This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3049784, IEEE Transactions on Pattern Analysis and Machine Intelligence

Differentiated Explanation of Deep Neural Networks with Skewed Distributions

Weijie Fu, Meng Wang, *Fellow, IEEE,*
Mengnan Du, Ninghao Liu, Shijie Hao, and Xia Hu, *Member, IEEE*

Abstract—Over the last decade, deep neural networks (DNNs) are regarded as black-box methods, and their decisions are criticized for the lack of explainability. Existing attempts based on local explanations offer each input a visual saliency map, where the supporting features that contribute to the decision are emphasized with high relevance scores. In this paper, we improve the saliency map based on differentiated explanations, of which the saliency map not only distinguishes the supporting features from backgrounds but also shows the different degrees of importance of the various parts within the supporting features. To do this, we propose to learn a differentiated relevance estimator called DRE, where a carefully-designed distribution controller is introduced to guide the relevance scores towards right-skewed distributions. DRE can be directly optimized under pure classification losses, enabling higher faithfulness of explanations and avoiding non-trivial hyper-parameter tuning. The experimental results on three real-world datasets demonstrate that our differentiated explanations significantly improve the faithfulness with high explainability. Our code and trained models are available at https://github.com/fuweijie/DRE.

Index Terms—deep neural networks, local explanation, relevance scores, differentiated saliency maps

1 Introduction

Deep neural networks (DNNs) have achieved high accuracies in a wide range of fields, such as image recognition [11], and natural language processing [10]. However, they often lack meaningful explanations about how specific decisions are made and are regarded as black-box methods. In particular, the explanation should take both faithfulness and explainability into account. The faithfulness estimates the fidelity between the explanation and the decision behavior of original DNNs, and the explainability quantifies how easy it is to understand the explanation for humans.

Local explanation methods are proposed to address this issue. They provide users an understandable rationale for each specific decision with a visual saliency map, where the relevance score of each feature indicates its contribution to the decision. For high faithfulness, the supporting features contributing to increase the probability of the target class are supposed to obtain high scores, and the remaining features regarded as backgrounds are expected to get almost zero scores. In particular, gradient-based explanations compute the partial derivative of the class probability with respect to input features via back-propagation [2], [20]. Besides, perturbation-based explanations aim to find the smallest region, which allows a confident decision directly or prevents a confident decision once being removed [8]. By applying

- W. Fu, M. Wang are with the School of Computer and Information, Hefei University of Technology, Hefei, 230009, China, and Intelligent Interconnected Systems Laboratory of Anhui Province (Hefei University of Technology) (e-mail: {fwj.edu, eric.mengwang}@gmail.com).
- S. Hao is with the School of Computer and Information, Hefei University of Technology, 230009, China (e-mail: hfut.hsj@gmail.com).
- M. Du, N. Liu, X. Hu are with the School of Computer Science and Engineering, Texas A&M University, TX 77840, U.S.A. (e-mail: {dumengnan, nhliu43}@tamu.com, and hu@cse.tamu.edu).
- M. Wang is the corresponding author.

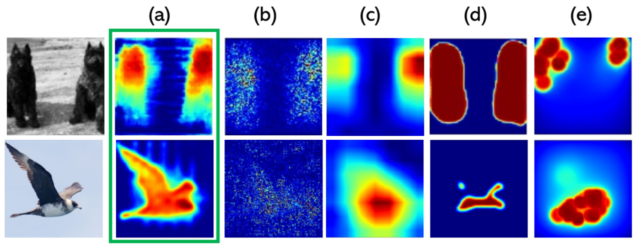


Fig. 1. Comparison of different saliency maps: (a) The proposed DRE, (b) Vanilla Gradient [20], (c) Grad-CAM++ [2], (d) Mask Generator [4], and (e) Extreme Perturbation [7].

various ad hoc constraints on the region and lowering the contributions of intricate supporting features, they maintain faithfulness and improve explainability.

To provide better explanations of the decisions, differentiated saliency maps are preferred. That is, *the strong supporting features that significantly contribute to the probability of the target class are highlighted with very high scores, the weak supporting features that slowly increase the probability obtain lower scores, and the other features regarded as the background have almost zero scores.* Based on the differentiated explanations, users not only can locate the whole set of supporting features but also figure out which parts of them are more important than the others. For illustration, two examples are shown in Fig.1(a), which not only capture the shapes of the whole animals but also provide detailed insights that their heads contribute more than the remaining parts.

However, the existing local methods fail to produce the differentiated explanations. For example, instead of directly addressing the basic question ”what makes this image belong to the target class”, the gradient-based methods answer the question ”what makes this instance more or

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3049784, IEEE Transactions on Pattern Analysis and Machine Intelligence

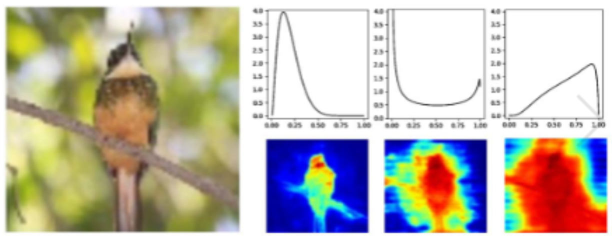


Fig. 2. The saliency maps obtained from the right-skewed distribution controller to the left-skewed distribution controller, illustrating the benefit of the right-skewed distribution for human-friendly explanations.

less similar to the target class" [16], leading to noisy results within the same region (Fig.1(b)). [2], [19] propose to create saliency maps by combining the gradients with the corresponding feature maps at high-level layers. However, they ignore the fine-grained information within low-level layers and bring coarse saliency maps (Fig.1(c)). In addition, the perturbation-based methods are formulated to highlight the supporting features directly, which ignore the different degrees of importance of these features [4] (Fig.1(d)). Recently, some explanation methods introduce soft ad hoc constraints and data augmentation techniques to improve their saliency maps [6], [24]. Nevertheless, they either significantly increase the number of iterations for optimizing each saliency map, or require users to carefully tune hyper-parameters to trade-off the constraints and the classification loss, leading to non-negligible costs. Although [7] introduces extreme perturbations with hard constraints to ease the setting of hyper-parameters, its saliency maps ignore some parts of the supporting features and still lose the differentiation on the detected supporting features (Fig.1(e)).

In this paper, we propose to learn a Differentiated Relevance Estimator (DRE) to construct differentiated explanations. Leveraging the quantitative observations found in [2], [7] that the occlusion of 5% (25%) pixels in natural images can bring nearly 50% (90%) drop in classification confidence, we present distribution controllers to guide the relevance scores towards right-skew distributions [3], so as to improve the consistency between the scores of input features and the actual contributions. Our qualitative experimental analysis on the skewness of distributions additionally shows the effectiveness of the right-skewed distribution for building human-friendly explanations, as displayed in Fig.2. We introduce the detailed setting of the controller by establishing the connections between its input and output distributions and then integrate it with a trainable mask generator to build the final estimator. Benefiting from the controller, we directly optimize DRE under classification losses, which avoids all ad hoc constraints and non-trivial hyper-parameter tuning. We further discuss a simple trick to improve saliency maps based on the ranking of relevance scores itself, which offers DRE more flexibility to address the various proportions of supporting features across instances.

The main contributions of our work are as follows.

- We introduce differentiated explanations and propose a novel relevance estimator DRE by integrating a distribution controller with a trainable mask generator. We develop a practical controller to guide relevance scores towards the desired right-skewed distributions, where the involved hyper-parameters can be easily set.
- We introduce classification losses to train DRE directly. It avoids the non-trivial hyper-parameter tuning on ad hoc constraints and also significantly improves the faithfulness of explanations.
- We empirically demonstrate the effectiveness of the above innovations with targeted ablation studies. Besides, the experimental comparison to other methods shows that DRE not only obtains better quantitative performance but also provides differentiated saliency maps for human-friendly explanations.
- We extend DRE with simple tricks with post hoc tuning. The results show that DRE can easily benefit from itself and be adaptive to different images.

2 Related Work

Gradient-based methods. Gradient-based methods leverage back-propagation to track information from the DNN's output back to its input [20]. In general, these methods are advantageous in their high computational efficiency, i.e., using a few forward-and-backward iterations is sufficient to generate saliency maps. However, the saliency maps based on the naive gradients are visually noisy and hard to understand. To address this issue, Smooth Grad [22] reduces the visual noise by introducing noise to inputs repeatedly, and Integrated Grad [24] estimates the global contribution of each feature rather than the local sensitivity. Guided backprorogation [23] modifies the gradients of ReLU functions by discarding negative values at the back-propagation process. Besides, recent methods propose to create saliency maps by combining the gradients with the corresponding feature maps. For example, Grad CAM [19] and Grad CAM ++ [2] take advantage of high-level feature maps to make saliency maps cleaner. Nevertheless, they inevitably sacrifice the detailed estimation of the contributions of input features and lead to coarse saliency maps.

Perturbation-based methods. Perturbation-based methods optimize the saliency map of each decision by perturbing its input features and observing the change in the output of DNNs. For example, [8] designs a preservation game to find the smallest region that significantly increases the probability of the target class. The authors also design a deletion game by preventing DNNs from recognizing objects. To improve the explainability, [6] regularizes saliency maps with middle-level feature maps and optimizes them by reconstructing higher-level feature maps. [7] further introduces extreme perturbations with a hard constraint on saliency maps, aiming to avoid the hyper-parameter tuning on soft ad hoc constraints. Nevertheless, to obtain a high-quality explanation, the above methods demand hundreds of iterations for optimizing the saliency map for each image, leading to non-negligible time costs. Recently, [4] proposes an efficient method for real-time saliency detection, which utilizes a trainable network to generate saliency maps.

Model-agnostic methods. To make the explanations compatible with more types of data and black-box classifiers, model-agnostic methods are proposed, such as LIME

0162-8828 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Science & Technology of China. Downloaded on January 10,2021 at 02:28:15 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3049784, IEEE Transactions on Pattern Analysis and Machine Intelligence

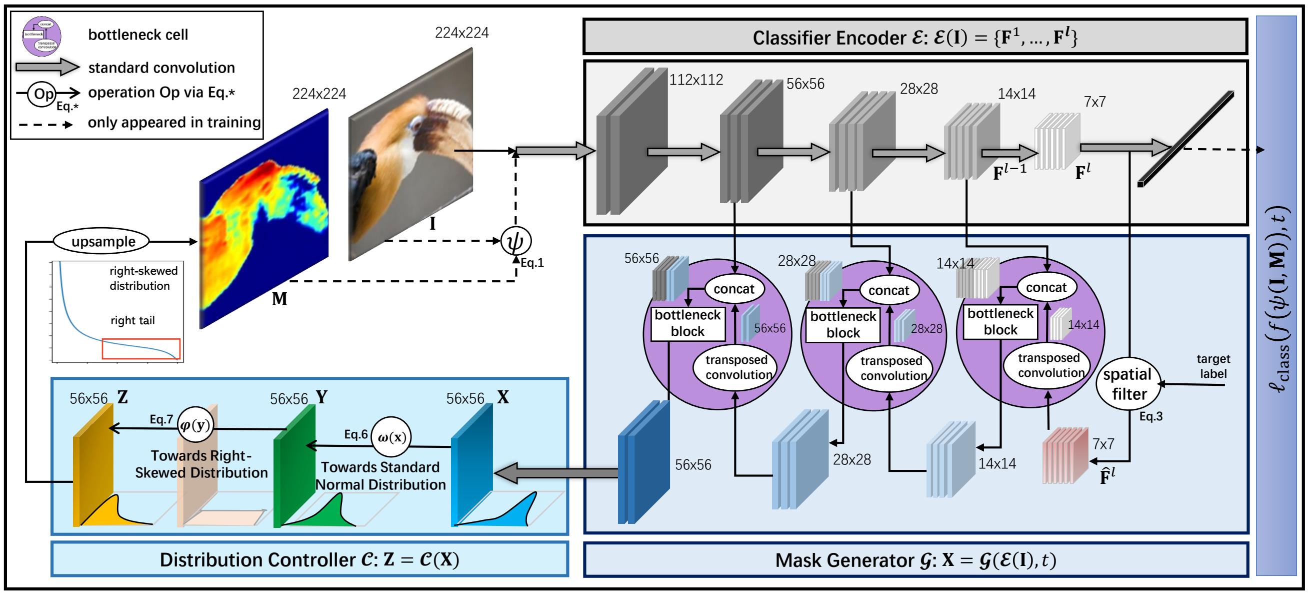

bottleneck cell
standard convolution
operation Op via Eq.*
only appeared in training
224x224
224x224
Classifier Encoder $\mathcal{E}$: $\mathcal{E}(\mathbf{I}) = \{\mathbf{F}^1, ..., \mathbf{F}^l\}$
112x112
56x56
28x28
14x14
7x7
$\mathbf{F}^{l-1}$
$\mathbf{F}^l$
upsample
right-skewed distribution
right tail
M
I
ψ
Eq.1
concat
bottleneck block
transposed convolution
spatial filter
target label
Eq.3
$\hat{\mathbf{F}}^l$
$\ell_{\text{class}}(f(\psi(\mathbf{I},\mathbf{M})),t)$
56x56 Z
56x56 Y
56x56 X
Eq.7 $\varphi(\mathbf{y})$
Eq.6 $\omega(\mathbf{x})$
Towards Right-Skewed Distribution
Towards Standard Normal Distribution
Distribution Controller $\mathcal{C}$: $\mathbf{Z} = \mathcal{C}(\mathbf{X})$
Mask Generator $\mathcal{G}$: $\mathbf{X} = \mathcal{G}(\mathcal{E}(\mathbf{I}), t)$

Fig. 3. The framework of our differentiated relevance estimator, where a distribution controller $\mathcal{C}$ is introduced right after the mask generator $\mathcal{G}$. For each instance $\mathbf{I}$, $\mathcal{G}$ takes the feature maps of the neural network $\mathcal{E}(\mathbf{I})$ as inputs and feeds the obtained mask $\mathbf{X}$ into the controller $\mathcal{C}$. The detailed process flow inside the mask generator $\mathcal{G}$ can be found in Sec.3.2.2. Then $\mathcal{C}$ guides the relevance scores towards the right-skewed distribution for a differentiated mask through $\mathbf{X}\to\mathbf{Y}\to\mathbf{Z}$. The final mask $\mathbf{M}$ with the original size is obtained via upsampling. In addition, we annotate the spatial sizes of feature maps and display the expected distributions of the scores within the controller.

[17], SHAP [15], and Anchor [18]. In particular, LIME employs more interpretable linear models to approximate the decisions of black-box classifiers. It assumes that each explanation can be derived from the points randomly generated around the neighborhood of the instance and their proximity measures. SHAP further introduces a united framework for interpreting decisions based on Shapley values and builds its connection to LIME. Nevertheless, these methods generally take much larger time costs to converge. For example, LIME takes around 10 minutes to explain each decision of Inception networks, and Shapley values take a more considerable time cost to compute [17].

3 Differentiated Explanation

3.1 Problem Statement

In a multi-class classification task, suppose a DNN classifier f is already trained over a training set. For each instance $\mathbf{I}$, local explanation aims to find out the contributions of its input features to the probability of the target class that we want to interpret. Take image classification as an example, where $I_{i,j}$ denotes to the pixel of $\mathbf{I}$ at the location of i,j. The corresponding local explanation is represented by a same-size mask[1] $\mathbf{M}$, in which each relevance score $M_{i,j}\in[0,1]$ represents the contribution of $I_{i,j}$ for the target probability. To improve explanations, differentiated masks are preferred.

We first analyze the perturbation-based methods [4], [8] in Sec. 3.2 and then introduce our method for differentiated explanations with skewed distributions in Sec. 3.3 - Sec. 3.5. Some important notations used in Sec. 3 are listed in Tab.1.

1. In this paper, we do not distinguish saliency maps and masks, as both indicate the permutation of relevance scores of an instance.

TABLE 1
Notations and definitions.

Notation	Definition
$\mathcal{E}$	The encoder of the classifier.
$\mathcal{G}$	The mask generator after the encoder.
$\mathcal{C}$	The distribution controller after the generator.
t	The label of the target (predicted) class.
l	The number of convolutional layers.
$\mathbf{I}$	The notation for images.
$\mathbf{B}$	The notation for background images.
$\mathbf{M}$	The notation for saliency maps or masks.
i,j,k	The symbols used for indexes.
$\mathbf{F}^k$	The feature maps at the k-th layer ($1 \le k \le l$).
$\mathbf{V}_k$	The embedding vector of the k-th class.
$\hat{\mathbf{F}}^l$	The last feature maps after the spatial attenuation.
$\mathbf{X}$	The output of the mask generator $\mathcal{G}$.
$\mathbf{Y}$	The intermediate variable inside the controller $\mathcal{C}$.
$\mathbf{Z}$	The output of the distribution controller $\mathcal{C}$.
$\ell.$	The symbol for different losses.
$\psi(\cdot,\cdot)$	The image perturbation with Eq.1.
$\omega(x)$	The instance normalization on x with Eq.6.
$\varphi(y)$	The transformation function on y with Eq.7.
$p(z)$	The probability density function of the variable z.
$\mathbb{E}[x]$	The expectation of the variable x.
$\mathbb{V}[x]$	The variance of the variable x.
η, h	The parameters inside $\mathcal{C}$, set based on $p(z)$.

3.2 Perturbation Analysis

3.2.1 Formulation of Perturbation-based Methods

To find supporting pixels, these methods perturb $\mathbf{I}$ according to an initialized mask $\mathbf{M}$ and introduce an alternative background image $\mathbf{B}$ to reduce the amount of unwanted evidence. Specifically, the perturbation is defined as

$$\psi(\mathbf{I},\mathbf{M}) = \mathbf{I}\odot\mathbf{M} + \mathbf{B}\odot(\mathbf{1}-\mathbf{M}), \quad (1)$$

where $\odot$ denotes the Hadamard product. Then these methods feed the perturbed image into the classifier and optimize the mask to locate the supporting pixels that increase the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3049784, IEEE Transactions on Pattern Analysis and Machine Intelligence

probability of the target class [4]. Specifically, let t denote the label of the target class, and $f_t(\psi(\mathbf{I},\mathbf{M}))$ is the corresponding class probability of the above perturbed image. The objective ℓ_{pert} of these methods can be formulated as

$$\begin{aligned}\text{argmin}_{\mathbf{M}} - f_t(\psi(\mathbf{I},\mathbf{M})) + \lambda_{\text{bg}} f_t(\psi(\mathbf{I},\mathbf{1}-\mathbf{M})) \\ + \lambda_{\text{av}}\Theta_{\text{av}}(\mathbf{M}) + \lambda_{\text{tv}}\Theta_{\text{tv}}(\mathbf{M}),\end{aligned} \tag{2}$$

where $-f_t(\psi(\mathbf{I},\mathbf{M}))$ encourages the supporting pixels to obtain high relevance scores, and $f_t(\psi(\mathbf{I},\mathbf{1}\text{-}\mathbf{M}))$ aims to avoid the supporting pixels being regarded as the background. Besides, the constraint $\Theta_{\text{av}}(\cdot)$ is used to minimize the area of the mask, and $\Theta_{\text{tv}}(\cdot)$ enforces it to be smooth.

3.2.2 *Real-time Mask Generator*

The iterative optimization for the above problem results in a considerable time cost for each test image. Thus, a mask generator $\mathcal{G}$ that produces real-time saliency maps is proposed in [4]. The simplified architecture is displayed at the bottom-right of Fig.3, which consists of a class-related spatial filter, three bottleneck cells, and a standard convolutional layer.

During the mask generation, the classifier first produces raw feature maps $\{\mathbf{F}^k\}_{k=1}^{l}$ at multiple layers for each image based on the encoder $\mathcal{E}$ and obtains its label t=argmax$_k f_k$. Then for the last feature maps $\mathbf{F}^l$, the spatial filter uses its class-related embedding vectors to attenuate the spatial locations whose feature vectors are dissimilar to the embedding of the target class. Let $\mathbf{V}_t$ be the above embedding vector. The output of the spatial filter at the location i,j denoted as $\hat{\mathbf{F}}^l_{ij}$ is calculated as

$$\hat{\mathbf{F}}^l_{ij} = \mathbf{F}^l_{ij}\text{sigmoid}({\mathbf{F}^l_{ij}}^{\text{T}}\mathbf{V}_t). \tag{3}$$

where $\mathbf{F}^l_{ij}$ denotes the feature vector of $\mathbf{F}^l$ at the location i,j. The first bottleneck cell then upsamples the filtered maps $\hat{\mathbf{F}}^l$ by a factor of two using transposed convolutions [29]. It introduces its bottleneck block [11] to generate new feature maps based on the concatenation of the upsampled maps and the higher-resolution raw feature maps $\mathbf{F}^{l-1}$. The following two bottleneck cells repeat this process as shown in Fig.3, and the channel numbers of their generated feature maps are the same as those of the corresponding upsampled feature maps. The standard convolutional layer takes the outputs of the final bottleneck cell and produces a one-channel feature map $\mathbf{X}$ at a coarse scale such as 56×56. Finally, for this coarse mask $\mathbf{X}$=$\mathcal{G}(\mathcal{E}(\mathbf{I}),t)$, the upsampling based on bilinear interpolation is employed to obtain a smoother mask at the image scale as $\mathbf{M}$=upsample($\mathbf{X}$).

Now we consider the optimization of the spatial filter and the remaining parts of the mask generator. Similar to metric learning [14], the class-related embedding vectors in the spatial filter can be gradually updated by maximizing the similarity to the feature vectors of the same-class images while minimizing the similarity to those of different-class images. Thus, we assign training images with true labels (k=t) and fake labels (k≠t) iteratively and update the embedding vectors $\mathbf{V}_k$ by minimizing the following loss:

$$\ell_{\text{embed}}(\mathbf{F}^l,\mathbf{V}_k) = \begin{cases} -\sum(\text{sigmoid}({\mathbf{F}^l_{ij}}^{\text{T}}\mathbf{V}_k)),\ k=t; \\ \sum(\text{sigmoid}({\mathbf{F}^l_{ij}}^{\text{T}}\mathbf{V}_k)),\ k\neq t. \end{cases} \tag{4}$$

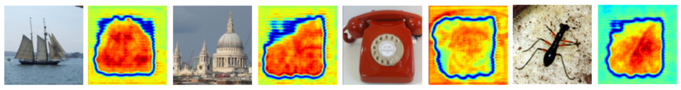


Fig. 4. The examples of the masks obtained with non-monotonic mappings, where higher scores can not guarantee larger contributions.

After that, the remaining parts of the mask generator can be optimized based on its generated mask $\mathbf{M}$ via Eq.2. Since the mask generator is trained offline, we can obtain a real-time explanation based on a single forward-pass.

3.2.3 *Limitation Analysis*

Lack of differentiation. The perturbation-based explanations are formulated to distinguish the supporting features from the background and are generally optimized based on a large number of iterations. Although mask generators significantly accelerate the explanations, they still fail to consider the different degrees of importance of the supporting features, and their obtained masks are lack of differentiation.

Sensitive hyper-parameter tuning. During the training phase, balancing the trade-offs between the classification loss and the additional soft constraints, e.g., the smoothing term in Eq.2, involves a non-trivial hyper-parameter tuning process. Since the quality of masks is subjective for evaluation, it increases the burden of learning a good generator where the Bayesian optimization is hard to employ [28].

3.3 Principles of Controllers

To produce differentiated saliency maps, we first introduce the concept of distribution controllers $\mathcal{C}$, which guides the relevance scores towards desired distributions. We place the controller right after the generator to together compose the differentiated relevance estimator DRE. Suppose $\mathbf{X}$ is the initial output of the generator (as we mentioned in Sec. 3.2.2), and $\mathbf{Z}$ denotes the output of $\mathcal{C}$. The output of the distribution controller is expressed as

$$\mathbf{Z} = \mathcal{C}(\mathbf{X}). \tag{5}$$

Besides, we follow [4], [6] to upsample $\mathbf{Z}$ with interpolation, aiming to improve the smoothness at the image scale. An overview of our framework is provided in Fig.3.

We investigate the principles for the controller design.

Principle 1. The hyper-parameters in $\mathcal{C}$ can be *easily set* without prior knowledge of classifiers and datasets.

Principle 2. The output relevance scores of $\mathcal{C}$ approach a *right-skewed distribution* over (0,1) for each decision.

Principle 3. The mapping function from the distribution controller's input $\mathbf{X}$ to its output $\mathbf{Z}$ is *monotonic*.

For the illustration of the last two principles, two examples are shown in Fig.2 and Fig.4, respectively. In the first figure, by modifying the expected distributions of the outputs of $\mathcal{C}$ from the right-skewed distribution to the left-skewed distribution, the differentiation of the saliency map is remarkably reduced. Besides, the proportions of pixels highlighted with high scores are positively correlated with the area at the right part of pre-configured distributions. It is worthwhile noting that the quantitative observation in [2], [7] shows that the occlusion of 5% (25%) pixels in natural images can bring nearly 50% (90%) drop in classification

0162-8828 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Science & Technology of China. Downloaded on January 10,2021 at 02:28:15 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3049784, IEEE Transactions on Pattern Analysis and Machine Intelligence

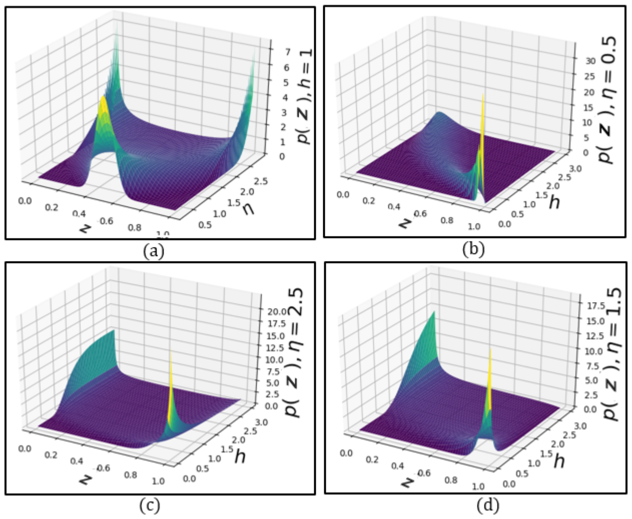


Fig. 5. The PDFs with the different settings of hyper-parameters.

confidence, which again demonstrates the effectiveness of the right-skewed distributions. In the second figure, a non-monotonic transform $g(x)$=x^2 is used in $\mathcal{C}$, making the scores deviate from the expected meaning. That is, a higher score implies a larger contribution. In contrast, a monotonic mapping enables to enhance the differentiation of a saliency map without changing the ranking of its relevance scores.

3.4 Controller Design

Following the above principles, we introduce a simple design of the controller. Since the sum of two independent random variables is more normal (Gaussian) than the original variables [12], [13], we first assume that the distribution of the inputs of the controller $\mathcal{C}$ (the outputs of the generator $\mathcal{G}$) is nearly normal for convenience. Later we show that the proposed controller built upon this assumption also guides other distributions towards the right-skewed ones.

3.4.1 *From the Normal to the Standard Normal*

To obtain the desired distributions, we first introduce instance normalization [26] to guide the normal distribution towards the standard normal distribution. The goal of this step is to shift the scores around the opposite sides of zero.

Specifically, let $x_{ij}\in\mathbf{X}$ be the input entry at the location (i,j), and $y_{ij}\in\mathbf{Y}$ denotes the expected variable following a standard normal distribution. The mapping can be expressed as

$$y_{ij} = \omega(x)_{ij} = (x_{ij} - \mathbb{E}[x_{ij}])/(\sqrt{\mathbb{V}[x_{ij}]}), \tag{6}$$

where the expectation $\mathbb{E}[\cdot]$ and variance $\mathbb{V}[\cdot]$ are computed over the entries of each $\mathbf{X}$.

3.4.2 *From the Standard Normal to the Right-skewed*

Now we guide the above scores towards right-skewed distributions monotonically. To do this, we introduce a customized transformation function with easy-to-set hyper-parameters.

To produce the relevance scores with a right tail in (0,1), we first transform the normal distribution towards a uniform distribution based on sigmoid functions and then change the skewness of the distribution based on power functions. An illustrative example is shown in the bottom-left in Fig.3. Specifically, the output z_{ij} of $\mathcal{C}$ is obtained as

$$z_{ij} = \varphi(y_{ij}) = (\text{sigmoid}(\eta \cdot y_{ij}))^h = (\frac{1}{1+\mathrm{e}^{-\eta \cdot y_{ij}}})^h \tag{7}$$

where η aims to guide the new scores approach the uniform distribution [27], and h determines the skewness of the final distribution.

In particular, the hyper-parameters η and h can be easily set according to their effects on the transformed probability density function (PDF) $p(z)$. To do this, we introduce the probability density transformation [9] and obtain $p(z)$ as

$$p(z) = \frac{1}{\sqrt{2\pi}h\eta} \cdot \frac{1}{z(1-z^{1/h})} \cdot \mathrm{e}^{-\frac{(\ln(z^{(-1/h)}-1))^2}{2\eta^2}}, \tag{8}$$

The detailed proof can be found in Appendix A.

Analysis. Now we set the hyper-parameters based on their effects on the intuitive geometry of $p(z)$, which corresponds to the distribution of relevance scores.

Firstly, we fix h=1 and observe the effect of η. The corresponding PDFs are displayed in Fig.5(a). By changing η within (0.5,2.5), $p(z)$ remains its skewness and changes from the concave to the convex for $z\in$(0,1). In particular, η=1.5 approximately leads $p(z)$ to an uniform distribution. Considering 1.5$\approx$0.9$\sqrt{\pi}$, it is consistent to the sigmoid approximation of the cumulative probabilities of the standard normal distribution [27].

Secondly, we set η=$\{0.5,1.5,2.5\}$ and observe the effect of h. Three PDF figures are displayed in Figs.5(b-d), where all $p(z)$s are able to obtain right-skewed distributions under a large h. However, the geometries of these $p(z)$s are significantly different. Fig.5(b) shows that $p(z)$ with η=0.5 obtains extremely low probabilities for $z\in$(0.5,1). Once a relevance score larger than 0.5 appears and becomes an outlier, this range for highlighting strong supporting features is likely to be wasted. Fig.5(c) shows that $p(z)$ with η=2.5 continues the undesired convexity and leads to more high scores than middle scores. $p(z)$ with η=1.5 can lead to a clear tail over the range of (0,1), as shown in Fig.5(d).

Above all, η=1.5 enables the sigmoid approximation of the cumulative probability for the standard normal distribution and leads it to a uniform distribution [27]. With h=2.5, we can further obtain the scores under the right-skewed distribution with a clear tail over (0,1). Note that other transformations with monotonicity can also be considered.

3.4.3 *Effects on Other Distributions.*

Now we relax the normal distribution on $\mathbf{X}$ and analyze the effects of the above controller (η=1.5 and h=2.5) on other typical distributions, including uniform distributions, the mixtures of normal distributions, and skew normal distributions. For simplicity, we only show the results based on the synthetic data and leave the detailed analysis in Appendix B. The original distributions (the 1st row) and their transformed distributions (the 2nd row) are displayed in Fig.6. As we can see, although the controller may not transform them into the right-skewed distributions completely, it still shifts a majority of relevance scores towards lower values and remains a minority of the scores at high values, which makes the scores away from the left-skewed distributions.

0162-8828 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Science & Technology of China. Downloaded on January 10,2021 at 02:28:15 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3049784, IEEE Transactions on Pattern Analysis and Machine Intelligence

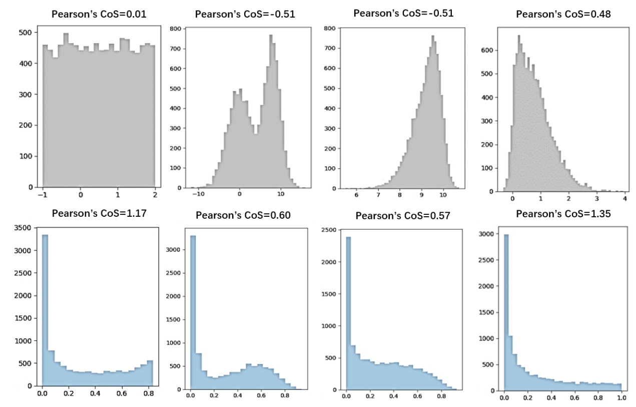


Fig. 6. The effects of the controller on the synthetic data with different distributions. The 1st and the 2nd rows show original distributions and their transformed distributions, respectively. For a quantitative comparison, we also list their Pearson's Coefficients of Skewness (CoS) [5].

From a quantitative perspective, we also calculate their Pearson's Coefficients of Skewness (CoS) [5] for comparison. Specifically, if skewness is positive (negative), the relevance scores are right- (left-) skewed, meaning that the right (left) tail of the distribution is longer than the left (right). As we observe, our controller consistently increases the values of the skewness for the above typical distributions.

3.5 Estimator Optimization

This section pays attention to the optimization of the above differentiated relevance estimator DRE. Denote ℓ_{class} as the classification loss used for training DNNs, such as the commonly used cross-entropy loss. Considering the explanation is already guided to be differentiated, DRE can be directly optimized without the non-trivial hyper-parameter tuning:

$$\text{argmin}_{\mathcal{G}} \ell_{\text{class}}(f(\psi(\mathbf{I}, \mathbf{M})), t), \text{ where } \mathbf{M} = \text{upsample}(\mathcal{C}(\mathbf{X})). \tag{9}$$

In particular, Eq.9 enables us to improve the faithfulness of explanations, since it is simplified to find the region that maximizes the target probability under the expected distribution. It is also valuable to train the classifier and the relevance estimator at the same time. As this paper focuses on post hoc explanations, we leave it for our future work.

4 Experiments

This section investigates the performance of DRE. We first evaluate the faithfulness and the explainability based on object recognition and scene recognition tasks. Then, we introduce simple tricks to further improve the performance. Finally, we perform targeted ablation studies to empirically demonstrate the effectiveness of the proposed innovations and discuss the results for misclassifications.

4.1 Setup

To demonstrate the broad applicability, we apply the proposed DRE to 3 types of CNNs, including ResNet50 [11], VGG19 [21], and GoogleNet [25]. The following 9 methods are used for comparison: (1) Mask Generator (MGnet) [4], (2) Meaningful Perturbation (MPert) [8], (3) Grad CAM (GCAM) [19], (4) Grad CAM++ (GCAM++) [2], (5) Feature Inversion (FInv) [6], (6) Extreme Perturbation (XPert) [7], (7) Vanilla Gradient (VGrad) [20], (8) Smoothness Gradient (SMGrad) [22], (9) Integrated Gradient (ITGrad) [24]. Of note, most of them require a large number of forward-and-backward iterations to build each mask and involve sensitive hyper-parameters in their objective functions. A summary is shown in Tab.2. We do not regard easy-to-set hyper-parameters as sensitive ones, such as the number of iterations[2] in MPert. We empirically tune the sensitive hyper-parameters around their suggested values.

