diagonal matrix, respectively. It is worth noting that we use the smooth and differentiable nonlinear activation function named as "*Double tanh*" (Gregor and LeCun 2010) instead of the shrinkage function for following two reasons. On one hand, the cell recurrent connection needs a function whose second derivative sustains for a long span to address the vanishing gradient problem (Chung et al. 2014). On the other hand, the *Double tanh* function could approximate the shrinkage function well within the interval of $[-\mathbf{u},\ \mathbf{u}]$. Figure 2 illustrates the difference between the *Double tanh* function and the shrinkage function.



(a) Unsupervised SLTSM. (b) Supervised SLTSM.

Figure 3: The Architecture of SC2Net.

Recall that both ISTA and LISTA generate the output of sparse codes exclusively on the previous output. This kind of architecture leads to the so-called "error propagation phenomenon". More specifically, the errors in the first few layers are propagated and further amplified in the upcoming layers. Furthermore, once the useful information is discarded in a previous layer, the upcoming layers will have no chance to utilize the discarded information. Fortunately, this issue can be alleviated with the usage of the "cell" state $\mathbf{C}^{(t)}$ in our SLSTM. The "cell" plays as another "eye" to supervise the optimization, which brings two major advantages. 1) It captures long-term dependence from the previous outputs. 2) It automatically accumulates important information and forgets useless or redundant information in the dynamics of neural networks.

Unsupervised SLTSM Network

LISTA requires additional cost to pre-compute $\mathbf{s}^*$, which could be computationally expensive. Furthermore, the performance of LISTA largely depends on the quality of $\mathbf{s}^*$. To address this issue, we propose a novel optimization framework for SC, named as SC2Net in short, which reformulates SC as a LSTM network instead of a simple RNN. Specifically, the sparsity loss and the reconstruction loss are incorporated into our SC2Net to supervise the optimization process. With the sparsity loss, SC2Net could generate sparse codes in parallel. With the reconstruction error, SC2Net is no longer an approximation to existing SC methods. In other words, SC2Net does not require any prior knowledge to achieve sparse codes.

For any data point $\mathbf{x}$, the reconstruction loss is defined as:

$$\left\|\mathbf{x} - \frac{1}{\tau}\mathbf{W}_d^T\mathbf{s}\right\|_F^2 \tag{9}$$

where $\mathbf{s}$ is the output of the encoder in the network w.r.t. $\mathbf{x}$ and $\mathbf{B} = \frac{1}{\tau}\mathbf{W}_d^T$ as defined in (6). Here, we do not learn the individual decoding matrix. Instead, we reuse the encoding matrix $\mathbf{W}_d$. Such a strategy has two advantages: 1) It maintains the physical meaning of the original formulation (6), i.e., the encoding matrix is the transpose of the decoding matrix. 2) Different from (Rolfe and Lecun 2013), as the encoding and decoding matrix is shared, the computation cost to train LISTA is reduced.

To further control the sparsity of the solution, the ℓ_1 loss is also considered in our formulation. The overall cost function for SC2Net is defined as follows,[1]

$$\left\|\mathbf{x} - \frac{1}{\tau}\mathbf{W}_d^T\mathbf{s}\right\|_F^2 + \lambda\|\mathbf{s}\|_1. \tag{10}$$

The architecture of SC2Net in the unsupervised learning manner is illustrated in Figure 3(a). In experiments, our numerical experiments will verify the effectiveness of SC2Net over RNN-based optimization solvers.

Supervised SLSTM Network

In this section, we further develop a supervised version of SC2Net with the proposed SLSTM unit. The architecture of supervised SC2Net is shown in Figure 3(b). Comparing with unsupervised SC2Net, the loss of supervised SC2Net is with an additive term to incorporate label information. In this work, we adopt the popular softmax$(\cdot)$ function

$$\text{softmax}(\mathbf{Cs}), \tag{11}$$

where $\mathbf{C} \in \mathbb{R}^{d_s \times c}$ is a classification matrix to be learned.

Once SC2Net converges, the predicted label of a give data point $\mathbf{x}$ could be obtained by first passing $\mathbf{x}$ into SC2Net and then solving the following problem:

$$\arg\max_j \text{softmax}(\mathbf{C}_j\mathbf{s}), \tag{12}$$

where $\mathbf{C}_j$ is the j-th row of $\mathbf{C}$.

