WARWICK
THE UNIVERSITY OF WARWICK

Manuscript version: Author's Accepted Manuscript
The version presented in WRAP is the author's accepted manuscript and may differ from the published version or Version of Record.

Persistent WRAP URL:
http://wrap.warwick.ac.uk/129925

How to cite:
Please refer to published version for the most recent bibliographic citation information. If a published version is known of, the repository item page linked to above, will contain details on accessing it.

Copyright and reuse:
The Warwick Research Archive Portal (WRAP) makes this work by researchers of the University of Warwick available open access under the following conditions.

Copyright © and all moral rights to the version of the paper presented here belong to the individual author(s) and/or other copyright owners. To the extent reasonable and practicable the material made available in WRAP has been checked for eligibility before being made available.

Copies of full items can be used for personal research or study, educational, or not-for-profit purposes without prior permission or charge. Provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way.

Publisher's statement:
Please refer to the repository item page, publisher's statement section, for further information.

For more information, please contact the WRAP Team at: wrap@warwick.ac.uk.

warwick.ac.uk/lib-publications

The Structure Transfer Machine Theory and Applications

Baochang Zhang, *Member, IEEE,* Wankou Yang, *Member, IEEE,* Ze Wang, Lian Zhuo, Xiantong Zhen, *Member, IEEE,* Jungong Han

***Abstract*—Representation learning is a fundamental but challenging problem, especially when the distribution of data is unknown. In this paper, we propose a new representation learning method, named Structure Transfer Machine (STM), which enables feature learning process to converge at the representation expectation in a probabilistic way. We theoretically show that such an expected value of the representation (mean) is achievable if the manifold structure can be transferred from the data space to the feature space. The resulting structure regularization term, named manifold loss, is incorporated into the loss function of the typical deep learning pipeline. The STM architecture is constructed to enforce the learned deep representation to satisfy the intrinsic manifold structure from the data, which results in robust features that suit various application scenarios, such as digit recognition, image classification and object tracking. Compared with state-of-the-art CNN architectures, we achieve better results on several commonly used public benchmarks.**

***Index Terms*—Transfer learning, convolutional neural networks, manifold loss, learning theory.**

I. Introduction

Human perception system abstracts the correct concept, when the relationship or the compactness of the intra-class data (small structure variations) is maintained after the perception; otherwise it will cause conceptual errors [10]. Analogously, data-driven learning approaches become a trend which by all means aim to maintain the class-specific feature compactness (perception) of the input data [45], [36], [44]. For a learning algorithm, such a compactness can be accomplished if the expected presentation is achieved for an unbiased estimator (classifier) [25], [2].

Traditional hand-crafted features often require human expert knowledge, thereby making themselves domain specific. In contrast, deep learning based features can be learned automatically by composing multiple nonlinear transformations, yielding more abstract and useful representations. However, typically no distribution prior is embedded into the learning of deep features, making such schemes uncontrollable for certain circumstances. Recently, a center loss regularization term which is in essence a Gaussian prior is successfully exploited in deep learning to improve face recognition performance [44]. However, such a system does not work properly when the data is of complicated structure. Considering the fact that the conventional deep learning features are able to better distinguish the between-class variability [24], [28], we attempt to breakthrough the restriction of simple Gaussian prior [44] into a better prior so as to tolerate large intra-class variations. Thus, the inter-class samples can be still well separated even with the large intra-class variance due to super discriminant capability of deep learning. As a result, the generalization ability of the learned feature is expected to be significantly improved.

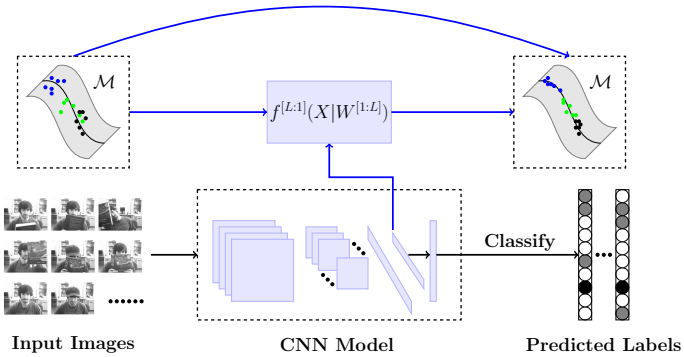


Fig. 1: Basic idea of the structure transferred machine (STM) method. By incorporating the manifold structure calculated in the input space into CNNs' loss function, termed as manifold loss. The expectation of the representation is theoretically obtained in a probabilistic way, as a result the variations among local neighbors of the data are mitigated due to converging into the expected representation. Function f is the feature mapping by CNNs (not necessarily the entire net) which will be introduced in section IV-B.

In this paper, we discover that a desired representation in deep learning can actually be achieved and the local neighborhood with no constraint of the data structure required is able to converge at its expectation during the learning process. More importantly, it is noticed that the features describing the local structure enable representing the data better than the global ones, since the global features tend to be inaccurate when the data variation is usually large in real-world applications. The above observations inspire us to integrate a nonlinear manifold

The work was supported in part by National Natural Science Foundation of China under Grants 61971004, 61773117, 614730867 and by Shenzhen Science and Technology Program, Shenzhen Peacock Plan KQTD2016112515134654 and National Key R & D Plan (2017YFC0821102). (Corresponding author: Jungong Han)

B. Zhang Z. Wang and Z. Zhuo are with Beihang University, Beijing, China. B. Zhang is also with Shenzhen Academy of Aerospace Technology, Shenzhen 100083, China. E-mail: Bczhang@139.com

W. Yang is with the School of Automation, Southeast University, Nanjing 210096, China.

X. Zhen are with Inception Institute of Artificial Intelligence (IIAI), United Arab Emirates.

J. Han is with WMG Data Science at University of Warwick, Coventry, UK. E-mail: jungonghan77@gmail.com.

structure encoding more flexible structure of the data than the center loss [44] into the objective function of deep feature learning. Thus, we can accommodate variations among local neighbours such as rotations, rescalings, and translations so as to gain system robustness. However, directly embedding such data distribution into the deep learning framework is not an easy task at all, because formulating the underlying concept into appropriate training criteria is problematic. In this paper, we theoretically show that the expected representation can be achieved in a probabilistic way as long as the property of manifold structure is revealed in the objective function of deep learning.

Upon such a proven, we present a novel structure transfer machine (STM) to learn structured deep features, the framework of which is illustrated in Fig. 1. Our STM starts with the incorporation of manifold structure into Convolutional Neural Network (CNNs) by calculating the manifold structure based on existing algorithms (i.e., local linear embedding (LLE) or Laplacian) in the input space. Such a manifold structure is in turn transferred to feature space and integrated into the loss function of the CNNs. Afterwards, the new CNN models are used to extract constitutional feature maps, where the intrinsic manifold structure is preserved even if it changes from the data space to the feature space. Experimental results demonstrate that these learned features can yield state-of-the-art performance in various computer vision tasks (e.g., digit recognition, natural object recognition, image classification (ImageNet) and object tracking) on commonly used benchmarks.

A. Our Contributions

Below, we summarize our major contributions in this paper.

- A theorem is developed to reveal that the expectation of representation can be obtained in a probabilistic way if a structure regularization is incorporated into the deep learning pipeline. With such structure regularization, we revise typical deep learning networks to a Structure Transfer Machine, which gains state-of-the-art performance on image classification and object tracking.
- With the aid of manifold, the structure of the input data is transferred into the feature space (output) with the intention to alleviate the unstructured problem in the higher-dimensional space, which eventually transfers the data structure into constraints in CNNs and leads to manifold loss. It is also demonstrated that in the deep feature space, the proposed manifold loss indeed improves the performance over the intra-class compactness methods such as the center loss.

B. Organization of This Paper

The rest of this paper is organized as follows. In section II, we analyze the related works. We introduce the definitions of the basic concepts and notations required to understand the structure transfer machine in section III. Subsequently, we propose the deep STM architecture and its corresponding theoretical analysis in section IV. In section V, we provide the experimental results as well as analysis on digit recognition, natural object recognition, large size image classification and object tracking. Finally, the last section draws conclusions.

II. Related Work

Increasing the discriminating performance of the features learned by CNNs for images has received extensive attention in recent years. We roughly divide the related works into the following three parts.

Manifold learning. Manifold learning methods [37], [33], [1], [15], [34] assume that high dimensional data can be viewed as a set of geometrically related points lying on (or close to) the surface of a smooth low dimensional manifold. There are some manifold based learning methods published recently [23], [8], such as region manifold [19], graph manifold [31], product manifold [38], Grassmannia manifold [42], or its application to zero-shot [7]. We find that all of them are different from ours because the features in these methods are designed for specific tasks, such as region manifold exploring manifold for image retrieval, product manifold introducing the product manifold filter for the problem of bijective correspondence recovery, and Grassmannia manifold for spectral clustering. We focus on structure transferring in CNNs for general feature learning and actually provide a new theoretical investigation into CNNs.

Manifold regularization. Highly relevant works contain the structure related regularization techniques [28] as well as the techniques that embed the prior knowledge, such as 2D topological structure of input data [23], both of which reveal that regularizing data structure is pretty useful when dealing with the image classification task. In [24], the adversarial examples suffer from performance degradation caused by small perturbations, manifold regularized networks (MRnet) that utilize a new training objective function aiming to minimize the difference between them. However, none of the existing works discuss the manifold constraint from theoretical perspective. Noted that in [28], a manifold deep learning method is carried out for set classification. However, the difference between our work and their work is clear: we provide a theoretical investigation into the structure based deep learning, whereas the work in [28] is more empirical. From application perspective, we consider the intra-class information and focus on single image based classification, while [28] is designed for set based classification by considering the inter-class information.