TABLE 2
Characteristics of compared methods. The easy-to-set hyper-parameters are not regraded as sensitive ones.

	number of sensitive hyper-parameters	number of iterations per explanation
DRE	-	1
MGnet	3	1
MPert	2	300
FInv	3	80
XPert	-	300
GCAM	-	1
GCAM++	-	1
VGrad	-	1
SMGrad	1	50
ITGrad	-	200

Implementation. For each of the above CNNs, we divide its convolutional layers into a few groups based on the resolutions of their outputs. We then introduce the three bottleneck cells at the intermediate positions to get the raw feature maps. We use varying channel numbers for different CNNs, aiming to propagate sufficient information between the cells while keeping efficiency. Specifically, for ResNet50 which consists of the intermediate layers with {256,512,1024} channels, we introduce one quarter channels for the high-to-low-resolution cells, namely {64,128,256}; for VGG19 and GoogleNet that contain the intermediate layers with {128,256,512} and {192,480,832} channels, we half the channel numbers for the corresponding cells, namely {64,128,256} and {96,240,416}, respectively. We use a two-stage scheme to train the relevance estimator. We first train class-related spatial filters based on the sampled images from the training set and then optimize other parts of the relevance estimator for 10 epochs. Of note, no ground truth is introduced, and only the outputs of the classifiers are utilized. We set the batch size to 64 and use Adam with the initialized learning rate of 10^{-2}. We apply the step decay and reduce the learning rate by half every three epochs. During the second stage, 50% background images are set to the Gaussian blurred version of raw images with the variance of 10, and the remaining ones are set to random color images with the addition of Gaussian noise. For a fair comparison, all perturbation-based methods apply the same strategy for adding perturbations.

Quantitative metrics. The faithfulness and the explainability of saliency maps are supposed to be evaluated based on the relevance scores of all pixels. Here we introduce two generalized metrics based on the ranking of pixels.

2. A larger iteration number empirically brings in better performance.

0162-8828 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Science & Technology of China. Downloaded on January 10,2021 at 02:28:15 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3049784, IEEE Transactions on Pattern Analysis and Machine Intelligence

TABLE 3
Ranking-based quantitative evaluation on faithfulness $\mathcal{M}_{\mathcal{F}}$.

	ImageNet				Birds-200-2011			
	ResNet50	VGG19	GoogleNet	MEAN	ResNet50	VGG19	GoogleNet	MEAN
DRE	77.13	76.75	69.24	74.37	84.62	85.25	82.13	84.00
MGnet	56.61	60.09	50.57	55.75	64.97	75.90	75.32	72.06
MPert	72.40	73.92	64.00	70.10	76.08	82.31	75.09	77.83
FInv	66.73	67.31	61.50	65.18	75.14	78.53	78.52	77.40
XPert	62.07	67.55	52.98	60.87	76.90	80.82	75.84	77.85
GCAM	70.13	64.56	67.84	67.51	77.67	80.49	79.43	79.20
GCAM++	68.34	69.96	62.51	66.94	78.42	79.75	78.34	78.84
VGrad	18.80	13.57	15.43	15.93	12.23	11.51	10.12	11.29
SMGrad	29.38	35.47	40.26	35.03	46.97	48.80	56.58	50.78
ITGrad	16.47	20.35	43.37	26.73	20.71	22.03	35.19	25.98

TABLE 4
Ranking-based quantitative evaluation on explainability $\mathcal{M}_{\mathcal{E}}$.

	ImageNet			
	ResNet50	VGG19	GoogleNet	MEAN
DRE	83.02	83.91	81.73	82.89
MGnet	82.62	83.53	81.87	82.67
MPert	75.74	72.50	70.98	73.07
FInv	75.20	72.63	75.35	74.39
XPert	78.73	80.92	71.26	76.97
GCAM	79.28	74.93	82.15	78.79
GCAM++	82.86	84.46	83.20	83.51
VGrad	66.23	70.90	66.78	67.97
SMGrad	73.17	74.03	70.31	72.50
ITGrad	66.92	66.74	63.25	65.64

Faithfulness. The traditional metrics use heuristic segmentation strategies on the scores of each mask and calculate the probability of the target class based on a fixed ratio of clean high-score pixels, which reduces the fairness for comparing different methods [8]. We instead utilize the ranking of pixels and perform the evaluation based on the target class probabilities corresponding to a number of ratios. Suppose Δ is the interval for the ratio of clean high-score pixels, and $\mathcal{S}_i$ denotes the set of locations with the top $i\times\Delta$ highest scores. For each image, we first estimate the probability of its fully blurred version as $Q_0{=}f_t(\psi(\mathbf{I},\mathbf{0}))$. Then we replace its blurred pixels within $\mathcal{S}_i$ by the corresponding $i\times\Delta$ pixels in the clean image and estimate the probability of the new image as Q_i. We repeat this step by increasing the ratio of clean pixels, until reaching the fully clean image and obtaining $Q_m{=}f_t(\psi(\mathbf{I},\mathbf{1}))$ $(m\times\Delta{=}1)$. With the intervals of Δ, the area under the curve (AUC) of the probability vs. the ratio is used as the measure of faithfulness:

$$\mathcal{M}_{\mathcal{F}} = 100\% \times \sum_i Q_i \cdot \Delta,\ i = 1, 2, \ldots, m. \tag{10}$$

Explainability. The explanation with high explainability should provide clear reasons that are easy to understand. Since it is time-consuming for users to detect the locations of all meaningful features (including bias features), we generally use bounding boxes as an alternative for its evaluation [6], [8]. For example, weakly-supervised object localization evaluates masks by calculating the intersection over the union between their binary variants and bounding boxes. Nevertheless, it faces the issue of choosing thresholds. Thus, we introduce a new metric by regarding the relevance scores as the results of retrieval tasks [1]. Specifically, let $\mathcal{S}_i$ be the set of locations that obtain the top i highest relevance scores, and $\mathcal{S}_b$ indicates the set of locations within the bounding box. We calculate the precision $P_i{=}\frac{|\mathcal{S}_b\cap\mathcal{S}_i|}{|\mathcal{S}_i|}$ as the fraction of these i locations retrieved within bounding boxes, and the recall $R_i{=}\frac{|\mathcal{S}_b\cap\mathcal{S}_i|}{|\mathcal{S}_b|}$ as the fraction of the within-bounding-box locations that are retrieved within these i locations. By computing P_i and R_i for all $\mathcal{S}_i$s, we can evaluate the explainability by the AUC of the precision vs. the recall as

$$\mathcal{M}_{\mathcal{E}} = 100\% \times \sum_i P_i \cdot (R_i - R_{i-1}). \tag{11}$$

4.2 On Object Recognition

This section investigates the effectiveness of the proposed method in object recognition tasks, which is the primary motivation of introducing right-skewed distributions. We use two real-world image datasets ImageNet and Birds-200-2011 for evaluation, where the latter is a fine-grained dataset of 200 bird species. In particular, we load pre-trained CNNs from torchvision for ImageNet and train ResNet50, VGG19, and GoogleNet to build the classifiers for Birds-200-2011.

4.2.1 *Ranking-based Quantitative Evaluation*

On the faithfulness with $\mathcal{M}_{\mathcal{F}}$. To evaluate the faithfulness of the proposed relevance estimator, we calculate the mean of the metric $\mathcal{M}_{\mathcal{F}}$ based on 10,000 and 2,000 sampled images for ImageNet and Birds-200-2011, respectively. We set $\Delta{=}\frac{1}{32}$ as the interval. Besides, we add a smoothed mask over the original one with a small weight. We introduce min-max normalization on $\mathcal{M}_{\mathcal{F}}$s of all methods for each image, balancing the effects of different images.

The results of the average $\mathcal{M}_{\mathcal{F}}$s are displayed in Tab.3, where the following observations can be obtained. *Firstly*, VGrad, SMGrad, and ITGrad are generally worse than the others with a large gap. It is understandable that, these methods search sensitive pixels based on the gradients, and the pixels with high scores will be discretely distributed in each image. As a result, it becomes harder for them to gather sufficient supporting information in a local receptive field and recover a high class probability. *Secondly*, GCAM obtains comparable performance to GCAM++ on average, and MPert that directly optimizes masks in a high resolution also brings satisfying faithfulness. *Thirdly*, DRE outperforms all the other methods and enjoys much better performance

0162-8828 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Science & Technology of China. Downloaded on January 10,2021 at 02:28:15 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3049784, IEEE Transactions on Pattern Analysis and Machine Intelligence

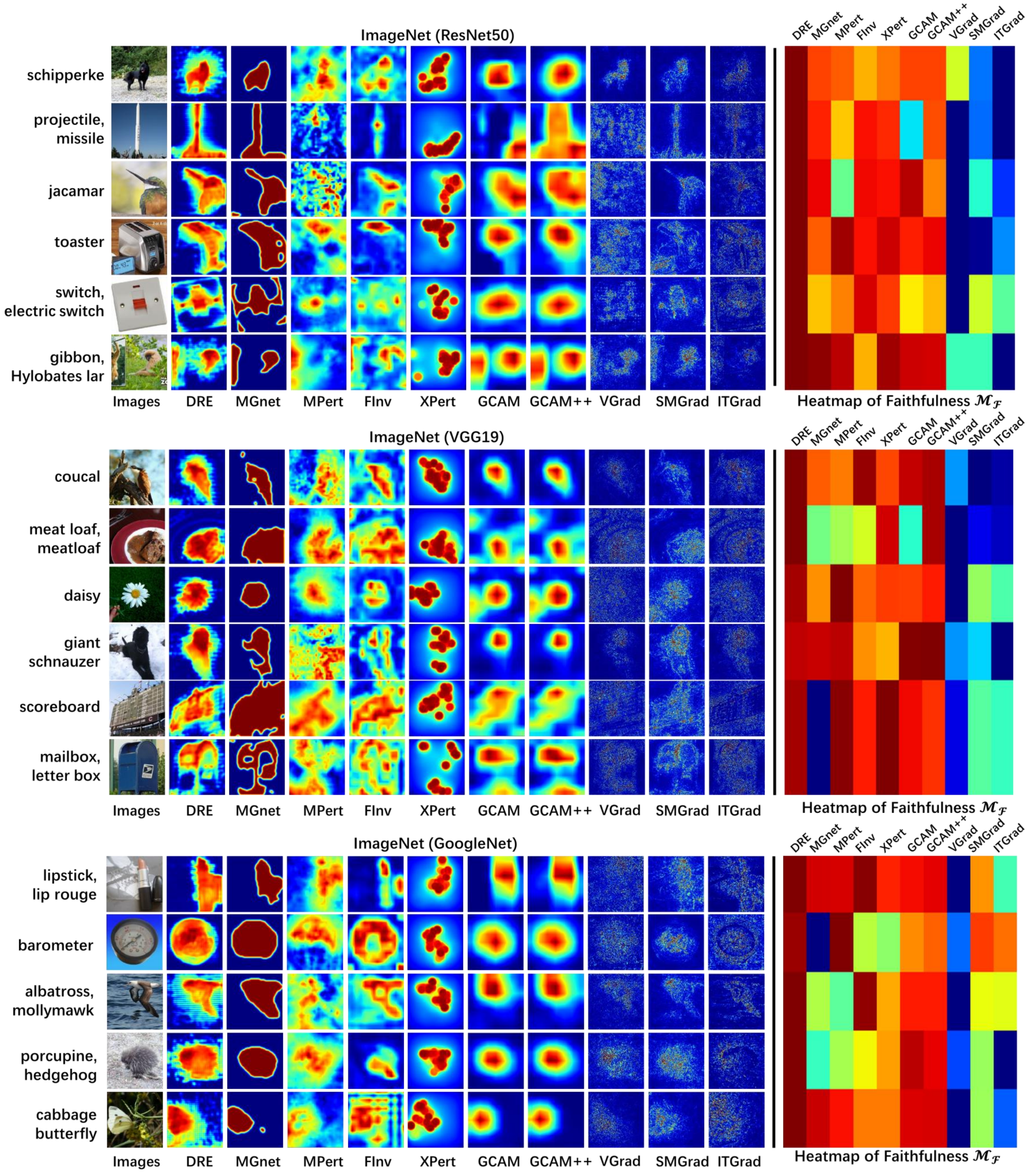


Fig. 7. The saliency maps of different explanation methods for the CNNs trained on ImageNet, in which these sampled images obtain correct classifications. We also display the heatmap of their normalized $\mathcal{M}_\mathcal{F}$ values.

0162-8828 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Science & Technology of China. Downloaded on January 10,2021 at 02:28:15 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3049784, IEEE Transactions on Pattern Analysis and Machine Intelligence

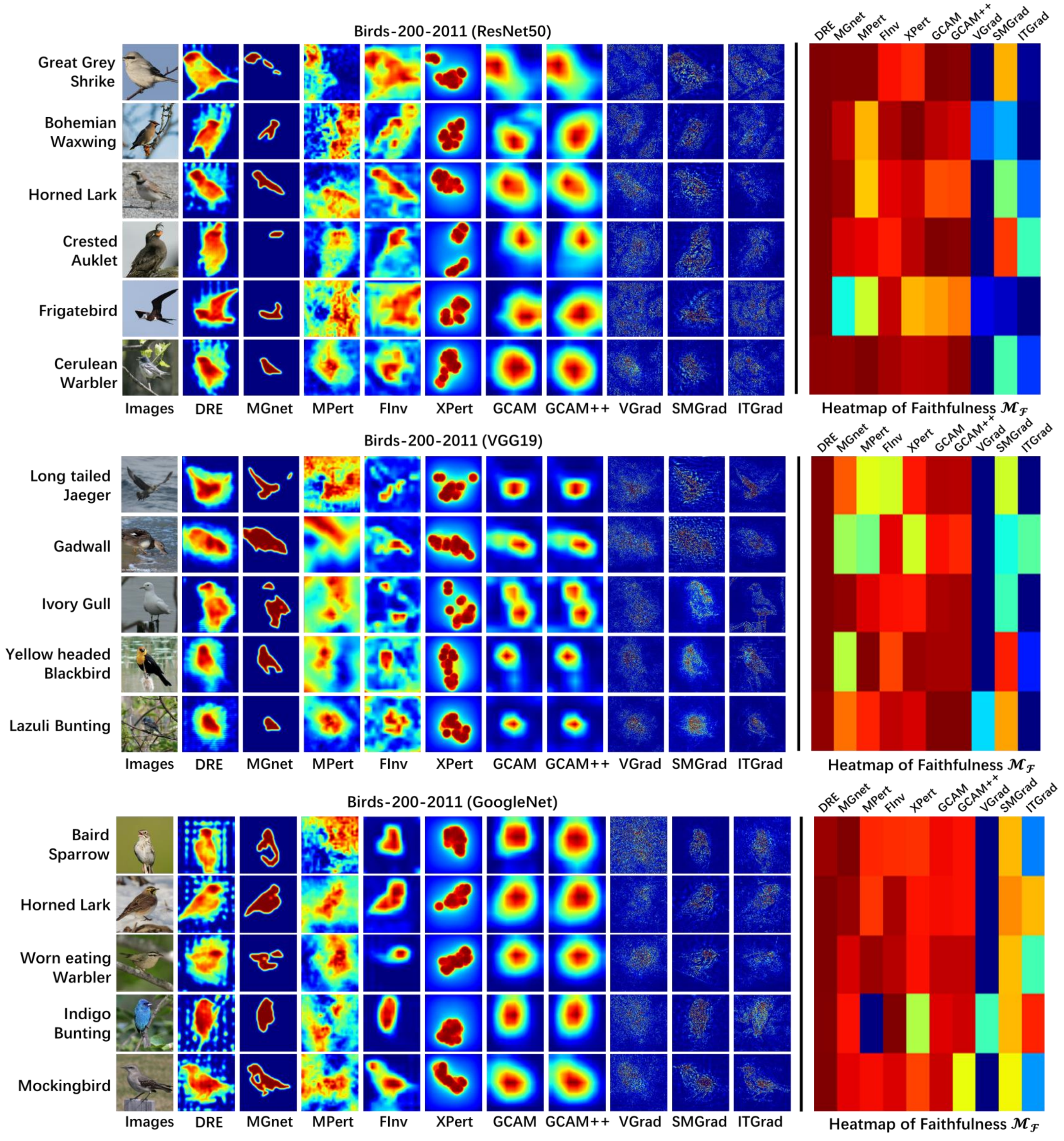


Fig. 8. The saliency maps of different explanation methods for the CNNs trained on Birds-200-2011, in which these sampled images obtain correct classifications. We also display the heatmap of their normalized $\mathcal{M}_{\mathcal{F}}$ values.

0162-8828 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Science & Technology of China. Downloaded on January 10,2021 at 02:28:15 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3049784, IEEE Transactions on Pattern Analysis and Machine Intelligence

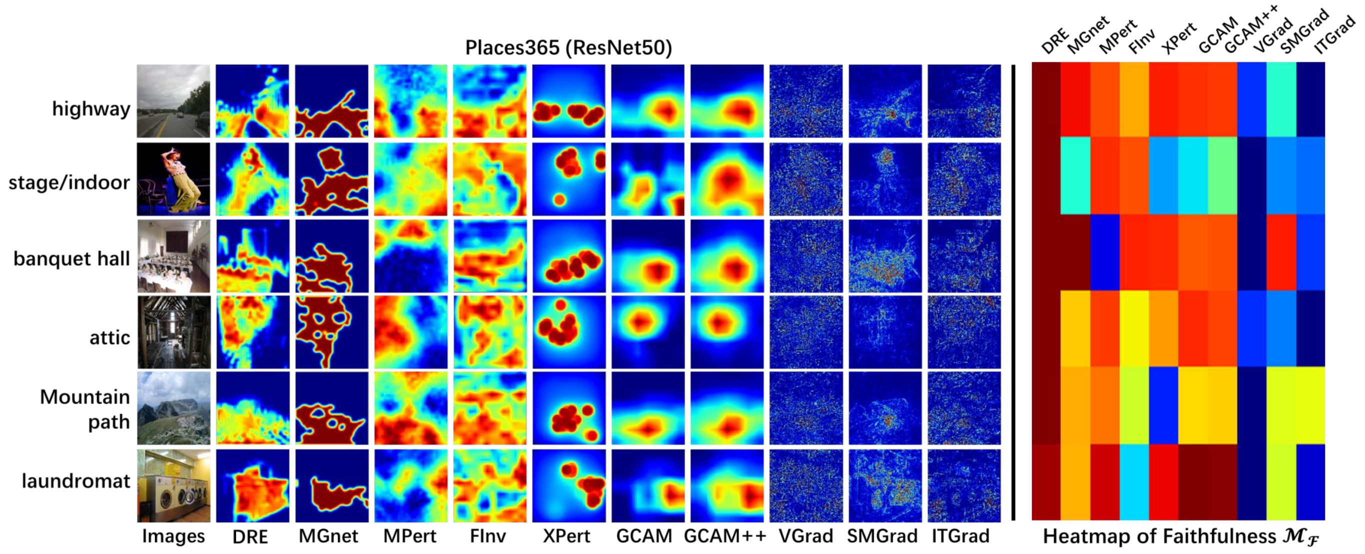


Fig. 9. The saliency maps of different explanation methods for the ResNet50 trained on Places365, in which these sampled images obtain correct classifications. We also display the heatmap of their normalized $\mathcal{M}_{\mathcal{F}}$ values.

than MGnet. Since the two methods apply the same architectures for their mask generators, the results demonstrate that guiding the relevance scores towards right-skewed distributions improves the ranking of supporting features.

On the explainability with $\mathcal{M}_{\mathcal{E}}$. To reveal the explainability, we introduce $\mathcal{M}_{\mathcal{E}}$ by evaluating the performance of object localization. For this, we resize and crop bounding boxes to the size of 224×224, leading to the same size of test images. The experiments are performed on 10,000 validation images of ImageNet with bounding box annotations.

The results of average $\mathcal{M}_{\mathcal{E}}$s are listed in Tab.4, where we obtain the following observations. *Firstly*, the last three gradient-based methods generally perform worse than the others. The possible reason is that, gradients are insensitive to the smooth supporting regions, which makes these regions ignored and harms the ranking of pixels. *Secondly*, GCAM++ obtains better performance than GCAM and the other methods. Understandably, the former is designed to detect multiple objects in the image and assign them high relevance scores. *Finally*, by replacing all constraints with a simple distribution controller, DRE outperforms MGnet with a small gap, and both of them enjoy better performance than most other methods. The reason is that, benefiting from the training with large-scale images, they tend to generate high relevance scores for the supporting features that are robust to the target class. If the bias features do not play the main role in the classification, the corresponding supporting regions prefer the locations inside the objects.

To sum up, DRE obtains comparable or even better explainability to others but achieves much higher faithfulness. Therefore, although we are not able to analyze the effects of bias features without more human intervention, the synthetic results still demonstrate its effectiveness empirically.

4.2.2 Visualization-based Qualitative Comparison

Below we visually compare different explanation methods based on their obtained saliency maps. The red and blue colors denote the high and low scores, respectively. We sample images from ImageNet and Birds-200-2011, and show their results in Fig.7 and Fig.8. In particular, we also display their normalized $\mathcal{M}_{\mathcal{F}}$ values via a heatmap, in which the color of each rectangle represents the faithfulness of the saliency map at the corresponding location.

From these results, we have the following observations. *Firstly*, the gradient-based methods bring more low relevance scores for the pixels insides the objects and high relevance scores for the pixels outside the objects. Considering their lower faithfulness, this observation implies that the supporting pixels generally locate inside the objects as expected, demonstrating the effectiveness of estimating the explainability with bounding boxes. *Secondly*, since GCAM and GCAM++ only take high-level feature maps into account, they fail to provide a detailed estimation of relevance scores. This issue becomes more obvious for fine-grained images, where they can only detect the locations of the objects. Besides, GCAM and GCAM++ may still miss a majority of supporting pixels of objects, such as the 4th row in Fig.8. *Thirdly*, MGnet is case-sensitive, which will either detect all supporting pixels or a few most important ones. Although we can carefully adjust the hyper-parameters for each kind of datasets and networks, the mask generators need to be re-trained for each setting, reducing its practicability significantly. Similarly, MPert and FInv contain sensitive hyper-parameters that need to be tuned for each instance individually. *Fourthly*, MGnet assigns the detected pixels similar relevance scores and loses the differentiation. We have attempted to perform the controller on its obtained masks in a post hoc manner, which still fails to build differentiated masks. The reason is that its relevance scores are already stacked at 0 and 1. *Finally*, DRE not only obtains high scores for the supporting pixels inside the objects and brings higher faithfulness, but also displays the different degrees of importance of these pixels, e.g., the supporting pixels within the animals' heads are more important than those within the other parts.

0162-8828 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Science & Technology of China. Downloaded on January 10,2021 at 02:28:15 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3049784, IEEE Transactions on Pattern Analysis and Machine Intelligence

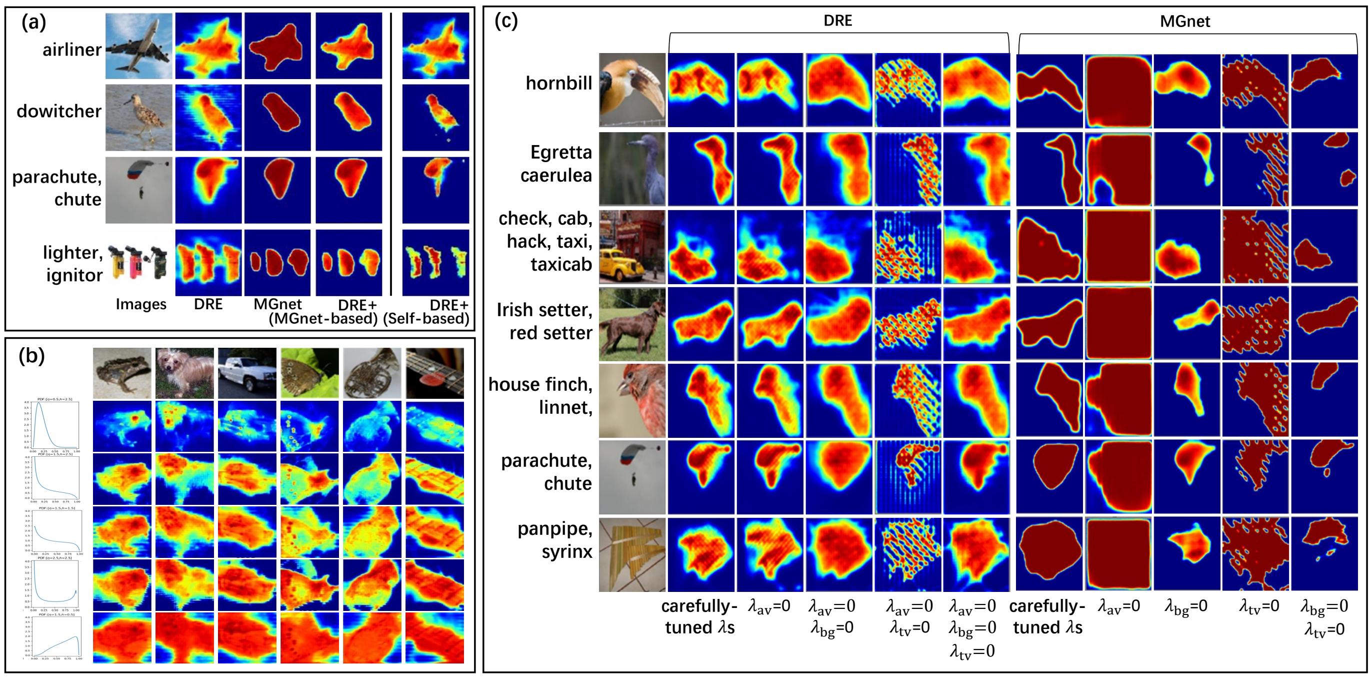


Fig. 10. The illustrative examples for Sec. 4.4, which are built upon ResNet50 with ImageNet. (a) The effects of post hoc tuning tricks on saliency maps. (b) The effects of different pre-configured distributions (the 1st column) on saliency maps (the 2nd-7th columns). Each row of masks corresponds to a pre-configured distribution. (c) The effects of the ad hoc constraints on DRE and MGnet [4].

TABLE 5
Ranking-based quantitative evaluation on faithfulness $\mathcal{M}_{\mathcal{F}}$.

Places365 (ResNet50)				
DRE	MGnet	MPert	FInv	XPert
70.45	53.85	64.61	64.84	55.71
GCAM	GCAM++	VGrad	SMGrad	ITGrad
68.60	60.89	7.73	38.07	45.20

4.3 On Scene Recognition

Now we introduce the explanation methods for the CNNs trained on scene images. Specifically, we first train ResNet50 on Places365 and then regard it as the black-box classifier. We list their faithfulness metric $\mathcal{M}_{\mathcal{F}}$ in Tab.5 and show some of their obtained saliency maps in Fig.9.

From these results, we obtain the following observations. *Firstly*, comparing to the masks for object images, the high scores of DRE and MGnet for scene images may locate at more than one region of an image. Understandably, scenes are composed of objects, and the conception of scenes is more comprehensive than that of objects. *Secondly*, although the masks of DRE tend to be visually noisy compared with GCAM and GCAM++, they still detect the important regions clearly and lead to high faithfulness. From the quantitative perspective, we also observe that DRE obtains higher faithfulness than all the other methods.

4.4 Discussion

In this section, we first introduce some simple tricks to further improve the proposed method. After that, since the right-skewed distributions are used and the ad hoc constraints are ignored, their impacts are investigated via targeted ablation studies. Finally, we discuss the saliency maps corresponding to misclassifications.

4.4.1 *On Improvements with Simple Tricks*

Although DRE has achieved stratifying performance without the non-trivial hyper-parameter tuning, constraining the scores of various images towards the same pre-configured distribution may lead to low but redundant scores on backgrounds, especially when the supporting pixels only take a tiny part of all pixels. Thus, we present two simple tricks to further improve the differentiation of DRE, including one self-based and one MGnet-based. For convenience, only ResNet50 is used as the classifier.

DRE+ (Self-based). This trick improves DRE based on the saliency map itself. Given an estimated mask with the relevance scores of pixels, we first follow the process in the faithfulness metric to iteratively calculate the probability Q_i based on the top $i\times\Delta$ pixels. Next, we modify these probabilities with Q_i=max $Q_{j\leq i}$ to obtain the monotonicity. For efficiency, we only calculate the probability Q_i at the $i\times\Delta$-th pixels and infer the probabilities at the remaining pixels with linear interpolation. Since the k-th pixel is not likely to be the supporting one if Q_k already approaches the maximum $Q_{\max}$ ($\max_k Q_k$), we perform the post hoc tuning on the relevance scores as M'_k=$M_k\times W_k$, where W_k=$(Q_{\max}-Q_k)$, and M_k is the original relevance score.

DRE+ (MGnet-based). It is natural to cooperate the proposed DRE with MGnet [4], since the latter can obtain clearer boundaries between objects and backgrounds. Thus, we introduce another mask as $\mathbf{M}' = \mathbf{M}_{\text{DRE}} \odot \mathbf{M}_{\text{MGnet}}$. For simplicity, we perform the post hoc combination without training them together with the shared mask generator.

The obtained masks showing the effects of DRE+ (Self-based) and DRE+ (MGnet-based) are displayed in Fig.10(a),

0162-8828 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Science & Technology of China. Downloaded on January 10,2021 at 02:28:15 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3049784, IEEE Transactions on Pattern Analysis and Machine Intelligence

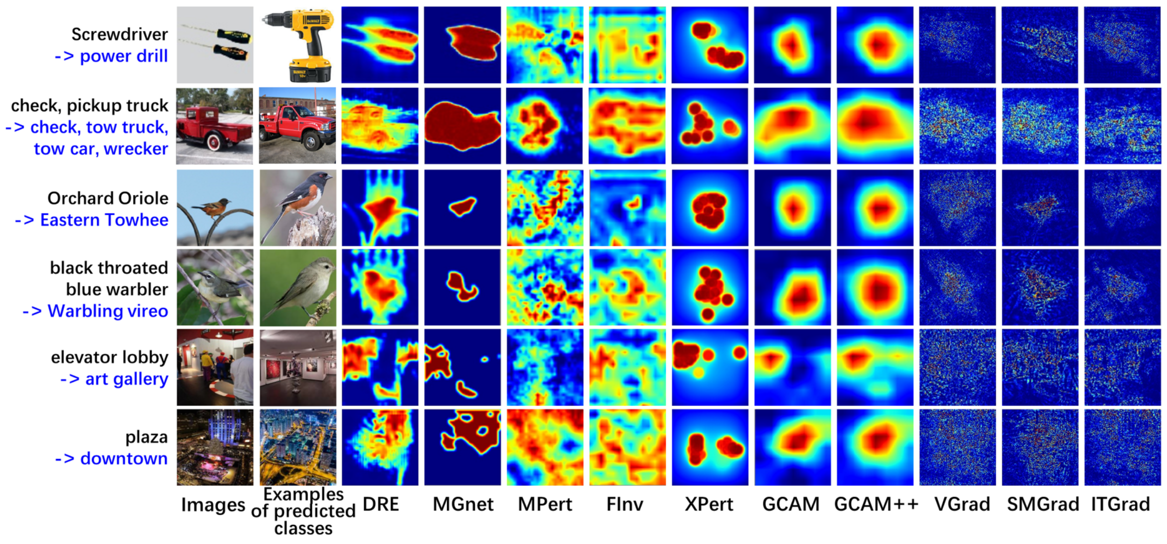


Fig. 11. The examples of the masks of different methods on ImageNet (1st-2nd rows), Birds-200-2011 (3rd-4th rows), and Places365 (5th-6th rows), where ResNet50 is used as the classifier. Of note, labels in black indicate the ground truths, and those in blue denote the predicted classes.

where the masks of original DRE and MGnet are also provided. From the results, the following observations can be made. *Firstly*, compared with MGnet, DRE without post hoc tuning sometimes sacrifices the boundaries between objects and backgrounds. *Secondly*, by combing the masks of DRE and MGnet, DRE+ (MGnet-based) improves the boundaries while keeping the differentiation to some degree. *Thirdly*, owing to the better ranking of the scores over all pixels (demonstrated by $\mathcal{M}_{\mathcal{F}}$ in Sec. 4.2.1), DRE+ (Self-based) can effectively utilize DRE's masks to reduce the noise around the boundaries and highlight the supporting pixels with differentiated scores. Since it does not introduce extra hyper-parameters, DRE+ (Self-based) is more practical. We further introduce flexible variants of self-based tuning in Appendix C, which could be more beneficial to DRE.

4.4.2 On Distribution Controller: the Effects of Distributions.

To show the effectiveness of the proposed controller, we first change the setting of its hyper-parameters, which leads the input with normal distributions to different types of pre-configured distributions. Specifically, we set (η,h) to (0.5,2.5), (1.5,2.5), (1.5,1.5), (2.5,2.5), (1.5,0.5) and the 5 corresponding distributions are shown in the 1st column in Fig.10(b). For convenience, we only use ResNet50 as the classifier and train the relevance estimators for 3 epochs. The examples of the corresponding saliency maps are shown in the following columns in the same figure.

As we can see, although it is hard to expect that all masks obtain relevance scores with the same distributions as the pre-configured ones, the controllers consistently enforce them towards these distributions. In general, the estimators built upon the right-skewed controllers obtain the differentiated masks, and the estimators built upon the left-skewed controllers reduce the explainability significantly.

4.4.3 On Optimization: the Effects of Ad Hoc Constraints.

To evaluate the impact of our simplified objective function, we introduce an ablation study to analyze how the ad hoc constraints affect the proposed relevance estimator. Following the formulation of MGnet with Eq.2, we add them back to Eq.9 and train the estimator using the following hyper-parameter settings: (1) carefully-tuned λs, (2) λ_{av}=0, (3) λ_{av}=λ_{bg}=0, (4) λ_{av}=λ_{tv}=0, (5) λ_{av}=λ_{bg}=λ_{tv}=0. For comparison, an ablation study is also performed on MGnet with the setting of (1) carefully-tuned λs, (2) λ_{av}=0, (3) λ_{bg}=0, (4) λ_{tv}=0, (5) λ_{bg}=λ_{tv}=0. For efficiency, we only use ResNet50 as the black-box classifier and train the corresponding estimators for 3 epochs. The masks corresponding to various settings are displayed in Fig.10(c).