Experiments

In this section, we carry out experiments to verify the effectiveness of the proposed SLSTM for classification on several real-world datasets.

Setup and Datasets

For fair comparisons, we employ the same optimization solver, Adadelta (Zeiler 2012), to train all neural-network-based approaches with a GPU of NVIDIA TITAN X using Keras. Regarding implementations, we experimentally find

[1] In the experiments, the network is often learned through minimizing the average cost over a set of training samples using the stochastic gradient descend method.

Table 1: Overall Comparison in terms of Classification Acc.

Datasets	ISTA	FISTA	LISTA	LFISTA	SLSTM
Unsupervised Setting					
MNIST	85.25	86.65	85.75	85.55	**89.81**
CIFAR-10	35.05	36.75	42.12	42.75	**65.40**
Supervised Setting					
MNIST	86.05	86.75	87.55	87.95	**94.31**
CIFAR-10	51.05	51.75	52.35	53.10	**79.53**

that all evaluated methods perform stable when λ ranges between 0.01 and 1. Thus, we fix the sparsity parameter $\lambda = 0.1$ for all evaluated methods in the unsupervised learning setting. In other words, all evaluated methods including our proposed method have the same objective function with the fixed λ. The only difference among them are the choices of the optimization approaches.

In experiments, we compare the proposed SLSTM with ISTA (Blumensath and Davies 2008), FISTA (Blumensath and Davies 2008), LISTA (Gregor and LeCun 2010), and LFISTA (Moreau and Bruna 2017). FISTA is an accelerated version of ISTA that converges faster, both in theory and practice. It considers the difference of the last two outputs of the shrinkage function. The non-neural-network-based methods (i.e. ISTA and FISTA) adopt the algorithm proposed in (Mairal et al. 2010) to learn a dictionary. LISTA and LFISTA achieve sparse coding by unfolding the ISTA and FISTA into a simple RNN, respectively.

We evaluate the performance of these methods using MINST [2] and CIFAR-10 (Krizhevsky 2009). MINST contains 60,000 training images and 10,000 testing images sampled from the digits (0-9), where each image is of the size of 28×28. The CIFAR-10 dataset is another widely used benchmark, which contains 60,000 $32 \times 32 \times 3$ color images distributed over 10 subjects. For computational efficiency and better performance, we extract features from the CIFAR-10 raw data using an open access CNN model[3], where the last fully connected layer is removed since it is used to perform classification in the original setting. The extracted features are with 2,034 dimensions for CIFAR-10. Following the experimental setting in (Gregor and LeCun 2010; Li and others 2014), we normalize all images into $[0, 1]$ and set the size of the dictionary as 100. Note that, we have also investigated other values for the dictionary size, and the proposed method consistently outperforms the baselines in these settings.

Performance Comparison

In this section, we evaluate the performance of our method for classification in supervised and unsupervised setting. In the supervised learning setting, we utilize the classification result to guide the learning of sparse codes. In contrast, in the unsupervised setting, we do not utilize the label information to learn sparse codes. As LISTA and LFISTA can also be extended as a cascade model to give discriminative codes by incorporating the label information into the loss function. For fair comparisons, we therefore evaluate the performance of supervised variants of these methods in the same way. More specifically, after getting sparse codes with the evaluated methods, we train a logistics regression classifier with the extracted sparse codes. Table 1 reports the results, from which we observe that all the methods perform better on MNIST than on CIFAR-10 because classification on the latter is more challenging. Furthermore, SLSTM outperforms all the baselines by a large margin on the both datasets. This verifies to the advantages of SLSTM, i.e., the long-term dependence, nonlinearity, and data-driven coding optimization procedure.