Metric learning / Loss function. Metric learning [21], [43], [48] usually learns a matrix for a distance metric based on the given features. Recently, some state-of-the-art image classification (face recognition) models usually adopt ideas from metric learning. These methods [17], [32], [44] use deep neural networks to automatically learn discriminative features followed by a simple distance metric such as Euclidean distance. For example, contrastive loss [4], [11] and triplet loss [40], [16], [35] are typical examples which borrow ideas from metric learning to increase the Euclidean margin for better feature embedding. Center loss [44] forces CNNs to learn centers for the features of each label and uses the learned centers to reduce intra-class variance. Compared with the Euclidean margin (contrastive loss) or intra-class variance reduction (center loss), angular softmax (A-Softmax) loss (SphereFace) [27], ring loss [52], and cosine loss [39]

implicitly involve the concept of angular margin. The angular margin is preferred because the cosine of the angle has intrinsic consistency with softmax. Our STM is different from these works: 1) the regularization item of deep features in these losses is predefined and independent of the structure of original data, while our manifold regularization item is dependent on the structure of original data; 2) the regularization item in these losses is defined on single data (image), while in our model it is defined on the distribution of all the original data, including both local information and global information.

III. Preliminaries

In this section, we first introduce the basic notations used in this paper, and then provide a formalization of the deep feature in CNNs and the algorithm for calculating the manifold structure.

A. Notation

The detailed descriptions of the notations used in this paper can be found in TABLE I. Matrices are bold upright roman letters; vectors (scalars) are bold (non-bold) lowercase letters.

TABLE I: Frequently used notations.

Notation	Description
(X_i, Y_i)	training sample, which consists of input image and output label, where $i = 1, 2, ..., N$
D_N	training set with N training sample
$\hat{Y}_i$	predicted label of X_i
F_i^l	feature of X_i at l-th layer of a L-layers CNN, where $l = 1, 2, ..., L+1$
$\hat{F}$	feature buffer of D_B for calculating manifold loss
D_B	data buffer with B input images for estimating manifold structure
$\mathcal{J}_\lambda(\cdot)$	cross-entropy loss function
$\mathcal{J}_\gamma(\cdot)$	manifold loss function
$\mathcal{J}_{\lambda,\gamma}(\cdot,\cdot)$	loss function of STM
$\mathcal{L}(\cdot,\cdot)$	loss function of label
$\Omega(\cdot)$	regularization function of weights
λ	weight decay coefficient ($\lambda \geq 0$)
γ	coefficient of manifold loss ($\gamma \geq 0$)
$\mathcal{M}$	specific manifold
$\alpha_{i,j}$	linear coefficient between sample X_i and X_j which calculated by LLE algorithm, and $\alpha_{i,j} \geq 0$
$B_{i,j}^{\mathcal{M}}$	linear coefficient between sample X_i and X_j which calculated by Laplacian algorithm, and $B_{i,j}^{\mathcal{M}} \geq 0$
k-NN(X_i)	k nearest neighbors of X_i
$\mathcal{T}_a$	time complexity of algorithm a, where $a \in \{r, lle, lap\}$
$\mathcal{S}_a$	space complexity of algorithm a, where $a \in \{r, lle, lap\}$
$E(\cdot)$	expectation operator of random variables
$\mathcal{N}(u, \sigma)$	Gaussian distribution with u mean and $sigma$ standard variation
$P(\cdot)$	probability of random events
$\lvert\cdot\rvert$	size of variables

B. The multi-layer features in CNNs

Let $D_N = \{(X_i, Y_i) | i = 1, 2, ..., N\}$ be a training dataset, where N is the total number of samples.

As shown in Fig. 1, we define $\{F_i^l | l = 1, 2, ..., L\}$ as a series of features for an image X_i with the label Y_i by a L-layers CNN,

$$X_i \longmapsto F_i^1 \longmapsto F_i^2 \longmapsto \cdots \longmapsto F_i^L \longmapsto \hat{Y}_i, \tag{1}$$

where $\hat{Y}_i$ is the predicted label of X_i. The feature transfer functions of the CNN are defined as:

$$F_i^l = f^l(F_i^{l-1}|W^l), \text{for } l = 1, 2, ..., L+1, \tag{2}$$

where $F_i^0 = X_i$; $F_i^{L+1} = \hat{Y}_i$ and W^l is the corresponding weight at l-th layer. Then, the relationship between feature F_i^l and X_i can be formulated as,

$$F_i^l = f^l(f^{l-1}(\cdots f^1(X_i|W^1)\cdots|W^{l-1})|W^l). \tag{3}$$

For the sake of simplicity, we have,

$$F_i^l = f^{[l:1]}(X_i|W^{[1:l]}), \tag{4}$$

where $l = 1, 2, ..., L+1$.

C. Algorithm for calculating manifold

In our STM, we need to know manifold $\mathcal{M}$ to calculate the manifold loss. We here first introduce the basic knowledge of LLE algorithm (or Laplacian in the next section) for estimating the manifold structure.

According to LLE, it is assumed that each data point and its neighbors lie on a locally linear patch of the manifold. Hence we compute the linear coefficients $A^{\mathcal{M}}$ to reconstruct each data from its neighbors by minimizing the reconstruction error:

$$\begin{aligned} \varepsilon(A^{\mathcal{M}}) &= \sum_{i=1}^{N} \|X_i - \sum_{X_j \in k\text{-NN}(X_i)} \alpha_{ij} X_j\|^2 \\ &= \sum_{i=1}^{N} \|X_i - X A_i^{\mathcal{M}}\|^2, \end{aligned} \tag{5}$$

which is a manifold loss. Here we define $A_i^{\mathcal{M}} = [\alpha_{i1}, \alpha_{i2}, ..., \alpha_{iN}]^T$ with α_{ij} being the corresponding weights of neighborhood data X, which is actually the feature buffer set, as shown in Fig. 2, for the i-th data X_i in the original data space. We enforce $\alpha_{ij} = 0$ if X_j does not belong to the neighborhood of X_i (i.e., $X_j \notin k$-NN(X_i)), such that each point is only reconstructed by its neighbors. The optimal weights $A^{\mathcal{M}}$ can be found by solving a least square problem with constraint $\sum_j \alpha_{ij} = 1$.

IV. Deep Structure Transfer Machine

It is reasonable to assume that the data lies on a manifold, whose intrinsic structure is expected to be embedded into the objective function of deep model. This is achieved by developing a generic representation learning method without any prior applied to the classification model. The proposed method is elaborated below, followed by the theoretical analysis.

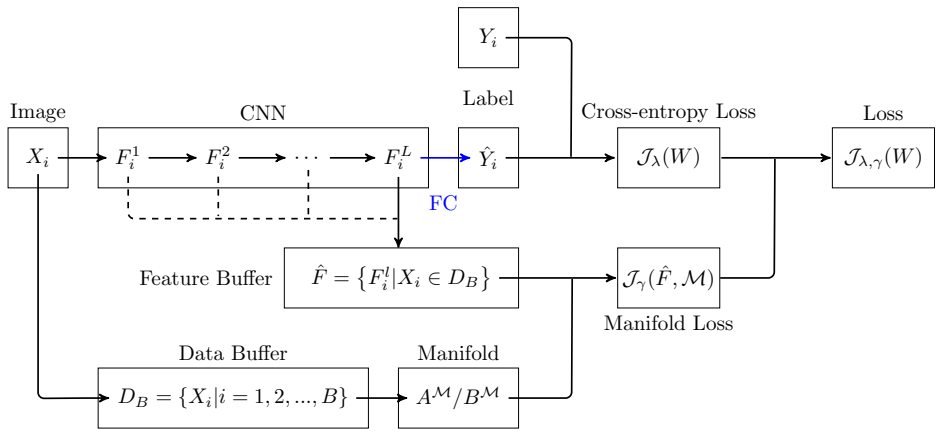


Fig. 2: The architecture of STM.

A. The cross-entropy loss

As show in Fig. 1, a L-layers CNN often defines $\{F_i^l | l = 1, 2, ..., L\}$ as a series of features for an image X_i by transfer function $f^{[l:1]}(\cdot)$ as defined in Eq. (4). The transfer function of the output layer is usually a fully connected (FC) layer followed by a softmax function (as shown in Fig. 2), and thus the conventional objective function for the network is formulated by cross-entropy function plus a regularization item:

$$\begin{aligned}\mathcal{J}_\lambda(W) &= \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}\left(Y_i, \hat{Y}_i\right) + \lambda\Omega(W) \\ &= \frac{1}{N}\sum_{i=1}^{N} -\log\frac{\exp(\theta_{Y_i}^T F_i^L)}{\sum_{j=1}^{m}\exp(\theta_j^T F_i^L)} + \lambda\Omega(W),\end{aligned} \tag{6}$$

where $\theta = W^{L+1}$ denotes the weight matrix in the last FC layer; for simplicity we only use one FC layer as an example, with θ_j as its j-th column; m is the number of classes; the scalar λ is a weight decay coefficient with $\lambda \geq 0$; and $\Omega(W)$ is a regularization function (e.g., $\Omega(W) = 1/2\|W\|^2$) of all the the weights $W = \{W^l | l = 1, 2, ..., L+1\}$.

B. Problem formulation

The conventional objective function for classification in Eq. (6) does not consider the property that the data usually lies in a specific manifold $\mathcal{M}$ [50], which reveals the nonlinear dependency of the data. Modeling this property can actually generate better solutions for lots of existing problems [28], [24]. In the deep learning approach with error propagation from the top layer, it is more favorable to impose the manifold constraint on the top layer features. Our inspiration also comes from the idea of preserving manifold structure in different spaces, i.e., the high dimensional and the low dimensional spaces. Similarly, the manifold structure of X is assumed to be preserved in the resulting deep features F^l of our model in order to reduce variation in the higher-dimensional feature space (Fig. 1). We resort to a new manifold constraint in deep learning, and achieve a new problem (P_1),

$$\begin{aligned}\mathcal{J}_\lambda(W) &= \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}\left(Y_i, \hat{Y}_i\right) + \lambda\Omega(W) \\ \text{s.t.} \quad \hat{F}_i &\in \mathcal{M}, \quad \text{for } i = 1, 2, ..., N,\end{aligned} \tag{P_1}$$

where $\hat{F}_i$ can be the deep feature of any layer, i.e., $\hat{F}_i = F_i^l$, $l \in \{1, 2, ..., L\}$. It is noticed that the objective shown in problem (P_1) is learnable if $\mathcal{M}$ is given, because $\hat{F}_i$ is directly related to the learned filters (Eq. (1) and Eq. (2)). How to solve the constraint $\hat{F}_i \in \mathcal{M}$ is elaborated in the next subsection.