From the results, the following observations can be obtained. *Firstly*, comparing the 6th column with the 2nd column, we can see that by adding all the constraints back, the masks of DRE can be improved to some degree. However, we also observe from the 3rd column that λ_{av}=0 has few effects on our relevance estimator. Besides, λ_{bg}=0 will increase the noise around the boundaries owing to the smoothness constraint (in the 4th column), and λ_{tv}=0 causes the holes in the masks (in the 5th column). Nevertheless, by further removing these two terms, these shortcomings can be alleviated, and the final masks of DRE remain the satisfying quality (in the 6th column). *Secondly*, all the constraints in MGnet, however, result in remarkable impacts on its masks, especially λ_{av}=0 (in the 8th column). Although λ_{bg}=0 can relax the masks and improve their differentiation, it would ignore a part of supporting features, such as the 4th-5th rows in the 9th column. Besides, it enhances the sensitiveness of λ_{av}. For example, while λ_{bg} is carefully set, λ_{av} varying within (2,12) consistently leads to acceptable results for ResNet50. Once λ_{bg}=0, the quality of masks is acceptable only for $\lambda_{\mathrm{av}}\in$(2,4). The reason is that, minimizing $f_t(\psi(\mathbf{I},\mathbf{1}\text{-}\mathbf{M}))$ can avoid the supporting pixels being regarded as backgrounds. Once this term is removed, the size of the supporting pixels is totally controlled by the hyper-parameter λ_{av}. When it is slightly larger, a part of supporting pixels will be regarded as backgrounds. In short, benefiting from the distribution controllers, DRE can

0162-8828 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Science & Technology of China. Downloaded on January 10,2021 at 02:28:15 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3049784, IEEE Transactions on Pattern Analysis and Machine Intelligence

be insensitive to the ad hoc constraints.

4.4.4 On Misclassifications

As the objective of explanation methods aims to explain all decisions, the explanations on misclassifications also need to be investigated. We thus show the masks of different methods on misclassified images. We take ResNet50 as an example and display the masks of the images from different datasets in Fig.11. As we can see, most explanation methods still target the typical parts of objects or the representative objects of scenes. It implies that different classes may share the same visual features, which leads to misclassifications. For example, the object images in the 1st-2nd rows share the similar shapes with the predicted classes, the bird images in the 3rd-4th rows share the similar colors with the wrong classes, and the scene images in the 5th-6th rows focus on the similar objects to the predicted classes. This observation is in line with the finding in [30], where different classes of scene images share the same object-level features.

5 Conclusion and Future Work

In this paper, we introduce a simple but effective relevance estimator called DRE to provide differentiated explanations for the decisions of DNNs. Specifically, we present the concept of distribution controllers on relevance scores and integrate it with a trainable mask generator to directly guide the relevance scores. By analyzing the effects of the skewness of the pre-configured distributions, we develop a simple distribution controller with the right-skewed distribution. We optimize DRE under the classification loss without non-trivial hyper-parameter tuning, which also improves the faithfulness of explanations. For each of the above innovations, we perform the targeted experiments to investigate their effectiveness. Finally, we compare DRE with state-of-the-art methods, and the experimental results demonstrate that DRE significantly improves faithfulness with high explainability.

There are some aspects needing further investigations. Firstly, although this paper provides an intuitive comparison of the transformed distributions for setting hyper-parameters, a quantitative analysis of the ratio of features at the tail is preferable. Secondly, since self-based tuning can improve saliency maps, it is worth incorporating it into the original models for an end-to-end optimization. Thirdly, benefiting from the simpleness of using distribution controllers, explaining the decisions of graph neural networks based on the controllers becomes another possible direction.

Appendix A
The proof of Eq.8 in Sec.3.4.2

Let $p(y)$ denote the probability density function (PDF) of the variable y, and the transformed variable z is calculated as $z = \varphi(y)$. According to the probability density transformation [9], the transformed PDF $p(z)$ can be obtained as

$$p(z) = p_y(\varphi_y^{-1}(z)) \cdot |\frac{\partial \varphi_y^{-1}(z)}{\partial z}|, \tag{12}$$

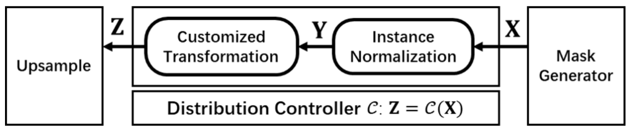

Upsample
Z
Customized Transformation
Y
Instance Normalization
X
Mask Generator
Distribution Controller $\mathcal{C}$: $\mathbf{Z} = \mathcal{C}(\mathbf{X})$

Fig. 12. The data flow inside the distribution controller with variables $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$.

where $\varphi_y^{-1}(z)$ denotes the inverse function of z on y, and $p_y(\varphi_y^{-1}(z))$ means substituting the above result into the PDF of y. Based on Eq.7 in the paper, we obtain

$$y = \varphi_y^{-1}(z) = -\frac{1}{\eta}\ln(z^{-1/h} - 1), \tag{13}$$

where ($z^{-1/h}$-1)>0. With simple derivations, we obtain:

$$|\frac{\partial \varphi_y^{-1}(z)}{\partial z}| = \frac{1}{\eta h} \cdot \frac{1}{z(1 - z^{(1/h)})}. \tag{14}$$

In addition, $p(y)$ follows the standard normal distribution, which can be formulated as

$$p(y) = \frac{1}{\sqrt{2\pi}} \mathrm{e}^{-\frac{y^2}{2}}. \tag{15}$$

By substituting Eq.13 into Eq.15, and then substituting Eqs.14-15 into Eq.12, we finally obtain

$$p(z) = \frac{1}{\sqrt{2\pi} h\eta} \cdot \frac{1}{z(1 - z^{1/h})} \cdot \mathrm{e}^{-\frac{(\ln(z^{(-1/h)} - 1))^2}{2\eta^2}}, \tag{16}$$

which completes the proof.

Appendix B
The Controller on Other Typical Distributions in Sec.3.4.3

Now we consider the effects of the distribution controller beyond normal distributions. Of note, $x \in \mathbf{X}$ denotes the input of the controller (output of the generator), $y \in \mathbf{Y}$ denotes the intermediate variable after instance normalization, and $z \in \mathbf{Z}$ denotes the output of the distribution controller. For convenience, we show the illustrative positions of these variables in Fig.12.

Recall that we have y=$\omega(x)$ via Eq.6 and z=$\varphi(y)$ via Eq.7. According to the probability density transformation [9], the transformed PDFs $p(y)$ and $p(z)$ can be obtained as

$$p(y) = p_x(\omega_x^{-1}(y)) \cdot |\frac{\partial \omega_x^{-1}(y)}{\partial y}| \tag{17}$$

and

$$p(z) = p_y(\varphi_y^{-1}(z)) \cdot |\frac{\partial \varphi_y^{-1}(z)}{\partial z}|, \tag{18}$$

where $\omega_x^{-1}(y)$ is the inverse function of y on x, and $\varphi_y^{-1}(z)$ is the inverse function of z on y. In particular, since $\omega(x)$ in Eq.6 denotes the instance normalization, we obtain

$$x = \omega_x^{-1}(y) = y\sqrt{\mathbb{V}[x]} + \mathbb{E}[x], \quad |\frac{\partial \omega_x^{-1}(y)}{\partial y}| = \sqrt{\mathbb{V}[x]}. \tag{19}$$

0162-8828 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Science & Technology of China. Downloaded on January 10,2021 at 02:28:15 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3049784, IEEE Transactions on Pattern Analysis and Machine Intelligence

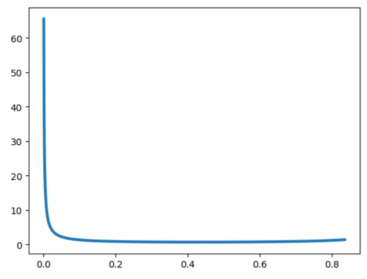


Fig. 13. The transformed PDF curve for inputs with uniform distributions.

B.1 On Uniform Distributions

Firstly, we consider the variable x with uniform distributions, whose PDF can be formulated as

$$p(x) = \begin{cases} \frac{1}{b-a}, & a < x < b \\ 0 & , \text{ others.} \end{cases} \tag{20}$$

For uniform distributions, we can calculate the expectation $\mathbb{E}[x]$ as $\frac{a+b}{2}$, and the variance $\mathbb{V}[x]$ as $\frac{(b-a)^2}{12}$. According to Eq.19, we obtain

$$x = \omega_x^{-1}(y) = \frac{b-a}{2\sqrt{3}} \times y + \frac{a+b}{2} \tag{21}$$

and

$$|\frac{\partial \omega_x^{-1}(y)}{\partial y}| = \frac{b-a}{2\sqrt{3}}. \tag{22}$$

By substituting Eqs.20-22 into Eq.17, $p(y)$ is obtained as

$$p(y) = \begin{cases} \frac{1}{2\sqrt{3}}, & a < \frac{b-a}{2\sqrt{3}} \times y + \frac{a+b}{2} < b \\ 0 & , \text{ others,} \end{cases} \tag{23}$$

which is equal to

$$p(y) = \begin{cases} \frac{1}{2\sqrt{3}}, & -\sqrt{3} < y < \sqrt{3} \\ 0 & , \text{ others.} \end{cases} \tag{24}$$

Furthermore, by substituting Eqs.13-14 and Eq.24 into Eq.18, we obtain $p(z)$ as

$$\begin{cases} \frac{1}{2\sqrt{3}\eta h z(1-z^{(1/h)})}, -\sqrt{3} < -\frac{1}{\eta}\ln(z^{-1/h}-1) < \sqrt{3} \\ \qquad 0 \quad , \text{others,} \end{cases} \tag{25}$$

which is equal to

$$\begin{cases} \frac{1}{2\sqrt{3}\eta h z(1-z^{(1/h)})}, (\mathrm{e}^{\sqrt{3}\eta}+1)^{-h} < z < (\mathrm{e}^{-\sqrt{3}\eta}+1)^{-h} \\ \qquad 0 \quad , \text{others.} \end{cases} \tag{26}$$

With η=1.5 and h=2.5 in Eq.26, we finally obtain the PDF curve of $p(z)$, as shown in Fig.13.

From this figure, we observe that $p(z)$ has a clear right tail over (0,1). Since we aim to obtain right-skewed distributions on the final output z, this observation demonstrates the effectiveness of our controller for the inputs with uniform distributions.

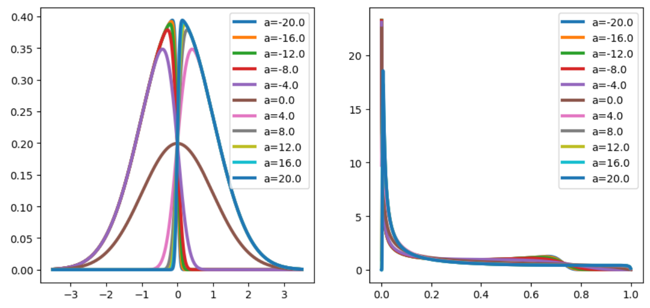


Fig. 14. The transformed PDF curves for inputs with skew normal distributions. The left shows the original distributions, and the right shows the transformed distributions.

B.2 On Skew Normal Distributions

Below we analyze the effects on skewed distributions. For simplicity, we focus on the skew normal distribution, which can be formulated as

$$p(x) = 2\phi(x)\Phi(ax), \tag{27}$$

where $\phi(x)$=$\frac{1}{\sqrt{2\pi}}\mathrm{e}^{(-\frac{x^2}{2})}$, $\Phi(x)$=$\int_{-\infty}^{x}\phi(t)\mathrm{d}t$=$\frac{1}{2}[1$+$\mathrm{erf}(\frac{x}{\sqrt{2}})]$ (erf denotes "error function"). Similarly, by substituting Eq.19 and Eq.27 into Eq.17, we can obtain $p(y)$:

$$2\phi(y\sqrt{\mathbb{V}[x]}+\mathbb{E}[x])\Phi(a(y\sqrt{\mathbb{V}[x]}+\mathbb{E}[x])) \times \sqrt{\mathbb{V}[x]}, \tag{28}$$

where we have $\mathbb{E}[x]$=$\frac{\sqrt{2}a}{\sqrt{(1+a)\pi}}$ and $\mathbb{V}[x]$=(1-$\frac{2a^2}{\pi(1+a^2)}$) for the skew normal distributions. By substituting Eqs.13-14 and Eq.28 into Eq.18, we obtain the transformed PDF $p(z)$:

$$\begin{aligned} 2\phi(\varphi_y^{-1}(z)\sqrt{\mathbb{V}[x]}+\mathbb{E}[x])\Phi(a(\varphi_y^{-1}(z)\sqrt{\mathbb{V}[x]}+\mathbb{E}[x])) \\ \times\sqrt{\mathbb{V}[x]}\times|\frac{\partial\varphi_y^{-1}(z)}{\partial z}|, \end{aligned} \tag{29}$$

where $\varphi_y^{-1}(z)$=$-\frac{1}{\eta}\ln(z^{-1/h}$-1) and $|\frac{\partial\varphi_y^{-1}(z)}{\partial z}|$=$\frac{1}{\eta h z(1-z^{(1/h)})}$. We substitute η=1.5 and h=2.5 into Eq.29 to obtain the final PDF. In particular, we change a from -20 to 20, and display their corresponding transformed PDF curves in Fig.14.

As we can see, for the different skewness of the original distributions (the right-skewed distribution with a>0 or the left-skewed distribution with a<0), their corresponding transformed distributions $p(z)$s have clear right tails over (0,1). As before, it demonstrates the effectiveness of the proposed controller for the inputs with skew normal distributions.

B.3 On Mixture of Normal Distributions

Now we consider the mixture of normal distributions. Suppose $\Psi(x,\sigma_i,\mu_i)$=$\frac{1}{\sqrt{2\pi}}\mathrm{e}^{-\frac{(x-\mu_i)^2}{2\sigma_i^2}}$ and r_i denotes the weight of the i-th component with $\sum_i r_i$=1. The PDF for x with the mixture of normal distributions is formulated as

$$p(x) = \sum_i r_i \Psi(x,\sigma_i,\mu_i). \tag{30}$$

By substituting Eq.19 and Eq.30 into Eq.17, we obtain

$$p(y) = \sum_i r_i \Psi(y\sqrt{\mathbb{V}[x]}+\mathbb{E}[x],\sigma_i,\mu_i)\sqrt{\mathbb{V}[x]}. \tag{31}$$

0162-8828 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Science & Technology of China. Downloaded on January 10,2021 at 02:28:15 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3049784, IEEE Transactions on Pattern Analysis and Machine Intelligence

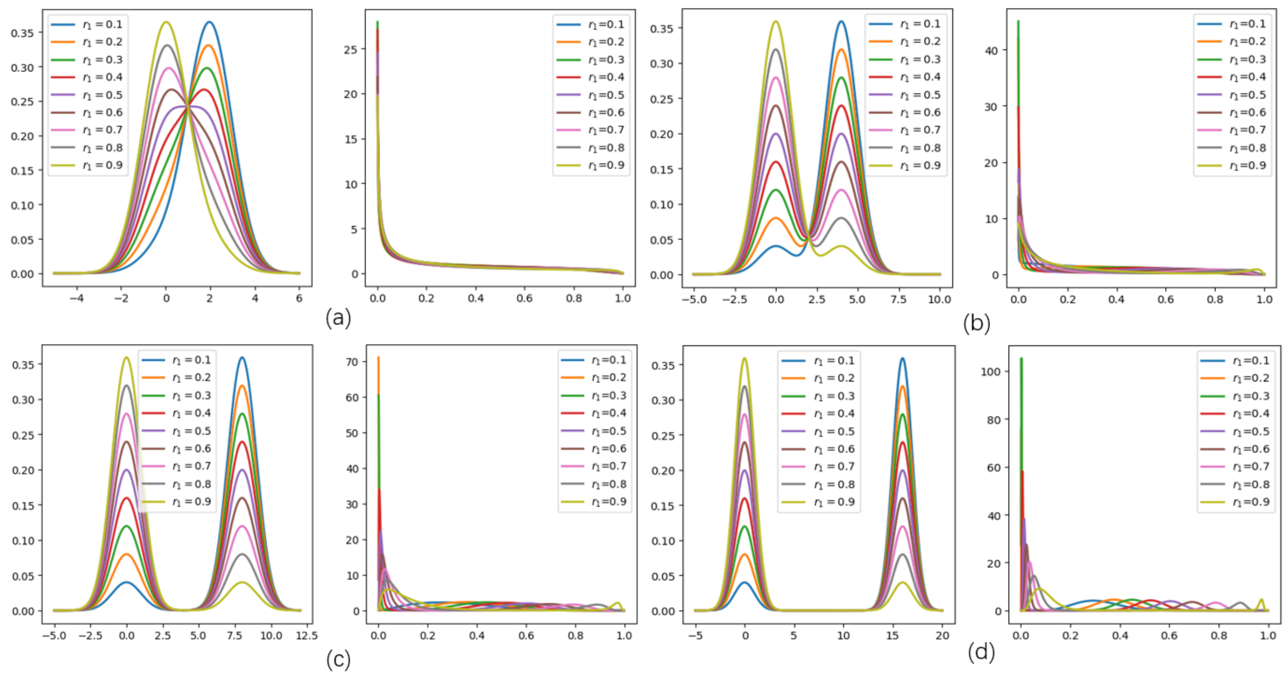


Fig. 15. The transformed PDF curves for inputs with the mixture of normal distributions. The left shows the original distributions, and the right shows the transformed distributions. Specifically, for these sub-figures, we have (a) $\mu_2 = 2$, (b) $\mu_2 = 4$, (c) $\mu_2 = 8$, (d) $\mu_2 = 16$.

In particular, we focus on the mixture of two normal distributions, where we have $\mathbb{E}(x)$=$r_1\mu_1$+$r_2\mu_2$ and $\mathbb{V}[x]$=$r_1\sigma_1^2$+$r_2\sigma_2^2$+$r_1r_2(\mu_1$-$\mu_2)^2$. Similarly, we substitute Eqs.13-14 and Eq.31 into Eq.18 and obtain $p(z)$ as

$$\sum_i r_i \Psi(\varphi_y^{-1}(z)\sqrt{\mathbb{V}[x]}+\mathbb{E}[x], \sigma_i, \mu_i)\sqrt{\mathbb{V}[x]}|\frac{\partial \varphi_y^{-1}(z)}{\partial z}|, \quad (32)$$

where $\varphi_y^{-1}(z)$=$-\frac{1}{\eta}\ln(z^{-1/h}$-1) and $|\frac{\partial \varphi_y^{-1}(z)}{\partial z}|$=$\frac{1}{\eta h z(1-z^{(1/h)})}$.

We set η=1.5 and h=2.5 in Eq.32 as before to build the final transformed PDF. For simplicity, we set σ_1=σ_2=1 and μ_1=0. We change r_1 within $\{0.1, 0.2, \ldots, 0.9\}$ and μ_2 within $\{2, 4, 8, 16\}$. The curves of the corresponding transformed PDFs are displayed in Fig.15. Of note, the mixtures of the above normal distributions with r_1=$\{0, 1\}$ or μ_2=0 equal to single normal distributions.

From this figure, we can obtain the following observations. Firstly, a large μ_2 brings a significant characteristic of bimodal distributions, namely two distinct peaks, which increases the difficulty of transformation. For example, with μ_2=16 and r_1=0.9, a part of values between (0.2,0.8) will be wasted due to the extremely low probabilities. However, if the two distributions are not too far away from each other (with respect to their variances), we observe that only a minority of the transformed scores are close to 1, and a majority of the scores are much lower and different. Thus, although this figure does not cover all kinds of mixtures of normal distributions, it still shows the effectiveness of our controller on the mixtures to some degree. Secondly, small r_1s result in the left tails for the original distributions. Benefiting from our controller, their transformed distributions are guided towards the right-skewed ones.

Appendix C
The variants of self-based tuning

In DRE+ (Self-based), we take advantage of the ranking of scores and use the accumulated class probabilities to improve masks. However, its post hoc trick is independent of the model training. Below we propose potential variants of the self-based tuning with an end-to-end optimization, which could probably help the optimization of the proposed relevance estimator and make it more adaptable. We leave the experimental investigations as future work.

Note that during the self-based tuning, we first estimate the extra weight W_k=$Q_{\max}$-Q_k for each pixel, where $Q_{\max}$ denotes the maximal probability and Q_k is the current accumulated probability. Then we combine it with the original score M_k for the final relevance score in the saliency map, represented as M'_k=$M_k$$\times$$W_k$. More details can be found in Sec. 4.4.1. Although the variables W_ks are untrainable for a fixed DNN, they keep being updated during the optimization of M_k. Therefore, we can directly introduce $\mathbf{M}'$ as the mask in Eq.9 for model training. Compared with the original version where all W_ks are equal to 1, it weakens the obtained relevance scores, especially the pixels within the background. Since the number of high scores is limited for each mask, this variant could enforce the relevance estimator to take more effort to detect important supporting pixels while ignoring the role of background, which further improves the differentiation. The potential issue is that calculating all W_ks for each image can lead to huge time costs in model training. Therefore, the trade-off between the performance and the training efficiency built upon the linear interpolation of W_ks needs further investigations. Further improvement can be made for the generation of W_ks. That is, to avoid the considerable time costs of estimating W_ks for testing images, we can predict them by learning to fit W_ks

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on January 10,2021 at 02:28:15 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3049784, IEEE Transactions on Pattern Analysis and Machine Intelligence

of training data. However, it inevitably introduces an extra hyper-parameter in the objective and requires more effort for sensitive analysis.

Acknowledgement

We are grateful to the associate editor and the reviewers for their great efforts in improving the quality of this paper. This work is partially supported by the National Nature Science Foundation of China under grants 61725203, 62020106007 and 61772171, the Fundamental Research Funds for the Central Universities under grants PA2020GDKC0023 and PA2019GDZC0095, and the China Scholarship Council.

References

[1] L. Azzopardi, P. Thomas, and A. Moffat, "cwl_eval: An evaluation tool for information retrieval," in *Proceedings of SIGIR*, 2019, pp. 1321–1324.
[2] A. Chattopadhay, A. Sarkar, P. Howlader, and V. N. Balasubramanian, "Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks," in *Proceedings of WACV*. IEEE, 2018, pp. 839–847.
[3] A. Clauset, C. R. Shalizi, and M. E. Newman, "Power-law distributions in empirical data," *SIAM review*, vol. 51, no. 4, pp. 661–703, 2009.
[4] P. Dabkowski and Y. Gal, "Real time image saliency for black box classifiers," in *Proceedings of NeurIPS*, 2017, pp. 6967–6976.
[5] D. P. Doane and L. E. Seward, "Measuring skewness: a forgotten statistic?" *Journal of statistics education*, vol. 19, no. 2, 2011.
[6] M. Du, N. Liu, Q. Song, and X. Hu, "Towards explanation of dnn-based prediction with guided feature inversion," in *Proceedings of SIGKDD*, 2018.
[7] R. Fong, M. Patrick, and A. Vedaldi, "Understanding deep networks via extremal perturbations and smooth masks," in *Proceedings of ICCV*, 2019, pp. 2950–2958.
[8] R. C. Fong and A. Vedaldi, "Interpretable explanations of black boxes by meaningful perturbation," in *Proceedings of ICCV*, 2017, pp. 3429–3437.
[9] C. Forbes, M. Evans, N. Hastings, and B. Peacock, "Statistical distributions," *John Wiley & Sons*, 2011.
[10] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *IEEE TNNLS*, vol. 28, no. 10, pp. 2222–2232, 2016.
[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of CVPR*, 2016, pp. 770–778.
[12] A. Hyvärinen and E. Oja, "Independent component analysis: algorithms and applications," *Neural networks*, vol. 13, no. 4-5, pp. 411–430, 2000.
[13] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of ICML*, 2015, pp. 448–456.
[14] B. Kulis *et al.*, "Metric learning: A survey," *Foundations and trends in machine learning*, vol. 5, no. 4, pp. 287–364, 2012.
[15] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proceedings of NeurIPS*, 2017, pp. 4765–4774.
[16] G. Montavon, W. Samek, and K.-R. Müller, "Methods for interpreting and understanding deep neural networks," *Digital Signal Processing*, vol. 73, pp. 1–15, 2018.
[17] M. T. Ribeiro, S. Singh, and C. Guestrin, "'why should i trust you?' explaining the predictions of any classifier," in *Proceedings of SIGKDD*, 2016, pp. 1135–1144.
[18] M. T. Ribeiro., S. Singh, and C. Guestrin, "Anchors: High-precision model-agnostic explanations," in *Proceedings of AAAI*, 2018.
[19] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of ICCV*, 2017, pp. 618–626.
[20] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *Proceedings of ICLR Workshop*, 2014.
[21] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
[22] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg, "Smoothgrad: removing noise by adding noise," *ICML Workshop*, 2017.
[23] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *Proceedings of ICLR Workshop*, 2014.
[24] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *Proceedings of ICML*. JMLR. org, 2017, pp. 3319–3328.
[25] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of CVPR*, 2015, pp. 1–9.
[26] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *arXiv preprint arXiv:1607.08022*, 2016.
[27] G. Waissi and D. F. Rossin, "A sigmoid approximation of the standard normal integral," *Applied Mathematics and Computation*, vol. 77, no. 1, pp. 91–95, 1996.
[28] J. Wu, M. Poloczek, A. G. Wilson, and P. Frazier, "Bayesian optimization with gradients," in *Proceedings of NeurIPS*, 2017, pp. 5267–5278.
[29] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks."
[30] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Object detectors emerge in deep scene cnns," *Proceedings of ICLR*, 2015.

Weijie Fu is currently working toward the PhD degree in the School of Computer and Information, Hefei University of Technology (HFUT). His current research interest focuses on data mining and interpretable machine learning.

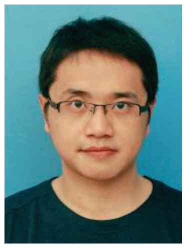
Meng Wang is a professor at the Hefei University of Technology, China. He received his B.E. degree and Ph.D. degree in the Special Class for the Gifted Young and the Department of Electronic Engineering and Information Science from the University of Science and Technology of China (USTC), Hefei, China, in 2003 and 2008, respectively. His current research interests include multimedia content analysis, computer vision, and pattern recognition. He has authored more than 200 book chapters, journal and conference papers in these areas. He is the recipient of the ACM SIGMM Rising Star Award 2014. He is an associate editor of IEEE Transactions on Knowledge and Data Engineering (IEEE TKDE), IEEE Transactions on Circuits and Systems for Video Technology (IEEE TCSVT), and IEEE Transactions on Neural Networks and Learning Systems (IEEE TNNLS).

Mengnan Du is currently a Ph.D. student at the Department of Computer Science and Engineering of Texas A&M University (TAMU). His research interests include interpretable machine learning and fairness in deep learning.

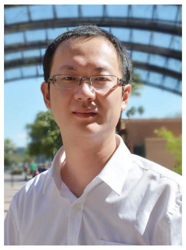
0162-8828 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Science & Technology of China. Downloaded on January 10,2021 at 02:28:15 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3049784, IEEE Transactions on Pattern Analysis and Machine Intelligence

Ninghao Liu is a Ph.D. student in Computer Science at the Department of Computer Science and Engineering of Texas A&M University (TAMU). His research interests include explainable artificial intelligence, network embedding and anomaly detection.



Shijie Hao is an associate professor at School of Computer Science and Information Engineering, Hefei University of Technology (HFUT). He is also with Key Laboratory of Knowledge Engineering with Big Data (Hefei University of technology), Ministry of Education. He received his Ph.D. degree at HFUT in 2012. His research interests include image processing and pattern recognition.



Xia Hu is an associate professor and Lynn '84 and Bill Crane '83 Faculty Fellow at Texas A&M University (TAMU) in the Department of Computer Science and Engineering. He directs the Data Analytics at Texas A&M (DATA) Lab, and has published over 100 papers in several major academic venues, including KDD, WWW, SIGIR, IJCAI, AAAI, etc. His papers have received several awards, including WWW 2019 Best Paper Shortlist, INFORMS 2019 Best Poster Award, WSDM 2013 Best Paper Shortlist, IJCAI 2017 BOOM workshop Best Paper Award. He is the recipient of JP Morgan AI Faculty Award, Adobe Data Science Award, NSF CAREER Award, and ASU President Award for Innovation. He was the conference General Co-Chair for WSDM 2020.

0162-8828 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Science & Technology of China. Downloaded on January 10,2021 at 02:28:15 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3015777, IEEE Transactions on Knowledge and Data Engineering

A Survey on Large-scale Machine Learning

Meng Wang, *Senior Member, IEEE,* Weijie Fu, Xiangnan He, Shijie Hao, and Xindong Wu, *Fellow, IEEE*

Abstract—Machine learning can provide deep insights into data, allowing machines to make high-quality predictions and having been widely used in real-world applications, such as text mining, visual classification, and recommender systems. However, most sophisticated machine learning approaches suffer from huge time costs when operating on large-scale data. This issue calls for the need of Large-scale Machine Learning (LML), which aims to learn patterns from big data with comparable performance efficiently. In this paper, we offer a systematic survey on existing LML methods to provide a blueprint for the future developments of this area. We first divide these LML methods according to the ways of improving the scalability: 1) model simplification on computational complexities, 2) optimization approximation on computational efficiency, and 3) computation parallelism on computational capabilities. Then we categorize the methods in each perspective according to their targeted scenarios and introduce representative methods in line with intrinsic strategies. Lastly, we analyze their limitations and discuss potential directions as well as open issues that are promising to address in the future.

Index Terms—large-scale machine learning, efficient machine learning, big data analysis, efficiency, survey

1 Introduction

Machine learning endows machines the intelligence to learn patterns from data, eliminating the need for manually discovering and encoding the patterns. Nevertheless, many effective machine learning methods face quadratic time complexities with respect to the number of training instances or model parameters [70]. With the rapidly increasing scale of data in recent years [207], these machine learning methods become overwhelmed and difficult to serve for real-world applications. To exploit the gold mines of big data, Large-scale Machine Learning (LML) is therefore proposed. It aims to address the general machine learning tasks on available computing resources, with a particular focus on dealing with large-scale data. LML can handle the tasks with nearly linear (or even lower) time complexities while obtaining comparable accuracies. Thus, it has become the core of big data analysis for actionable insights. For example, self-driving cars such as Waymo and Tesla Autopilot apply convolutional networks in computer vision to perceive their surroundings with real-time images [115]; online media and E-commerce sites such as Netflix and Amazon build efficient collaborative filtering models from users' histories to make product recommendations [18]. All in all, LML has been playing a vital and indispensable role in our daily lives.

Given the increasing demand for learning from big data, a systematic survey on this area becomes highly scientific and practical. Although some surveys have been published in the area of big data analysis [12], [33], [54], [193], they are less comprehensive in the following aspects. Firstly, most of them only concentrate on one perspective of LML and overlook the complementarity. It limits their values for understanding this area and promoting future developments. For example, [12] focuses on predictive models without covering the optimization, [33] reviews stochastic optimization algorithms while ignoring the parallelization, and [193] only pays attention to processing systems for big data and discusses the machine learning methods that the systems support. Secondly, most surveys either lose the insights into their reviewed methods or overlook the latest high-quality literature. For example, [12] lacks discussions on the computational complexities of the reviewed models, [33] neglects the optimization algorithms that address the data with high dimensionality, and [120] limits its investigation to distributed data analysis in the Hadoop ecosystem.

In this paper, we thoroughly review over 200 papers on LML from computational perspectives with more in-depth analysis and discuss future research directions. We provide practitioners lookup tables to choose predictive models, optimization algorithms, and processing systems based on their demands and resources. Besides, we offer researchers guidance to develop the next generation of LML more effectively with the insights of current strategies. We summarize the contributions as follows.

Firstly, we present a comprehensive overview of LML according to three computational perspectives. Specifically, it consists of: 1) model simplification, which reduces computational complexities by simplifying predictive models; 2) optimization approximation, which enhances computational efficiency by designing better optimization algorithms; and 3) computation parallelism, which improves computational capabilities by scheduling multiple computing devices.

Secondly, we provide an in-depth analysis of existing LML methods. To this end, we divide the methods in each perspective into finer categories according to targeted scenarios. We analyze their motivations and intrinsic strategies for accelerating the machine learning process. We then introduce the characteristics of representative achievements. In addition, we review the hybrid methods that jointly improve multiple perspectives for synergy effects.

Thirdly, we analyze the limitations of the LML methods in each perspective and present the potential directions

- *Meng Wang, Weijie Fu, Shijie Hao and Xindong Wu are with the Key Laboratory of Knowledge Engineering with Big Data (Hefei University of Technology), Ministry of Education, and the School of Computer Science and Information Engineering, Hefei University of Technology, Hefei, Anhui, 230601, China. E-mail: {eric.mengwang, fwj.edu, hfut.hsj}@gmail.com and xwu@hfut.edu.cn.*
- *Xiangnan He is with the University of Science and Technology of China, Hefei, Anhui, 230031, China. E-mail: xiangnanhe@gmail.com.*

1041-4347 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on August 12,2020 at 03:51:46 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3015777, IEEE Transactions on Knowledge and Data Engineering

based on their extensions. Besides, we discuss some open issues in related areas for the future development of LML.

The paper is organized as follows. We first present a general framework of machine learning in Sec.2, followed by a high-level discussion on its effectiveness and efficiency. In Sec.3, we comprehensively review state-of-the-art LML methods and provide in-depth insights into their benefits and limitations. Lastly, we discuss the future directions to address the limitations and other promising open issues in Sec.4, before concluding the paper in Sec.5.

2 From Effectiveness to Efficiency

In this section, we present three perspectives to quantify the efficiency based on an acknowledged error decomposition on effectiveness. Then we provide a brief of LML.

2.1 Overview of Machine Learning

Notations. We consider a general setting of machine learning tasks: given n instances $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ sampled from a d-dimension space, and $\{\mathbf{y}_i\}_{i=1}^{n_L}$ indicate to the labels of the first n_L instances ($n_L \geq 0$) within c distinct classes. The goal of machine learning is to learn an instance-to-label mapping model $f : \mathbf{x} \rightarrow \mathbf{y}$ from a family of functions $\mathcal{F}$, which can handle both existing and future data.