[2] http://yann.lecun.com/exdb/mnist/

[3] https://github.com/fchollet/keras/blob/master/examples/cifar10_cnn.py

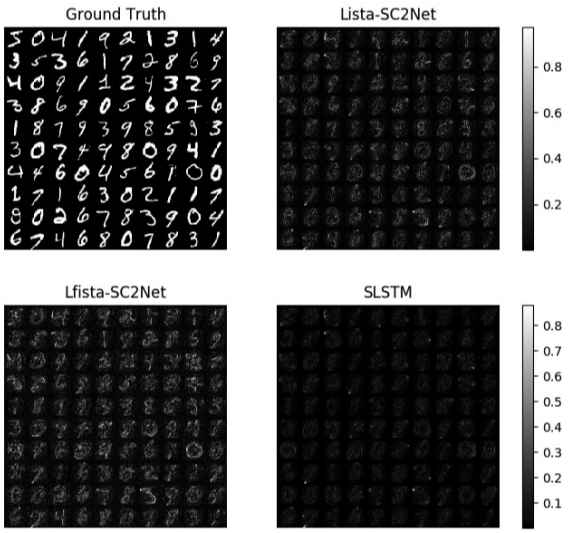


Figure 4: Reconstructed images on MNIST: The more black is, the lower error is.

We further evaluate the performance of our method in terms of reconstruction errors in the the unsupervised setting. Figures 4 and 5 show the reconstructed images on MINIST and CIFAR-10, respectively. From the results, one could observe that SLSTM gives better reconstructions than those obtained by the baselines. Either on MNIST or CIFAR-10, the proposed SLSTM recovers more details than the evaluated simple RNN based optimization methods (i.e., LISTA and LFISTA).

Effectiveness of The Proposed Architecture

The proposed SC2Net can also be used to improve some existing LISTA-like methods. To verify this claim, we take LISTA and LFISTA as two showcases and denote the corresponding variants by LISTA-SC2Net and LFISTA-SC2Net. The performance comparisons are reported in Table 2. One could find that LISTA-SC2Net and LFISTA-SC2Net outperforms their counterparts LISTA and LFISTA consistently, which provides another evidence to the effectiveness of our framework.

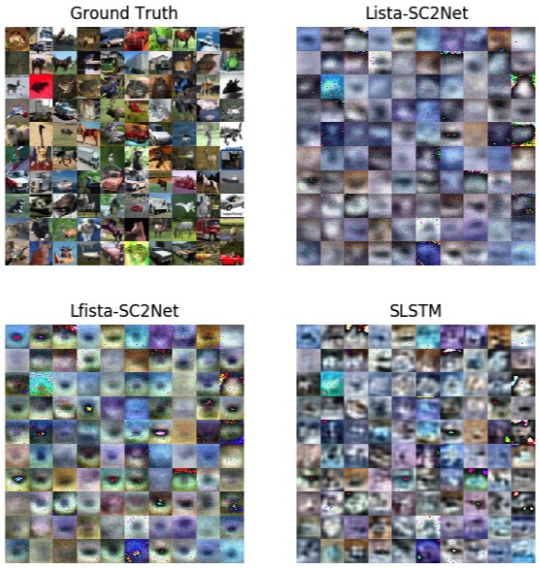
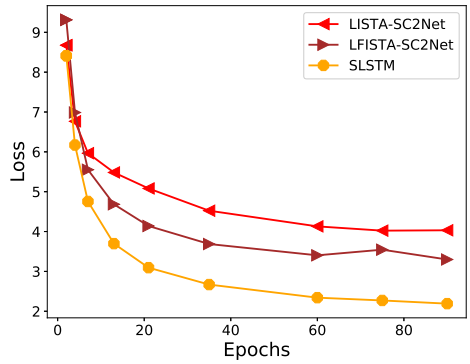


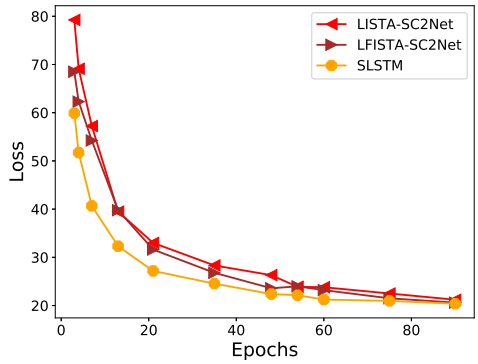
Figure 5: Reconstructed images on CIFAR-10

Table 2: Impact of SC2Net on Classification Accuracy

Datasets	LISTA	LFISTA	LISTA-SC2Net	LFISTA-SC2Net
Unsupervised Setting				
MNIST	85.25	86.65	88.75	88.95
CIFAR-10	42.12	42.75	60.20	60.50
Supervised Setting				
MNIST	87.55	87.95	90.85	91.60
CIFAR-10	52.35	53.10	71.10	72.48



(a) Loss vs. Epochs on MNIST.