C. The manifold loss

Solving the above problem needs to know manifold $\mathcal{M}$ for calculating manifold loss. We calculate it by LLE or Laplacian.

Calculating manifold loss by LLE. As assumed, a linear embedding process for neighborhood preserving in the feature space is given by:

$$\hat{F}_i = \hat{F}A_i^{\mathcal{M}}, \tag{7}$$

where $\hat{F}_i$ is the deep feature from the current layer for the i-th input sample, and the feature of its neighbors or feature buffer (as shown in Fig. 2) are denoted by $\hat{F}$. In this process, the feature of each sample is linearly reconstructed from $\hat{F}$ by linear coefficients. The reconstruction weight $A^{\mathcal{M}}$ is obtained by minimizing Eq. (5), which characterizes intrinsic geometric properties of the data that are invariant to rotations, rescalings, and translations of that data point and its neighbors [33], is related to the manifold $\mathcal{M}$. This is the key part of the proposed algorithm where the constraint manifold $\mathcal{M}$ arises. As assumed, replacing $\mathcal{M}$ equals incorporating Eq. (7) into our objective. This is the modularity alluded previously. Based on the Lagrangian multiplier method, Eq. (7) is introduced to solve problem (P_1) by a new objective as:

$$\begin{aligned}\mathcal{J}_{\lambda,\gamma}(W) &= \mathcal{J}_\lambda(W) + \mathcal{J}_\gamma(\hat{F}, \mathcal{M}) \\ &= \mathcal{J}_\lambda(W) + \frac{\gamma}{2N}\sum_{i=1}^{N}\|\hat{F}_i - \hat{F}A_i^{\mathcal{M}}\|^2,\end{aligned} \tag{8}$$

where the scalar γ is adopted to balance these two terms of the objective. See **Algorithm 3** in Appendix A-B for the details of calculating the manifold loss by LLE.

Calculating manifold loss by Laplacian. Similar to LLE, we can also exploit the Laplacian manifold [1] to our problem. As shown in [1], we can deduce such a manifold loss:

$$\mathcal{J}_\gamma(\hat{F}, \mathcal{M}) = \sum_{i=1}^{N}\sum_{j=1}^{N} \|\hat{F}_i - \hat{F}_j\|^2 B_{ij}^{\mathcal{M}}, \tag{9}$$

where $B_{ij}^{\mathcal{M}}$ is defined to be exponential distance between the i-th and j-th sample in the input space [1], which is actually used to normalize the feature buffer set, as shown in Fig. 2, to improve the efficiency. Similarly, we obtain the following objective as:

$$\begin{aligned}\mathcal{J}_{\lambda,\gamma}(W) &= \mathcal{J}_\lambda(W) + \mathcal{J}_\gamma(\hat{F}, \mathcal{M}) \\ &= \mathcal{J}_\lambda(W) + \frac{\gamma}{2N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} \|\hat{F}_i - \hat{F}_j\|^2 B_{ij}^{\mathcal{M}}.\end{aligned} \tag{10}$$

Algorithm 4 in Appendix A-C shows the details of calculating the manifold loss by Laplacian.

D. The training algorithm of STM

Now, we have the training algorithm of STM to solve problem (P_1), which is summarized in **Algorithm 1**. Regarding the convergence of the proposed algorithm, our learning procedures never hurt the convergence of the back propagation, because newly added variables related to the manifold loss (convex) are solved following the similar pipeline. As shown in **Algorithm 1**, the procedures for training STM are as follows.

The input of the training algorithm includes a set of image-label pairs $D_N = \{(X_i, Y_i)|i = 1, 2, ..., N\}$, the number of mini-batch size N_b, the learning rate μ, the size of the feature buffer size k_0 as shown in Fig. 2, the number of neighbors in LLE k_l, the weight decay parameter λ, the manifold loss balance parameter γ, the indicator $m \in \{lle, lap\}$ for denoting which manifold method is used in training, the maximum number of iterations $\tau_{\max}$, and the minimum value of the objection function $\mathcal{J}_{\min}$. The output of this algorithm is the trainable parameter W.

Step 1. Initialization, line 1. Initialize t with 0, D_B with ϕ, $\hat{F}$ with ϕ and W with random values [9], [12], [18], where ϕ means empty set.

Step 2. Picking up mini-batch samples, line 3. Randomly pick N_b input-output pairs (X_i, Y_i) as a mini-batch sample set D_{N_b}. The time complexity and space complexity of this step are both equal to the sample complexity, i.e., $O(N_b \times |D|)$, where $|D| = |X| + |Y|$ denotes the capacity of one input-output pair.

Step 3. Forward-propagation (i.e., calculating features and labels for the mini-batch samples), line 4-8. The space complexity of this step is $O(1)$, and the time complexity of this step is $O(N_b \times |W|)$.

Step 4. Refreshing the data buffer and feature buffer, line 9. The details of function $refresh(\cdot)$ and its corresponding complexities are shown in **Algorithm 2** in Appendix A-A.

Algorithm 1: Training algorithm of STM.

Input: D_N, N_b, μ, k_0, k_l, λ, γ, m, $\tau_{\max}$, $\mathcal{J}_{\min}$;
Output: W;
1 Initialize t, D_B, $\hat{F}$ and W with default or random values;
2 **repeat**
3 $\quad D_{N_b} \sim \mathcal{G}(D_N)$;
4 $\quad$ **for** $X_i \in D_{N_b}$ **do**
5 $\quad\quad$ **for** $l \in \{1, 2, ..., L+1\}$ **do**
6 $\quad\quad\quad F_i^l \leftarrow f^{[l:1]}(X_i|W^{[1:l]})$;
7 $\quad\quad$ **end**
8 $\quad$ **end**
9 $\quad \{D_B, \hat{F}\} \leftarrow$ *refresh*$(D_B, \hat{F}, D_{N_b}, \{F_i^L|X_i \in D_{N_b}\}, \gamma, k_l, m)$;
10 $\quad \mathcal{J}_\lambda(W) \leftarrow \frac{1}{N_b}\sum_{i=1}^{N_b} \mathcal{L}\left(Y_i, \hat{Y}_i\right) + \lambda\Omega(W)$;
11 $\quad \mathcal{J}_\gamma(\hat{F}, \mathcal{M}) \leftarrow m(D_B, \hat{F}, \gamma, k_l)$;
12 $\quad \mathcal{J}_{\lambda,\gamma}(W) \leftarrow \mathcal{J}_\lambda(W) + \mathcal{J}_\gamma(\hat{F}, \mathcal{M})$;
13 $\quad \nabla_W \leftarrow \frac{\partial \mathcal{J}_\lambda(W)}{\partial W} + \frac{\partial \hat{F}}{\partial W}\frac{\partial \mathcal{J}_\gamma(\hat{F}, \mathcal{M})}{\partial \hat{F}}$;
14 $\quad W \leftarrow W - \mu\nabla_W$;
15 $\quad t \leftarrow t + 1$;
16 **until** $t > \tau_{\max} \vee \mathcal{J}_{\lambda,\gamma}(W) < \mathcal{J}_{\min}$;
17 **return** W;

We denote the time complexity and space complexity of $refresh(\cdot)$ by $\mathcal{T}_r$ and $\mathcal{S}_r$.

Step 5. Calculation of the objective function, line 10-12. The total value of the objective function (line 12) consists of two items: ordinary loss (line 10) and manifold loss (line 11). The time complexity of this step is $O(|W|) + \mathcal{T}_{lle}$ if we using LLE manifold, or $O(|W|)+\mathcal{T}_{lap}$ if Laplacian manifold is used; and the space complexity is $\mathcal{S}_{lle}$ or $\mathcal{S}_{lap}$. The $\mathcal{T}_{lle}$, $\mathcal{S}_{lle}$, $\mathcal{T}_{lap}$ and $\mathcal{S}_{lap}$ are used to denote the time complexity, space complexity of function $lle(\cdot)$ and function $lap(\cdot)$. The details for calculating these functions are shown in **Algorithm 3** in Appendix A-B, and in **Algorithm 4** in Appendix A-C.

Step 6. Back-propagation (i.e., calculating the gradient), line 13. Calculate the gradient ∇_W with back-propagation. The space complexity and time complexity of this step are $O(1)$ and $O(|W|)$, respectively.

Step 7. Updating the trainable parameters, line 14. The time complexity of this step is $O(|W|)$; and the space complexity of this step is $O(1)$.

Step 8. Not reaching the end condition, line 15-16. Increase the number of iterations $t \leftarrow t + 1$. If $t \leq \tau_{\max}$ and $\mathcal{J}_{\lambda,\gamma}(W) \geq \mathcal{J}_{\min}$, go to **Step 2**.

Step 9. Return the output, line 17. Return the trained parameters W.

In summary, the time complexity of the training algorithm of STM in each mini-batch training is

$$\mathcal{T} = O(N_b \times (|W| + |D|) + 3|W|) + \mathcal{T}_r + \mathcal{T}_m, \tag{11}$$

and the space complexity in each mini-batch training is

$$\mathcal{S} = O(N_b \times (|W| + |D|)) + \mathcal{S}_r + \mathcal{S}_m, \tag{12}$$

where $m \in \{lle, lap\}$ is used for denoting which manifold loss function is used. From Eq. (11) and Eq. (12), we know that the last two items are the additional complexities which come from two parts: the function of refreshing the data buffer and features buffer, and the function of calculating manifold loss.

E. Theoretical analysis

In this section, we theoretically show that the manifold loss can lead a convergence process to the expectation of data representation, based on assumption that data lies on a manifold. More specifically, **Theorem 1** provides a foundation of feature learning that the expected value can be achieved, if manifold structure is transferred from the input space to the feature space. Such a proof is very useful to guide the feature design in various practical applications. Notably, many machine learning tasks often require that features are compact and stable during the model learning process. In other words, the variances among the data are mitigated in the learning process, as they are converging into a single expected value. In the following, we will address how our theorem can be involved in the learning stage.