Effectiveness with error decomposition. Let $Q(f)_n$ denote the empirical risk of a machine learning model trained over n instances, and $Q(f)$ be the corresponding expected risk where the number of instances is infinite. Then we introduce the following specific functions obtained under different settings. Specifically, let f^*=$\mathrm{argmin}_f Q(f)$ be the optimal function that may not belong to the function family $\mathcal{F}$, and $f^*_{\mathcal{F}}$=$\mathrm{argmin}_{f\in\mathcal{F}} Q(f)$ be the optimal solution in $\mathcal{F}$. Suppose f_n=$\mathrm{argmin}_{f\in\mathcal{F}} Q(f)_n$ is the optimal solution that minimizes the empirical risk $Q(f)_n$, and $\tilde{f}_n$ refers to its approximation obtained by iterative optimization.

Let $T(\mathcal{F}, n, \rho)$ be the computational time for an expected tolerance ρ with $Q(\tilde{f}_n)_n$−$Q(f_n)_n$<ρ. Based on the above definitions, the excess error $\mathcal{E}$ obtained within an allotted time cost T_{max} can be decomposed into three terms [32]:

$$\mathrm{argmin}_{\mathcal{F},n,\rho}\ \mathcal{E}_{app} + \mathcal{E}_{est} + \mathcal{E}_{opt},\ \text{s.t.}\ T(\mathcal{F}, n, \rho) \leq T_{max} \quad (1)$$

where $\mathcal{E}_{app}$=$\mathbb{E}[Q(f^*_{\mathcal{F}})$−$Q(f^*)]$ denotes the approximation error, measuring how closely the functions in $\mathcal{F}$ can approximate the optimal solution beyond $\mathcal{F}$; $\mathcal{E}_{est}$=$\mathbb{E}[Q(f_n)-Q(f^*_{\mathcal{F}})]$ is the estimation error, which evaluates the effect of minimizing the empirical risk instead of the expected risk; $\mathcal{E}_{opt}$=$\mathbb{E}[Q(\tilde{f}_n)$−$Q(f_n)]$ indicates the optimization error, which measures the impact of the approximate optimization on the generalization performance.

Efficiency from three perspectives. Based on a low degree of reduction of the above decomposition, we show the three perspectives for improving machine learning efficiency.

Firstly, we focus on the effect of $\mathcal{E}_{app}$, which is predefined by the function family $\mathcal{F}$ of predictive models. By tuning the size of $\mathcal{F}$, we can make a trade-off between $\mathcal{E}_{app}$ and the computational complexity. Secondly, we consider the influence of $\mathcal{E}_{est}$. According to the probably approximately correct theory, the required number of instances for optimizing models in a large-size function family can be much larger [91]. To make use of available data and reduce the estimation error $\mathcal{E}_{est}$, we suppose all n instances are traversed at least once during optimization. Thus, we ignore $\mathcal{E}_{est}$ and the factor n. Thirdly, we pay attention to $\mathcal{E}_{opt}$, which is affiliated with optimization algorithms and processing systems. Specifically, the algorithms play a crucial role in computational efficiency, which attempts to increase the reduction in the optimization error per computation unit. The processing systems determine the computational capacity based on the hardware for computing and the software for scheduling. With a powerful system, plenty of iterations in optimization can be performed within allotted time costs.

Above all, the machine learning efficiency can be improved from three perspectives, including 1) the computational complexities of predictive models, 2) the computational efficiency of optimization algorithms, and 3) the computational capabilities of processing systems.

2.2 Brief of Large-scale Machine Learning

Now we present a brief of LML based on the above analysis.

1) Reduce the computational complexity based on model simplification. Machine learning models can be optimized based on matrix operations, which generally take cubic computational complexities for square matrices. Besides, most non-linear methods require extra quadratic complexities for estimating the similarity between pairs of instances to formulate their models. If these models are simplified by being constructed and optimized with smaller or sparser involved matrices, we can reduce their complexities significantly.

2) Improve the computational efficiency based on optimization approximation. Gradient descent algorithms cannot always compensate for their huge computations by selecting larger learning rates [33]. Therefore, a more effective manner is splitting data or parameters into multiple subsets and then updating models with these small subsets. As these approximate algorithms can obtain relatively reliable gradients with fewer computations, the reduction in the optimization error can be increased per computation unit.

3) Enhance the computational capacity based on computation parallelism. During the procedure of model construction and optimization, a high number of computation-intensive operations can be performed simultaneously, such as the calculations that are repeated on different mini-batches. Built upon parallel processing systems, we can break up these intensive computations and accomplish them on multiple computing devices with a shorter runtime.

3 Review on Large-scale Machine Learning

In this section, we review LML in detail. Specifically, we present the methods in the above three perspectives in Sec.3.1-Sec.3.3, and discuss their collaboration in Sec.3.4. For each part, we categorize the related methods according to targeted scenarios and introduce the methods based on their intrinsic strategies. We also provide experimental evidence to demonstrate the effectiveness of these strategies and summarize their pros and cons. For convenience, Fig.1 provides a coarse-to-fine overview for the structure of this section.

1041-4347 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on August 12,2020 at 03:51:46 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3015777, IEEE Transactions on Knowledge and Data Engineering

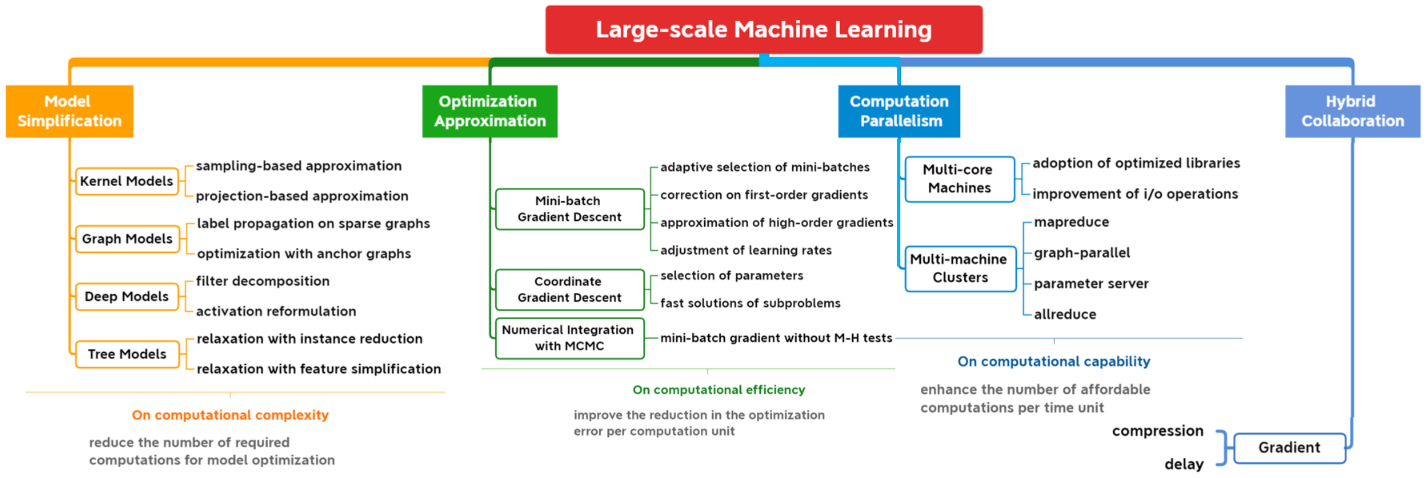

Large-scale Machine Learning
Model Simplification
Optimization Approximation
Computation Parallelism
Hybrid Collaboration
Kernel Models
sampling-based approximation
projection-based approximation
Graph Models
label propagation on sparse graphs
optimization with anchor graphs
Deep Models
filter decomposition
activation reformulation
Tree Models
relaxation with instance reduction
relaxation with feature simplification
On computational complexity
reduce the number of required computations for model optimization
Mini-batch Gradient Descent
adaptive selection of mini-batches
correction on first-order gradients
approximation of high-order gradients
adjustment of learning rates
Coordinate Gradient Descent
selection of parameters
fast solutions of subproblems
Numerical Integration with MCMC
mini-batch gradient without M-H tests
On computational efficiency
improve the reduction in the optimization error per computation unit
Multi-core Machines
adoption of optimized libraries
improvement of i/o operations
Multi-machine Clusters
mapreduce
graph-parallel
parameter server
allreduce
On computational capability
enhance the number of affordable computations per time unit
compression
delay
Gradient

Fig. 1. The framework of Sec.3, which shows the ways of scaling up machine learning to large-scale machine learning from three perspectives.

TABLE 1
A brief lookup table for LML methods based on Model Simplification.

Categories	Strategies	Representative Methods
Kernel-based Models	sampling-based approximation	uniformly sampling [118], [214], incremental sampling [34], [73].
	projection-based approximation	Gaussian [139], orthonormal transforms [215], ramdom features [134], [161].
Graph-based Models	label propagation on sparse graph	approximate search [110], [227], [232], division and conquer [45], [197].
	optimization with anchor graph	single layer [132], [201], [228], hierarchical layers [77], [200].
Deep Models	filter decomposition	on channels [97], [117], [181], [189], on spatial fields [182], [190], [218].
	activation reformulation	at hidden layers [44], [128], [137], [149], at the output layer [143], [144].
Tree-based Models	relaxation with instance reduction	random sampling [76], sparse-based [52], importance-based [112].
	relaxation with feature simplification	random sampling [36], histogram-based [21], [52], exclusiveness-based [112].

3.1 Model Simplification

Model simplification methods improve machine learning efficiency from the perspective of computational complexities. To reformulate predictive models and lower computational complexities, researchers introduce reliable domain knowledge for a small $\mathcal{E}_{app}$, such as the structures or distributions of instances and the objectives of learning tasks [24]. According to targeted scenarios, the reviewed scalable predictive models consist of four categories, including kernel-based, graph-based, deep-learning-based, and tree-based. For convenience, a brief overview is provided in Tab.1.

3.1.1 For Kernel-based Models

Kernel methods play a central role in machine learning and have demonstrated huge success in modeling real-world data with highly complex, nonlinear distributions [146]. The key element of these methods is to project instances into a kernel-induced Hilbert space $\phi(\mathbf{x})$, where dot products between instances can be computed equivalently through the kernel evaluation as $K_{ij}{=}{<}\phi(\mathbf{x}_i), \phi(\mathbf{x}_j){>}$.

Motivation. Given n instances, the computational complexity of constructing a kernel matrix $\mathbf{K}{\in}\mathbb{R}^{n\times n}$ scales as $O(n^2d)$. Supposing all instances are labeled with a label matrix of $\mathbf{Y}{\in}\mathbb{R}^{n\times c}$ ($n{=}n_L$), most kernel-based methods can be solved based on matrix inversion as

$$(\mathbf{K}+\sigma\mathbf{I})^{-1}\mathbf{Y}, \tag{2}$$

which requires a computational complexity of $O(n^3+n^2c)$. Examples include the Gaussian process, kernel ridge regression, and least-square support vector machine [81].

To alleviate the above computational burdens for a large n, a powerful solution is performing low-rank approximation based on SPSD sketching models [83] and solving the matrix inversion with Woodbury matrix identity [94]. Specifically, let $\mathbf{S}\in\mathbb{R}^{n\times m}$ with $n\geq m$ indicate the sketching matrix. Take $\mathbf{C}=\mathbf{KS}$, and $\mathbf{W}=\mathbf{S}^{\mathrm{T}}\mathbf{KS}$. Then $\mathbf{CW}^{\dagger}\mathbf{C}^{\mathrm{T}}$ becomes a low-rank approximation to $\mathbf{K}$ with the rank at most m. Based on the matrix inversion lemma [94], Eq.2 can be rewritten into $\frac{1}{\sigma}[\mathbf{Y}-\mathbf{C}(\sigma\mathbf{W}+\mathbf{C}^{\mathrm{T}}\mathbf{C})^{-1}(\mathbf{C}^{\mathrm{T}}\mathbf{Y})]$, which reduces the computational complexity to $O(m^3{+}nmc)$.

For scaling up kernel-based models without hurting the performance, the choice of $\mathbf{S}$ or the efficient construction of $\mathbf{C}$ becomes crucial. Below we introduce two strategies for low-rank approximation based on sketching matrices.

Sampling-based approximation. With this strategy, $\mathbf{S}$ represents a sparse matrix that contains one nonzero in each column. A direct method is sampling columns of kernel matrices at random with replacement [118], [214], which is equivalent to the Nyström approximation. By assuming that potential clusters are convex, one can select columns corresponding to kmeans centers with the cost of $O(nmdt)$ [228], where t is the number of iterations. To weigh the complexity and the performance, a few incremental sampling methods are proposed. At each iteration, these methods first randomly sample a subset of columns and then pick the column either with the smallest variance of the similarity matrix between the sampled columns and the remaining ones or the lowest sum of squared similarity between selected ones [34], [73], [229]. In general, the computational complexities of these methods scale as $O(nmp)$, where p is the size of

1041-4347 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on August 12,2020 at 03:51:46 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3015777, IEEE Transactions on Knowledge and Data Engineering

the subset. Besides, the structures of clustered blocks can be exploited to reduce storage costs [180].

Projection-based approximation. Based on this strategy, $\mathbf{S}$ is a dense matrix, which consists of random linear combinations of all columns of kernel matrices [90]. For example, one can introduce Gaussian distributions to build a data-independent random projection [139]. Besides, the sketch can be improved with orthonormal columns that span uniformly random subspaces. To do this, we can randomly sample and then rescale the rows of a fixed orthonormal matrix, such as Fourier transform matrices, and Hadamard matrices [215]. These orthonormal sketches can additionally speed up the matrix product to $O(n^2\text{log}m)$, as opposed to $O(n^2m)$ required for the same operation with general dense sketches [215]. On the other hand, some methods construct kernel spaces directly by mapping data non-linearly to new low-dimension subspaces [134], [148], [161].

Discussion based on experimental results. Benefit from matrix identity lemma on the low-rank approximation, time costs can be significantly reduced. For example, [210] sped up the matrix inversion nearly 100+ times without scarifying the accuracy for COREL images. Below we discuss the choice of the above methods in practical applications.

Firstly, for sampling-based methods, uniform sampling is the fastest solution. Although its performance is generally worse than others with a small m, it can be mitigated with the increase of m [118]. Besides, the effectiveness and efficiency of incremental sampling are weighed based on q. However, its time cost could be 50% lower than kmeans, especially when the selected number of columns was small [73]. Secondly, for projection-based methods, Gaussian matrices lead to comparable or better performance than orthonormal sketches [83], [215]. Thirdly, projection-based sketches are consistently better than uniformly sampling, while the latter outperforms naive random features [134], [214]. Finally, since sampling-based methods only need to compute the involved similarity, the computation of a full kernel matrix is avoided. However, if data comes in streaming with varying distributions, these methods are no longer suitable [134]. In contrast, random features can be solved in the primal form through fast linear solvers, thereby enabling to handle large-scale data with acceptable performance [161], [186].

3.1.2 For Graph-based Models

Graph-based methods define a graph where the nodes represent instances in the dataset, and the weighted edges reflect their similarity. For classification tasks, their underlying assumption is label smoothness over the graph [146].

Motivation. Given n instances, a graph is first constructed to estimate the similarity between all instances, measured as W_{ij}=$\text{RBF}(x_i, x_j)$. Then graph-based models constrain the labels of nearby instances to be similar and the predicted labels towards the ground truths. Let $\mathbf{Y}$ be the label matrix where only n_L rows contain nonzero elements. Consequently, the soft label matrix $\mathbf{F}$ can be solved by

$$\mathbf{F} = (\mathbf{I} + \alpha\mathbf{L})^{-1}\mathbf{Y}, \tag{3}$$

where $\mathbf{L}$=$\text{diag}(\mathbf{1}^{\text{T}}\mathbf{W})$-$\mathbf{W}$ denotes the graph Laplacian matrix. The computational cost of graph models comes from two aspects: the graph construction with the cost of $O(n^2d)$ and the matrix inversion with the cost of $O(n^3)$. Thus, there are two types of strategies for improving scalability, including label propagation on sparse graphs and optimization with anchor graphs.

Label propagation on sparse graphs. Unlike kernels, most graphs prefer the sparse similarity, which has much less spurious connections between dissimilar points [132]. Thus, we can conduct iterative label propagation to accelerate the spread of labels, represented as $\mathbf{F}^{t+1}$=$\alpha(\mathbf{W})\mathbf{F}^t$+$(1$-$\alpha)\mathbf{Y}$, where $\mathbf{F}^0$=$\mathbf{Y}$. Let k denote the average nonzero element in each row of $\mathbf{W}$. Then the number of necessary computations in each iteration scales as $O(nkC)$, taking up a tiny part of the original amount $O(n^2C)$. As a result, a huge part of the computational complexity now comes from graph construction, which can be efficiently solved based on approximate sparse graph construction.

These graph construction methods introduce a hierarchical division on datasets to find the neighbors of each instance and estimate their similarity, which reduces the cost to $O(n\log(n)d)$. For example, approximate nearest neighbor search (ANNS) first builds a structured index with all instances [110] and then searches the approximate neighbors of each instance on the obtained index, such as hierarchical trees [200] and hashing tables [199], [227]. To enhance the quality, we can repeat the above procedure to generate multiple basic graphs and combine them to yield a high-quality one [232]. Besides, as sparse graph construction is much easier than the nearest neighbor search, divide-and-conquer methods become more popular. These methods first divide all instances into two or three overlapped subsets and then unite the sub-graphs constructed from these subsets multiple times with neighbor propagation [45], [197].

Optimization with anchor graphs. Anchor graphs sample m instances as anchors and measure the similarity between all instances and these anchors as $\mathbf{Z} \in \mathbb{R}^{n\times m}$. Then the labels of instances are inferred from these anchors, leading to a small set of to-be-optimize parameters. They use anchors as transition nodes to build the similarity between instances for label smoothness, and their soft label matrix can be obtained as $\mathbf{F}$=$\mathbf{Z}(\mathbf{Z}^{\text{T}}\mathbf{Z}$+$\alpha\tilde{\mathbf{L}})^{-1}\mathbf{Z}^{\text{T}}\mathbf{Y}$, where $\tilde{\mathbf{L}}$$\in$$\mathbb{R}^{m\times m}$ is a reduced Laplacian over anchors. Clearly, the cost of graph construction is reduced to $O(nmd)$ or even $O(nd\log(m))$ with ANNS, and the cost of optimization scales as $O(nm^2 + m^3)$.

The original anchor graph models introduce sparse adjacency between instances and anchors [132]. After that, hierarchical anchor graphs propose to retain sparse similarities over all instances while keeping a small number of anchors for label inference [200]. In case that the smallest set of anchors still needs to be large and brings considerable computations, FLAG developes label optimizers for further acceleration [77]. Besides, EAGR proposes to perform label smoothness over anchors with pruned adjacency [201].

Discussion based on experimental results. Now we show the advantages of the two classes of methods. On the one hand, label propagation is more efficient than the original matrix inversion. Without accuracy reduction, it could obtain $10\times$ to $100\times$ acceleration over the matrix inversion [79]. As for approximate graph construction, although hierarchical trees and hashing tables are easy to be used for ANNS

1041-4347 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on August 12,2020 at 03:51:46 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3015777, IEEE Transactions on Knowledge and Data Engineering

[198], they generally ignore the fact that each query must be one of the instances in the graph construction. Thus, they put redundancy efforts on giving a good result for unnecessary future queries. In contrast, divide-and-conquer methods can obtain higher consistency to the exact graphs with less time cost, which was demonstrated on both Caltech101, Imagenet, and TinyImage [45]. On the other hand, although the accuracy of anchor graphs is lightly worse than sparse graphs [132], they are more potent on handling very large datasets. The reason is that, once a set of anchors can be stored in the memory, anchor graphs can be efficiently constructed with the memory cost of $O(md+nk)$ rather than $O(nd)$. For example, with hierarchical anchor graphs, the classification on 8 million instances could be implemented on a personal computer within 2 mins [77], [200].

3.1.3 For Deep Models

Deep models introduce layered architectures for data representation [66]. Instead of fully-connected networks (FCNs) where any pair of input features in each grid-like data are relevant, convolutional neural networks (CNNs) and recurrent neural networks (RNNs) utilize the structures of data by using small-size filters on local receptive fields [117].

Motivation. Take CNN as an example, where each layer of convolution kernels can be viewed as a 4D tensor. Consider input feature maps with the spatial size of $d_I \times d_I$ and the channel number of m_I. A standard convolutional layer on the input can be parameterized by convolution kernels with the size of $d_K \times d_K \times m_I \times m_O$, where d_K is the spatial dimension of the square kernel and m_O is the number of output channels. Suppose the stride equals 1. The computational complexity of this convolutional layer becomes

$$d_I \times d_I \times d_K \times d_K \times m_I \times m_O. \tag{4}$$

Generally, deep models involve a large number of hidden layers. Although the sizes of filters are much smaller than the sizes of input features, CNNs still contain millions of parameters [117], and the convolutions contribute to the bulk of most computations. Besides, deep models contain two types of bounded activation functions, including those used in hidden layers for feature extraction, and the ones used at output layers for probability prediction. The former type is used for all neurons, and the latter type involves normalization operations. When handling big models or a large number of output neurons, e.g., NLP tasks, both can lead to huge computations during back-propagation [141].

To address these issues, we introduce the filter decomposition for simplifying convolution layers and improving the overall speedup. After that, we review efficient activation functions where gradients can be easily estimated.

Filter decomposition. Filter decomposition is derived by the intuition that there is a significant amount of redundancy in 4D tensors. Thus, some methods reduce computations by gathering information from different channels hierarchically. For example, Alexnet develops group convolutions to avoid the computations between two groups of channels [117]. Mobilenet introduces separable depthwise convolutions [97], which factorize the filters into purely spatial convolutions followed by a pointwise convolution along with the depth variable [181], leading to 70% reduction of parameters. Inception v1 in turn first implements the dimensionality reduction on the channels of input features [189], followed by the filters with the original receptive fields. After that, Shufflenet generalizes group convolutions and depthwise convolutions based on the channel shuffle to further reduce the redundancy [231].

One the other hand, some methods reduce the computations based on the hierarchical information integration over the spatial side. For example, VGG replaces one layer of large-size filters with the two layers of smaller-size filters [182], reducing nearly 28% computations. Besides, Inception v3 introduces asymmetric convolutions that decompose the convolution with 3×3 filters into two cascaded ones with the spatial dimensions of 1×3 and 3×1 [190], and nearly 33% parameters can be saved. Dilated convolutions further support the exponential expansion of receptive fields without the loss of resolution [218]. It shows that a 7×7 receptive field could be explored by two dilated convolutions with 3×3 parameters, reducing 80% computations.

Activation reformulation. The bounded activation functions such as sigmoid and tanh bring expensive exponential operations at each neuron. Besides, both of them face vanishing gradient problems. To address these issues, ReLU releases the bound by simply picking the outputs as max(0, x_{input}) [149]. Since too many activations being below zero will make ReLU neurons inactive, leaklyReLU additionally introduces small gradients over the negative domain [137]. Besides, to improve the training stability by constraining the outputs of activations, a few hard bounded functions upon ReLU were developed, such as bounded ReLU and bounded leakyReLU [128]. Moreover, [44] proposes to replace all the multiplications with adds to reduce the computations.

For probability prediction, the implementation of softmax needs to compute a normalization factor based on all output neurons that could be in millions or billions. To reduce the computations, hierarchical softmax reorganizes output probabilities based on a binary tree. Specifically, each parent node divides the own probability to its children nodes, and each leaf corresponds to a probabilistic output [144]. When an input-output pair is available, one can only maximize the probability of the path in the binary tree, which reduces the computational complexity logarithmically. When Huffman coding is adopted for an optimal hierarchy, the speedup can be further enhanced [143].

Discussion based on experimental results. Gradient descent is widely used for training various networks. Thus, filter decomposition successfully reduces the number of computations by lowering the scale of parameters. For example, Inception v1 was able to become 2× to 3× faster than those without dimensionality reduction on channels [189]. Mobilenet with depthwise convolutions could build a 32× smaller and 27× less compute-intensive networks than VGG16, which still obtained comparable accuracies [97]. Benefit from spatial decomposition, Inception v3 used at least 5× fewer parameters and achieved 6× cheaper computationally [190]. Meanwhile, by improving the activations, the computations can be significantly reduced. For example, by using hierarchical softmax on vocabulary with 10,000 words, [144] sped up network training more than 250×. If there was no overhead and no constant term, the speedup

1041-4347 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on August 12,2020 at 03:51:46 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3015777, IEEE Transactions on Knowledge and Data Engineering

could be 750×. Besides, the network with ReLU could reach a 25% training error rate on CIFAR-10 while being 6× faster than the one with tanh [117].

Above all, it is necessary to combine the above strategies to reduce the computations in iterative optimization. However, to maintain the effectiveness with filter decomposition, it is essential to analyze the complexity of each operation carefully and modify a model progressively [49], [92]. We also note that, model compression also plays a vital role in deep learning. More details can be found in [55], [230].

3.1.4 For Tree-based Models

Tree-based models build hierarchical trees from a root to leaves by recursively splitting the instances at each node using different decision rules [172], such as Gini index and entropy. After that, random forest (RF) and gradient boosting decision trees (GBDT) introduce ensemble learning to improve the robustness of classifiers and enhance their accuracies. The former trains each tree independently. The latter learns trees sequentially by correcting errors, and often applies second-order approximation for custom losses [75].

Motivation. Given n instances with d features, finding the best split for each node generally leads to the computational complexity of $O(nd)$ by going through all features of each instance. Thus, the datasets with millions of instances and features will lead to huge computational burdens when growing trees. To address this issue, the relaxation of decision rules becomes significantly important, which can be performed from the views of instances and features.

Relaxation with instance reduction. Implementing instances sampling while growing trees is the simplest method of relaxing the rules and reducing costs [65], which benefits both trees, RF, and GBDT. Besides, one can take advantage of sparse features to ignore invalid instances when evaluating each split, making the complexity linear to the number of non-missing instances [52]. To improve the representation of sampled instances for GBDT, GOSS prefers the instances with large gradients [112].

Relaxation with feature simplification. Feature sampling is an alternative for rule relaxation, which considers only a subset of features at each split node [36]. Besides, histogram-based boosting groups features into a few bins using quantile sketches in either a global or local manner and then perform the splitting based on these bins directly [21], [52]. Since a part of features rarely take nonzero values simultaneously, EFB develops a greedy bundling method to locate these features and then merges them by only extending the range of exclusive features [112].

Discussion based on experimental results. Below we discuss the strengths and weaknesses of the relaxation of decision rules. Firstly, randomly sampling 30% to 20% instances could speed up GBDT 3× to 5× while improving its performance [76]. Sparsity-aware splitting also ran 50 × faster than the naive version on Allstate-10K [52]. On the other hand, local histogram aggregation resulted in smaller numbers of iterations than the global one by refining the histogram strategy [52]. Meanwhile, EFB additionally merged implicitly exclusive features into much fewer features, leading to better performance than sparsity-aware splitting [112]. Secondly, although the relaxed rules significantly speed up the training of tree-based models, it may result in extra costs. For example, GOSS requires the cost of computing gradients, and sparsity-aware splitting has to maintain a nonzero data table with additional memory costs.

3.1.5 Summary

We have reviewed various LML methods from the perspective of computational complexities. Now we discuss both the advantages and disadvantages of the above methods.

Firstly, kernel and graph-based models can be scaled up and optimized more efficiently than deep models. Besides, experts can introduce their domain knowledge on input features and develop specific similarities for these two models. Since the sum of positive semidefinite matrices is still positive semidefinite, it is also easier to merge the similarities of different types of features into a single model [211]. Of note, although graphs generally lead to smaller memory costs than kernels, they are only able to handle the data that is satisfied with the cluster assumption [228].

Secondly, benefit from the hierarchical feature extraction on structural instances, deep models can obtain much higher classification accuracies [17], [171]. However, these methods in turn require huge time costs for training the over-parameterized models. Although filter decomposition methods reduce the computations remarkably, the architectures demand careful designs [190]. Besides, some deep models are mathematically equivalent to kernelized ridge regressions that learn their own kernels from the data [170]. However, they can only build kernels in finite-dimension spaces, and their representations are generally uninterpretable, making predictions hard to be explained [20]. In contrast, tree-based models with hierarchical splitting are more interpretable. Besides, these models can be directly integrated into many other methods for acceleration, such as label trees with binary classifiers [22].

3.2 Optimization Approximation

Optimization approximation scales up machine learning from the perspective of computational efficiency. In each iteration, these methods only compute the gradients over a few instances or parameters to avoid most useless computations [220]. As a result, they increase the reduction in the optimization error per computation unit and obtain an approximate solution with fewer computations. For a small $\mathcal{E}_{opt}$, advanced mathematical techniques must be used to guarantee the effectiveness of approximation. According to targeted scenarios, we further categorize them into mini-batch gradient descent, coordinate gradient descent, and numerical integration based on Markov chain Monte Carlo. For convenience, an overview is provided in Tab.2.

3.2.1 For Mini-batch Gradient Descent

The methods of mini-batch gradient descent (MGD)[1] aim to solve the problems with a modest number of parameters but a large number of instances. Compared with stochastic gradient descent, MGD utilizes better gradients estimated over more instances per iteration and generally obtains fast local convergence with lower variances.

1. MGD in this paper is equal to mini-batch SGD in other papers.

1041-4347 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on August 12,2020 at 03:51:46 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3015777, IEEE Transactions on Knowledge and Data Engineering

TABLE 2
A brief lookup table for LML methods based on Optimization Approximation.

Categories	Strategies	Representative Methods
Mini-batch Gradient Descent	adaptive sampling of mini-batches	dynamic batch sizes [183], proportional instance sampling [13], [86].
	correction of first-order gradients	momentum [152], [160], weighted historical gradients, [64], [108], [174].
	approximation of higher-order gradients	mini-batch Quasi-Newton [38], mini-batch Gauss-Newton [72].
	adjustment of learning rates	refer to iterations [212], to gradients [69], [225], to moments [116], [164].
Coordinate Gradient Descent	selection of parameters	Gauss-Seidel manner [41], [98], [151], Gauss-Southwel rules [67], [153], [178].
	fast solutions of subproblems	using extrapolated points [19], [124], [129], [151], [175], caching [35], [178].
Numerical Integration	mini-batch gradient without M-H tests	1st-order Langevin dynamics [158], [202], 2nd-order Langevin dynamics [51].

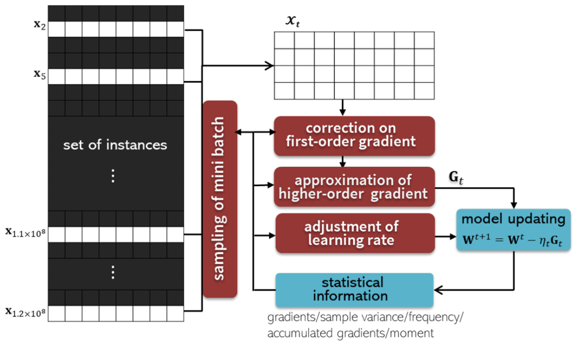

$\mathbf{x}_2$
$\mathbf{x}_5$
set of instances
$\mathbf{x}_{1.1\times10^8}$
$\mathbf{x}_{1.2\times10^8}$
$\mathcal{X}_t$
sampling of mini batch
correction on first-order gradient
approximation of higher-order gradient
$\mathbf{G}_t$
adjustment of learning rate
model updating $\mathbf{W}^{t+1} = \mathbf{W}^t - \eta_t\mathbf{G}_t$
statistical information
gradients/sample variance/frequency/ accumulated gradients/moment

Fig. 2. The diagram of mini-batch gradient descent, where the steps in red are key points for improving the computational efficiency.

Motivation. We first review the basic formulation of MGD. Suppose $\mathcal{X}_t$ refers to a mini-batch of instances with the size of m_t, and Q denotes the objective function built upon the parameter matrix $\mathbf{W}$. Let $\partial Q(\mathbf{W};\mathbf{x}_i)$ be the stochastic gradient on $\mathbf{x}_i$ and $\mathbf{G}_t = \frac{1}{m_t}\sum_{i\in\mathcal{X}_t}\partial Q(\mathbf{W}^t;\mathbf{x}_i)$ indicates the aggregated stochastic gradient on $\mathcal{X}_t$. Similar to gradient descent, the parameter $\mathbf{W}$ can be updated by

$$\mathbf{W}^{t+1} = \mathbf{W}^t - \eta\mathbf{G}_t \tag{5}$$

with a learning rate of η. For large-scale datasets, updating parameters based a few instances leads to large variances of gradients and makes optimization unstable. Although we can estimate gradients with large and fixed batch sizes, it remarkably increases per-iteration costs [33].

To address this issue, many LML methods have been proposed by improving gradient information in each iteration with a few extra computations. According to the roles in MGD, we review the methods from four complementary aspects. An illustrative diagram is shown in Fig.2.

Adaptive sampling of mini-batches. We take account of both the sizes of mini-batches and the sampling of instances. Firstly, since the naive algorithms that partially employ gradient information sacrifice local convergence, we can increase batch sizes gradually via a prescribed sequence [183]. Besides, linear scaling is effective for large mini-batches [87]. Secondly, as randomly selected instances are independent of optimization, it is worth considering both data distributions and gradient contributions. Specifically, we can enforce the sampling weights of instances to be proportional to the L2 norm of their gradients [13]. Besides, by maintaining a distribution over bins and learning the distribution per t iterations, computations can be further reduced [86].

Correction of first-order gradients. Performing the correction on mini-batch gradients provides an alternative to improve the quality of search directions with lower variances, which enables a larger learning rate for accelerations. On the one hand, gradient descent with momentum stores the latest gradients and conducts the next update based on a linear combination of the gradient and previous updates [160]. Gradients descent with Nesterov momentum first performs a simple step towards the direction of the previous gradient and then estimates the gradient based on this lookahead position [152]. On the other hand, SAG utilizes the average of its gradients over time to reduce the variances of current gradients [174], and SVRG develops a memory-efficient version which only needs to reserve the scalars to constrict the gradients at subsequent iterations [108].