(b) Loss vs. Epochs on CIFAR-10.

Figure 6: Algorithm convergence analysis.

Convergence Analysis

In this section, we investigate the influence of the SLSTM unit on the convergence performance. Moreau and Bruna (2017) has shown that RNN based optimization approaches (LISTA and FISTA) converge faster than traditional iterative approaches (ISTA and LFISTA). For fair comparisons, we use LISTA-SC2Net and LFISTA-SC2Net as baselines in our experiments. In other words, the only difference among all the evaluated methods is the neural unit. Figure 6 shows the average objective loss (10) with increasing training epochs. From the results, we observe that SLSTM enjoys faster convergence speed and reaches a lower overall cost than the other methods. With more epochs, the performance gap between our method and other tested methods becomes larger.

Sparsity Parameter Analysis

In the objective function of our method, there is only one user-specified parameter, i.e., the sparsity parameter λ, which controls the trade-off between the sparsity level and the reconstruction error. In this section, we investigate the influence of the parameter w.r.t. the reconstruction loss by varying the values of λ in Figure 7. The results show that ISTA and FISTA achieve almost the same loss after around epoches. Furthermore, SLSTM consistently produces the smallest loss and outperforms other baselines in all tests.

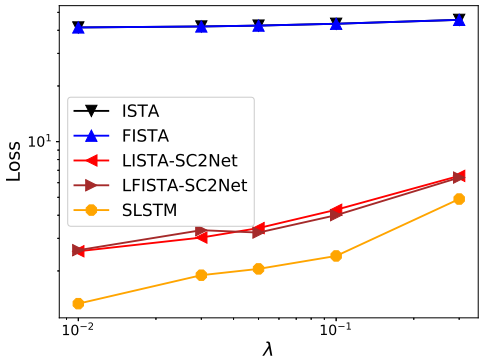
Related Work

Our proposed architecture SC2Net and the Sparse LSTM unit are highly related to two topics, i.e., RNN-based optimization solvers (especially, LISTAs) and Long Short Term Memory Networks. We briefly review them in this section.

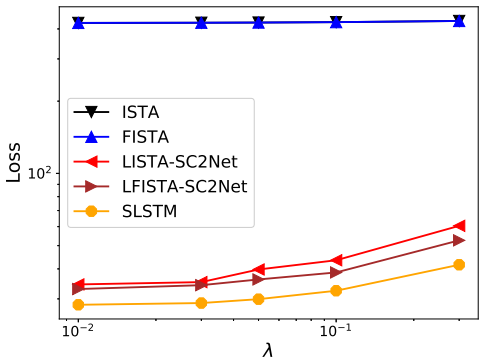
LISTAs

Recently, Gregor and LeCun (2010) proposed Learned ISTA (LISTA) to recast the well-known ISTA as a simple recurrent neural network. The work has attracted a lot of interests from the community (Rolfe and Lecun 2013; Sprechmann, Bronstein, and Sapiro 2015; Zuo et al. 2015; Yang et al. 2016; Diamond et al. 2017; Wang, Ling, and Huang 2016; Zuo et al. 2016; Wang et al. 2016) since it gives a feasible way to bridge statistical inference methods and neural networks.

With the simple RNN-based reformulation, sparse coding is achieved by optimizing a RNN. One major advantage of these works is that they perform inference by simply



(a) Reconstruction loss vs. λ on MNIST.



(b) Reconstruction loss vs. λ on CIFAR-10.

Figure 7: Codes sparsity analysis.

passing the input through a neural network, thus remarkably improving the inference speed for sparse coding. Motivated by the success of these works, a lot of LISTA-like methods (Moreau and Bruna 2017; Rolfe and Lecun 2013) have been proposed, but they suffer from two limitations. First, LISTA is actually a supervised method, which uses the precomputed sparse codes of ISTA as supervisors. However, this is impractical in real-world applications to obtain supervisors. Second, the performance of LISTA is upper bounded by that of ISTA in theory as the former is an approximation of the latter.