Definition 1: For $x_1, x_2, ..., x_n$, we define:

$$\|x_{i-1} - x_i\| \leq c_{i-1}, \tag{13}$$

where x_i is a random scalar variable and $c = (c_0, c_1, ..., c_{n-1})$. Further we have $x_i = x_{i-1} + c_{i-1}$, if define

$$c_{i-1} \sim \mathcal{N}(0, \sigma^2),$$

where $\mathcal{N}$ is a Gaussian distribution with 0 mean and σ standard variation.

From the above definition, we know that x_i and x_{i-1} are bounded by c_{i-1}. The expectation of x_i given x_{i-1} is chosen to be x_{i-1}, which means that x_i is sampling from a distribution that is quite related to x_{i-1}. That is to say, x_i is chosen around x_{i-1} (mean).

Lemma 1: If $x_1, x_2, ..., x_n$ satisfies **Definition 1**, then:

$$P(|\bar{x} - E(\bar{x})| \geq \frac{\lambda}{n}) \leq 2\exp(\frac{-\lambda^2}{2\sum_{i=1}^{n} c_i^2}), \tag{14}$$

where $\bar{x}$ is the average of $x_1, x_2, ..., x_n$.

Proof: The details are shown in Appendix B. ■

Before proving the theorem, we first introduce the following two propositions.

Proposition 1: The most popular approaches, such as LLE [33] and ISOMAP [37], are with the underlying idea that a high dimensional vector representing the data that can be mapped into a lower dimension space preserving, as much as possible, the metric of the original space. The distances of all the pairs of data points in the embedding space is bounded [3], [26]. Thus, it is claimed that:

$$\|F_i(j) - F_{i-1}(j)\| \leq \|F_i - F_{i-1}\| \leq L\|X_{i-1} - X_i\|, \tag{15}$$

where $F_i = f(X_i)$, and $f(\cdot)$ denotes the projection from the original sample X_i to F_i. And $F_i(j)$ denotes the j-th dimension of F_i in the manifold feature space, i.e., the deep feature space obtained based on manifold loss in this work; L is a constant. Due to $\|X_i - X_{i-1}\|$ is controlled by the input sample, so that it is reasonable to claim that F_i:

$$\|F_i(j) - F_{i-1}(j)\| \leq \boldsymbol{c}_{i-1}(j), \tag{16}$$

where, $\boldsymbol{c}_i(j)$ is the j-th dimension of vector $\boldsymbol{c}_i$. Our theoretical result is general, which is however founded on Eq. (15). That is to say, it is satisfied only when the distance between two samples in the feature space is bounded to that in the input space, which is guaranteed by manifold.

Proposition 2: For any vector $v = [v_1, v_2, ..., v_n]$, we have:

$$\begin{aligned} P(\sum_{i=1}^{n} |v_i| \geq \sum_{i=1} \lambda_i) &\leq P(\bigcup_{i=1}^{n} |v_i| \geq \lambda_i) \\ &\leq \sum_{i}^{n} P(|v_i| \geq \lambda_i). \end{aligned} \tag{17}$$

Now, we have the following theorem.

Theorem 1: If vectors $F_1, F_2, ..., F_n \in \mathcal{M}$, then:

$$P(|\bar{F} - E(\bar{F})| \geq \sum_{i} \lambda_i) \leq C, \tag{18}$$

where $\bar{F}$ is the average vector; $\lambda = [\lambda_1, \lambda_2, ..., \lambda_n]$ with $\lambda_i \leq 1$ and C is a constant.

Proof: **Theorem 1** means that the expectation of $\bar{F}$ is achieved in a probabilistic way. According to Eq. (14) in **Lemma 1** and Eq. (16) in **Proposition 1**, we have:

$$P(|Z(j)| \geq \lambda_j) \leq 2\exp(\frac{-\lambda_j^2}{2\sum_{i=1}^{n} \boldsymbol{c}_{i-1}^2(j)}), \tag{19}$$

where Z is defined as:

$$Z = \bar{F} - E(\bar{F}). \tag{20}$$

We set $a_j = 2\exp(-\lambda_j^2/(2\sum_{i=1}^{n} \boldsymbol{c}_{i-1}^2(j))$, based on Eq. (17) in **Proposition 2**, we have:

$$P(|\bar{F} - E(\bar{F})| \geq \sum_{j=1}^{n} \lambda_j) \leq \sum_{j=1}^{n} a_j = C. \tag{21}$$

Thus, **Theorem 1** is proved. ■

V. Experiment

In this section, we first present the details about how to implement our method with a deep learning pipeline. We then use the digit recognition (MNIST) and the natural object recognition (CIFAR) experiments to show the superiority of our method. We finally validate the effectiveness of our method with large-scale visual tasks including image classification and object tracking.

A. Implementation details

Comparison. We validate our method on various CNN base models, including ResNet [13], WideResNet [49], and then compare the performance with state-of-the-art networks. Center loss [44], A-Softmax loss (SphereFace) [27], ring loss [52], and cosine loss [39] are also evaluated equally as comparison. For the unavailability of the training face database used in [44], we choose other testbeds, such as MNIST, CIFAR,

ImageNet and the large scale OTB-50 tracking database for a fair comparison.

Manifold. We introduce manifold loss or the constraint term for structure preserving based on LLE or Laplacian. We build a feature buffer (Fig. 2) consisting of its k_0 (e.g., 30) previous samples from the same class, which denotes the maximum number of nearest neighbors used to calculate the reconstruction weights ($A^{\mathcal{M}}$, $B^{\mathcal{M}}$) exactly as that in LLE or Laplacian manifold. Notice that we then obtain the feature from this sample mapped by the network in each iteration and its corresponding subset of features from its neighbor samples, and learn the model by the proposed loss function as the reconstruction weights ($A^{\mathcal{M}}$, $B^{\mathcal{M}}$) are introduced. The above process is used for the MNIST, CIFAR and ImageNet datasets, but in the tracking task we divide each sequence into batch sets, which are then used to calculate the manifold for further learning process.

Settings in CNN. The proposed models are implemented on common libraries (i.e., Caffe, TensorFlow and PyTorch) with our modifications, and can still be trained end-to-end by SGD without introducing many parameters compared to their base model. We train our STMs via the algorithm shown in section IV-D. The manifold loss is added before the FC layer as shown in Fig. 2. To further understand how the elements of the framework affect the performance, we test our models when the manifold data structure is extracted by different techniques, such as LLE or Laplacian. For fair comparison, most of settings in our networks follow the ones in their base model, except for the learning rate and policy, because the modified object function is determined on validation set. More detailed settings in each experiment are described in corresponding subsection.

B. Digit recognition

MNIST dataset of handwritten digits[1] contains a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.

We use a weight decay (λ) of 0.0001 and momentum of 0.9 in our model with a mini-batch size (N_b) of 500. The learning rate (μ) is started from 0.1, and divided by 10 at 32k and 48k iterations, and the training procedure is terminated at 64k iterations. We do not conduct any data augmentation for training. The LeNet++ [44] architecture is used for base-CNN, our STM, center loss [44], A-Softmax loss [27], ring loss [52], and cosine loss [39] for a fair comparison.

(1) Parameter evaluation and performance comparison. There are several parameters affecting the performance of the proposed method, i.e., k_0 denoting the size of the feature buffer set for each class, k_l denoting number of neighbors in LLE or Laplacian. The results in TABLE II show that STM with LLE achieves the best performance when $k_l = 14$. The performances of LLE and Laplacian based STMs are very similar, but STM with LLE needs more computation as shown in Algorithm 3, in comparison with STM with Laplacian (Algorithm 4). For all the following experiments the buffer size and neighbor size are set as, $k_0 = 30$, $k_l = 14$. The parameter γ is also evaluated in TABLE II, which show that in the certain scope the parameter affect little on the final performance.

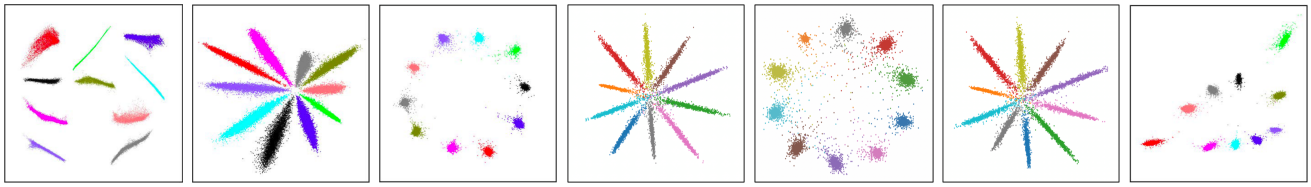
[1]http://yann.lecun.com/exdb/mnist/

TABLE II: Comparisons with different hyper-parameters of STM on the MNIST dataset.

Models	**Hyper-parameter setting**			**Error (%)**
	k_l	k_0	γ	
STM with LLE	14	30	0.000	0.73
	14	30	0.001	0.65
	14	30	0.010	0.45
	14	30	0.100	0.40
	14	30	0.200	0.37
	14	30	0.300	**0.36**
	14	30	0.400	**0.36**
	14	30	0.500	0.37
	14	30	0.800	0.37
	14	30	0.900	0.38
	14	30	1.000	0.38
	22	30	0.000	0.73
	22	30	0.100	0.45
	22	30	0.200	0.42
	22	30	0.300	0.41
	22	30	0.800	0.42
	22	30	1.000	0.42
STM with Laplacian	14	30	0.000	0.73
	14	30	0.001	0.60
	14	30	0.010	0.41
	14	30	0.100	0.38
	14	30	0.200	**0.36**
	14	30	0.300	**0.36**
	14	30	0.400	**0.36**
	14	30	0.800	0.38
	14	30	1.000	0.38
	22	20	0.200	0.38
	22	20	0.300	0.37
	22	20	0.400	0.38

(2) Illustration. We first conduct experiments to illustrate how the STM method influences the distribution in Fig. 3. Without the special note, STM means using Laplacian to calculate $B^{\mathcal{M}}$ from the original data space for learning. Fig. 3 shows that the distributions of STM deep features (g) appear to be simpler (parsimony) than the original one (a) because of its approaching to the expectation. This is even more profound in the sense that the compactness did not conflict with the structure preservation, which can be viewed that STM obtains a more similar structure as that of the original manifold than the center loss [44]. A-Softmax loss [27] feature in sub-figure (d) and cosine loss [39] feature in sub-figure (f) have a similar distribution like the baseline CNN feature in sub-figure (a). In addition, the center loss in sub-figure (c) or ring loss [52] in sub-figure (e) appears more scattered, meaning our structure preservation is a better strategy to achieve a good representation. In a summary, manifold reveals the nonlinear structure, which is more robust to accommodate variations in the input data, especially compared to the center loss (Gaussian) in Fig. 3.