Approximation of higher-order gradients. When the condition numbers of objective functions become larger, the optimization can be extremely hard owing to ill-conditioning [61]. To solve this issue, MGD methods thus introduce the approximation of second-order information with successive re-scaling [122], [135], [187]. Similar to inexact Newton algorithms, a basic solution is to employ conjugate gradient algorithms to estimate Hessian matrices. However, as the mini-batch size is much smaller, these algorithms implemented in a nearly stochastic manner can lead to very noisy results. To this end, mini-batch-based Quasi-Newton (MQN) and Gauss-Newton algorithms (MGN) are proposed, which improve the approximation by only using first-order information. In particular, MQN introduces online L-BFGS to approximate the inverse of Hessian based on the latest parameters and a few gradients at previous mini-batches [38]. MGN builds the Hessian approximation based on the Jacobian of the predictive function inside quadratic objective functions [72]. When logarithmic losses are used in probability estimation, it enables faster implementation without the explicit Jacobian matrices. It is worthwhile noting that we usually can offset the costs of these approximations when the size of mini-batches is not too small [33].

Adjustment of learning rates. The learning rate plays an important role in the convergence [169]. Specifically, a small rate may slow down the convergence, while a large rate can hinder convergence and cause the objective function to fluctuate around its minimum. Although the rate decayed by the number of iterations can alleviate this issue [212], recent studies propose to adjust it more carefully according to optimization processes. For example, Adagrad prefers smaller rates for the parameters associated with frequently occurring dimensions and larger rates for the ones associated with infrequent dimensions [69]. Besides, Adadelta

1041-4347 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on August 12,2020 at 03:51:46 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3015777, IEEE Transactions on Knowledge and Data Engineering

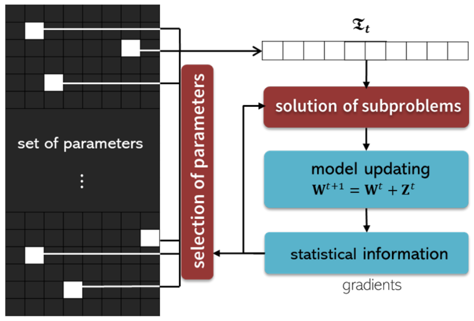


Fig. 3. The diagram of coordinate gradient descent, where the steps in red are key points for improving computational efficiency.

takes account of the decaying average over past squared gradients [225]. The motivation is that, by restricting the length of accumulated gradients to some fixed sizes, we can reduce the aggressive when decreasing learning rates [23]. Meanwhile, learning rates can also be updated based on the second moment. For example, Adam prefers flat minima in error surfaces [116]. Besides, Nadam and AMSGrad introduce Nesterov-accelerated gradients and the maximum of past squared gradients into Adam, respectively [68], [164].

Discussion based on experimental results. Now we discuss the effectiveness of the above strategies. Firstly, adaptive sampling of mini-batch can clearly improve computational efficiency. By doubling batch sizes during training, [183] reduced the time cost from 45 mins to 30 mins on ImageNet. Besides, [86] introduced adaptive sampling and only used 30% epochs to obtain the same accuracy of uniform sampling. Since updating the statistic information does not need to be frequent, its extra computational cost is insignificant. Secondly, by correcting gradients for lower variance, SVRG converged faster than the MGD using learning rate scheduling [108]. Thirdly, by approximating the second-order information, [135] achieved 6× to 35× faster when solving eigensystems. MQN also reached a lower objective value with competitive computing time to the original MGD [38]. Finally, by taking the frequency of parameters and the decaying of gradients into account for learning rates, Adagrad and Adadelta could only use 20% epochs to achieve the same result of normal optimizers [225]. Based on adaptive moment estimation, Adam further reduced more than 50% computations [116]. A comprehensive comparison of learning rate schemes can be found in [113]. Of note, since these strategies play different roles in optimization, one can apply all for further acceleration [80].

3.2.2 *For Coordinate Gradient Descent*

The LML methods built upon coordinate gradient descent (CGD) aim at addressing the problems with a modest number of instances in a high dimension. These problems frequently arise in areas like natural language processing [220] and recommender systems [18], [93].

Motivation. We first present the basic formulation of CGD [204]. Let $Q(\mathbf{W})$ denote the objective function with the parameter matrix of $\mathbf{W}$, and $\mathcal{I}_t$ contain the indexes of the selected parameters in the t-th iteration. As only a few parameters need to be updated, CGD can take all instances into account and optimize the selected parameters to optimal solutions in each iteration. Specifically, let $\mathbf{W}+\mathbf{Z}^t$ refer to the expected solution on selected parameters, namely $\mathbf{Z}^t$ only contains non-zero values at the indexes in $\mathcal{I}_t$. In each iteration, CGD minimizes the following subproblem

$$g(\mathbf{Z}^t) = Q(\mathbf{W}+\mathbf{Z}^t) - Q(\mathbf{W}), \tag{6}$$

where $Z^t_{i,j} = 0, \forall\{i,j\} \notin \mathcal{I}_t$. Although the scales of the reduced subproblems are small and many available solvers can be directly used for the optimization [11], CGD can still lead to an expensive time cost for exact solutions during iterative optimization. Besides, updating each parameter with the same number of iterations causes a massive amount of redundant computations.

To address these issues, we review LML methods from two aspects: the selection of parameters and the fast solution of subproblems. For convenience, an illustrative diagram showing their roles is displayed in Fig.3.

Selection of parameters. For convergence with a smaller number of iterations, the indexes must be selected carefully. Typically, the selection of these parameters follows a Gauss-Seidel manner [217], namely, each parameter is updated at least once within a fixed number of consecutive iterations. As features may be correlated, performing the traversal with a random order of parameters in each iteration has empirically shown the ability of acceleration [41], [98], [151]. On the other hand, Gauss-Southwell (GS) rules propose to take advantage of gradient information for the selection, which can be regarded as performing the steepest descent [178]. To avoid the expensive cost in GS rules, we can connect them to the nearest neighbor search and introduce a tree structure to approximate the rules [67], [153]. Besides, by introducing the quadratic approximation on objective functions and performing the diagonal approximation on Hessian matrices, the selection of multiple indexes can be reduced to separable problems [221].

Fast solutions of subproblems. Similar to accelerated gradient descent, extrapolation steps can be employed for accelerated coordinate descent methods [124], [151]. Suppose $Q(\mathbf{W})$=$p(\mathbf{W})$+$q(\mathbf{W})$, where $p(\mathbf{W})$ and $q(\mathbf{W})$ are smooth and nonsmooth, respectively. Accelerated proximal gradient (APG) first replaces $p(\mathbf{W})$ with the first-order approximation regularized by a trust region penalty and then uses the information at an extrapolated point to update the next iterate [129]. With appropriate positive weights, it can improve the convergence remarkably while remaining almost the same per-iteration complexity to the proximal gradient [19], [175]. To address nonconvex problems, the monotone APG method proposed in [125] introduces sufficient descent conditions with a line search. It reduces the average number of proximal mappings in each iteration and speeds up the convergence. Moreover, if both $p(\mathbf{W})$ and $q(\mathbf{W})$ are nonsmooth, ADMM introduces Douglas-Rachford splitting to solve the problem, which is extremely useful when the proximal operators can be evaluated efficiently [35]. Of note, similar to standard iterative optimization, common heuristics can be used for speedup, such as caching variable-dependent eigendecomposition [35] and pre-computation of non-variable quantities [178].

1041-4347 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on August 12,2020 at 03:51:46 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3015777, IEEE Transactions on Knowledge and Data Engineering

Discussion based on experimental results. Now we discuss the efficiency of the above strategies for solving primal and dual problems. On the one hand, when optimizing L2-SVM with correlated features, the CGD based on random permutation could obtain the same relative difference to the optimum with 2× to 8× fewer training costs [41]. Besides, due to the sparsity of the solutions of L1 and L2 regularized least squares problems, CGD based on GS rules not only outperformed random and cyclic selections with 5× to 10× fewer epochs for faster convergence but also used 2× less running time including the estimation of statistical information [153]. On the other hand, for solving lasso, APG only required 20% time of the basic proximal gradient, and ADMM further saved 50% time costs [155]. Besides, when solving linear inverse problems like image deblurring, the result of the proximal algorithm ISTA after 10,000 iterations could be obtained by its accelerated version with 275 iterations [19], [60]. For nonconvex logistic regression with capped L1 penalties, nonmonotone APG with released conditions obtained a higher accuracy with 75% fewer costs than monotone APG [125]. In addition, by caching eigendecomposition, ADMM solved problems 20× faster than computing it repeatedly in each iteration [35].

3.2.3 For Numerical Integration with MCMC

The methods based on Markov chain Monte Carlo (MCMC) are widely used in Bayesian posterior inference on $p(\mathbf{W}|\mathcal{X})$ [27], such as M-H and Gibbs sampling [57], [89]. Unlike the gradient-descent optimization with multi-dimensional integrals on $p(\mathcal{X})=\int_{\mathbf{W}} p(\mathcal{X}|\mathbf{W})p(\mathbf{W})d\mathbf{W}$, they introduce numerical approximations by recording samples from the chain and avoid the huge time costs in high-dimensional models. Specifically, they first generate candidate samples by picking from distributions and then accept or reject them based on the corresponding acceptance ratios. To enhance acceptance probabilities and improve efficiency, more methods introduce the gradient of the density of target distributions to generate samples at high-density regions.

Motivation: Given a dataset of n instances, most gradient-based sampling methods [150] introduce the Langevin dynamic and update parameters based on both the gradient and the Gaussian noise as

$$\mathbf{W}_t + \frac{\epsilon_t}{2}\{\partial_{\mathbf{W}_t}\text{log}p(\mathbf{W}_t) + \sum_{x\in\mathcal{X}} \partial_{\mathbf{W}_t}\text{log}p(x|\mathbf{W}_t)\} + v_t, \quad (7)$$

where the step size ϵ_t and the variances of noise $v_t \sim N(0, \epsilon_t)$ are balanced. As a result, their trajectories of parameters can converge to the full posterior distribution rather than just the maximum a posteriori mode. However, the calculation of gradients per interaction faces the computation over the whole dataset, and expensive M-H accept/reject tests are required to correct the discretization error. Although Gibbs sampling is free from M-H tests, it is still limited to the exact sampling from conditional posterior distributions. Thus, many stochastic gradient MCMC methods based on mini-batches are proposed to address the above issues.

Mini-batch gradient without M-H tests. SGLD first introduces the scaled gradients that estimated from mini-batch instances to approximate the gradient of the log-likelihood [202]. Besides, it discards the M-H test and accepts all the generated samples, since its discretization error disappears when $\epsilon_t \to 0$. To reduce the number of iterations, SGFS prefers samples from approximated Gaussian distributions for large step sizes and switches to samples from the non-Gaussian approximation of posterior distributions for small step sizes [10]. Meanwhile, SGHMC introduces a second-order Langevin dynamics with a friction term to improve the exploration of distant space [51]. In addition, SGRLD extends SGLD for models on probability simplices by using Riemannian geometry of parameter spaces [158], and SGRHMC further takes advantages of both the momentum of SGHMC and the geometry of SGLRD [136].

Discussion based on experimental results. Now we show the effectiveness of the above MCMC methods for Bayesian posterior inference. Firstly, these methods enable efficient inference for large-scale datasets. For example, SGRLD could perform LDA on Wikipedia corpus while Gibbs sampling was not able to run [158]. Secondly, different mini-batch gradient variants can inherit the properties of their original versions. For example, SGHMC could generate high-quality distant samples and reduce more than 50% iterations than SGLD for training Bayesian neural networks [51]. Besides, SGRHMC resulted in lower perplexities of LDA than both SGRLD and SGHMC while remaining similar costs [136].

3.2.4 Summary

To enhance the computational efficiency and remain reliable solutions, the above methods prefer the computations that produce higher reduction in optimization errors. Firstly, both MGD and CGD methods take the importance of involved instances or parameters into account and introduce the careful selections for reducing the time cost. Meanwhile, both can take advantage of Nesterov's extrapolation steps and tune learning rates with accumulated gradients or use a line search for faster convergence. Secondly, CGD is more suitable for optimizing models with a large number of parameters, especially for linear models where data is stored with inverted indexes. In contrast, if the number of features is much smaller than the number of instances, one should solve problems based on MGD. Thirdly, for handling large models with a massive amount of instances, it is natural to combine two strategies together, i.e., updating a subset of parameters with a few instances in each iteration. For example, a recently proposed method called mini-batch randomized block coordinate descent estimates the partial gradient of selected parameters based on a random mini-batch in each iteration [235]. Finally, mini-batch gradient MCMC algorithms improve qualities of samples and accelerate Bayesian inference. Besides, these algorithms generally resemble MGD from different views, such as additive noise and momentum. However, since samples from the dynamics of native stochastic variants may not always converge to the desired distribution, some necessary modification terms must be introduced to substitute the correction steps [51], [136]. Besides, the theory on how these additions differentiate a Bayesian algorithm from its optimization remains to be studied [43].

Meanwhile, a significant part of the above LML methods focuses on handling convex problems, while only a few methods can address nonconvex problems. Besides, most

1041-4347 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on August 12,2020 at 03:51:46 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3015777, IEEE Transactions on Knowledge and Data Engineering

methods solve target problems directly. Instead, one may seek the transformed forms of their problems and develop more appropriate algorithms by referring related work in other domains. Details will be discussed in Sec.4.1.2.

3.3 Computation Parallelism

Computation parallelism reduces practical time costs from the perspective of computational capabilities. The motivation is, mutually-independent subtasks can be processed simultaneously over multiple computing devices. For this purpose, many parallel processing systems have been developed. On the one hand, these systems increase the number of computations per time unit based on powerful hardware; on the other hand, they introduce advanced software to schedule hardware efficiently. According to the scenarios of hardware, we review the systems from two categories, as shown in Tab.3, including the systems with multi-core machines and those with multi-machine clusters. Below we introduce their main characteristics, aiming at utilizing their developments to benefit efficient machine learning methods.

3.3.1 For Multi-core Machines

The systems here take advantage of multi-core machines for parallelism. Specifically, all cores can access common memories, and each of them execute subtasks independently.

Motivation. To enhance the computational ability of a single machine, recent processing systems increase the number of parallelable instructions by introducing multiple processors into the machine, such as multiple-core CPUs and GPUs. Although their theoretical capabilities could be improved almost linearly with respect to the number of cores, the practical speedup is much lower owing to the overhead. To improve the effectiveness, users have to handle tedious works such as the scheduling of cores and memory.

To address this issue, we review representative works that provide the interfaces with different levels of simplicity and flexibility. Consequently, one can take their advantages and smoothly implement LML models in a single machine.

Adoption of highly-optimized libraries. When the memory is sufficient, we can directly utilize multi-core CPUs or GPUs to perform the optimization with available highly-optimized libraries, e.g., Mkl for Intel CPUs and Cuda for Nvidia GPUs. On the one hand, CPUs can use relatively cheap RAMs and handle complex operations in parallel. For example, Liblinear and FPSG provide the efficient solutions to L1-regularized linear classification and parallel matrix factorization [56], [58], respectively. For general purposes, Shogun is compatible with support vector machines and multiple kernel learning [185]. Shark additionally supports evolutionary algorithms for multi-objective optimization [99]. Of note, lock-less asynchronous parallel (ASP) can further take advantage of CPU cores and reduce the overhead for parallelizing MGD [163]. On the other hand, GPUs built upon thousands of stream cores are better at handling simple computations in parallel. In particular, deep learning systems are highly benefited from GPU-accelerated tensor computations and generally provide automatic differentiation, such as Theano [25], Caffe [103], MXNet [53], TensorFlow [5] and PyTorch [156]. For example, TensorFlow optimizes execution phase based on the global information of programs and achieves the high utilization of GPUs [5]. PyTorch introduces easy-to-use dynamic dataflow graphs and offers the interface to wrap any module to be parallelized over batch dimension [156].

Improvement of I/O accesses. When the memory cannot handle a LML model at once, we have to break it into multiple parts, where each part is computed in parallel and different parts are implemented sequentially. In this case, the I/O accesses result in an in-negligible overhead for overall computations, and the effective utilization of memory and extra storage becomes essential. For example, to handle a large graph with n_e edges, GraphChi divides its vertices into many intervals, where each is associated with a shard of ordered edges [119]. When performing updating from an interval to the next, it slides a window over each of the shards for reading and writing, implementing the asynchronous model with $O(2n_e)$ accesses. To avoid expensive pre-sorting of edges, X-Stream instead proposes an edge-centric implementation in synchronous by streaming unordered edge lists [168]. However, it needs to read edges and generate updates in the scatter phase and read updates in the gather phase, leading to $O(3n_e)$ accesses. Recently, GridGraph introduces a two-level partitioning, which streams every edge and applies the generated update instantly [238]. It only requires one read pass over edges and several read/write passes over vertices, leading to nearly $O(n_e)$ accesses. Considering that the memory of GPU is expensive and intermediate feature maps account for the majority of usage, vDNN moves the intermediate data between GPU and CPU, which can train larger networks beyond the limits of the GPU's memory [165].

Discussion based on experiments. The systems based on multi-core machines have shown their power to accelerate the implementation of LML. In particular, when the memory is sufficient, FPSG could speed up matrix factorization nearly 7$\times$ with a 12-core CPU [58]. Besides, Theano was able to achieve 5$\times$ faster for training CNNs on GPUs rather than CPUs and 6$\times$ faster for optimizing DBNs [25]. Even with limited memory, by scheduling I/O accesses efficiently, we can still utilize the parallelism and train large models within acceptable time costs. For example, GraphChi used a mac mini and successfully solved large Pagerank problems reported by distributed systems [119]. For the same number of iterations, GridGraph only required fewer than 50% time costs of others [238]. Besides, with the support of CPU memory, vDNN could optimize a deep network with the memory requirement of 67 GB based on a 12GB GPU [165].

3.3.2 For Multi-machine Clusters

With the explosion of data size, processing systems based on distributed clusters have attracted increasing attention. These systems utilize local area networks to integrate a set of machines, where each runs its own tasks.

Motivation. Distributed processing systems introduce parallelism at two levels, including a number of nodes in a cluster and multiple cores within the individual node. One can also construct huge models for complex tasks if model parallelism is applied. Nevertheless, the main challenge of applying a cluster is managing (heterogeneous) machines,

1041-4347 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on August 12,2020 at 03:51:46 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3015777, IEEE Transactions on Knowledge and Data Engineering

TABLE 3
A brief lookup table for LML methods based on Computation Parallelism.

Categories	Strategies	Representative Methods
Multi-Core Machines	adoption of optimized libraries	acceleration with CPUs [56], [58], [99], [185], with GPUs [5], [25], [103], [156].
	improvement of i/o accesses	balance memory and disks [119], [168], [238], balance CPU and GPU memory [165].
Multi-Machine Clusters	mapreduce abstraction	mapreduce [167], [179], [29], spark [223], supporting libraries [28], [142], [224].
	graph-parallel abstraction	bulk synchronous [138], asynchronous [133], GAS decomposition [48], [84].
	parameter server abstraction	distributed networks [53], [62], gradient boosting [104], general-purpose system [209].
	allreduce abstraction	tree-based [52], butterfly-based [3], [112], ring-based [2], [4], [102], [177], hybrid [87].

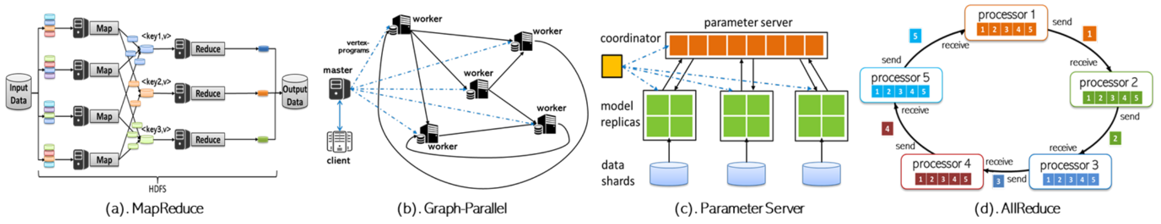


Fig. 4. The toy examples of distributed processing systems. (a) the structure of a mapreduce system [63], (b) the structure of graph-parallel systems [138], (c) a sketch of parameter-server-based cluster [126], and (d) a step of data communication of the ring-based allreduce abstraction [177].

such as data communication across the nodes. These tedious works may lead to low resource utilization.

Below we review typical distributed systems to simplify the developments of parallelism [39]. We present the introduction according to their top-level topology configurations.

MapReduce abstraction. The MapReduce abstraction divides computations among multiple machines [63], where each works with a part of tasks in parallel with its locality-distributed data. As we can see from Fig.4(a), MapReduce systems reformulate complex models into a series of simple Map and Reduce subtasks. Specifically, Map transforms its local data into a set of intermediate key/value pairs, and Reduce merges all intermediate values associated with the same intermediate key. The systems take care of the details of data partitioning, execution scheduling, and communication managing. As a result, the methods that meet this abstraction can take advantage of locality-distributed data and be executed on clusters smoothly.

Hadoop MapReduce is the first open-source implementation, which introduces HDFS files to store the data over disks [179]. Its flexibility allows new data-analysis software to either complement or replace its original elements, such as YARN [195]. Besides, to support iterative machine learning methods, Iterative MapReduce introduces a loop operator as a fundamental extension [167], and Hybrid MapReduce presents a cost-based optimization framework to combine both the task and data parallelism [29]. On the other hand, Spark employs the immutable distributed collection of objects RDD as its architectural foundation [222] and distributes the data over a cluster in the memory to speed up iterative computations [223]. Besides, a stack of libraries are developed to further simplify its programming. For example, MLlib provides a variety of linear algebra and optimization primitives [142] and Spark Streaming eases the streaming learning [224]. Recently, SystemML enables Spark to compile sophisticated machine learning methods into efficient execution plans automatically [28].

Graph-parallel abstraction. The graph-parallel abstraction shown in Fig.4(b) exploits the structure of sparse graphs to improve communication between different machines [111]. Specifically, it consists of a sparse graph and a vertex-program. The former is partitioned into different subsets of vertexes over multiple machines. The latter is executed in parallel on each vertex and can interact with the neighboring vertexes with the same edge. Compared with MapReduce, these systems constrain the interaction based on sparse adjacency and reduces the cost of communication.

Pregel is a fundamental graph-parallel system running in bulk synchronous parallel (BSP) [138]. It consists of a sequence of supersteps, where messages sent during one superstep are guaranteed to be delivered at the beginning of the next superstep. However, since the slowest machine determines the runtime in each iteration, Pregel can result in idling problems. GraphLab thus introduces ASP by releasing the constraints of supersteps [133], which updates the vertexes using the most recent values. Moreover, PowerGraph introduces a gather, apply, and scatter (GAS) decomposition to factor vertex-programs for power-law graphs [84] and enables the computations in both BSP and ASP. Recently, PowerLyra uses centralized computation for low-degree vertices to avoid frequent communications and follows GAS to distribute the computation for high-degree vertices [48]. Similarly, it supports both BSP and ASP.

Parameter server abstraction. The parameter server abstraction distributes data and workloads over worker nodes and maintains distributed shared memory for parameters on server nodes [126]. An example is shown in Fig.4(c). This abstraction enables an easy-to-use shared interface for I/O access to models, namely, workers can update and retrieve the different parts of parameters as needed.

DistBelief first utilizes computing clusters with the parameter server [62], where the model replicas in workers asynchronously fetch parameters and push gradients with the parameter server. It thus can train deep networks with billions of parameters using thousands of CPU cores. After that, MXNet adopts a two-level structure to reduce the bandwidth requirement [53], where one level of servers

1041-4347 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on August 12,2020 at 03:51:46 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3015777, IEEE Transactions on Knowledge and Data Engineering

manages the data synchronization within a single machine, and the other level of servers manages intermachine synchronization. Moreover, the naive distributed Tensorflow introduces dataflow with the mutable state to mimic the functionality of a parameter server, which provides additional flexibility on optimization algorithms and consistency schemes [5]. Besides, to handle GBDT with high dimensions, DimBoost transforms each local histogram to a low-precision histogram before sending it to the parameter server [104]. Once these local histograms are merged on the servers, a task scheduler assigns active tree nodes among all workers to calculate the best split for each tree node. Meanwhile, distributed MCMC methods are also developed for large-scale Bayesian matrix factorization [9], where the parameter server updates its global copy with new sub-parameter states from workers. In addition, as a general-purpose system, Petuum is also benefited from this abstraction [209] and simplifies the distributed development of various scalable machine learning models.

Allreduce abstraction. The allreduce abstraction reduces the target arrays in all m machines to a single array and returns the resultant array to all machines. It retains the convergence of gradient descent in BSP and can be divided into tree-based [50], butterfly-based [206], and ring-based [1].

XGBoost introduces tree-allreduce by organizing all the workers as a binomial tree [52]. Its aggregation steps follow a bottom-to-up scheme starting from the leaves and ending at the root. However, these steps cannot overlap in its implementation. LightGBM thus introduces butterfly-allreduce with a recursive halving strategy [3], [112]. Specifically, at the k-th step, each worker exchanges $\frac{n}{2k}$ data with a worker that is $\frac{m}{2k}$ distance away. This process iterates until the nearest workers exchange their data.

Recently, ring-allreduce has attracted more attention by employing bandwidth-optimal algorithms to reduce communication overhead [157]. Fig.4(d) displays an illustrative example, where m machines communicate with their neighbors $2\times(m\text{-}1)$ times by sending and receiving parameter buffers. Specifically, in the first (second) m-1 iterations, each machine receives a buffer and adds it to (substitutes it for) its own value at the corresponding location. These new values will be sent in the next iteration. To ease the use of ring-allreduce on GPUs, Nvidia develops Nccl for its devices [102]. After that, Horovod integrates Nccl into deep learning and allows users to reduce the modification of their single-GPU programs for the distributed implementation [177]. Pytorch utilizes Nccl to operate heavily-parallel programs on independent GPUs [4]. Recently, Tensorflow also provides two different experimental implementations for ring-allreduce [2]. In particular, Caffe2 introduces butterfly-allreduce for inter-machine communication while using ring-allreduce for local GPUs within each machine [87].

Discussion based on experiments. Effective distributed systems not only allow us to handle larger-scale datasets and models, but also make data communication more efficient. However, different abstractions show their characteristics in specific tasks. For example, for PageRank, Hadoop and Spark took 198 and 97 seconds for each iteration, while PowerGraph only used 3.6 seconds and was significantly faster [84]. Based on the differentiated partition for low-degree and high-degree vertices, PowerLyra further outperformed PowerGraph with a speedup ranging from 1.4× to 2×. For matrix factorization, Petuum was faster than Spark and GraphLab [209]. Besides, when the number of workers was not a power of two, DimBoost could outperform XGBoost and LightGBM significantly by merging parameters with data sparsity [104]. Based on ring-allreduce, Horovod makes TensorFlow more scalable on a large number of GPUs. In particular, it could half the time cost of naive distributed Tensorflow when training CNNs with 128 GPUs [177].

3.3.3 Summary

Now we provide rough guidance of parallelism strategies. Firstly, CPUs are optimized for handling the models with complex sequential operations. For example, by introducing locality sensitive hashing for the sparse selection of activate neurons, the training of FCNs on CPUs was 3.5× faster than the best available GPUs [42]. On the contrary, GPUs could enhance the computational capability greatly for the intensive computational models with huge numbers of dense matrix and vector operations [74]. Secondly, although machine learning methods may be compatible with multiple abstractions [154], it is crucial to choose a proper abstraction for a specific setting. In general, graph-parallel and ring-allreduce are more suitable for graph models and deep learning models, respectively. Thirdly, we can ease the development of higher-level systems by leveraging the existing systems. For example, built upon YARN, Angel develops a parameter server system for general-purpose LML [106], which supports both model and data parallelism. Similarly, Glint applies Spark for convenient data processing in memory and introduces an asynchronous parameter server for large topic models [100]. Moreover, PS2 launches Spark and parameter servers as two separate applications without hacking Spark [233], which makes it compatible with any version of Spark. GraphX also introduces the graph-parallel to accelerate graph-based models on Spark [85].

However, the best choice of practical developments varies for the size of datasets and need further studies. For example, for training SVMs, the efficiency of GPUs only became higher than CPUs when the sizes of datasets increased to some degree [127]. Besides, owing to the overhead, the return of distributed systems becomes lower with the increasing number of processors, and we have to control the return on investment regardless of abstractions. For example, although larger CNNs was supposed to benefit more with multiple machines, a model with 1.7 billion parameters only had 12× speedup using 81 machines [62].

3.4 Hybrid Collaboration

The methods in the above sections scale up LML from the perspective of computational complexities, computational efficiency, and computational capabilities, respectively. Thus, they are independent of each other, and any partial enhancement enables the overall improvement.

Motivation. In general, we can enhance machine learning efficiency by jointly using multiple strategies directly [40], [71], [88]. The most typical work is deep learning, which has been undergoing unprecedented development over the past decades [47], [54], [123]. The key reasons that facilitate

1041-4347 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on August 12,2020 at 03:51:46 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3015777, IEEE Transactions on Knowledge and Data Engineering

TABLE 4
A brief lookup table for LML methods based on Hybrid Collaboration.

Categories	**Strategies**	**Representative Methods**
Gradient	Compression	quantize component values [14], [105], [176], [203], [226], remove redundancy gradients [130].
	Delay	limited delay with stale synchronous [7], [101], [121], [166], [234], [236], infinite delay [240].

its rapid development are detailed as follows. Firstly, deep learning methods introduce prior knowledge such as filter decomposition to simplify predictive models with fewer parameters [216]. Secondly, owing to its mini-batch-based optimization, various advanced MGD algorithms can be employed to train neural networks, leading to faster converges [116]. Thirdly, since GPUs are paired with a highly-optimized implementation of 2D convolutions and well-suited to cross-GPU parallelization [117], it is efficient for researchers to debug their advanced networks [219].

However, the above perspectives can be further improved for synergy effects. In particular, for simplified models, we can modify optimization algorithms to improve parallelism. We divided them into two strategies as listed in Tab.4, including gradient compression and gradient delay.

Gradient compression. To further accelerate the data communication in distributed optimization, many compression methods have been proposed, where only a small number of data is communicated across machines. For example, 1-bit MGD quantizes each component of gradients aggressively to one bit per value and feeds back the quantization error across mini-batches [176]. On the contrary, QSGD directly quantizes the components by randomized rounding to a discrete set of values while preserving the expectation with minimal variance [14]. As a result, it makes a trade-off between the number of bits communicated per iteration and the added variance. To exploit valuable gradients with outliers, TernGrad introduces layer-wise ternarizing and gradient clipping [203]. Meanwhile, DGC only transmits gradients larger than a threshold for sparsification while accumulating the rest of gradients locally [130]. In particular, when gradients are sparse and their distribution is nonuniform, the nonzero elements in a gradient are generally stored as key-value pairs to save space. SketchML thus takes the distribution into account and uses a quantile sketch to summarize gradient values into several buckets for encoding [105]. Besides, it stores the keys with a delta format and transfers them with fewer bytes. Furthermore, ZipML proposes to optimize the compression on gradients, models, and instances and then perform an end-to-end low-precision learning [226]. Of note, the above compression methods are compatible with various parallel systems, such as multiple GPUs [14], [176] and parameter servers [105], [130], [203].

Gradient delay. ASP-based MGD and CGD algorithms [131], [163] minimize the overhead of locking and alleviate the idling issues. However, they remain huge costs of data communication. The stale synchronous parallel (SSP) algorithms with gradient delays thus get more attention [96]. For example, [121] orders CPU cores where each updates variables in a round-robin fashion. Its delay reduces the cost of reading parameters of the latest models. [7] introduces tree-allreduce in distributed settings, where each parent averages the gradients of the children nodes from the previous round with its own gradient, and then passes the result back up the tree. In addition, CoCoA and Hydra use workers to perform some steps of optimization with their local instances and variables in each iteration [101], [166], respectively. EASGD introduces an elastic force between central and local models for more exploration to avoid many local optima [234]. Moreover, [240] proposes to solve subproblems exactly on each machine without communication between machines before the end, namely infinite delay. Since the delayed gradient is just a zero-order approximation of the correct version, DCASGD leverages the Taylor expansion of gradient functions and the approximation of the Hessian matrix of the loss function to compensate for delay [236]. On the other hand, delayed gradients can also be utilized to improve the convergence, which in turn reduces the communication. For example, ECQSGD and DoubleSqueeze accumulate all the previous quantization errors rather than only the last one for error feedback [191], [205]. Besides, [140] introduces the delayed gradients to chooses adaptive learning rates in parallel settings.

Discussion based on experiments. Now we give a brief discussion on the methods of hybrid collaboration. Firstly, gradient compression can reduce not only the time cost per iteration but also the overall cost. For example, compared with full 32-bit gradients, the training of 16-GPU AlexNet with 4-bit QSGD led to more than 4× speedup on communications, and 2.5× speedup on overall costs for the same accuracy [14]. In particular, for linear models, SketchML could accelerate the original optimizer and achieve more than 4× improvements [105]. Of note, models with larger communication-to-computation ratios and networks with s-maller bandwidth can benefit more from compression [203]. Secondly, gradient delay also reduces the amount of data communication and remarkably increases the proportion of time workers spend on the computation. For example, based on local updates, CoCoA was able to converge to a more accurate solution with 25× faster than the best non-locally updating competitor [101]. Besides, the reduction of communication cost could be nearly linear with respect to the delay under a large number of workers [234].

4 Discussions

Existing LML methods have established a solid foundation for big data analysis. Below we outline some promising extension directions and discuss important open issues.

4.1 Extension Directions

4.1.1 *For Model Simplification*

The methods of model simplification reduce computational complexities with adequate prior knowledge. Therefore,

1041-4347 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on August 12,2020 at 03:51:46 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3015777, IEEE Transactions on Knowledge and Data Engineering

we may further explore the distributions and structures of instances to enhance the scalability of predictive models.

Explore distributions of data. The low-rank approximation for kernels assumes that mapped instances in RKHS nearly lie on a manifold of a much lower dimension, and their inner products can be estimated on the low-dimension space. Graph-based methods are also scaled up based on the assumption that nearby points are more likely to share labels. Thus, to make use of large-scale unstructured data, the underlying global and local data distributions must be considered [82]. Based on this motivation, HSE and AER build hierarchical indexes on unlabeled instances for coarse-to-fine query selection [15], [78]. As a result, although the cost of the quality estimation of each candidate is not reduced, they lower computational complexities significantly by decreasing the number of candidates.