The most related work with our method is LFISTA (Moreau and Bruna 2017) which accelerates LISTA by building a shorter path between the next iteration and the previous two iterations. However, our method is different from LFISTA in following aspects: 1) The architectures are different. LFISTA is built upon a simple RNN like LISTA, which adds a shorter but fixed-length path between the $t-1$ and $t+1$ layer. In contrast, SLSTM is built upon a LSTM, which employs a memory unit to capture longer range dependency. 2) The definition of momentum is different. LFISTA defines a momentum term using the difference between the last two outputs. In contrast, the proposed SLSTM borrows the concept "gates" in LSTM to model momentum and the "cell" state stores all the historical information. In other words, only SLSTM considers per-dimension combination of all the stored historical outputs and current update. 3) Our method does not require using the precomputed sparse codes as the prior and performs sparse coding in a data driven way.

Long Short Term Memory Networks

LSTMs have been successfully applied to deal with various sequential data (Gers, Schraudolph, and Schmidhuber 2002; Chung et al. 2014; Yao et al. 2015). Different from these existing methods, our proposed SLSTM is specifically designed for sparse coding. Another major difference is that we introduce a smooth sparse activation function (i.e. double tanh) and modify cell states and output gate to learn sparse representation. In contrast, LSTMs is quite general and cannot give sparse codes. To the best of our knowledge, this is one of the first works to bridge sparse coding and LSTMs.

Conclusion

In this paper, we propose a novel formulation of ISTA, adaptive ISTA, for sparse coding. The proposed adaptive ISTA employs the per-dimension update strategy and takes the historical information into optimization. With the help of the RNN-based architecture, we show that the optimization of adaptive ISTA could be casted as a LSTM with a novel computational unit. To verify the efficacy of our idea, we build a model, termed SC2Net, which enjoys the end-to-end learning and independence on the sparse codes prior given by traditional ℓ_1-solvers. Extensive experimental results show the effectiveness of our method comparing with several famous methods including ISTA, LISTA, FISTA, and LFISTA. In future, we plan to investigate the effectiveness of our method for supervised applications such as action recognition (Yang et al. 2017) and identification (Zhu et al. 2016).

Acknowledgments

This work was supported by Programmatic grant no. A1687b0033 from the Singapore governments Research, Innovation and Enterprise 2020 plan (Advanced Manufacturing and Engineering domain). Xi Peng is funded by National Nature Science Foundation of China under grant No.61432012 and No.U1435213, and the Fundamental Research Funds for the Central Universities under grant No.YJ201748. Ivor W. Tsang is grateful for the support from the ARC Future Fellowship FT130100746, ARC grant LP150100671, and DP180100106. Sinno Jialin Pan thanks the support from NTU Singapore Nanyang Assistant Professorship (NAP) grant M4081532.020, Singapore MOE AcRF Tier-2 grant MOE2016-T2-2-060, and Singapore MOE AcRF Tier-1 grant 2016-T1-001-159.

References

Blumensath, T., and Davies, M. E. 2008. Iterative thresholding for sparse approximations. *Journal of Fourier Analysis and Applications* 14(5):629–654.

Cevher, V.; Sankaranarayanan, A. C.; Duarte, M. F.; Reddy, D.; Baraniuk, R. G.; and Chellappa, R. 2008. Compressive sensing for background subtraction. 155–168.

Chung, J.; Gulcehre, C.; Cho, K.; and Bengio, Y. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Diamond, S.; Sitzmann, V.; Heide, F.; and Wetzstein, G. 2017. Unrolled optimization with deep priors. *arXiv preprint arXiv:1705.08041*.

Duchi, J.; Hazan, E.; and Singer, Y. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12(Jul):2121–2159.

Gers, F. A.; Schraudolph, N. N.; and Schmidhuber, J. 2002. Learning precise timing with lstm recurrent networks. *Journal of Machine Learning Research* 3(Aug):115–143.

Gregor, K., and LeCun, Y. 2010. Learning fast approximations of sparse coding. In *ICML-10, June 21-24, 2010, Haifa, Israel*, 399–406.

Guntuku, S. C.; Zhou, J. T.; Roy, S.; Lin, W.; and Tsang, I. W. 2016. Understanding deep representations learned in modeling users likes. *IEEE Transactions on Image Processing* 25(8):3762–3774.