In TABLE III, we report the error rates obtained by six different approaches. It can be seen that ours is far lower than the existing approaches including center loss, A-Softmax loss, ring loss and cosine loss, indicating that the manifold loss term indeed increases the discriminative power of the deeply learned features. In addition, it seems that STM with LLE



(a) (b) (c) (d) (e) (f) (g)

Fig. 3: Distribution illustration of learned features, where (a) is the manifold, and we create each structure separately but show in one figure; (b) is baseline CNN feature; (c) is center loss [44] feature; (d) is A-Softmax loss (SphereFace) [27] feature; (e) is ring loss [52] feature; (f) is cosine loss [39] feature; and (g) is the STM feature.

TABLE III: Comparisons with CNNs on the MNIST dataset.

Models	Results (error rate (%))
Base-CNN (LeNet++ [44])	0.73
STN(affine) [20]	0.61
Center loss [44]	0.61
A-Softmax loss [27]	0.49
Ring loss [52]	0.46
Cosine loss [39]	0.45
STM with LLE (ours)	0.36
STM with Laplacian (ours)	0.36

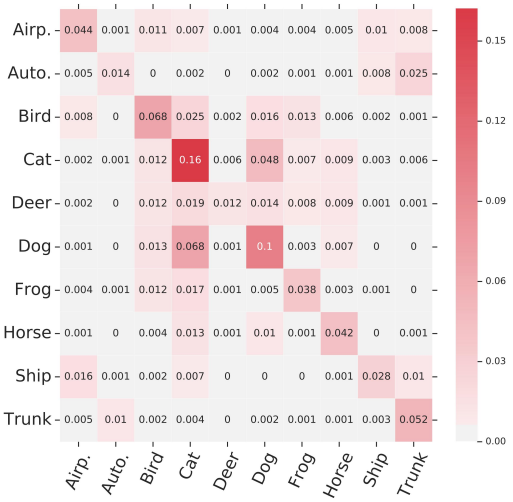
performs slightly better depending on parameter selections than STM with Laplacian. Moreover, the weight calculation for Laplacian is much easier than that of LLE so that we change the notation of STM with Laplacian to STM and use it in the following experiments.

C. Natural object recognition

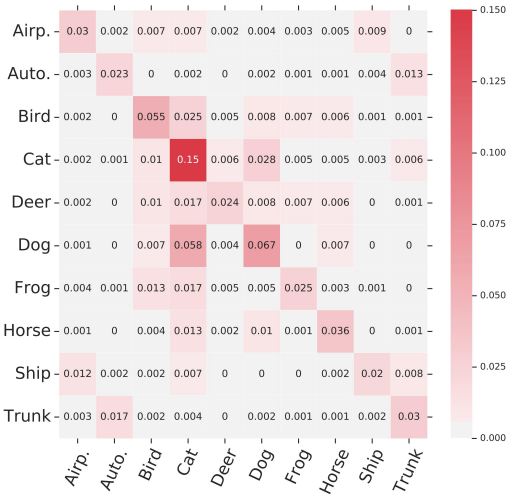
CIFAR [22] dataset is a famous natural image classification benchmark which consists of 60000 32x32 color images in 10 or 100 classes, with 6000 images per class. There are 50000 training images and 10000 test images. We follow the same protocol as that of [49]. Seven CNNs including VGG [36], ResNet [13], WideResNet [49] (or baseline CNN), center loss [44], A-Softmax loss [27], ring loss [52], and cosine loss [39] are used as baselines on these datasets.

We use a weight decay (λ) of 0.0001 and momentum of 0.9. These models are trained on two GPUs (Titan XP) with a mini-batch size (N_b) of 128. The learning rate (μ) is started from 0.1, and divided by 10 at 32k and 48k iterations, and the training procedure is terminated at 64k iterations, which is determined on a 45k/5k train/val split. We follow the same data augmentation in [13] for training: horizontal flipping is adopted, and a 32×32 crop is sampled randomly from the image padded by 4 pixels on each side. For testing, we only evaluate the single view of the original 32×32 image.

Our algorithm is also compared with the state-of-the-art algorithms when carrying out the task of image classification. To be fair, the settings for all the algorithms follow WideResNet [49], which was implemented by us. The results in TABLE IV again show that STM significantly improves the baselines (e.g., WideResNet) on both CIFAR10 and CIFAR100 datasets. In Fig. 4, we notice that the top-2 classes of being improved in CIFAR10 are dog (34% higher than baseline



(a) WideResNet (baseline).



(b) STM (ours).

Fig. 4: Error distribution analyzing on the CIFAR10 dataset.

WideResNet[2]), and horse (14%), in which significant image variations take place. This implies considering the manifold structure in feature learning enhances the capability of handling image variations. In addition, the center loss method (or other three loss methods) performs worse than STM due to

[2]Our implementation in Tensorflow.

TABLE IV: Comparisons with CNNs on the CIFAR dataset.

Models	**Results on (error rate (%))**	
	CIFAR-10	**CIFAR-100**
VGG [36]	6.32	28.49
ResNet [13]	6.43	25.16
WideResNet [49]	5.61	22.07
Center loss [44]	5.58	22.08
A-Softmax loss [27]	5.56	22.07
Ring loss [52]	5.54	22.01
Cosine loss [39]	5.30	21.62
STM (ours)	4.60	20.2

severe variations in the CIFAR datasets.

D. Large size image classification

The previous experiments are conducted on datasets with small size images. To further show the effectiveness of the proposed STM method, we evaluate it on the ImageNet [6] dataset. Different from MNIST and CIFAR, ImageNet consists of images with a much higher resolution. In addition, the images usually contain more than one attribute per image, which may have a large impact on the classification accuracy. In this experiment, we firstly choose a 100-class ImageNet 2012 [6] subset for reducing the time for the training a deep model. The 100 classes are selected from the full ImageNet dataset at a step of 10. Similar subset is also applied in [47]. In order to make a more general validation of effectiveness of our STM on large-sized images, we also take the full ImageNet dataset for another test.

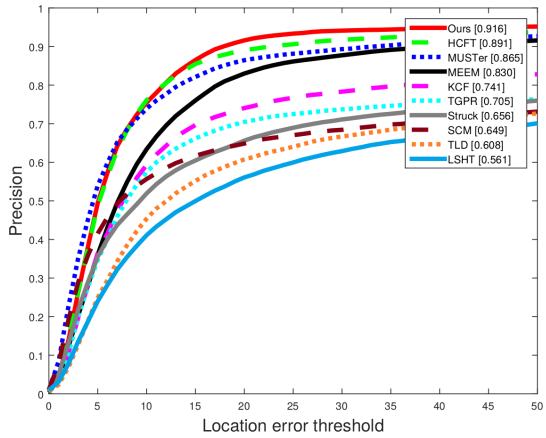
TABLE V: Comparisons with CNNs on the ImageNet dataset.

Models	**Results on (error rate (%))**			
	ImageNet-100		**ImageNet-Full**	
	Top-1	Top-5	Top-1	Top-5
ResNet (ResNet-101 [13])	11.94	3.16	22.44	6.21
Center loss [44]	11.92	3.15	22.31	6.18
A-Softmax loss [27]	11.28	3.12	22.15	6.16
Ring loss [52]	11.13	3.10	22.10	6.09
Cosine loss [39]	11.28	3.09	22.14	6.15
STM (ours)	10.67	2.94	21.56	5.77

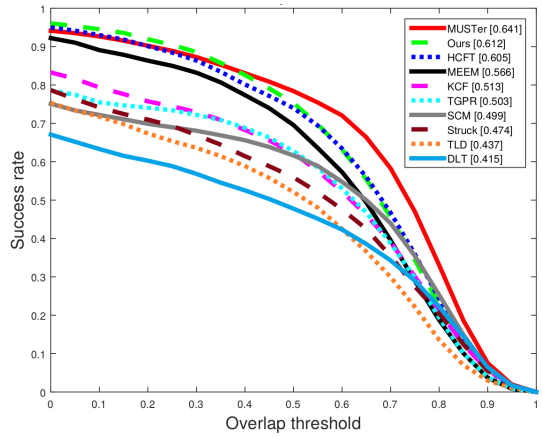
For the ImageNet-100 and ImageNet-Full experiment, we use the same model as the baseline ResNet (ResNet-101 [13]) model, and the setting is the same as the previous experiments. Both methods are trained after 120 epochs. The learning rate (μ) is initialized as 0.1 and decreases to 1/10 times per 30 epochs. Top-1 and Top-5 errors are used as evaluation metrics. The test errors are shown in TABLE V. Compared to the ResNet baseline, our STM achieves a better classification performances (i.e., Top-5 error: 2.94% vs. 3.16% for ImageNet-100, 5.77% vs. 6.21% for ImageNet-Full, Top-1 error: 10.67% vs. 11.94% for ImageNet-100, 21.56% vs. 22.44% for ImageNet-Full) with almost the same parameters (44.54M). With respect to other baselines, such as center loss, A-Softmax loss, ring loss, and cosine loss, STM also achieves remarkable improvements. Considering the large variations in ImageNet, STM can still achieve a better performance than ResNet, and we believe that manifold loss is really effective.

E. Object tracking

In this section, we evaluate the performance of STM on the tracking problem based on 50 sequences from the commonly used tracking *object tracking benchmark* (OTB) dataset [46].



(a) Precision plots.



(b) Success plots.

Fig. 5: Precision and success plots on the OTB dataset.