Exploit structures within instances. Various types of convolutions based on filter decomposition have been proposed to reduce the sizes of deep models. These works demonstrate that to make receptive fields more effectively, a critical direction is to take the features with multi-scale invariance into account and make use of the structural information of instances. Following this motivation, DCNs thus develop deformable convolutions to model larger transformations efficiently [59], [239], which only add a few parameters.

Analyze objective tasks. A more in-depth analysis of objective tasks provides another powerful way for model simplification. For example, inspired by the coarse-to-fine categorization of visual scenes in scene-selective cortex [147], hierarchical convolutional networks are developed for image classification [213]. These networks decrease the number of parameters and correspondingly reduce the involved computations per iteration. In addition, distributing parameters and computations according to the frequencies of objective predictions also reduces computational complexities [143].

4.1.2 For Optimization Approximation

To enhance computational efficiency, the methods of optimization approximation prefer the computations that bring significant reduction in optimization errors. However, most of them focus on convex problems. Besides, researchers generally follow stereotyped routines to address their emerging problems without seeking more appropriate solutions. To further speed up the optimization for a broader range of scenarios, the following extensions can be considered.

Optimization for nonconvex problems. The optimization may fall into poor local minima when problems are nonconvex [187], leading to unstable performances in real-world tasks. Therefore, it is necessary to enhance the ability of algorithms to escape or bypass the poor minima. To this end, various randomized operations can be considered, such as randomized weight initialization and random noise on gradients [188]. Besides, inspired by curriculum learning [114], we may first utilize instances that are easy to fit and then gradually turn to difficult ones, such as boundary instances. As a result, it allows algorithms to obtain a good solution at early stages and then tune it for a better minimum.

Solutions inspired by mathematical models. By transforming target problems into classical models in other mathematical areas, such as geometry, we can borrow the experiences of handling these models to accelerate the optimization. For example, a simple iterative scheme used to find the minimum enclosing ball (MEB) can be used to solve large-scale SVM problems [194]. The motivation is, the dual of both MEB and SVM problems can be transformed into the same form. Similarly, [107] introduces cutting-plane algorithms that iteratively refine a feasible set to solve SVM on high-dimension sparse features.

4.1.3 For Computation Parallelism

Computation parallelism enhances computational capacities based on multiple computing devices. Benefit from existing systems, we can take advantage of flexible programming interfaces and low data communication for parallel data processing. However, with the fast upgrading of hardware, software, and predictive models, the following issues recently attract much attention.

Flexible hardware abstractions. Highly-efficient parallel computations require fast data communication between different nodes. Although allreduce is able to minimize the communication overhead for most methods, it supposes that the memory and the computational ability of different nodes are at the same level. Otherwise, the utilization of resources will be limited by the weakest one. Considering the real-world scenarios where distributed systems generally consist of heterogeneous devices, more flexible abstractions need to be studied. A possible solution is to utilize domain-specific knowledge such as intra-job predictability for cluster scheduling [159], [208]. Besides, we may further explore hybrid abstractions and take account of traffic optimization to balance computational resources [46], [87].

Modularized open-source software. There are increasing interests in supporting open-source projects. Several motivations are as follows. The developments of parallel systems on general-purpose machine learning can be challenging even for large companies. Besides, open-source software can benefit more communities and democratize data science. To this end, normalized and modularized learning frameworks (similar to computer architectures [196]) are preferred. As a consequence, any localized enhancement can be achieved by contributors, improving the overall efficiency. In addition, to construct a vibrant and lively community, it is necessary to provide easy-to-use interfaces with different programming languages to benefit more researchers [145], as well as the simple postprocessing for productization.

4.2 Open Issues

4.2.1 Tighter Bounds of Complexities

Various complexities during the procedure of data analysis can be used to evaluate required computational resources in real-world tasks. Below we discuss some important ones.

Bounds of computational complexities. The computational complexity determines the numbers of computing devices and the spaces of memory. However, many existing bounds of computational complexities only provide general guarantees for any probability distribution on instances and for any optimization algorithm that minimizes disagreement on training data. To estimate the complexities more precisely, it is necessary to establish tighter bounds for real-world data

1041-4347 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on August 12,2020 at 03:51:46 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3015777, IEEE Transactions on Knowledge and Data Engineering

distributions and specific optimization algorithms, which may be achieved with the assistance of heuristical arguments from statistical physics [6].

Other related complexities. The measures of many other complexities also need to be studied. For example, over-parameterized networks can be well trained with a much smaller set of instances than the one estimated based on the current sample complexity [184]. Besides, more instances are required for adversarial training to build robustness [173]. In addition, the complexity of communication and the number of gates in a circuit can be used to estimate the practical limits on what machines can and cannot do [16].

4.2.2 *Collection of valuable data*

Large-scale datasets play a crucial role in LML, as the patterns for making decisions are learned from data without being explicitly programmed. Therefore, our machine learning community is in great need of large-scale instance-level datasets, and efficient methods for generating such data either in a supervised or unsupervised manner. To this end, we highlight the following directions.

Annotation tools. Annotation tools aim at building large-scale annotated datasets by collecting contributions from a lot of people, such as LabelMe in computer vision [192]. To reduce the costs of obtaining high-quality labels and simplify the annotation procedure, it is essential to ease the modes of annotation and interaction for oracles with scalable active learning methods [109]. Besides, it is difficult to collect clean instances with categorical and strong supervision information. The reasons are as follows. Firstly, there exist a large number of instances whose classes are fuzzy themselves. The one-hot or multi-hot encoding that groups them into specific classes can significantly harm their supervision information. Secondly, the inconsistent and inaccurate labels are ubiquitous in real-world situations due to subjective data-labeling processes. To address these issues, efficient weakly-supervised learning recently obtains urgent attention [237].

Data augmentation. Data augmentation applies transformations to training sets to increase their scales. Based on the augmented data, we can improve the performance of predictive models, especially when original datasets are imbalanced or their instances are insufficient. These transformations can be as simple as flipping or rotating an image, or as complex as applying generative adversarial learning. Recently, BigGAN augments image datasets successfully by synthesizing high fidelity natural instances [37].

4.2.3 *Moderate Specialization of Hardware*

Over the past years, the design of computer architectures focuses on computation over communication. However, with the progress of semiconductor, we observed a new reality that data communication across processors becomes more expensive than the computation. Besides, the flexibility of general-purpose computing devices also makes the computation energy-inefficient in many emerging tasks. Although some special-purpose accelerators have been employed to be orders of magnitude improvements such as GPUs, FPGA, and TPUs, most of them are customized to a single- or narrow-type of machine learning methods. Therefore, for future hardware, it is necessary to exploit both the performance and energy-efficiency of specialization while broadening compatible methods [95].

4.2.4 *Quantum Machine Learning*

Quantum computers exploit superposition and entanglement principles of quantum mechanics, obtaining an immense number of calculations in parallel. Compared to digital computers, they can reduce both execution time and energy consumption dramatically, such as IBM-Q. However, since the programming models of these quantum computers are fundamentally different from those of digital computers, more attention should be paid to redesign the corresponding machine learning methods [26], [162].

4.2.5 *Privacy Protection*

With the development of LML, increasing importance has been attached to privacy protection. Specifically, when personal and sensitive data are analyzed in the cloud or other distributed environments [8], it is necessary to ensure that the analysis will not violate the privacy of individuals. Recently, federate learning provides an alternative by bringing codes to edge devices and updating the global model with secure aggregation [30], [31].

5 Conclusions

Large-scale machine learning (LML) has significantly facilitated the data analysis in a mass scale over the past decades. However, despite these advances, the current LML requires further improvements to handle rapidly increasing data. In this paper, we first surveyed existing LML methods from three independent perspectives, namely, model simplification to reduce the computational complexity, optimization approximation to improve computational efficiency, and computation parallelism to enhance the computational capability. After that, we discussed the limitations of these methods and the possible extensions that can make further improvements. Besides, some important open issues in related areas are presented. We hope that this survey can provide a clean sketch on LML, and the discussions will advance the developments for next-generation methods.

Acknowledgement

This work is partially supported by the National Key Research and Development Program of China under grant 2016YFB1000901, and the National Nature Science Foundation of China under grants 61725203, 61732008, 61772171, 91746209, and U19A2079.

References

[1] "Baidu-allreduce," https://github.com/baidu-research/baidu-allreduce, 2017.

[2] "Multi-worker mirrored strategy," https://www.tensorflow.org/guide/distributed_training#multiworkermirroredstrategy, 2020.

[3] "Parallel learning of lightgbm," https://lightgbm.readthedocs.io/en/latest/Parallel-Learning-Guide.html, 2020.

[4] "Torch.distributed," https://pytorch.org/docs/stable/distributed, 2020.

Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on August 12,2020 at 03:51:46 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3015777, IEEE Transactions on Knowledge and Data Engineering

[5] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning." in *Proceedings of OSDI*, vol. 16, 2016, pp. 265–283.

[6] A. Agarwal and L. Bottou, "A lower bound for the optimization of finite sums," in *Proceedings of ICML*. ACM, 2015, pp. 78–86.

[7] A. Agarwal and J. C. Duchi, "Distributed delayed stochastic optimization," in *Proceedings of NeurIPS*, 2011, pp. 873–881.

[8] N. Agarwal, A. T. Suresh, F. Yu, S. Kumar, and H. B. Mcmahan, "cpsgd: Communication-efficient and differentially-private distributed sgd," in *Proceedings of NeurIPS*, 2018.

[9] S. Ahn, A. Korattikara, N. Liu, S. Rajan, and M. Welling, "Large-scale distributed bayesian matrix factorization using stochastic gradient mcmc," in *Proceedings of SIGKDD*, 2015, pp. 9–18.

[10] S. Ahn, A. Korattikara, and M. Welling, "Bayesian posterior sampling via stochastic gradient fisher scoring," *arXiv preprint arXiv:1206.6380*, 2012.

[11] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid, "Good practice in large-scale learning for image classification," *IEEE TPAMI*, vol. 36, no. 3, pp. 507–520, 2014.

[12] O. Y. Al-Jarrah, P. D. Yoo, S. Muhaidat, G. K. Karagiannidis, and K. Taha, "Efficient machine learning for big data: A review," *Big Data Research*, vol. 2, no. 3, pp. 87–93, 2015.

[13] G. Alain, A. Lamb, C. Sankar, A. Courville, and Y. Bengio, "Variance reduction in sgd by distributed importance sampling," *arXiv preprint arXiv:1511.06481*, 2015.

[14] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "Qsgd: Communication-efficient sgd via gradient quantization and encoding," in *Proceedings of NeurIPS*, 2017, pp. 1709–1720.

[15] O. M. Aodha, N. D. F. Campbell, J. Kautz, and G. J. Brostow, "Hierarchical subquery evaluation for active learning on a graph," in *Proceedings of CVPR*, 2014, pp. 564–571.

[16] Y. Arjevani and O. Shamir, "Communication complexity of distributed convex learning and optimization," in *Proceedings of NeurIPS*, 2015, pp. 1756–1764.

[17] Y. Baveye, E. Dellandréa, C. Chamaret, and L. Chen, "Deep learning vs. kernel methods: Performance for emotion prediction in videos," in *Proceedings of ACII*, 2015, pp. 77–83.

[18] I. Bayer, X. He, B. Kanagal, and S. Rendle, "A generic coordinate descent framework for learning from implicit feedback," in *Proceedings of WWW*, 2017, pp. 1341–1350.

[19] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM journal on imaging sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[20] M. Belkin, S. Ma, and S. Mandal, "To understand deep learning we need to understand kernel learning," *arXiv preprint arXiv:1802.01396*, 2018.

[21] Y. Ben-Haim and E. Tom-Tov, "A streaming parallel decision tree algorithm." *JMLR*, vol. 11, no. 2, 2010.

[22] S. Bengio, J. Weston, and D. Grangier, "Label embedding trees for large multi-class tasks," in *Proceedings of NeurIPS*, 2010, pp. 163–171.

[23] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 437–478.

[24] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE TPAMI*, vol. 35, no. 8, pp. 1798–1828, 2013.

[25] J. Bergstra, F. Bastien, O. Breuleux, P. Lamblin, R. Pascanu, O. Delalleau, G. Desjardins, D. Warde-Farley, I. Goodfellow, A. Bergeron *et al.*, "Theano: Deep learning on gpus with python," in *Proceedings of NeurIPS*, vol. 3. Citeseer, 2011, pp. 1–48.

[26] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, no. 7671, p. 195, 2017.

[27] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *JMLR*, vol. 3, no. Jan, pp. 993–1022, 2003.

[28] M. Boehm, M. W. Dusenberry, D. Eriksson, A. V. Evfimievski, F. M. Manshadi, N. Pansare, B. Reinwald, F. R. Reiss, P. Sen, A. C. Surve *et al.*, "Systemml: Declarative machine learning on spark," in *Proceedings of VLDB*, vol. 9, no. 13, 2016, pp. 1425–1436.

[29] M. Boehm, S. Tatikonda, B. Reinwald, P. Sen, Y. Tian, D. R. Burdick, and S. Vaithyanathan, "Hybrid parallelization strategies for large-scale machine learning in systemml," *Proceedings of VLDB*, vol. 7, no. 7, pp. 553–564, 2014.

[30] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecny, S. Mazzocchi, H. B. McMahan *et al.*, "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046*, 2019.

[31] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of SIGSAC*, 2017, pp. 1175–1191.

[32] L. Bottou and O. Bousquet, "The tradeoffs of large scale learning," in *Proceedings of NeurIPS*, 2008, pp. 161–168.

[33] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Review*, vol. 60, no. 2, pp. 223–311, 2018.

[34] D. Bouneffouf and I. Birol, "Sampling with minimum sum of squared similarities for nystrom-based large scale spectral clustering." in *Proceedings of IJCAI*, 2015, pp. 2313–2319.

[35] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[36] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[37] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," *arXiv preprint arXiv:1809.11096*, 2018.

[38] R. H. Byrd, S. L. Hansen, J. Nocedal, and Y. Singer, "A stochastic quasi-newton method for large-scale optimization," *SIAM Journal on Optimization*, vol. 26, no. 2, pp. 1008–1031, 2016.

[39] Z. Cai, Z. J. Gao, S. Luo, L. L. Perez, Z. Vagena, and C. Jermaine, "A comparison of platforms for implementing and running very large scale machine learning algorithms," in *Proceedings of SIGMOD*, 2014, pp. 1371–1382.

[40] D. Calandriello, I. Koutis, A. Lazaric, and M. Valko, "Improved large-scale graph learning through ridge spectral sparsification," in *Proceedings of ICML*, 2018.

[41] K.-W. Chang, C.-J. Hsieh, and C.-J. Lin, "Coordinate descent method for large-scale l2-loss linear support vector machines," *Journal of Machine Learning Research*, vol. 9, no. Jul, pp. 1369–1398, 2008.

[42] B. Chen, T. Medini, and A. Shrivastava, "Slide: In defense of smart algorithms over hardware acceleration for large-scale deep learning systems," *arXiv preprint arXiv:1903.03129*, 2019.

[43] C. Chen, D. Carlson, Z. Gan, C. Li, and L. Carin, "Bridging the gap between stochastic gradient mcmc and stochastic optimization," in *Artificial Intelligence and Statistics*, 2016, pp. 1051–1060.

[44] H. Chen, Y. Wang, C. Xu, B. Shi, C. Xu, Q. Tian, and C. Xu, "Addernet: Do we really need multiplications in deep learning?" *arXiv preprint arXiv:1912.13200*, 2019.

[45] J. Chen, H. R. Fang, and Y. Saad, "Fast approximate k nn graph construction for high dimensional data via recursive lanczos bisection," *JMLR*, vol. 10, no. 5, pp. 1989–2012, 2009.

[46] L. Chen, J. Lingys, K. Chen, and F. Liu, "Auto: scaling deep reinforcement learning for datacenter-scale automatic traffic optimization," in *Proceedings of SIGCOMM*, 2018, pp. 191–205.

[47] M. Chen, S. Mao, and Y. Liu, "Big data: A survey," *Mobile Networks and Applications*, vol. 19, no. 2, pp. 171–209, 2014.

[48] R. Chen, J. Shi, Y. Chen, B. Zang, H. Guan, and H. Chen, "Powerlyra: Differentiated graph computation and partitioning on skewed graphs," *ACM Transactions on Parallel Computing (TOPC)*, vol. 5, no. 3, pp. 1–39, 2019.

[49] T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," in *Proceedings of NeurIPS*, 2018, pp. 6571–6583.

[50] T. Chen, I. Cano, and T. Zhou, "Rabit: A reliable allreduce and broadcast interface," *Transfer*, vol. 3, no. 2, 2015.

[51] T. Chen, E. Fox, and C. Guestrin, "Stochastic gradient hamiltonian monte carlo," in *Proceedings of ICML*, 2014, pp. 1683–1691.

[52] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of SIGKDD*, 2016, pp. 785–794.

[53] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, "Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems," *arXiv preprint arXiv:1512.01274*, 2015.

[54] X.-W. Chen and X. Lin, "Big data deep learning: challenges and perspectives," *IEEE Access*, vol. 2, pp. 514–525, 2014.

[55] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "A survey of model compression and acceleration for deep neural networks," *arXiv preprint arXiv:1710.09282*, 2017.

1041-4347 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on August 12,2020 at 03:51:46 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3015777, IEEE Transactions on Knowledge and Data Engineering

[56] W.-L. Chiang, M.-C. Lee, and C.-J. Lin, "Parallel dual coordinate descent method for large-scale linear classification in multi-core environments," in *Proceedings of SIGKDD*, 2016, pp. 1485–1494.
[57] S. Chib and E. Greenberg, "Understanding the metropolis-hastings algorithm," *The american statistician*, vol. 49, no. 4, pp. 327–335, 1995.
[58] W.-S. Chin, Y. Zhuang, Y.-C. Juan, and C.-J. Lin, "A fast parallel stochastic gradient method for matrix factorization in shared memory systems," *ACM TIST*, vol. 6, no. 1, pp. 1–24, 2015.
[59] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *Proceedings of CVPR*, 2017, pp. 764–773.
[60] I. Daubechies, M. Defrise, and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Communications on Pure and Applied Mathematics*, vol. 57, no. 11, pp. 1413–1457, 2004.
[61] Y. Dauphin, H. de Vries, and Y. Bengio, "Equilibrated adaptive learning rates for non-convex optimization," in *Proceedings of NeurIPS*. IEEE, 2015, pp. 1504–1512.
[62] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le *et al.*, "Large scale distributed deep networks," in *Proceedings of NeurIPS*, 2012, pp. 1223–1231.
[63] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.
[64] A. Defazio, F. Bach, and S. Lacoste-Julien, "Saga: A fast incremental gradient method with support for non-strongly convex composite objectives," in *Proceedings of NeurIPS*, 2014, pp. 1646–1654.
[65] J. Deng, S. Satheesh, A. C. Berg, and F. Li, "Fast and balanced: Efficient label tree learning for large scale object recognition," in *Proceedings of NeurIPS*, 2011, pp. 567–575.
[66] L. Deng, "A tutorial survey of architectures, algorithms, and applications for deep learning," *APSIPA TSIP*, vol. 3, 2014.
[67] I. S. Dhillon, P. K. Ravikumar, and A. Tewari, "Nearest neighbor based greedy coordinate descent," in *Proceedings of NeurIPS*, 2011, pp. 2160–2168.
[68] T. Dozat, "Incorporating nesterov momentum into adam," 2016.
[69] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *JMLR*, vol. 12, no. Jul, pp. 2121–2159, 2011.
[70] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*. John Wiley & Sons, 2012.
[71] C. Dünner, T. Parnell, D. Sarigiannis, N. Ioannou, A. Anghel, G. Ravi, M. Kandasamy, and H. Pozidis, "Snap ml: A hierarchical framework for machine learning," in *Proceedings of NeurIPS*, 2018, pp. 250–260.
[72] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *Proceedings of ECCV*, 2014, pp. 834–849.
[73] A. Farahat, A. Ghodsi, and M. Kamel, "A novel greedy algorithm for nyström approximation," in *Proceedings of AISTATS*, 2011, pp. 269–277.
[74] K. Fatahalian, J. Sugerman, and P. Hanrahan, "Understanding the efficiency of gpu algorithms for matrix-matrix multiplication," in *Proceedings of ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, 2004, pp. 133–137.
[75] J. Friedman, T. Hastie, R. Tibshirani *et al.*, "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)," *The annals of statistics*, vol. 28, no. 2, pp. 337–407, 2000.
[76] J. H. Friedman, "Stochastic gradient boosting," *Computational statistics & data analysis*, vol. 38, no. 4, pp. 367–378, 2002.
[77] W. Fu, M. Wang, S. Hao, and T. Mu, "Flag: Faster learning on anchor graph with label predictor optimization," *IEEE TBD*, no. 1, pp. 1–1, 2017.
[78] W. Fu, M. Wang, S. Hao, and X. Wu, "Scalable active learning by approximated error reduction," in *Proceedings of SIGKDD*, 2018, pp. 1396–1405.
[79] Y. Fujiwara and G. Irie, "Efficient label propagation," in *Proceedings of ICML*, 2014, pp. 784–792.
[80] N. Gazagnadou, R. M. Gower, and J. Salmon, "Optimal mini-batch and step sizes for saga," *arXiv preprint arXiv:1902.00071*, 2019.
[81] T. V. Gestel, J. A. Suykens, G. Lanckriet, A. Lambrechts, B. D. Moor, and J. Vandewalle, "Bayesian framework for least-squares support vector machine classifiers, gaussian processes, and kernel fisher discriminant analysis," *Neural computation*, vol. 14, no. 5, pp. 1115–1147, 2002.
[82] N. Gilardi and S. Bengio, "Local machine learning models for spatial data analysis," *Journal of Geographic Information and Decision Analysis*, vol. 4, no. ARTICLE, pp. 11–28, 2000.
[83] A. Gittens and M. W. Mahoney, "Revisiting the nyström method for improved large-scale machine learning," *JMLR*, vol. 17, no. 1, pp. 3977–4041, 2016.
[84] J. E. Gonzalez, Y. Low, H. Gu, D. Bickson, and C. Guestrin, "Powergraph: distributed graph-parallel computation on natural graphs." in *Proceedings of OSDI*, vol. 12, no. 1, 2012, p. 2.
[85] J. E. Gonzalez, R. S. Xin, A. Dave, D. Crankshaw, M. J. Franklin, and I. Stoica, "Graphx: graph processing in a distributed dataflow framework," in *Proceedings of OSDI*, vol. 14, 2014, pp. 599–613.
[86] S. Gopal, "Adaptive sampling for sgd by exploiting side information," in *Proceedings of ICML*, 2016, pp. 364–372.
[87] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large mini-batch sgd: Training imagenet in 1 hour," *arXiv preprint arXiv:1706.02677*, 2017.
[88] E. Grave, A. Joulin, M. Cissé, D. Grangier, and H. Jégou, "Efficient softmax approximation for gpus," in *Proceedings of ICML*, 2017.
[89] T. L. Griffiths and M. Steyvers, "Finding scientific topics," *PNAS*, vol. 101, no. suppl 1, pp. 5228–5235, 2004.
[90] N. Halko, P.-G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM review*, vol. 53, no. 2, pp. 217–288, 2011.
[91] S. Hanneke, "The optimal sample complexity of pac learning," *JMLR*, vol. 17, no. 1, pp. 1319–1333, 2016.
[92] K. He and J. Sun, "Convolutional neural networks at constrained time cost," in *Proceedings of CVPR*, 2015, pp. 5353–5360.
[93] X. He, J. Tang, X. Du, R. Hong, T. Ren, and T.-S. Chua, "Fast matrix factorization with non-uniform weights on missing data," *IEEE TNNLS, to appear*, 2018.
[94] N. J. Higham, *Accuracy and stability of numerical algorithms*. Siam, 2002, vol. 80.
[95] M. D. Hill, S. Adve, L. Ceze, M. J. Irwin, D. Kaeli, M. Martonosi, J. Torrellas, T. F. Wenisch, D. Wood, and K. Yelick, "21st century computer architecture," *arXiv preprint arXiv:1609.06756*, 2016.
[96] Q. Ho, J. Cipar, H. Cui, S. Lee, J. K. Kim, P. B. Gibbons, G. A. Gibson, G. Ganger, and E. P. Xing, "More effective distributed ml via a stale synchronous parallel parameter server," in *Proceedings of NeurIPS*, 2013, pp. 1223–1231.
[97] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.
[98] C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan, "A dual coordinate descent method for large-scale linear svm," in *Proceedings of ICML*, 2008, pp. 408–415.
[99] C. Igel, V. Heidrich-Meisner, and T. Glasmachers, "Shark," *JMLR*, vol. 9, no. Jun, pp. 993–996, 2008.
[100] R. Jagerman, C. Eickhoff, and M. de Rijke, "Computing web-scale topic models using an asynchronous parameter server," in *Proceedings of SIGIR*, 2017, pp. 1337–1340.
[101] M. Jaggi, V. Smith, M. Takác, J. Terhorst, S. Krishnan, T. Hofmann, and M. I. Jordan, "Communication-efficient distributed dual coordinate ascent," in *Proceedings of NeurIPS*, 2014, pp. 3068–3076.
[102] S. Jeaugey, "Nccl 2.0," 2017.
[103] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. B. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of ACM MM*, 2014, pp. 675–678.
[104] J. Jiang, B. Cui, C. Zhang, and F. Fu, "Dimboost: Boosting gradient boosting decision tree to higher dimensions," in *Proceedings of ICDM*, 2018, pp. 1363–1376.
[105] J. Jiang, F. Fu, T. Yang, and B. Cui, "Sketchml: Accelerating distributed machine learning with data sketches," in *Proceedings of SIGMOD*, 2018, pp. 1269–1284.
[106] J. Jiang, L. Yu, J. Jiang, Y. Liu, and B. Cui, "Angel: a new large-scale machine learning system," *National Science Review*, vol. 5, no. 2, pp. 216–236, 2018.
[107] T. Joachims, "Training linear svms in linear time," in *Proceedings of SIGKDD*, 2006, pp. 217–226.

1041-4347 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on August 12,2020 at 03:51:46 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3015777, IEEE Transactions on Knowledge and Data Engineering

[108] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *Proceedings of NeurIPS*, 2013, pp. 315–323.

[109] A. J. Joshi, F. Porikli, and N. P. Papanikolopoulos, "Scalable active learning for multiclass image classification," *IEEE TPAMI*, vol. 34, no. 11, pp. 2259–2273, 2012.

[110] Y. Kalantidis and Y. Avrithis, "Locally optimized product quantization for approximate nearest neighbor search," in *Proceedings of CVPR*, 2014, pp. 2321–2328.

[111] V. Kalavri, V. Vlassov, and S. Haridi, "High-level programming abstractions for distributed graph processing," *IEEE TKDE*, vol. 30, no. 2, pp. 305–324, 2018.

[112] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Proceedings of NeurIPS*, 2017, pp. 3146–3154.

[113] N. Ketkar *et al.*, *Deep Learning with Python*. Springer, 2017, vol. 1.

[114] F. Khan, B. Mutlu, and X. Zhu, "How do humans teach: On curriculum learning and teaching dimension," in *Proceedings of NeurIPS*, 2011, pp. 1449–1457.

[115] J. Kim and J. F. Canny, "Interpretable learning for self-driving cars by visualizing causal attention." in *Proceedings of ICCV*, 2017, pp. 2961–2969.

[116] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[117] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of NeurIPS*, 2012, pp. 1097–1105.

[118] S. Kumar, M. Mohri, and A. Talwalkar, "Sampling methods for the nyström method," *JMLR*, vol. 13, no. Apr, pp. 981–1006, 2012.

[119] A. Kyrola, G. Blelloch, and C. Guestrin, "Graphchi: Large-scale graph computation on just a {PC}," in *OSDI*, 2012, pp. 31–46.

[120] S. Landset, T. M. Khoshgoftaar, A. N. Richter, and T. Hasanin, "A survey of open source tools for machine learning with big data in the hadoop ecosystem," *Journal of Big Data*, vol. 2, no. 1, p. 24, 2015.

[121] J. Langford, A. Smola, and M. Zinkevich, "Slow learners are fast," *arXiv preprint arXiv:0911.0491*, 2009.

[122] Q. V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A. Y. Ng, "On optimization methods for deep learning," in *Proceedings of ICML*, 2011, pp. 265–272.

[123] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[124] Y. T. Lee and A. Sidford, "Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems," in *IEEE FOCS*, 2013, pp. 147–156.

[125] H. Li and Z. Lin, "Accelerated proximal gradient methods for nonconvex programming," in *Proceedings of NeurIPS*, 2015, pp. 379–387.

[126] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su, "Scaling distributed machine learning with the parameter server," in *Proceedings of OSDI*, vol. 14, 2014, pp. 583–598.

[127] Q. Li, "Fast parallel machine learning algorithms for large datasets using graphic processing unit," *Ph. D. thesis, Virginia Commonwealth University*, 2011.

[128] S. S. Liew, M. Khalil-Hani, and R. Bakhteri, "Bounded activation functions for enhanced training stability of deep neural networks on visual pattern recognition problems," *Neurocomputing*, vol. 216, pp. 718–734, 2016.

[129] Q. Lin, Z. Lu, and L. Xiao, "An accelerated proximal coordinate gradient method," in *Proceedings of NeurIPS*, 2014, pp. 3059–3067.

[130] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," *arXiv preprint arXiv:1712.01887*, 2017.

[131] J. Liu, S. Wright, C. Ré, V. Bittorf, and S. Sridhar, "An asynchronous parallel stochastic coordinate descent algorithm," in *Proceedings of ICML*, 2014, pp. 469–477.

[132] W. Liu, J. He, and S.-F. Chang, "Large graph construction for scalable semi-supervised learning," in *Proceedings of ICML*, 2010, pp. 679–686.

[133] Y. Low, D. Bickson, J. E. Gonzalez, C. Guestrin, A. Kyrola, and J. M. Hellerstein, "Distributed graphlab: a framework for machine learning and data mining in the cloud," in *Proceedings of VLDB*, vol. 5, no. 8, 2012, pp. 716–727.

[134] J. Lu, S. C. Hoi, J. Wang, P. Zhao, and Z.-Y. Liu, "Large scale online kernel learning," *JMLR*, vol. 17, no. 1, pp. 1613–1655, 2016.

[135] S. Ma and M. Belkin, "Diving into the shallows: a computational perspective on large-scale shallow learning," in *Proceedings of NeurIPS*, 2017, pp. 3778–3787.

[136] Y.-A. Ma, T. Chen, and E. Fox, "A complete recipe for stochastic gradient mcmc," in *Proceedings of NeurIPS*, 2015, pp. 2917–2925.

[137] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, vol. 30, no. 1, 2013, p. 3.

[138] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski, "Pregel: a system for large-scale graph processing," in *Proceedings of SIGMOD*, 2010, pp. 135–146.

[139] P.-G. Martinsson, V. Rokhlin, and M. Tygert, "A randomized algorithm for the decomposition of matrices," *Applied and Computational Harmonic Analysis*, vol. 30, no. 1, pp. 47–68, 2011.

[140] B. McMahan and M. Streeter, "Delay-tolerant algorithms for asynchronous distributed online learning," in *Proceedings of NeurIPS*, 2014, pp. 2915–2923.

[141] R. Memisevic, C. Zach, M. Pollefeys, and G. E. Hinton, "Gated softmax classification," in *Proceedings of NeurIPS*, 2010, pp. 1603–1611.

[142] X. Meng, J. K. Bradley, B. Yavuz, E. R. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. B. Tsai, M. Amde, S. Owen *et al.*, "Mllib: machine learning in apache spark," *JMLR*, vol. 17, no. 34, pp. 1–7, 2016.

[143] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of NeurIPS*, 2013, pp. 3111–3119.

[144] F. Morin and Y. Bengio, "Hierarchical probabilistic neural network language model." in *Aistats*, vol. 5. Citeseer, 2005, pp. 246–252.

[145] P. Moritz, R. Nishihara, S. Wang, A. Tumanov, R. Liaw, E. Liang, M. Elibol, Z. Yang, W. Paul, M. I. Jordan *et al.*, "Ray: A distributed framework for emerging {AI} applications," in *Proceedings of OSDI*, 2018, pp. 561–577.

[146] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[147] B. Musel, L. Kauffmann, S. Ramanoël, C. Giavarini, N. Guyader, A. Chauvin, and C. Peyrin, "Coarse-to-fine categorization of visual scenes in scene-selective cortex," *Journal of Cognitive Neuroscience*, vol. 26, no. 10, pp. 2287–2297, 2014.

[148] M. Mutny and A. Krause, "Efficient high dimensional bayesian optimization with additivity and quadrature fourier features," in *Proceedings of NeurIPS*, 2018, pp. 9018–9029.

[149] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of ICML*, 2010, pp. 807–814.

[150] R. M. Neal *et al.*, "Mcmc using hamiltonian dynamics," *Handbook of markov chain monte carlo*, vol. 2, no. 11, p. 2, 2011.

[151] Y. Nesterov, "Efficiency of coordinate descent methods on huge-scale optimization problems," *SIAM Journal on Optimization*, vol. 22, no. 2, pp. 341–362, 2012.

[152] ——, "Gradient methods for minimizing composite functions," *Mathematical Programming*, vol. 140, no. 1, pp. 125–161, 2013.

[153] J. Nutini, M. Schmidt, I. Laradji, M. Friedlander, and H. Koepke, "Coordinate descent converges faster with the gauss-southwell rule than random selection," in *Proceedings of ICML*, 2015, pp. 1632–1641.

[154] B. C. Ooi, K.-L. Tan, S. Wang, W. Wang, Q. Cai, G. Chen, J. Gao, Z. Luo, A. K. Tung, Y. Wang *et al.*, "Singa: A distributed deep learning platform," in *Proceedings of ACM MM*, 2015, pp. 685–688.

[155] N. Parikh, S. Boyd *et al.*, "Proximal algorithms," *Foundations and Trends® in Optimization*, vol. 1, no. 3, pp. 127–239, 2014.

[156] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Proceedings of NeurIPS*, 2019, pp. 8024–8035.

[157] P. Patarasuk and X. Yuan, "Bandwidth optimal all-reduce algorithms for clusters of workstations," *Elsevier JPDC*, vol. 69, no. 2, pp. 117–124, 2009.

[158] S. Patterson and Y. W. Teh, "Stochastic gradient riemannian langevin dynamics on the probability simplex," in *Proceedings of NeurIPS*, 2013, pp. 3102–3110.