Krizhevsky, A. 2009. Learning multiple layers of features from tiny images.

Li, Y., et al. 2014. A novel statistical algorithm for multiclass eeg signal classification. volume 34, 154–167. Elsevier.

Mairal, J.; Bach, F. R.; Ponce, J.; Sapiro, G.; and Zisserman, A. 2008. Discriminative learned dictionaries for local image analysis. In *CVPR 2008, 24-26 June 2008, Anchorage, Alaska, USA*.

Mairal, J.; Bach, F.; Ponce, J.; and Sapiro, G. 2010. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research* 11(Jan):19–60.

Moreau, T., and Bruna, J. 2017. Understanding neural sparse coding with matrix factorization.

Peng, X.; Lu, C.; Zhang, Y.; and Tang, H. 2016. Connections between nuclear norm and frobenius norm based representation. *IEEE Trans Neural Netw. Learn. Syst.* PP(99):1–7.

Peng, X.; Lu, J.; Yi, Z.; and Rui, Y. 2017. Automatic subspace learning via principal coefficients embedding. *IEEE Trans. Cybern.* 47(11):3583–3596.

Qian, N. 1999. On the momentum term in gradient descent learning algorithms. *Neural networks* 12(1):145–151.

Rolfe, J. T., and Lecun, Y. 2013. Discriminative recurrent sparse auto-encoders. In *ICLR 2013, May 2 - 4, 2013, Scottsdale, USA*.

Sprechmann, P.; Bronstein, A. M.; and Sapiro, G. 2015. Learning efficient sparse and low rank models. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37(9):1821–1833.

Wang, Z.; Chang, S.; Zhou, J.; Wang, M.; and Huang, T. S. 2016. Learning a task-specific deep architecture for clustering. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, 369–377. SIAM.

Wang, Q.; Gao, J.; and Yuan, Y. 2017. Embedding structured contour and location prior in siamesed fully convolutional networks for road detection. *IEEE Transactions on Intelligent Transportation Systems*.

Wang, Z.; Ling, Q.; and Huang, T. S. 2016. Learning deep l0 encoders. In *AAAI 2016, February 12-17, 2016, Phoenix, Arizona, USA.*, 2194–2200.

Yang, Y.; Sun, J.; Li, H.; and Xu, Z. 2016. Deep admm-net for compressive sensing MRI. In *NIPS 2016, December 5-10, 2016, Barcelona, Spain*, 10–18.

Yang, Y.; Deng, C.; Gao, S.; Liu, W.; Tao, D.; and Gao, X. 2017. Discriminative multi-instance multitask learning for 3d action recognition. *IEEE Trans. on Multimedia* 19(3):519–529.

Yao, K.; Cohn, T.; Vylomova, K.; Duh, K.; and Dyer, C. 2015. Depth-gated lstm. *arXiv preprint arXiv:1508.03790*.

Zeiler, M. D. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Zhong, W.; Lu, H.; and Yang, M. 2012. Robust object tracking via sparsity-based collaborative model. In *CVPR 2012, Providence, RI, USA, June 16-21, 2012*, 1838–1845.

Zhou, J. T.; Xu, X.; Pan, S. J.; Tsang, I. W.; Qin, Z.; and Goh, R. S. M. 2016. Transfer hashing with privileged information. In *IJCAI 2016, New York, NY, USA, 9-15 July 2016*, 2414–2420.

Zhu, H.; Lu, J.; Cai, J.; Zheng, J.; Lu, S.; and Thalmann, N. M. 2016. Multiple human identification and cosegmentation: A human-oriented crf approach with poselets. *IEEE Trans. on Multimedia* 18(8):1516–1530.

Zuo, W.; Ren, D.; Gu, S.; Lin, L.; and Zhang, L. 2015. Discriminative learning of iteration-wise priors for blind deconvolution. In *IEEE Conference on Computer Vision and Pattern Recognition*, 3232–3240.

Zuo, W.; Ren, D.; Zhang, D.; Gu, S.; and Zhang, L. 2016. Learning iteration-wise generalized shrinkage–thresholding operators for blind deconvolution. *IEEE Transaction on Image Processing* 25(4):1751–1764.