OTB [46] is a large dataset with ground-truth object positions and extents for tracking and introduces the sequence attributes for the performance analysis. They integrate most of the publicly available trackers into one code library with the uniform input and output formats to facilitate large-scale performance evaluation. The performances of most tracking algorithms are included on 50 sequences with different initialization settings. In this tracking benchmark [46], each sequence is manually tagged with different attributes, such as *illumination variations*, *scale variations*, *occlusions*, *deformations*, *motion blur*, *abrupt motion*, *in-plane rotation*, *out-of-plane rotation*, *out-of-view*, *background clutters* and *low resolution*, indicating what kind of challenges exist in the video.

We implement our STM model based on VGG-19 [36] with two outputs for each sequence separately. We randomly collect 50 positive and 200 negative samples for each frame from VOT13, VOT14, and VOT15[3], where the positive and negative examples have 0.7 and 0.5 IoU overlap ratios with ground-

[3]http://www.votchallenge.net/

truth bounding boxes, respectively. Noted that we remove the overlapped sequences with OTB from the trained databases. In the tracking, we used the same strategy as that of [29], [30], which learns a discriminative classifier and estimates the translation of target objects by searching for the maximum value of correlation response map. Similar to KCF [14], using the set of correlation response maps based on deep features can hierarchically infer the target translation at each layer, i.e., the location of the maximum value in the last layer is used as a regularization to search for the maximum value of the earlier layer. Our STM tracker can be generated by simply replacing the deep model of [29], [30]. Regarding the comparison, our baseline algorithms mainly consist of correlation filters or deep learning based trackers, such as KCF, FCNT, and Cf+CNN [30].

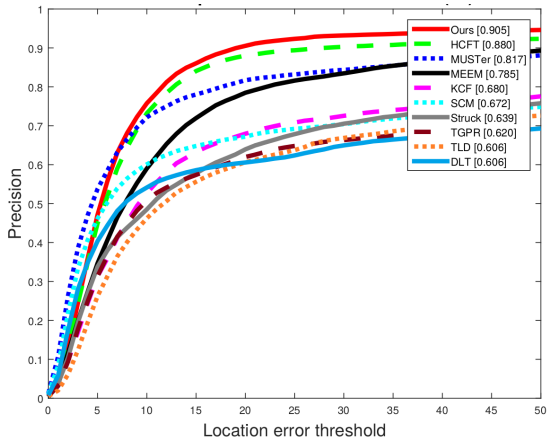
In Fig. 5, we plot the precision against location error curve, which measures the ratio of successful tracking frames when the threshold of allowed location errors is changed. Here, the location error (x-axis, in pixel) on the plot implies the distance between the bounding box center and the ground-truth. For ease of comparison, we also include the plots of several baseline trackers in the figure.

TABLE VI: Comparisons with state-of-the-art trackers on the OTB dataset.

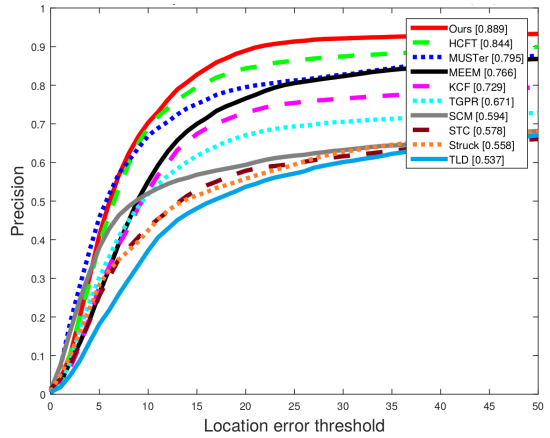
Models	**Results**	
	Precision (%)	**Success rate (%)**
FCNT [41]	85.7	47.2
KCF [14]	74.1	51.3
Cf+CNN [30]	90.7	61.1
HCFT [29]	89.1	60.5
MEEM [51]	83.0	56.6
DSST [5]	73.9	50.5
STM (ours)	91.6	61.2

As can be seen in TABLE VI, the STM and KCF achieve 61.2% and 51.3% based on the average success rate, while HCFT and MEEM trackers respectively achieve 60.5% and 56.6%. In terms of precision, STM and KCF respectively achieve 91.6% and 74.1% when the threshold is set to 20. Moreover, the STM and baseline HCFT obtain 91.6% and 89.1% respectively, which further confirms that the proposed deep model is effective on object tracking. We also compare with cf+CNN, one of the latest variants of KCF, and the results show that STM still achieves performance improvement in terms of precision. It is believed that the special strategy used in cf+CNN can also be used to further improve STM. All the above observations clearly demonstrate that imposing the manifold prior constraint during the feature learning helps generate more robust features for tracking, thus enabling its superiority over the state-of-the-art trackers.

Here, we also show the scale variation and lighting attributes in Fig. 6, where STM performs much better than other trackers again. The reason for this phenomenon is that the manifold structure of the learned features is data-dependent. This property leads the learned features can handle nonlinear of variations in object tracking. Again, STM shows its super capability of handling severe variations. The experimental results for full set of plots generated by the benchmark toolbox are reported in Fig. 7 and Fig. 8 in Appendix C.



(a) Precision plots for scale variation.



(b) Precision plots for illumination variation.

Fig. 6: Precision plots for two attributes (scale variation and illumination variation) on the OTB dataset.

VI. Conclusion

In this paper, we have presented a new concept for representation learning that data structure preservation can help feature learning process converge at the representation expectation. Thus, we open up a possible way to learn deep features which are robust to the variations of the input data because of theoretical convergence. The proposed STM method formulates the data structure preserving into an objective function optimization problem with constraint, which can be solved via the BP algorithm. Extensive experiments and comparisons on the commonly used benchmarks show that the proposed method significantly improved the CNN performance, and achieved a better performance than the state-of-the-arts.

References

[1] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[2] A. Birnbaum. A unified theory of estimation. *Annals of Mathematical Statistics*, 32(1):112–135, 1961.

[3] J. Bourgain. On lipschitz embedding of finite metric spaces in hilbert space. *Israel Journal of Mathematics*, 52(1-2):46–52, 1985.

[4] S. Chopra, R. Hadsell, Y. LeCun, et al. Learning a similarity metric discriminatively, with application to face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 539–546, 2005.

[5] M. Danelljan, G. Häger, F. Khan, and M. Felsberg. Accurate scale estimation for robust visual tracking. In *British Machine Vision Conference, Nottingham, September 1-5, 2014*. BMVA Press, 2014.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[7] S. Deutsch, S. Kolouri, K. Kim, Y. Owechko, and S. Soatto. Zero shot learning via multi-scale manifold regularization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7112–7119, 2017.

[8] P. Etyngier, F. Segonne, and R. Keriven. Shape priors using manifold learning techniques. In *Proceedings of the IEEE international conference on computer vision*, pages 1–8, 2007.

[9] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In Y. W. Teh and M. Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[10] R. L. Gregory. *Concepts and mechanisms of perception*. Charles Scribner's Sons, 1974.

[11] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, volume 2, pages 1735–1742, 2006.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[14] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, 2015.

[15] R. Hettiarachchi and J. F. Peters. Multi-manifold lle learning in pattern recognition. *Pattern Recognition*, 48(9):2947–2960, 2015.

[16] E. Hoffer and N. Ailon. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*, pages 84–92. Springer, 2015.

[17] J. Hu, J. Lu, and Y.-P. Tan. Discriminative deep metric learning for face verification in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1875–1882, 2014.

[18] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR.

[19] A. Iscen, G. Tolias, Y. Avrithis, T. Furon, and O. Chum. Efficient diffusion on region manifolds: Recovering small objects with compact cnn representations. In *Proceedings of the IEEE international conference on computer vision*, pages 2077–2086, 2017.

[20] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. pages 2017–2025, 2015.

[21] M. Koestinger, M. Hirzer, P. Wohlhart, P. M. Roth, and H. Bischof. Large scale metric learning from equivalence constraints. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2288–2295, 2012.

[22] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[23] Q. V. Le, W. Y. Zou, S. Y. Yeung, and A. Y. Ng. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3361–3368, 2011.

[24] T. Lee, M. Choi, and S. Yoon. Manifold regularized deep neural networks using adversarial examples. *arXiv preprint arXiv:1511.06381*, 2015.

[25] E. L. Lehmann. A general concept of unbiasedness. *Annals of Mathematical Statistics*, 22(4):587–592, 1951.

[26] N. Linial, E. London, and Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 1995.

[27] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song. Sphereface: Deep hypersphere embedding for face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 212–220, 2017.

[28] J. Lu, G. Wang, W. Deng, P. Moulin, and J. Zhou. Multi-manifold deep metric learning for image set classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1137–1145, 2015.

[29] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang. Hierarchical convolutional features for visual tracking. pages 3074–3082, 2015.

[30] C. Ma, Y. Xu, B. Ni, and X. Yang. When correlation filters meet convolutional neural networks for visual tracking. *IEEE Signal Processing Letters*, 23(10):1454–1458, 2016.

[31] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5115–5124, 2017.

[32] H. Oh Song, Y. Xiang, S. Jegelka, and S. Savarese. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4004–4012, 2016.

[33] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[34] L. K. Saul and S. T. Roweis. Think globally, fit locally: unsupervised learning of low dimensional manifolds. *Journal of Machine Learning Research*, 4(Jun):119–155, 2003.

[35] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[36] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. 2015.

[37] J. B. Tenenbaum, V. De Silva, and J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[38] M. Vestner, R. Litman, E. Rodolà, A. Bronstein, and D. Cremers. Product manifold filter: Non-rigid shape correspondence via kernel density estimation in the product space. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3327–3336, 2017.

[39] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, and W. Liu. Cosface: Large margin cosine loss for deep face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5265–5274, 2018.

[40] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu. Learning fine-grained image similarity with deep ranking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1386–1393, 2014.

[41] L. Wang, W. Ouyang, X. Wang, and H. Lu. Visual tracking with fully convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 3119–3127, 2015.

[42] Q. Wang, J. Gao, and H. Li. Grassmannian manifold optimization assisted sparse spectral clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5258–5266, 2017.

[43] K. Q. Weinberger and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(Feb):207–244, 2009.

[44] Y. Wen, K. Zhang, Z. Li, and Y. Qiao. A discriminative feature learning approach for deep face recognition. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *Computer Vision – ECCV 2016*, pages 499–515, Cham, 2016. Springer International Publishing.