[159] Y. Peng, K. Chen, G. Wang, W. Bai, Y. Zhao, H. Wang, Y. Geng, Z. Ma, and L. Gu, "Towards comprehensive traffic forecasting in cloud computing: Design and application," *IEEE/ACM TON*, vol. 24, no. 4, pp. 2210–2222, 2016.

1041-4347 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on August 12,2020 at 03:51:46 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3015777, IEEE Transactions on Knowledge and Data Engineering

[160] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural networks*, vol. 12, no. 1, pp. 145–151, 1999.

[161] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Proceedings of NeurIPS*, 2008, pp. 1177–1184.

[162] P. Rebentrost, M. Mohseni, and S. Lloyd, "Quantum support vector machine for big data classification," *Physical review letters*, vol. 113, no. 13, p. 130503, 2014.

[163] B. Recht, C. Re, S. Wright, and F. Niu, "Hogwild: A lock-free approach to parallelizing stochastic gradient descent," in *Proceedings of NeurIPS*, 2011, pp. 693–701.

[164] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," *arXiv preprint arXiv:1904.09237*, 2019.

[165] M. Rhu, N. Gimelshein, J. Clemons, A. Zulfiqar, and S. W. Keckler, "vdnn: Virtualized deep neural networks for scalable, memory-efficient neural network design," in *MICRO*. IEEE, 2016, pp. 1–13.

[166] P. Richtárik and M. Takáč, "Distributed coordinate descent method for learning with big data," *JMLR*, vol. 17, no. 1, pp. 2657–2681, 2016.

[167] J. Rosen, N. Polyzotis, V. Borkar, Y. Bu, M. J. Carey, M. Weimer, T. Condie, and R. Ramakrishnan, "Iterative mapreduce for large scale machine learning," *arXiv preprint arXiv:1303.3517*, 2013.

[168] A. Roy, I. Mihailovic, and W. Zwaenepoel, "X-stream: Edge-centric graph processing using streaming partitions," in *Proceedings of SOSP*, 2013, pp. 472–488.

[169] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[170] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature Machine Intelligence*, vol. 1, no. 5, pp. 206–215, 2019.

[171] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[172] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE TSMC*, vol. 21, no. 3, pp. 660–674, 1991.

[173] L. Schmidt, S. Santurkar, D. Tsipras, K. Talwar, and A. Madry, "Adversarially robust generalization requires more data," in *Proceedings of NeurIPS*, 2018, pp. 5014–5026.

[174] M. Schmidt, N. Le Roux, and F. Bach, "Minimizing finite sums with the stochastic average gradient," *Mathematical Programming*, vol. 162, no. 1-2, pp. 83–112, 2017.

[175] M. Schmidt, N. L. Roux, and F. R. Bach, "Convergence rates of inexact proximal-gradient methods for convex optimization," in *Proceedings of NeurIPS*, 2011, pp. 1458–1466.

[176] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu, "1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns," in *Proceedings of INTERSPEECH*, 2014.

[177] A. Sergeev and M. Del Balso, "Horovod: fast and easy distributed deep learning in tensorflow," *arXiv preprint arXiv:1802.05799*, 2018.

[178] H.-J. M. Shi, S. Tu, Y. Xu, and W. Yin, "A primer on coordinate descent algorithms," *arXiv preprint arXiv:1610.00040*, 2016.

[179] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Proceedings of MSST*, 2010, pp. 1–10.

[180] S. Si, C. J. Hsieh, and I. S. Dhillon, "Memory efficient kernel approximation," in *Proceedings of ICML*, 2017, pp. 701–709.

[181] L. Sifre and S. Mallat, "Rigid-motion scattering for image classification," *Ph. D. thesis*, 2014.

[182] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[183] S. L. Smith, P.-J. Kindermans, C. Ying, and Q. V. Le, "Don't decay the learning rate, increase the batch size," 2018.

[184] M. Soltanolkotabi, A. Javanmard, and J. D. Lee, "Theoretical insights into the optimization landscape of over-parameterized shallow neural networks," *IEEE TIT*, 2017.

[185] S. Sonnenburg, S. Henschel, C. Widmer, J. Behr, A. Zien, F. d. Bona, A. Binder, C. Gehl, V. Franc *et al.*, "The shogun machine learning toolbox," *JMLR*, vol. 11, no. 6, pp. 1799–1802, 2010.

[186] B. Sriperumbudur and Z. Szabó, "Optimal rates for random fourier features," in *Proceedings of NeurIPS*, 2015, pp. 1144–1152.

[187] S. Sun, Z. Cao, H. Zhu, and J. Zhao, "A survey of optimization methods from a machine learning perspective," *IEEE Trans on Cybernetics*, 2019.

[188] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proceedings of ICML*, 2013, pp. 1139–1147.

[189] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of CVPR*, 2015, pp. 1–9.

[190] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of CVPR*, 2016, pp. 2818–2826.

[191] H. Tang, X. Lian, T. Zhang, and J. Liu, "Doublesqueeze: Parallel stochastic gradient descent with double-pass error-compensated compression," *arXiv preprint arXiv:1905.05957*, 2019.

[192] A. Torralba, B. C. Russell, and J. Yuen, "Labelme: Online image annotation and applications," *Proceedings of IEEE*, vol. 98, no. 8, pp. 1467–1484, 2010.

[193] C.-W. Tsai, C.-F. Lai, H.-C. Chao, and A. V. Vasilakos, "Big data analytics: a survey," *Journal of Big data*, vol. 2, no. 1, p. 21, 2015.

[194] I. W. Tsang, A. Kocsor, and J. T. Kwok, "Simpler core vector machines with enclosing balls," in *Proceedings of ICML*, 2007, pp. 911–918.

[195] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth *et al.*, "Apache hadoop yarn: yet another resource negotiator," in *Proceedings of SOCC*, 2013, p. 5.

[196] J. Von Neumann, *The computer and the brain*. Yale University Press, 2012.

[197] J. Wang, J. Wang, G. Zeng, Z. Tu, R. Gan, and S. Li, "Scalable k-nn graph construction for visual descriptors," in *Proceedings of CVPR*, 2012, pp. 1106–1113.

[198] J. Wang, H. T. Shen, J. Song, and J. Ji, "Hashing for similarity search: A survey," *arXiv preprint arXiv:1408.2927*, 2014.

[199] J. Wang, T. Zhang, N. Sebe, H. T. Shen *et al.*, "A survey on learning to hash," *IEEE TPAMI*, vol. 40, no. 4, pp. 769–790, 2018.

[200] M. Wang, W. Fu, S. Hao, H. Liu, and X. Wu, "Learning on big graph: Label inference and regularization with anchor hierarchy," *IEEE TKDE*, vol. 29, no. 5, pp. 1101–1114, 2017.

[201] M. Wang, W. Fu, S. Hao, D. Tao, and X. Wu, "Scalable semi-supervised learning by efficient anchor graph regularization," *IEEE TKDE*, vol. 28, no. 7, pp. 1864–1877, 2016.

[202] M. Welling and Y. W. Teh, "Bayesian learning via stochastic gradient langevin dynamics," in *Proceedings of ICML*, 2011, pp. 681–688.

[203] W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, and H. Li, "Terngrad: Ternary gradients to reduce communication in distributed deep learning," in *Proceedings of NeurIPS*, 2017, pp. 1509–1519.

[204] S. J. Wright, "Coordinate descent algorithms," *Mathematical Programming*, vol. 151, no. 1, pp. 3–34, 2015.

[205] J. Wu, W. Huang, J. Huang, and T. Zhang, "Error compensated quantized sgd and its applications to large-scale distributed optimization," *arXiv preprint arXiv:1806.08054*, 2018.

[206] R. Wu, S. Yan, Y. Shan, Q. Dang, and G. Sun, "Deep image: Scaling up image recognition," *arXiv preprint arXiv:1501.02876*, vol. 7, no. 8, 2015.

[207] X. Wu, X. Zhu, G.-Q. Wu, and W. Ding, "Data mining with big data," *IEEE TKDE*, vol. 26, no. 1, pp. 97–107, 2014.

[208] W. Xiao, R. Bhardwaj, R. Ramjee, M. Sivathanu, N. Kwatra, Z. Han, P. Patel, X. Peng, H. Zhao, Q. Zhang *et al.*, "Gandiva: introspective cluster scheduling for deep learning," in *Proceedings of OSDI*, 2018, pp. 595–610.

[209] E. P. Xing, Q. Ho, W. Dai, J. K. Kim, J. Wei, S. Lee, X. Zheng, P. Xie, A. Kumar, and Y. Yu, "Petuum: A new platform for distributed machine learning on big data," *IEEE TBD*, vol. 1, no. 2, pp. 49–67, 2015.

[210] B. Xu, J. Bu, C. Chen, D. Cai, X. He, W. Liu, and J. Luo, "Efficient manifold ranking for image retrieval," in *Proceedings of SIGIR*, 2011, pp. 525–534.

[211] C. Xu, D. Tao, and C. Xu, "A survey on multi-view learning," *arXiv preprint arXiv:1304.5634*, 2013.

[212] W. Xu, "Towards optimal one pass large scale learning with averaged stochastic gradient descent," *arXiv preprint arXiv:1107.2490*, 2011.

[213] Z. Yan, H. Zhang, R. Piramuthu, V. Jagadeesh, D. DeCoste, W. Di, and Y. Yu, "Hd-cnn: hierarchical deep convolutional neural networks for large scale visual recognition," in *Proceedings of ICCV*, 2015, pp. 2740–2748.

[214] T. Yang, Y.-F. Li, M. Mahdavi, R. Jin, and Z.-H. Zhou, "Nyström method vs random fourier features: A theoretical and empirical

1041-4347 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on August 12,2020 at 03:51:46 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3015777, IEEE Transactions on Knowledge and Data Engineering

comparison," in *Advances in neural information processing systems*, 2012, pp. 476–484.

[215] Y. Yang, M. Pilanci, M. J. Wainwright *et al.*, "Randomized sketches for kernels: Fast and optimal nonparametric regression," *The Annals of Statistics*, vol. 45, no. 3, pp. 991–1023, 2017.

[216] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *Proceedings of NeurIPS*, 2018, pp. 4801–4811.

[217] D. M. Young, *Iterative solution of large linear systems*. Elsevier, 2014.

[218] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *Proceedings of ICLR*, 2016.

[219] M. Yu, Z. Lin, K. Narra, S. Li, Y. Li, N. S. Kim, A. Schwing, M. Annavaram, and S. Avestimehr, "Gradiveq: Vector quantization for bandwidth-efficient gradient aggregation in distributed cnn training," in *Proceedings of NeurIPS*, 2018, pp. 5125–5135.

[220] G.-X. Yuan, C.-H. Ho, and C.-J. Lin, "Recent advances of large-scale linear classification," *Proceedings of IEEE*, vol. 100, no. 9, pp. 2584–2603, 2012.

[221] S. Yun and K.-C. Toh, "A coordinate gradient descent method for l1-regularized convex minimization," *Computational Optimization and Applications*, vol. 48, no. 2, pp. 273–307, 2011.

[222] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. Mccauley, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing," in *Proceedings of NSDI*, 2012.

[223] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: cluster computing with working sets," in *Proceedings of Hot Topics in Cloud Computing*, 2010, pp. 10–10.

[224] M. Zaharia, T. Das, H. Li, T. Hunter, S. Shenker, and I. Stoica, "Discretized streams: fault-tolerant streaming computation at scale," in *Proceedings of SOSP*, 2013, pp. 423–438.

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

[226] H. Zhang, J. Li, K. Kara, D. Alistarh, J. Liu, and C. Zhang, "Zipml: Training linear models with end-to-end low precision, and a little bit of deep learning," in *Proceedings of ICML*, 2017, pp. 4035–4043.

[227] H. Zhang, F. Shen, W. Liu, X. He, H. Luan, and T.-S. Chua, "Discrete collaborative filtering," in *Proceedings of SIGIR*, 2016, pp. 325–334.

[228] K. Zhang, L. Lan, J. T. Kwok, S. Vucetic, and B. Parvin, "Scaling up graph-based semisupervised learning via prototype vector machines," *IEEE TNNLS*, vol. 26, no. 3, pp. 444–457, 2015.

[229] K. Zhang, I. W. Tsang, and J. T. Kwok, "Improved nyström low-rank approximation and error analysis," in *Proceedings of ICML*, 2008, pp. 1232–1239.

[230] Q. Zhang, L. T. Yang, Z. Chen, and P. Li, "A survey on deep learning for big data," *Information Fusion*, vol. 42, pp. 146–157, 2018.

[231] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proceedings of CVPR*, 2018, pp. 6848–6856.

[232] Y.-M. Zhang, K. Huang, G. Geng, and C.-L. Liu, "Fast knn graph construction with locality sensitive hashing," in *ECML PKDD*, 2013, pp. 660–674.

[233] Z. Zhang, B. Cui, Y. Shao, L. Yu, J. Jiang, and X. Miao, "Ps2: Parameter server on spark," in *Proceedings of ICDM*, 2019, pp. 376–388.

[234] L. Y. Zhang S, Choromanska AE, "Deep learning with elastic averaging sgd," in *Proceedings of NeurIPS*, 2015, pp. 685–693.

[235] T. Zhao, M. Yu, Y. Wang, R. Arora, and H. Liu, "Accelerated mini-batch randomized block coordinate descent method," in *Proceedings of NeurIPS*, 2014, pp. 3329–3337.

[236] S. Zheng, Q. Meng, T. Wang, W. Chen, N. Yu, Z.-M. Ma, and T.-Y. Liu, "Asynchronous stochastic gradient descent with delay compensation," in *Proceedings of ICML*, 2017, pp. 4120–4129.

[237] Z.-H. Zhou, "A brief introduction to weakly supervised learning," *National Science Review*, vol. 5, no. 1, pp. 44–53, 2017.

[238] X. Zhu, W. Han, and W. Chen, "Gridgraph: Large-scale graph processing on a single machine using 2-level hierarchical partitioning," in *USENIX ATC*, 2015, pp. 375–386.

[239] X. Zhu, H. Hu, S. Lin, and J. Dai, "Deformable convnets v2: More deformable, better results," in *Proceedings of CVPR*, 2019, pp. 9308–9316.

[240] M. Zinkevich, M. Weimer, L. Li, and A. J. Smola, "Parallelized stochastic gradient descent," in *Proceedings of NeurIPS*, 2010, pp. 2595–2603.

Meng Wang received the B.E. and Ph.D. degrees in special class for the gifted young from the Department of Electronic Engineering and Information Science, University of Science and Technology of China (USTC), Hefei, China, in 2003 and 2008, respectively. He is currently a professor with the Hefei University of Technology (HFUT), China. He has authored over 200 book chapters, journal, and conference papers in his research areas. His current research interests include multimedia content analysis, computer-vision, and pattern recognition. He was a recipient of the ACM SIGMM Rising Star Award in 2014. He is an Associate Editor of the IEEE TKDE, the IEEE TCSVT, and the IEEE TNNLS.

Weijie Fu is pursuing his Ph.D. degree in the School of Computer Science and Information Engineering, Hefei University of Technology (H-FUT). His current research interest focuses on machine learning and data mining.

Xiangnan He is currently a professor with the University of Science and Technology of China (USTC). He received his Ph.D. in Computer Science from National University of Singapore (NUS) in 2016, and did postdoctoral research in NUS until 2018. His research interests span information retrieval, data mining, and multimedia analytics. He has over 50 publications appeared in several top conferences such as SIGIR, WWW, and MM, and journals including TKDE, TOIS, and TMM. His work on recommender systems has received the Best Paper Award Honourable Mention in WWW 2018 and ACM SIGIR 2016. Moreover, he has served as the PC member for several top conferences including SIGIR, WWW, MM, KDD etc., and the regular reviewer for journals including TKDE, TOIS, TMM, TNNLS etc.

Shijie Hao is an associate professor at the Hefei University of Technology (HFUT), China. He received his B.E., M.S. and Ph.D. Degree in the School of Computer Science and Information Engineering from HFUT. His current research interests include machine learning and image processing.

Xindong Wu is Yangtze River Scholar in the School of Computer Science and Information Engineering at the Hefei University of Technology (HFUT), China, and a Fellow of the IEEE. He received his Bachelor's and Master's degrees in Computer Science from the Hefei University of Technology, China, and his Ph.D. degree in Artificial Intelligence from the University of Edinburgh, Britain. His research interests include data mining, knowledge-based systems, and Web information exploration. Dr. Wu is the Steering Committee Chair of the IEEE ICDM, the Editor-in-Chief of KAIS, by Springer, and a Series Editor of the Springer Book Series on AI&KP. He was the Editor-in-Chief of the IEEE TKDE between 2005 and 2008. He served as Program Committee Chair/Co-Chair for ICDM '03, KDD-07, and CIKM 2010.

1041-4347 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on August 12,2020 at 03:51:46 UTC from IEEE Xplore. Restrictions apply.

Scalable Active Learning by Approximated Error Reduction

Weijie Fu
Hefei University of Technology
Hefei, China
fwj.edu@gmail.com

Meng Wang
Hefei University of Technology
Hefei, China
eric.mengwang@gmail.com

Shijie Hao
Hefei University of Technology
Hefei, China
hfut.hsj@gmail.com

Xindong Wu
Hefei University of Technology
Hefei, China
xwu@hfut.edu.cn

ABSTRACT

We study the problem of active learning for multi-class classification on large-scale datasets. In this setting, the existing active learning approaches built upon uncertainty measures are ineffective for discovering unknown regions, and those based on expected error reduction are inefficient owing to their huge time costs. To overcome the above issues, this paper proposes a novel query selection criterion called approximated error reduction (AER). In AER, the error reduction of each candidate is estimated based on an expected impact over all datapoints and an approximated ratio between the error reduction and the impact over its nearby datapoints. In particular, we utilize hierarchical anchor graphs to construct the candidate set as well as the nearby datapoint sets of these candidates. The benefit of this strategy is that it enables a hierarchical expansion of candidates with the increase of labels, and allows us to further accelerate the AER estimation. We finally introduce AER into an efficient semi-supervised classifier for scalable active learning. Experiments on publicly available datasets with the sizes varying from thousands to millions demonstrate the effectiveness of our approach.

KEYWORDS

active learning, query selection, efficient algorithms

ACM Reference Format:
Weijie Fu, Meng Wang, Shijie Hao, and Xindong Wu. 2018. Scalable Active Learning by Approximated Error Reduction. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3219819.3219954

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
KDD '18, August 19–23, 2018, London, United Kingdom
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5552-0/18/08...$15.00
https://doi.org/10.1145/3219819.3219954

1 INTRODUCTION

With the explosive growth of datasets, supervised learning and semi-supervised learning have been broadly used in many multi-class classification tasks, such as speech recognition [6], image classification [4], and data mining [5]. The former directly employs labeled data to train its classifier, while the latter further exploits the prior knowledge from unlabeled data to improve the classification.

To obtain satisfactory performance, classifiers require high-quality labeled data. Active learning that selects valuable queries to label has been studied to address this problem [9], [18]. The uncertainty-based sampling is the simplest query selection criterion [12]. However, as the methods based on this criterion consider each datapoint independently, they ignore the accuracy improvement on other datapoints after labeling the selected query. Although several density-weighting approaches were developed to relieve this issue [14], [25], they are still insufficiently effective to discover unknown regions, especially at the early phase of query selection.

An alternative active learning criterion called expected error reduction (EER) was therefore proposed. In general, EER makes a tradeoff on the reduction in generalization errors achieved by either labeling an unknown region or tuning decision boundaries under its current classifier, which leads to impressive performance [1], [29]. Nevertheless, EER brings a huge time cost owing to its error reduction estimation. That is, for each datapoint, the classifier has to be re-optimized with its possible labels and the labels of other datapoints need to be re-inferred to calculate its expected generalization error. As a result, even scalable classifiers are employed [16], [23], [28], the EER-based query selection is still inefficient for active learning on large-scale datasets.

To overcome the above issues, this paper proposes a novel criterion called approximated error reduction (AER). According to AER, we estimate the error reduction of a candidate based on an expected impact over all datapoints, and an approximated ratio between the error reduction and the impact over its nearby datapoints. Meanwhile, a hierarchical anchor graph [23] is utilized to build the candidate set as well as the nearby datapoint sets of these candidates. Of note, the construction of the hierarchical anchor graph is efficient, and the anchor sets and the datapoint set on this graph establish coarse-to-fine coverings of the data distribution. As a consequence, it allows us to expand the candidate set hierarchically with the increase of labeled queries, which can further accelerate the AER estimation. Finally, by introducing the proposed AER criterion into a scalable semi-supervised classifier, we obtain an efficient and effective active learning approach for query selection on large-scale datasets. The promising results on benchmark datasets highlight the superior performance of our approach[1].

The main contributions of our work are as follows.

- We propose a novel AER criterion for query selection. Compared with EER, it enables an efficient estimation of the error reduction without re-inferring labels of massive datapoints. We also utilize a hierarchical anchor graph to construct a small candidate set, which allows us to further accelerate the AER estimation.
- We introduce the AER criterion into a scalable semi-supervised classifier for active learning. Meanwhile, we develop a fast algorithm to calculate the expected impact over all datapoints for all candidates, which can be performed via direct matrix operations rather than multiple iterations.
- We show that, apart from the similar time cost to that of the uncertainty-based sampling, the remaining time cost of our AER-based approach is independent of data sizes during the query selection. Furthermore, our AER-based approach can still achieve comparable or even higher accuracies than the EER-based approach. The experimental results on different types of datasets demonstrate the effectiveness of our approach.

The rest of this paper is organized as follows. In Section 2, we review the related work on active learning. In Section 3, we introduce the preliminaries of hierarchical anchor graphs and an efficient semi-supervised classifier. In Section 4, we propose the AER criterion and use it for scalable active learning. Section 5 validates the strengths of our approach on different-size datasets, and Section 6 concludes this paper.

2 RELATED WORK

Recent years have witnessed a number of studies on active learning for searching valuable queries and reducing manual labeling costs [15], [25]. In particular, discriminative active learning has obtained satisfactory performance in many real-world applications. Different from representative active learning that only considers the feature spaces of data distributions [2], [26], these approaches are prediction dependent and can query informative instances to facilitate the improvement of the classifier for a higher accuracy.

Uncertainty-based sampling is the simplest and most widely used discriminative criterion [13]. The methods built upon this criterion generally select the query that is the least certain, where different uncertainty measures can be used, such as entropy and ℓ_p loss. In particular, Joshi et al. [10] proposed to estimate the uncertainty by merely using the probabilities of the best and the second best classes. As these approaches are prone to outliers, some methods combining representativeness were also developed [9], [14], [21], [25]. For example, Settles et al. [21] proposed to select queries based on a density-weighting uncertainty with the cosine similarity. Li et al. [14] proposed to employ the mutual information rather than the marginal density. However, the former faces a challenge of combing two different measures, and the latter has to estimate the mutual information with a cubic time cost with respect to data sizes. Recently, Dasarathy et al. [7] proposed to select uncertain queries based on the structure of a graph, which is inefficient when the number of the datapoints along decision boundaries is large.

Instead of only considering the classifiers based on current labels, one can further exploit re-optimized classifiers by giving possible labels on unlabeled datapoints. EER therefore has become an effective query selection criterion by directly minimizing the generalization error [1], [29]. Nevertheless, it also leads to the most expensive query selection approaches, as classifiers have to be re-optimized with each possible label and the labels of massive datapoints need to be re-inferred. Although Zhu et al. [29] proposed a novel method to update the label matrix of unlabeled data, and Aodha et al. [1] presented a hierarchical subquery approach for the EER estimation, they are still impractical for large-size datasets, owing to the inevitable huge time cost of classifier initialization.

Another criterion that considers possible labels is expected model change, which selects the query with the greatest expected change on the parameters of a classifier [20]. Compared with EER, it does not require the label re-inference for datapoints, which remarkably reduces time costs. When a classifier is trained with gradient-based optimization, it is equal to select the query that creates the largest change on the gradient of the objective function [3]. However, this criterion ignores the importance of the parameters corresponding to different features, which in turn reduces its effectiveness.

In short, the above criteria either do not consider the error reduction over all datapoints, or face a huge time cost in estimating error reduction. In contrast, our AER criterion can obtain an efficient error reduction estimation, which brings significant advantages for scalable active learning.

3 PRELIMINARIES

To better present our work, we introduce the preliminaries of hierarchical anchor graphs and a scalable semi-supervised classifier. Some important notations are listed in Table 1.

3.1 Hierarchical Anchor Graph

We first introduce hierarchical anchor graphs [23]. An illustrative example of graphs is shown in Fig.1

Let $\mathcal{X}_0 \in \mathbb{R}^{N_0 \times d}$ indicate the set of datapoints, and each $\mathcal{X}_b \in \mathbb{R}^{N_b \times d}$ (b=1, ..., h) denote a small set of anchors (landmark datapoints) that roughly cover data distributions [16]. A hierarchical anchor graph can be constructed based on the following constraints: (1). *Fine-to-Coarse Coverings.* The sets of anchors share the same feature space of the datapoint set, and their sizes are gradually reduced with $N_1 > \ldots > N_h$.

[1] Both the datasets and codes are available at http://github.com/fuweijie/AER

Table 1: Notations and Definitions

Notation	Definition
$\mathcal{X}_0$	The set of datapoints with the dimension d.
N_0	The number of datapoints.
h	The number of anchor sets.
$\mathcal{X}_b$	The b-th set of anchors ($h \geq b \geq 1$).
N_b	The number of anchors in $\mathcal{X}_b$ ($h \geq b \geq 1$).
C	The number of classes.
$\mathbf{Z}^{b-1,b}$	The inter-set adjacency matrix between $\mathcal{X}_{b-1}$ and $\mathcal{X}_b$.
$\mathbf{Z}^{\mathrm{H}}$	The cascaded inter-set adjacency matrix.
$\lceil \cdot \rceil$	The nearest points in the connected coarser set.
$\mathbf{A}$	The soft label matrix of the coarsest anchor set.
$\mathbf{A}^{+\hat{y}_{qr}}$	The updated matrix with an extra label r on $\mathbf{x}_q$.
$\mathbf{F}$	The soft label matrix of the datapoint set.
$\mathbf{Y}_{\mathrm{L}}$	The class indicator matrix on labeled datapoints.
$\bar{\mathcal{E}}$	The average estimated error based on labeled data.
$\mathcal{S}_{\mathrm{AL}}$	The set of candidates for active learning.
N_q	The number of candidates in $\mathcal{S}_{\mathrm{AL}}$.
$\mathcal{I}_q$	The expected impact over all datapoints of $\mathbf{x}_q$.
$\langle q \rangle$	The indices of the nearby datapoints of $\mathbf{x}_q$.
$N_{\langle q \rangle}$	The number of the nearby datapoints of $\mathbf{x}_q$.
$\mathcal{E}r_{\langle q \rangle}$	The error reduction over the nearby datapoints of $\mathbf{x}_q$.
$\mathbb{E}(\mathcal{E}^{+\hat{y}_{qr}}_{\langle q \rangle})$	The generalization error over the nearby datapoints.
$\mathcal{I}_{\langle q \rangle}$	The impact over the nearby datapoints of $\mathbf{x}_q$.
$\lfloor q \rfloor$	The set of points whose nearest anchor in the connected coarser set is $\mathbf{x}_q$.

These anchor sets bring fine-to-coarse coverings of the data distribution. (2). *Pyramidal Structure.* Let $\mathcal{G}$ denote a multiple-set pyramidal graph. The original datapoints in $\mathcal{X}_0$ locate at the bottom layer of the pyramid, and the remaining layers are all composed of fine-to-coarse anchor sets with $\mathcal{X}_1$,...,$\mathcal{X}_h$. (3). *Inter-set Adjacency.* The datapoint set and all anchor sets are linked up to a complete graph with h sets of inter-set adjacency edges between the neighboring sets, such as $\mathbf{Z}^{b-1,b} \in \mathbb{R}^{N_{b-1} \times N_b}$ between $\mathcal{X}_{b-1}$ and $\mathcal{X}_b$.

In the above graph model, we denote anchors in $\mathcal{X}_1$ and $\mathcal{X}_h$ as the finest anchors and the coarsest anchors, respectively. In particular, if the graph only contains one anchor set (h=1), we denote anchors in $\mathcal{X}_1$ as the coarsest anchors for convenience. Besides, we use 'N_1-N_2-...-N_h-anchor-graph' to indicate a hierarchical anchor graph built upon h anchor sets with N_1,N_2,...,N_h anchors in different anchor sets. More details of the setting of anchor sets can be found in [23].

The remaining issues of the graph construction involve two aspects, including the generation of anchor sets and the inter-set adjacency estimation between the neighboring sets. To obtain an anchor set, we can perform a fast clustering algorithm on the datapoint set with a predetermined number of centers [16]. Here we briefly describe the adjacency estimation. Specifically, for each $\mathbf{Z}^{b-1,b} \in \mathbb{R}^{N_{b-1} \times N_b}$, its weights can be determined by the kernel regression [16]:

$$Z^{b-1,b}_{ij} = \frac{K_\sigma(\mathbf{x}_i, \mathbf{x}_j)}{\sum_{j' \in \lceil i \rceil} K_\sigma(\mathbf{x}_i, \mathbf{x}_{j'})}, \ \forall j \in \lceil i \rceil, \qquad (1)$$

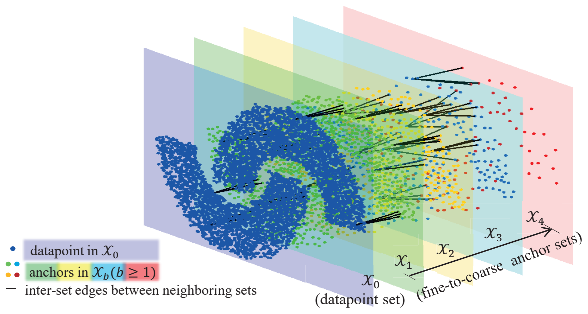

datapoint in $\mathcal{X}_0$
anchors in $\mathcal{X}_b (b \geq 1)$
inter-set edges between neighboring sets
$\mathcal{X}_0$
(datapoint set)
$\mathcal{X}_1$
$\mathcal{X}_2$
$\mathcal{X}_3$
$\mathcal{X}_4$
(fine-to-coarse anchor sets)

Figure 1: An example of hierarchical anchor graphs.

where σ is the bandwidth of the Gaussian kernel, $\mathbf{x}_i$ is the i-th point in $\mathcal{X}_{b-1}$, and $\lceil i \rceil$ is the set of indices of its s nearest anchors in the connected coarser set $\mathcal{X}_b$. For large-scale datasets, we can speed up the weight estimation with the approximate nearest neighbor search [17], which reduces the cost of the graph construction to $O(N_0 \log N_1)$.

3.2 Scalable Semi-Supervised Learning

Efficient semi-supervised classifiers have been proposed for large-scale classification [16], [23], [28]. Here we introduce a scalable one built upon the above graph, which has shown its effectiveness on many multi-class classification tasks [23].

Let $\mathcal{D}$=$\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{N_{\mathrm{L}}}, y_{N_{\mathrm{L}}}), \ldots, \mathbf{x}_{N_0}\}$ be the dataset where the first N_{L} datapoints are labeled from C distinct classes. Let $\mathbf{A}$ denote the soft label matrix on the anchors in $\mathcal{X}_h$ that needs to be optimized. Based on the inter-set adjacency in the hierarchical anchor graph, the soft label matrix on datapoints ($\mathbf{F}$) can be inferred in a hierarchical manner: $\mathbf{F}$=$\mathbf{Z}^{\mathrm{H}}\mathbf{A} \in \mathbb{R}^{N_0 \times C}$, where $\mathbf{Z}^{\mathrm{H}}$=$\mathbf{Z}^{0,1}(\ldots(\mathbf{Z}^{h-1,h})) \in \mathbb{R}^{N_0 \times N_h}$ is the cascaded inter-set matrix between $\mathcal{X}_0$ and $\mathcal{X}_h$. Let $\mathbf{\Lambda}$ be a diagonal matrix with Λ_{jj}=$\sum_{i=1}^{N_0} Z^{0,1}_{ij}$, and $\mathbf{rL}$=${\mathbf{Z}^{\mathrm{H}}}^{\mathrm{T}}\mathbf{Z}^{\mathrm{H}}$ $-{\mathbf{Z}^{\mathrm{H}}}^{\mathrm{T}}\mathbf{Z}^{0,1}\mathbf{\Lambda}^{-1}{\mathbf{Z}^{0,1}}^{\mathrm{T}}\mathbf{Z}^{\mathrm{H}}$ be the reduced Laplacian matrix on the graph. Besides, denote $\mathbf{Y}_{\mathrm{L}}$=$[\mathbf{y}_1; \ldots; \mathbf{y}_{N_{\mathrm{L}}}] \in \mathbb{R}^{N_{\mathrm{L}} \times C}$ as the class indicator matrix of the labeled data, where y_{ir}=1 if $\mathbf{x}_i$ belongs to the class r, and y_{ir}=0 otherwise. Let $\mathbf{Z}^{\mathrm{H}}_{\mathrm{L}}$ be the labeled part of $\mathbf{Z}^{\mathrm{H}}$, and λ be the parameter that weighs the fitting constraint against the smoothness constraint in manifold regularization. Hierarchial anchor graph regularization (HAGR) obtains an optimal solution of $\mathbf{A}$ in a closed form:

$$\mathbf{A} = \left({\mathbf{Z}^{\mathrm{H}}_{\mathrm{L}}}^{\mathrm{T}}\mathbf{Z}^{\mathrm{H}}_{\mathrm{L}} + \lambda \mathbf{rL}\right)^{-1}{\mathbf{Z}^{\mathrm{H}}_{\mathrm{L}}}^{\mathrm{T}}\mathbf{Y}_{\mathrm{L}} \in \mathbb{R}^{N_h \times C} \qquad (2)$$

with a time cost of $O({N_h}^3)$, where N_h is the size of $\mathcal{X}_h$.

Finally, HAGR employs the soft labels of the anchors in $\mathcal{X}_h$ to infer the label of any unlabeled datapoint in $\mathcal{X}_0$:

$$\mathrm{argmax}_{r \in \{1,\ldots,C\}} \frac{\mathbf{Z}^{\mathrm{H}}_{i\cdot} \times \mathbf{A}_{\cdot r}}{\pi_r}, i = N_{\mathrm{L}} + 1, \ldots, N_0, \qquad (3)$$

where $\mathbf{A}_{\cdot r}$ is the r-th column of $\mathbf{A}$, and π_r=$\mathbf{1}^{\mathrm{T}}\mathbf{Z}^{\mathrm{H}}\mathbf{A}_{\cdot r}$ is the normalization factor [16]. As we have obtained $\mathbf{Z}^{\mathrm{H}}$, this label inference can be performed with a time cost of $O(N_0 N_h C)$.