[45] J. Wright, A. Y. Yang, A. Ganesh, S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):210–227, 2009.

[46] Y. Wu, J. Lim, and M. Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1834–1848, 2015.

[47] L. Yao and J. Miller. Tiny imagenet classification with convolutional neural networks. *CS 231N*, 2(5):8, 2015.

[48] Y. Ying and P. Li. Distance metric learning with eigenvalue optimization. *Journal of Machine Learning Research*, 13(Jan):1–26, 2012.

[49] S. Zagoruyko and N. Komodakis. Wide residual networks. 2016.

[50] B. Zhang, A. Perina, V. Murino, and A. D. Bue. Sparse representation classification with manifold constraints transfer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4557–4565, 2015.

[51] J. Zhang, S. Ma, and S. Sclaroff. Meem: robust tracking via multiple experts using entropy minimization. In *European conference on computer vision*, pages 188–203, 2014.

[52] Y. Zheng, D. K. Pal, and M. Savvides. Ring loss: Convex feature normalization for face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5089–5097, 2018.

APPENDIX A
ALGORITHMS

A. Algorithm of refreshing data buffer and feature buffer

Algorithm 2 shows the algorithm of refreshing data buffer and feature buffer. The input of this algorithm includes the old data buffer D'_B, the old feature buffer $\hat{F}'$, the candidate data buffer D_{N_b}, the candidate feature buffer F, the manifold loss balance parameter γ, the number of neighbors k_l, the manifold loss balance parameter γ, the indicator $m \in \{lle, lap\}$ for denoting which manifold method is used in training. The output of this algorithm is the refreshed data buffer D_B and the refreshed feature buffer $\hat{F}$.

Algorithm 2: Refreshing data and feature buffer, denoted as $\{D_B, \hat{F}_B\} = \textit{refresh}(D'_B, \hat{F}'_B, D_{N_b}, F_{N_b}, \gamma, k_l, m)$.

Input: D'_B, $\hat{F}'_B$, D_{N_b}, F_{N_b}, γ, k_l, m;
Output: D_B, $\hat{F}$;
1 $\mathcal{J}' \leftarrow m(D'_B, \hat{F}', \gamma, k_l)$;
2 $\mathcal{J} \leftarrow m(D_{N_b}, F, \gamma, k_l)$;
3 **if** $\mathcal{J} \geq \mathcal{J}'$ **then**
4 $\quad \{D_B, \hat{F}_B\} \leftarrow \{D'_B, \hat{F}'_B\}$;
5 $\quad$ **go to line** 14;
6 **else**
7 $\quad$ **if** $B \leq N_b$ **then**
8 $\quad\quad \{D_B, \hat{F}_B\} \sim \mathcal{G}(D'_{N_b}, \hat{F}'_{N_b})$;
9 $\quad$ **else**
10 $\quad\quad D_B \leftarrow D'_{B-N_b} \bigcup D_{N_b}$;
11 $\quad\quad \hat{F}_B \leftarrow \hat{F}'_{B-N_b} \bigcup \hat{F}_{N_b}$;
12 $\quad$ **end**
13 **end**
14 **return** $\{D_B, \hat{F}_B\}$;

The procedures for refreshing the buffers include following three main steps.

Step 1. Calculation of the condition for updated buffers, line 1-2. Line 1 is used to calculate the manifold loss for the old data buffer D'_B and the old feature buffer $\hat{F}'$, and line 2 for the candidate buffers. The complexities of this step depends on the function $lle(\cdot)$ (as shown in **Algorithm 3**) or the function $lap(\cdot)$ (as shown in **Algorithm 4**). We denote the complexities of this step by $\mathcal{T}_m$ and $\mathcal{S}_m$.

Step 2. The buffers updating, line 3-13. If the manifold loss of the candidate data buffer and the candidate feature buffer is larger than or equal to the manifold loss of the old data buffer and the old feature buffer (line 3), then not change the buffers (line 4) and go to line 14, i.e., **Step 3**; otherwise, refresh the buffers (line 6-13). If the buffer size B (i.e., $B = k_0 \times |Y|$) is lesser than or equal to the mini-batch size N_b (line 12), then randomly choose B samples from candidate buffers (line 13); otherwise, remove N_b samples from the old buffer and set the new buffer with all samples from candidate buffer and the rest samples from the old buffer (line 10, 11). The time complexity and space complexity of this step are both equal to the buffer complexity, i.e., $O(B \times (|D| + |F|))$.

Step 3. Return the updated buffers, line 14. Return the refreshed buffers $\{D_B, \hat{F}_B\}$.

In summary, the time complexity of this algorithm is

$$\mathcal{T}_r = O(B \times (|D| + |F|)) + \mathcal{T}_m, \tag{22}$$

and the space complexity in each mini-batch training is

$$\mathcal{S}_r = O(B \times (|D| + |F|)) + \mathcal{S}_m. \tag{23}$$

B. Algorithm of calculating the manifold loss by LLE

Algorithm 3 shows the algorithm of calculating the manifold loss by LLE. The input of this algorithm includes data buffer D_B, feature buffer $\hat{F}$, the manifold loss balance parameter γ, and the number of neighbors k_l. The output of this algorithm is the LLE manifold loss $\mathcal{J}$.

Algorithm 3: Calculating the LLE manifold loss, denoted as $\mathcal{J} = lle(D_B, \hat{F}, \gamma, k_l)$.

Input: D_B, $\hat{F}$, γ, k_l;
Output: $\mathcal{J}$;
1 $N \leftarrow [N_i = k\text{-NN}(X_i, k_l) | X_i \in D_B]$, where $N_i = [n_{1,i}, n_{2,i}, ..., n_{k_l,i}]$;
2 **if** $k_l > |X_i|$ **then**
3 $\quad \epsilon \leftarrow 10^{-4}$;
4 **else**
5 $\quad \epsilon \leftarrow 0$;
6 **end**
7 $A^{\mathcal{M}} \leftarrow [\alpha_{i,j} = 0 | i, j = 1, 2, ..., B]$;
8 $W \leftarrow [w_{k,i} = 0 | k = 1, 2, ..., k_l; i = 1, 2, ..., B]$;
9 **for** $i = 1, 2, ..., B$ **do**
10 $\quad z = \{X_k - X_i | k = n_{1,i}, n_{2,i}, ..., n_{k_l,i}\}$;
11 $\quad Z \leftarrow z^\top z$;
12 $\quad Z \leftarrow Z + \epsilon \times \text{trace}(Z) \times \text{eye}(k_l, k_l)$;
13 $\quad W[:, i] \leftarrow Z \backslash \text{ones}(k_l, 1)$;
14 $\quad$ **for** $j = 1, 2, ..., k_l$ **do**
15 $\quad\quad w_{j,i} \leftarrow w_{j,i} / \sum_{j=1}^{k_l} w_{j,i}$;
16 $\quad\quad \alpha_{n_{j,i},i} \leftarrow w_{j,i}$;
17 $\quad$ **end**
18 **end**
19 $\mathcal{J} \leftarrow \frac{\gamma}{2|\hat{F}|} \sum_{\hat{F}_i \in \hat{F}} \|\hat{F}_i - \hat{F} A_i^{\mathcal{M}}\|^2$;
20 **return** $\mathcal{J}$;

The procedures for calculating the manifold loss by LLE include following four main steps.

Step 1. Setting the neighbors' ID, line 1. Set the neighbors' id by k-nearest neightbor algorithm k-NN$(\cdot)$. The time complexity and space complexity of this step are both equal to $O(B^2 \times |D|)$.

Step 2. Calculating the weight for each neighbor, line 2-18. Calculate the weight $\alpha_{i,j}$ for each neighbor pair X_i, X_j. The time complexity and space complexity of this step are both equal to $O(B^2 \times |D|)$.

Step 3. Calculating the manifold loss, line 19. Calculating the manifold loss $\mathcal{J}$ based on the feature buffer $\hat{F}$ and

the weight matrix $A^{\mathcal{M}}$. The time complexity of this step is $O(B^2 \times |F|)$, and the space complexity of this step is $O(1)$.

Step 4. Return the manifold loss, line 20. Return the manifold loss $\mathcal{J}$.

In summary, the time complexity of this algorithm is

$$\mathcal{T}_{lle} = O(B^2 \times (|D| + |F|)), \tag{24}$$

and the space complexity in each mini-batch training is

$$\mathcal{S}_{lle} = O(B^2 \times |D|). \tag{25}$$

C. Algorithm of calculating the manifold loss by Laplacian

Algorithm 4 shows the algorithm of calculating the manifold loss by Laplacian. The input of this algorithm includes data buffer D_B, feature buffer $\hat{F}$, the manifold loss balance parameter γ, and the number of neighbors k_l. The output of this algorithm is the Laplacian manifold loss $\mathcal{J}$.

Algorithm 4: Calculating the Laplacian manifold loss, denoted as $\mathcal{J} = lap(D_B, \hat{F}, \gamma, k_l)$.

Input: D_B, $\hat{F}$, γ, k_l;
Output: $\mathcal{J}$;
1 $N \leftarrow [N_i = k\text{-NN}(X_i, k_l) | X_i \in D_B]$, where $N_i = [n_{1,i}, n_{2,i}, ..., n_{k_l,i}]$;
2 $B^{\mathcal{M}} \leftarrow [\beta_{i,j} = 0 | i, j = 1, 2, ..., B]$;
3 $\epsilon \leftarrow \max_{X_i, X_j \in D_B} \|X_i - X_j\|^2 / B^2$;
4 **for** $X_i \in D_B$ **do**
5 **for** $j \in N_i$ **do**
6 $\beta_{j,i} \leftarrow \exp(\|X_i - X_j\|^2 / \epsilon)$;
7 **end**
8 $\beta_i \leftarrow \sum_{j \in N_i} \beta_{j,i}$;
9 **for** $j \in N_i$ **do**
10 $\beta_{j,i} \leftarrow \beta_{j,i} / \beta_i$;
11 **end**
12 **end**
13 $\mathcal{J} \leftarrow \frac{\gamma}{2|\hat{F}|^2} \sum_{\hat{F}_i, \hat{F}_j \in \hat{F}} \|\hat{F}_i - \hat{F}_j\|^2 B^{\mathcal{M}}_{ij}$;
14 **return** $\mathcal{J}$;

The procedures for calculating the manifold loss by Laplacian include following three main steps.

Step 1. Calculating the weight for each neighbor, line 1-12. Calculate the weight $\beta_{i,j}$ for each neighbor pair X_i, X_j. The time complexity and space complexity of this step are both equal to $O(B^2 \times |D|)$.

Step 2. Calculating the manifold loss, line 13. Calculating the manifold loss $\mathcal{J}$ based on the feature buffer $\hat{F}$ and the weight matrix $B^{\mathcal{M}}$. The time complexity of this step is $O(B^2 \times |F|)$, and the space complexity of this step is $O(1)$.

Step 3. Return the manifold loss, line 14. Return the manifold loss $\mathcal{J}$.

In summary, the time complexity of this algorithm is

$$\mathcal{T}_{lap} = O(B^2 \times (|D| + |F|)), \tag{26}$$

and the space complexity in each mini-batch training is

$$\mathcal{S}_{lap} = O(B^2 \times |D|). \tag{27}$$

Appendix B
Proof of Lemma 1

Lemma 1: If $x_1, x_2, ..., x_n$ satisfies Definition 1, then:

$$P(|\bar{x} - E(\bar{x})| \geq \frac{\lambda}{n}) \leq 2\exp(\frac{-\lambda^2}{2\sum_{i=1}^n c_i^2}), \tag{28}$$

where $\bar{x}$ is the average of $x_1, x_2, ..., x_n$.

Proof: For a fixed t ($t \geq 0$), the function e^{ty} of the variable y is convex in the interval $[-g, g]$ with $g \geq 0$.We draw a line between the two endpoints points $(-g, e^{-tg})$ and (g, e^{tg}). The curve of e^{ty} lies entirely below this line. Thus,

$$e^{ty} \leq \frac{g-y}{2g} e^{-tg} + \frac{g+y}{2g} e^{tg}. \tag{29}$$

According to Eq. (29) and $\|x_{i-1} - x_i\| \leq c_i$ (actually $|x_{i-1} - x_i| \leq c_i$), we have:

$$\begin{aligned} E(e^{t(x_i - x_{i-1})} | x_{[1:i-1]}) &\leq E(\frac{(e^{tc_i} - e^{-tc_i})(x_i - x_{i-1})}{2c_i} | x_{[1:i-1]}) + \\ &E(\frac{e^{tc_i} + e^{-tc_i}}{2} | x_{[1:i-1]}) \\ &= \frac{e^{tc_i} + e^{-tc_i}}{2}, \end{aligned} \tag{30}$$

where $x_{[1:i-1]} = \{x_1, x_2, ..., x_{i-1}\}$. Based on **Definition 1**, we have:

$$E(\frac{(e^{tc_i} - e^{-tc_i})(x_i - x_{i-1})}{2c_i} | x_{[1:i-1]}) = 0.$$

Using the Taylor expansion, we have:

$$\frac{e^{tc_i} + e^{-tc_i}}{2} \leq \exp(\frac{t^2 c_i^2}{2}). \tag{31}$$

With the condition $E(e^{tx_{i-1}} | x_{[1:i-1]}) = e^{tx_{i-1}}$, we have:

$$E(e^{tx_i} | x_{[1:i-1]}) \leq \exp(\frac{t^2 c_i^2}{2}) e^{tx_{i-1}}. \tag{32}$$

Inductively, we have:

$$\begin{aligned} E(e^{tx}) &= E(E(e^{tx_n} | x_{[1:n-1]})) \\ &\leq \exp(\frac{t^2 c_n^2}{2}) E(e^{tx_{n-1}}) \leq \cdots \leq \prod_{i=1}^n \exp(\frac{t^2 c_i^2}{2}) E(e^{tx_i}) \\ &= \exp(\frac{1}{2} t^2 \sum_{i=1}^n c_i^2) \exp(tE(x)), \end{aligned} \tag{33}$$

where x is the sum of the input samples. According to Markov's inequality, we have:

$$\begin{aligned} P(x \geq E(x) + \lambda) &= P(\exp(t(x - E(x))) \geq e^{t\lambda}) \\ &\leq e^{-t\lambda} E(e^{t(x - E(x))}) \\ &\leq e^{-t\lambda} \exp(\frac{t^2 \sum_{i=1}^n c_i^2}{2}) \\ &= \exp(-t\lambda + \frac{1}{2} t^2 \sum_{i=1}^n c_i^2). \end{aligned} \tag{34}$$

We choose $t = \lambda / \sum_{i=1}^{n} c_i^2$ (in order to minimize the above expression), and have:

$$\begin{aligned} P(x \geq E(x) + \lambda) &\leq \exp(-t\lambda + \frac{1}{2}t^2 \sum_{i=1}^{n} c_i^2) \\ &= \exp(\frac{-\lambda^2}{2\sum_{i=1}^{n} c_i^2}). \end{aligned} \tag{35}$$

To derive a similar lower bound, we consider $-x_i$ instead of x_i in the preceding proof. Then we obtain the following bound for the lower tail:

$$P(x \leq E(x) - \lambda) \leq \exp(\frac{-\lambda^2}{2\sum_{i=1}^{n} c_i^2}). \tag{36}$$

So, we have:

$$P(|\bar{x} - E(\bar{x})| \geq \frac{\lambda}{n}) \leq 2\exp(\frac{-\lambda^2}{2\sum_{i=1}^{n} c_i^2}), \tag{37}$$

where $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ is the average. Thus, the theorem is proved. ■

Appendix C
Comparing Precision and Success Plots for All Attributes of the OTB Dataset

The experimental results (precision and success) for full set of plots generated by the benchmark toolbox of OTB [46] are reported in Fig. 7 and Fig. 8, where STM performs better than other trackers.

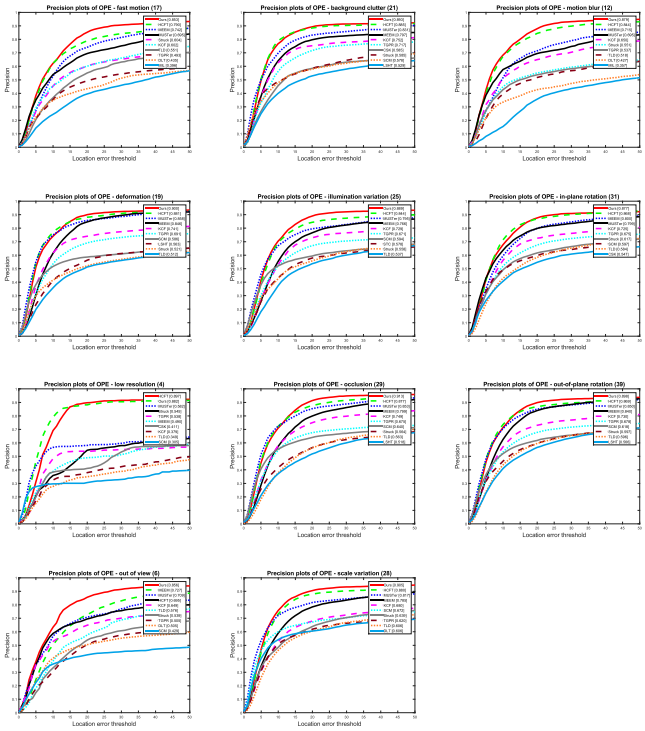


Fig. 7: Precision plots for all attributes of the OTB dataset.

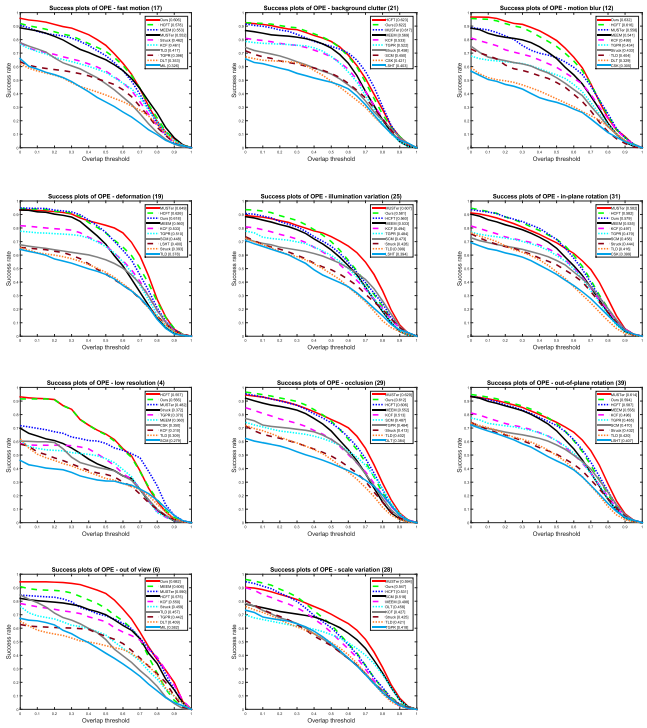


Fig. 8: Success plots for all attributes of the OTB dataset.

Baochang Zhang is currently an associate professor with the Science and Technology on Aircraft Control Laboratory, School of Automation Science and Electrical Engineering, Beihang University, Beijing, China. His research interests include pattern recognition, machine learning, face recognition, and wavelets.

Wankou Yang is an associate professor in the School of Automation, Southeast University, P.R. China. His research interests include pattern recognition and computer vision.

Ze Wang is currently a Ph.D. student at the Electrical and Computer Engineering department, Duke University, USA. His research interests include machine learning and computer vision.

Lian Zhuo received his B.S. and M.S. degrees at the School of Automation Science and Electrical Engineering, Beihang University (BUAA), PR.China, 2018 and 2020 respectively. His research interests include pattern recognition and computer vision.

Xiantong Zhen is currently a senior scientist with Inception Institute of Artificial Intelligence (IIAI) in the United Arab Emirates. His research interests include machine learning and computer vision.

Jungong Han is currently an associate professor with WMG Data Science at University of Warwick, UK. His research interests span the fields of video analysis, computer vision and applied machine learning.