4 PROPOSED APPROACH

In Section 4.1, we propose a novel query selection criterion called approximated error reduction (AER). We also introduce its implementing details based on hierarchical anchor graphs for scalable active learning. In Section 4.2, we present our AER-based approach. Section 4.3 analyzes its time cost, and Section 4.4 makes a comparison to other criteria.

4.1 Approximated Error Reduction

In AER, to obtain valuable queries, the error reduction of a candidate is estimated based on an expected impact over all datapoints and an approximated ratio between the error reduction and the impact over its nearby datapoints.

Suppose $\mathbf{f}_i$ is the soft label assignment of $\mathbf{x}_i$ based on current labeled datapoints, and $\widehat{\mathbf{f}}_i$ is the hard indicator vector with $\widehat{f}_{ir}$=1 if r=argmax$_r f_{ir}$ and $\widehat{f}_{ir}$=0 otherwise. Let $\mathcal{S}_{\mathrm{AL}}$ be the candidate set. For each candidate $\mathbf{x}_q \in \mathcal{S}_{\mathrm{AL}}$, we first calculate its expected impact over all datapoints:

$$\mathcal{I}_q = \sum_{r=1}^{C} f_{qr} \sum_{i=1}^{N_0} \ell(\mathbf{f}_i, \mathbf{f}_i^{+\hat{y}_{qr}}), \tag{4}$$

where $\mathbf{f}_i^{+\hat{y}_{qr}}$s are the re-inferred soft labels based on the current labeled datapoints and $\mathbf{x}_q$ with the label r, and ℓ denotes the l_2 loss. As we can see, $\mathcal{I}_q$ calculates the change on the soft labels of all datapoints by assuming new labels on $\mathbf{x}_q$.

Then, we consider the ratio between the error reduction and the impact of $\mathbf{x}_q$. Instead of an exact ratio built upon all datapoints, AER only requires an approximated one for $\mathbf{x}_q$ based on its error reduction and impact over nearby datapoints. Let $\langle q \rangle$ be the indices of the nearby datapoints with the size of $N_{\langle q \rangle}$. The approximated ratio can be calculated:

$$\frac{\mathcal{E}r_{\langle q \rangle}}{\mathcal{I}_{\langle q \rangle}} = \frac{\mathcal{E}_{\langle q \rangle} - \mathbb{E}(\mathcal{E}_{\langle q \rangle}^{+\hat{y}_{qr}})}{\mathcal{I}_{\langle q \rangle}}, \tag{5}$$

where

$$\begin{cases} \mathcal{E}_{\langle q \rangle} = \sum_{i=1}^{N_{\langle q \rangle}} \ell(\mathbf{f}_i, \widehat{\mathbf{f}}_i), \\ \mathcal{I}_{\langle q \rangle} = \sum_{r=1}^{C} f_{qr} \sum_{i=1}^{N_{\langle q \rangle}} \ell(\mathbf{f}_i, \mathbf{f}_i^{+\hat{y}_{qr}}), \\ \mathbb{E}(\mathcal{E}_{\langle q \rangle}^{+\hat{y}_{qr}}) = \sum_{r=1}^{C} f_{qr} \sum_{i=1}^{N_{\langle q \rangle}} \ell(\mathbf{f}_i^{+\hat{y}_{qr}}, \widehat{\mathbf{f}}_i^{+\hat{y}_{qr}}), \end{cases}$$

are the accumulated estimated error, the impact and the generalization error over these nearby datapoints, respectively.

As $\frac{\mathcal{E}r_{\langle q \rangle}}{\mathcal{I}_{\langle q \rangle}}$ is estimated based on nearby datapoints, its confidence is lower than that built upon all datapoints. Therefore, instead of applying the same confidence to $\mathcal{I}_q$ and $\frac{\mathcal{E}r_{\langle q \rangle}}{\mathcal{I}_{\langle q \rangle}}$, the objective function of the AER criterion is formulated as

$$\mathrm{argmax}_{\mathbf{x}_q} \mathcal{I}_q \times \left(\frac{\mathcal{E}r_{\langle q \rangle}}{\mathcal{I}_{\langle q \rangle}}\right)^{1-\epsilon}, \ \mathbf{x}_q \in \mathcal{S}_{\mathrm{AL}}, \tag{6}$$

where $\epsilon \in (0{:}1)$ aims to control the confidence of $\frac{\mathcal{E}r_{\langle q \rangle}}{\mathcal{I}_{\langle q \rangle}}$. Of note, the above formulation leads to the following conclusion.

Proposition.1: Suppose $\frac{\mathcal{E}r_{\langle q \rangle}}{\mathcal{I}_{\langle q \rangle}}>0$ and $\mathcal{I}_q>0$. For Eq.6 with $\epsilon \in (0,1)$, the influence of $\frac{\mathcal{E}r_{\langle q \rangle}}{\mathcal{I}_{\langle q \rangle}}$ on the objective value is reduced, and that of $\mathcal{I}_q$ is relatively increased.

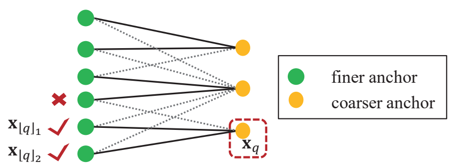

finer anchor
coarser anchor
$\mathbf{x}_{\lfloor q \rfloor_1}$
$\mathbf{x}_{\lfloor q \rfloor_2}$
$\mathbf{x}_q$

Figure 2: An example of $\mathcal{S}_{\lfloor q \rfloor}$ that denotes the set of finer anchors whose nearest connected coarser anchor is $\mathbf{x}_q$. For simplification, only a tiny fraction of inter-set edges of the graph are shown.

We leave the proof of the proposition to the Appendix. Besides, when ϵ is closer to 1, the reduction will be larger. For example, if ϵ=1, the influence of $\frac{\mathcal{E}r_{\langle q \rangle}}{\mathcal{I}_{\langle q \rangle}}$ will be zero.

In this paper, we simply set ϵ to the average estimated error as $\epsilon = \bar{\mathcal{E}} = \frac{1}{N_0}\sum_{i=1}^{N_0} \ell(\mathbf{f}_i, \widehat{\mathbf{f}}_i)$. The idea behind is that, when the error is large, the classification result is often instable. As a result, labeled candidates can affect more datapoints rather than their nearby ones, which reduces the confidence of the approximated ratio. Later we show its effectiveness via the experimental comparison to another strategy [14].

In short, for each $\mathbf{x}_q$ in $\mathcal{S}_{AL}$, if its expected impact over all datapoints ($\mathcal{I}_q$) and its ratio over nearby datapoints ($\frac{\mathcal{E}r_{\langle q \rangle}}{\mathcal{I}_{\langle q \rangle}}$) can be efficiently obtained, we can perform scalable active learning via Eq.6. Below we introduce these implementing details based on hierarchical anchor graphs.

4.1.1 *Hierarchical Expansion of Candidates ($\mathcal{S}_{AL}$)*

We first introduce a hierarchical expansion technique to construct the candidate set by employing both the coarse-to-fine anchors and datapoints on a hierarchical anchor graph.

Instead of using all unlabeled datapoints as candidates, we initialize a candidate set with the coarsest anchors in $\mathcal{X}_h$:

$$\mathcal{S}_{\mathrm{AL}} \Leftarrow \mathcal{X}_h. \tag{7}$$

As the size of $\mathcal{X}_h$ is much smaller than that of the unlabeled data, it can significantly reduce the time cost of the AER estimation. Besides, this candidate set is sufficiently representative for query selection at the early stage.

With the increase of labels, a finer candidate set is needed to tune decision boundaries. Let $\mathcal{S}_{\lfloor q \rfloor} = \{\mathbf{x}_{\lfloor q \rfloor_1}, \ldots\}$ be the set of finer points whose nearest connected coarser anchor is $\mathbf{x}_q$. After $\mathbf{x}_q \in \mathcal{S}_{\mathrm{AL}}$ is labeled, we expand candidates with $\mathcal{S}_{\lfloor q \rfloor}$:

$$\mathcal{S}_{\mathrm{AL}} \Leftarrow (\mathcal{S}_{\mathrm{AL}} \ominus \mathbf{x}_q) \cup \mathcal{S}_{\lfloor q \rfloor}, \tag{8}$$

where $\mathcal{S}_{\mathrm{AL}} \ominus \mathbf{x}_q$ denotes the operation that removes $\mathbf{x}_q$ from $\mathcal{S}_{\mathrm{AL}}$. Different from employing all connected finer points, we alleviate the rapid expansion of the candidate set based on a few candidates in $\mathcal{S}_{\lfloor q \rfloor}$. An illustrative example is shown in Fig.2, where only $\mathbf{x}_{\lfloor q \rfloor_1}$, $\mathbf{x}_{\lfloor q \rfloor_2}$ are added into the candidate set, while the other connected finer points of $\mathbf{x}_q$ are ignored.

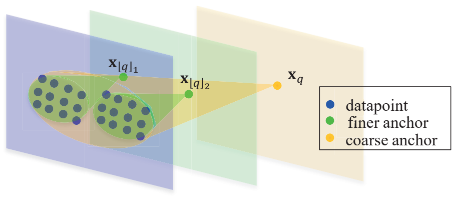

$\mathbf{x}_{\lfloor q \rfloor_1}$
$\mathbf{x}_{\lfloor q \rfloor_2}$
$\mathbf{x}_q$
● datapoint
● finer anchor
● coarse anchor

Figure 3: An example of the hierarchical assignment of nearby datapoints. In this example, the query $\mathbf{x}_q$ is labeled and $\mathbf{x}_{\lfloor q \rfloor_1}$, $\mathbf{x}_{\lfloor q \rfloor_2}$ are added into the candidate set. We only assign the datapoints in $\langle q \rangle$ to their approximate nearest candidate $\mathbf{x}_{\lfloor q \rfloor_1}$ or $\mathbf{x}_{\lfloor q \rfloor_2}$.

4.1.2 *Hierarchical Assignment of Nearby Datapoints ($\langle q \rangle$)*

Then, we consider the assignment of the nearby datapoints. Similar to the candidate expansion, we build the nearby datapoint sets for all candidates in a hierarchical manner.

Specifically, we first build nearby sets for the candidates in $\mathcal{X}_h$ by assigning all datapoints in $\mathcal{X}_0$ to their approximate nearest candidates. Denoting $\langle q \rangle$ as the nearby datapoints of the q-th candidate, we have:

$$\begin{cases} \langle 1 \rangle \cup \langle 2 \rangle \cup \ldots = \mathcal{X}_0, \\ \langle 1 \rangle \cap \langle 2 \rangle \cap \ldots = \emptyset. \end{cases}$$

When a candidate $\mathbf{x}_q \in \mathcal{S}_{\mathrm{AL}}$ is labeled and $\mathcal{S}_{\lfloor q \rfloor}$ is added into the candidate set, we re-build nearby sets for these new candidates in $\mathcal{S}_{\lfloor q \rfloor}$ in a similar way based on the nearby datapoints of $\mathbf{x}_q$ (See Fig.3). As a consequence, we obtain:

$$\begin{cases} \langle \lfloor q \rfloor_1 \rangle \cup \langle \lfloor q \rfloor_2 \rangle \cup \ldots = \langle q \rangle, \\ \langle \lfloor q \rfloor_1 \rangle \cap \langle \lfloor q \rfloor_2 \rangle \cap \ldots = \emptyset. \end{cases}$$

In general, for each finer candidate, the number of its nearby datapoints is smaller than those of coarser ones.

4.1.3 *Fast Computation of the Expected Impact($\mathcal{I}_q$)*

Here, we focus on the computation of the expected impact over all datapoints. Let $\mathbf{A}^{+\hat{y}_{qr}}$ denote the updated classifier with an extra label r on $\mathbf{x}_q$. Taking HAGR as an example with $\mathbf{F}=\mathbf{Z}^{\mathrm{H}}\mathbf{A}$, we can therefore obtain:

$$\mathcal{I}_q = \sum_{r=1}^{C} f_{qr} \| \mathbf{Z}^{\mathrm{H}}(\mathbf{A}^{+\hat{y}_{qr}} - \mathbf{A}) \|_{\mathrm{F}}^2. \tag{9}$$

To calculate Eq.9, we focus on its Frobenius norm:

$$\mathrm{trace}((\mathbf{A}^{+\hat{y}_{qr}} - \mathbf{A})^{\mathrm{T}} \boldsymbol{\Delta} (\mathbf{A}^{+\hat{y}_{qr}} - \mathbf{A})), \tag{10}$$

where $\boldsymbol{\Delta}=\mathbf{Z}^{\mathrm{H}^{\mathrm{T}}}\mathbf{Z}^{\mathrm{H}}$. As $\boldsymbol{\Delta}$ only needs to be calculated once, the time cost here is much smaller than that of the generalization error in EER, where the labels of all datapoints must be incrementally re-inferred. However, as the time cost of each $\mathbf{A}^{+\hat{y}_{qr}}$ scales as $O(N_h^3)$, for N_q candidates within C classes, the total cost of $O(N_h^3 N_q C)$ can still be expensive.

The remaining challenge is how to compute Eq.10 for all candidates efficiently. To solve this issue, we present an efficient method for the impact estimation in the Appendix, which only involves fast matrix operations. In this way, the time cost can be drastically reduced to $O(N_h^2 N_q)$.

4.1.4 *Fast Estimation of the Approximated Ratio ($\frac{\mathcal{E}r_{\langle q \rangle}}{\mathcal{I}_{\langle q \rangle}}$)*

Now, we estimate the ratio between the error reduction and the impact over nearby datapoints. A naive solution is to calculate the two involved terms directly. However, it is practically inefficient to incrementally calculate them for all candidates or store all the relevant matrices in the memory, such as $\boldsymbol{\Delta}_{\langle q \rangle}=\mathbf{Z}_{\langle q \rangle}^{\mathrm{H}}{}^{\mathrm{T}}\mathbf{Z}_{\langle q \rangle}^{\mathrm{H}}$ for the impact estimation over nearby datapoints, where $\mathbf{Z}_{\langle q \rangle}^{\mathrm{H}}$ denotes the nearby part of $\mathbf{x}_q$ in $\mathbf{Z}^{\mathrm{H}}$.

Below we introduce an alternative to accelerate this step. First, for the candidate $\mathbf{x}_q$, we approximate its expected impact over its nearby datapoints ($\mathcal{I}_{\langle q \rangle}$) based on its expected impact over all datapoints ($\mathcal{I}_q$):

$$\mathcal{I}_{\langle q \rangle} \approx \frac{\mathcal{I}_q}{1+\mu}, \tag{11}$$

in which the auxiliary parameter $\mu \geq 0$ describes the degree that the impact overflows the nearby datapoints.

Then we consider the error reduction over nearby datapoints ($\mathcal{E}r_{\langle q \rangle}$). Instead of the repeated calculation, we evaluate it based on the estimated errors of nearby datapoints:

$$\mathcal{E}r_{\langle q \rangle} = \sum_{i=1}^{N_{\langle q \rangle}} \eta_i \times \ell(\mathbf{f}_i, \widehat{\mathbf{f}}_i), \quad i \in \langle q \rangle, \tag{12}$$

where the auxiliary parameter $\eta_i \in [0{:}1]$. It describes the degree that the expected error of the i-th nearby datapoint will be reduced from its current estimated error. Since these parameters of the nearby datapoints of each candidate η_is tend to be similar, we can approximate Eq.12 with the accumulated error over these nearby datapoints ($\mathcal{E}_{\langle q \rangle}$) in Eq.5:

$$\mathcal{E}r_{\langle q \rangle} \approx \eta \times \sum_{i=1}^{N_{\langle q \rangle}} \ell(\mathbf{f}_i, \widehat{\mathbf{f}}_i) = \eta \times \mathcal{E}_{\langle q \rangle}. \tag{13}$$

Note that the auxiliary parameters μ and η can be different for each candidate, which may involve complex relationships with respect to the uncertainty over their nearby datapoints. However, such an evaluation is difficult to be described. To circumvent this problem, in this work, we propose to apply the same settings for all candidates. Our reasoning here is as follows: as the candidate set is expanded hierarchically, we expect that the influence on the nearby datapoints of one candidate will not be much larger than that of the other candidates. With this simplification, based on Eq.11 and Eq.13, we can directly obtain the ratio in Eq.6:

$$\frac{\mathcal{E}r_{\langle q \rangle}}{\mathcal{I}_{\langle q \rangle}} = (\eta \times \mathcal{E}_{\langle q \rangle})/(\frac{\mathcal{I}_q}{1+\mu}) = \eta(1+\mu) \times \frac{\mathcal{E}_{\langle q \rangle}}{\mathcal{I}_q}. \tag{14}$$

4.2 Scalable Active Learning

Now we present the AER-based scalable active learning.

Given a dataset $\mathcal{D}$ with N_{L} labeled datapoints, we first construct a hierarchical anchor graph with Eq.1 and build a candidate set $\mathcal{S}_{\mathrm{AL}}$ via Eq.7. We initialize the classifier based on all datapoints with a few labels via Eq.2, and infer the labels of unlabeled datapoints with Eq.3.

Then for each candidate $\mathbf{x}_q \in \mathcal{S}_{\mathrm{AL}}$, we compute its expected impact over all datapoints ($\mathcal{I}_q$) based on the proposed fast

Table 2: Our Approach for Scalable Active Learning

```
Input: datapoint set $\mathcal{X}_0$, anchor sets $\mathcal{X}_b$s, the parameters $s$
    and $\lambda$, the number of labeled datapoints $T$.
# Initialization
1. Construct a hierarchical anchor graph with Eq.1.
2. Build a set of candidates $\mathcal{S}_{\mathrm{AL}}$ with Eq.7 and find the
    nearby datapoint sets of these candidates.
3. Initialize a scalable classifier, such as HAGR.
# Efficient Query Selection
    Repeat the following steps until $T$ queries are labeled:
1. Obtain the estimated value in Eq.16 for each candidate based
    on the proposed fast algorithm and the labels of datapoints.
2. Ask an oracle for the label of the selected query.
3. Re-train the classifier via Eq.2 and re-infer the labels of
    datapoints via Eq.3.
4. Expand the candidate set $\mathcal{S}_{\mathrm{AL}}$ via Eq.8.
Output: The classifier and the labels of all datapoints.
```

algorithm mentioned in Section 4.1.3. We employ its nearby datapoints to estimate its approximated ratio between the error reduction and the impact ($\frac{\mathcal{E}r_{\langle q\rangle}}{\mathcal{I}_{\langle q\rangle}}$). We substitute these two terms of $\mathbf{x}_q$ into Eq.6 and obtain its approximated error reduction over all datapoints ($\mathcal{E}r_q$):

$$\begin{aligned}\mathcal{E}r_q = &\ \ \mathcal{I}_q \times \left(\frac{\mathcal{E}r_{\langle q\rangle}}{\mathcal{I}_{\langle q\rangle}}\right)^{1-\epsilon}\\ = &\ \ \mathcal{I}_q \times \left(\eta(1+\mu)\times\frac{\mathcal{E}_{\langle q\rangle}}{\mathcal{I}_q}\right)^{1-\epsilon}\\ = &\ \ \mathcal{I}_q^{\epsilon}\times{\mathcal{E}_{\langle q\rangle}}^{1-\epsilon}\times(\eta(1+\mu))^{1-\epsilon},\end{aligned} \tag{15}$$

where $\epsilon{=}\bar{\mathcal{E}}$. As $(\eta(1{+}\mu))^{1-\epsilon}$ in Eq.15 is a constant for all candidates, it can be removed without changing the solution of the optimization problem. Therefore, according to AER, the following query can be selected:

$$\mathrm{argmax}_{\mathbf{x}_q}\ \mathcal{I}_q^{\epsilon}\times{\mathcal{E}_{\langle q\rangle}}^{1-\epsilon}, \mathbf{x}_q\in\mathcal{S}_{\mathrm{AL}}. \tag{16}$$

Once the query is labeled, we re-infer the labels of datapoints and update candidates via Eq.8. In our experiments, we conduct the query selection until T queries are labeled.

The overall active learning approach is given in Table 2.

4.3 Computational Cost Analysis

Below we analyze the time cost of the proposed approach.

During the initialization step, the time cost of graph construction is $O(N_0\log N_1)$. The total cost of computing $\mathbf{Z}^{\mathrm{H}}$, $\mathbf{\Delta}$, and $\mathbf{rL}$ scales as $O(N_0N_hs)$ with the sparse matrix multiplication [27]. We optimize HAGR with a cost of $O(N_h^3)$. In short, the time cost here scales as $O(N_0\log N_1{+}N_0N_hs{+}N_h^3)$.

Then, in each iteration of query selection, the following time costs are required. Firstly, we infer the labels of datapoints in $O(N_0N_hC)$, and calculate their estimated errors in $O(N_0C)$. Secondly, to estimate the error reduction for N_q candidates, we compute their expected impact values over all datapoints in $O(N_h^2N_q)$ and the approximated ratios in $O(N_0{+}N_q)$. Finally, we select the query based on AER in $O(N_q)$. As we have $N_q{\geq}N_h$, the time cost here can be simplified as $O(N_0N_hC{+}N_h^2N_q{+}N_h^3){\approx}O(N_0N_hC{+}N_h^2N_q)$.

As we can see, apart from the similar time cost to that of the uncertainty-based sampling, namely $O(N_0N_hC)$, the remaining time cost of our AER-based query selection is independent of data sizes, which highlights its superiority for large-scale active learning.

4.4 Discussion on AER

In this section, we discuss the relationships and differences between the proposed AER criterion and other criteria.

4.4.1 Comparison to Density-Weighting Uncertainty

This learning criterion [21] selects the datapoint that is both uncertain and representative, which can be formulated as

$$\mathrm{argmax}_{\mathbf{x}_q} d(\mathbf{x}_q)\times\ell^{NSE}(\mathbf{f}_q), \tag{17}$$

where $d(\mathbf{x}_q){=}\sum_{i=1}^{N_{\mathrm{U}}}\mathrm{sim}_{cos}(\mathbf{x}_q,\mathbf{x}_i)$ denotes the cosine similarity of $\mathbf{x}_q$ over N_{U} unlabeled datapoints, and $\ell^{NSE}(\mathbf{f}_q)$ denotes the N-best sequence entropy [10]. Eq.17 can be rewritten as

$$\mathrm{argmax}_{\mathbf{x}_q} d(\mathbf{x}_q)\times\frac{\ell^{NSE}(\mathbf{f}_q)}{\mathrm{sim}_{cos}(\mathbf{x}_q,\mathbf{x}_q)}, \tag{18}$$

where $\mathrm{sim}_{cos}(\mathbf{x}_q,\mathbf{x}_q){=}1$. Similar to Eq.6, the first term evaluates the similarity of $\mathbf{x}_q$ over massive datapoints, and the second term is the approximated ratio between the uncertainty reduction and the similarity of $\mathbf{x}_q$ itself. Compared with AER, Eq.18 estimates the ratio based on a single datapoint, which can be insufficiently effective.

4.4.2 Comparison to Expected Model Change

This learning criterion [3] selects the datapoint that brings the greatest expected change on the parameters of a classifier. For HAGR, its query selection can be formulated as

$$\mathrm{argmax}_{\mathbf{x}_q}\sum_{r=1}^{C} f_{qr}\|\ \mathbf{A}^{+\hat{y}_{qr}}-\mathbf{A}\ \|_{\mathrm{F}}^2. \tag{19}$$

The change of HAGR is equal to the change on the soft labels of the coarsest anchors, which is far from the error reduction over massive datapoints. In contrast, our impact in Eq.9 calculates the change on the soft labels of all datapoints, which narrows the above gap by introducing data distributions.

4.4.3 Comparison to Expected Error Reduction

In EER, if labeling the candidate $\mathbf{x}_q$ only reduces the errors of its nearby datapoints $\langle q\rangle$, we have $\mathcal{E}r_q{=}\mathcal{E}r_{\langle q\rangle}$. Below we show that when $\mu{=}0$ and $\eta{=}1$, our AER value is equal to EER, which is independent of the average estimated error $\bar{\mathcal{E}}$.

Since $\eta{=}1$, we obtain $\mathcal{E}_{\langle q\rangle}{=}\mathcal{E}r_{\langle q\rangle}$ via Eq.13. As all errors of nearby datapoints are changed to 0, we obtain $\mathcal{I}_{\langle q\rangle}{=}\mathcal{E}r_{\langle q\rangle}$. Then as $\mu{=}0$, we have $\mathcal{I}_q{=}\mathcal{I}_{\langle q\rangle}$ via Eq.11. By substituting the above results into Eq.15, we finally obtain:

$$\mathcal{E}r_q=\mathcal{I}_q^{\epsilon}\times{\mathcal{E}_{\langle q\rangle}}^{1-\epsilon}\times 1={\mathcal{E}r_{\langle q\rangle}}^{\epsilon}\times{\mathcal{E}r_{\langle q\rangle}}^{1-\epsilon}=\mathcal{E}r_{\langle q\rangle}. \tag{20}$$

When $\eta{\neq}1$, and $\mu{\neq}0$, AER first focuses on the expected impact over all datapoints for rapid accuracy improvements, and then adaptively pays attention to the error reduction over a few nearby datapoints for tuning decision boundaries.

Table 3: Details of the Datasets in Our Experiments.

Dataset	Num of instances	Num of categories	Num of dimensions
Alphadigits	1,404	36	320
Semeion	1,593	10	256
USPS	7,291	10	256
ISOLET	7,797	26	617
Letter	20,000	26	16
MNIST	70,000	10	784
ImageNet	256,091	200	512
MNIST8M	8,100,000	10	86

5 EXPERIMENTS

In this section, we investigate the effectiveness of our AER criterion. The experiments are implemented on a PC with i7-5820K CPU @ 3.30GHz and 64G RAM. We use the following datasets with varying sizes, including Alphadigits[2], Semeion[3], ISOLET[4], Letter[3], USPS[5], MNIST[6], ImageNet[7], and MNIST8M[23]. Some statistics of them are listed in Table 3. For convenience, we regard the first six as medium-size datasets, and the last two as large-size datasets.

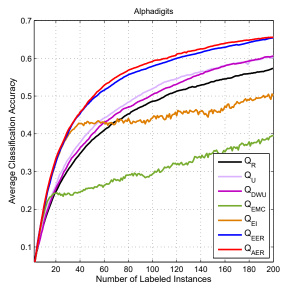
5.1 Comparison to Other Methods

We first compare the proposed AER-based approach with the methods built upon several state-of-the-art active learning criteria. For scalability and fair comparisons, we use HAGR as the classifier for all active learners, which has shown its impressive performance on semi-supervised classification tasks. The methods for comparison are as follows:

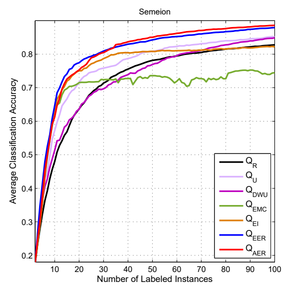
1. Random Sampling: This method randomly selects queries for labeling. We denote it as 'Q_R'.
2. Maximal Uncertainty: This method selects queries based on the 2-best sequence entropy [10]. We denote it as 'Q_U'.
3. Maximal Density-Weighting Uncertainty: This method selects queries based on the density-weighting entropy with the cosine similarity [21]. We denote it as 'Q_{DWU}'
4. Maximal Expected Model Change: It selects queries based on the expected change on the parameters of a classifier [3]. This method is denoted as 'Q_{EMC}'.
5. Maximal Expected Impact: This method selects queries based on the proposed expected impact over all datapoints. We denote it as 'Q_{EI}'.
6. Maximal EER: This method selects queries with the EER evaluation [29]. We denote it as 'Q_{EER}'.
7. Maximal AER: This proposed method chooses queries based AER. It is denoted as 'Q_{AER}'.

Note that besides Q_{AER}, the candidate sets in Q_{DWU}, Q_{EMC}, Q_{EI} are also expended in a hierarchical manner for

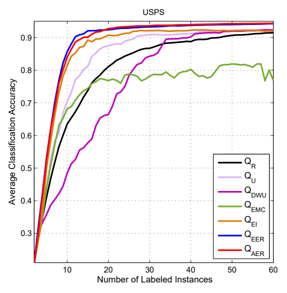
[2]available at http://www.cs.nyu.edu/∼roweis/data.html
[3]available at http://archive.ics.uci.edu/ml
[4]available at http://www.cad.zju.edu.cn/home/dengcai/
[5]available at http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets
[6]available at http://yann.lecun.com/exdb/mnist
[7]We randomly select 200 classes from ImageNet [19] and build a subset with 256,091 images. We extract their 4,096-D CNN features via AlexNet [11] and perform PCA to reduce the dimension to 512.



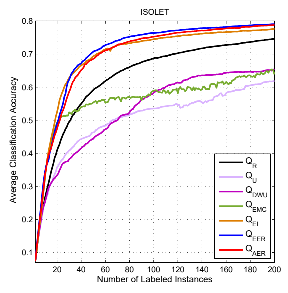
(a)



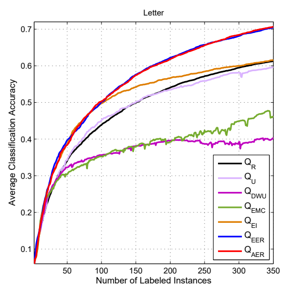
(b)



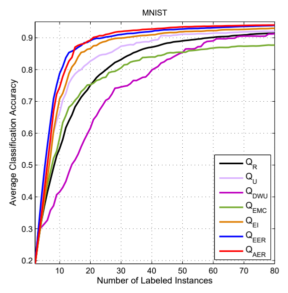
(c)



(d)



(e)



(f)

Figure 4: Average performance curves with respect to the number of labels on medium-size datasets.

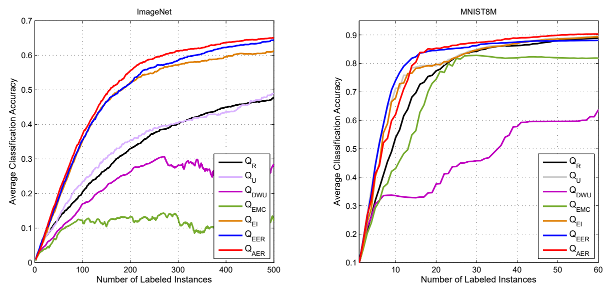
efficient implementation (see Section 4.1.1). Meanwhile, as the huge time cost of the EER estimation, we only employ the coarsest anchors as the candidates for Q_{EER}. We enlarge the numbers of anchor sets with the increase of datapoints, where the sizes of these anchor sets follow the proportion suggested in hierarchical anchor graph models [16], [23], [24].

5.1.1 On Medium-size Datasets

For Alphadigits, Semeion, ISOLET, and Letter, we follow [24] and build 500-anchor-graphs. We build 2,000-500-anchor-graphs and 5,000-1,250-anchor-graphs for USPS and MNIST, respectively. We empirically set λ to 0.1. For active learning, only 2 instances are randomly sampled as the initial labeled data. Based on 20 trials, the average accuracy curves are displayed in Fig.4, and the time costs are listed in Table 4.

Table 4: Comparison of average time costs (in seconds per query) on medium-size datasets.

Dataset	Q_U	Q_{DWU}	Q_{EMC}	Q_{EI}	Q_{EER}	Q_{AER}
Alphadigits	0.01	0.03	0.04	0.04	1.80	0.04
Semeion	0.01	0.01	0.02	0.02	0.67	0.02
USPS	0.01	0.02	0.02	0.02	2.45	0.02
ISOLET	0.02	0.04	0.03	0.04	7.94	0.05
Letter	0.03	0.10	0.11	0.17	13.81	0.20
MNIST	0.04	0.18	0.15	0.17	30.16	0.21



(a) (b)

Figure 5: Average accuracy curves with respect to the number of labels on large-size datasets.

Table 5: Comparison of average time costs (in seconds per query) on large-size datasets.

Dataset	Q_U	Q_{DWU}	Q_{EMC}	Q_{EI}	Q_{EER}	Q_{AER}
ImageNet	2.15	10.05	4.19	7.61	90.73	9.35
MNIST8M	4.22	14.47	10.97	12.19	198.73	16.15

From these results, we can obtain the following observations. Firstly, compared with Q_R, Q_U obtains higher accuracies on Semeion, USPS, and MNIST, and receives comparable or even worse performance on Alphadigits, ISOLET, and Letter. The reason behind is that, the uncertainty-based sampling is insufficiently effective to discover queries that actually belong to new classes, especially when the number of classes is large. Secondly, Q_{DWU} performs worse than Q_U in most cases, which indicates that directly combing the density and the entropy is not generally suitable for all classifiers. Thirdly, as Q_{EMC} only considers the changes on the parameters of the classifier, it obtains worse performance than Q_R with the increase of labels. In contrast, by introducing the data distribution, Q_{EI} can achieve much higher accuracies, which is consistent with the theoretical analysis in Section 4.4.2. Fourthly, Q_{AER} obtains comparable or even better performance than Q_{EER}, and its performance is more consistent to that of Q_{EER} than others. This result empirically demonstrates the effectiveness of our AER criterion on the error reduction estimation. Finally, when considering both the efficiency and the effectiveness, Q_{AER} highlights its strengths over other compared methods.

5.1.2 *On Large-size Datasets*

We also conduct experiments on ImageNet and MNIST8M with 100,000-10,000-2,500-anchor-graphs. We empirically set λ to 0.01 in HAGR. By repeating the similar process, we report the average classification accuracies over 10 trials in Fig.5 and list the time costs in Table 5.

From these results, the following observations can be made. Firstly, similar to the pervious results, Q_{EI} outperforms Q_{EMC} by giving consideration to data distributions. Secondly, compared with Q_{EI} and Q_{EER}, Q_{AER} can obtain higher accuracies after a few iterations of query selection. It means that introducing uncertainty into active learning criteria will be beneficial to tune decision boundaries. Thirdly, when taking account into the time cost in Table 5, we further confirm the superior performance of Q_{AER} for scalable active learning.

5.2 Effectiveness Analysis

5.2.1 *On the Formulation of AER*

In AER, ϵ is fixed to $\bar{\mathcal{E}}$ to control the confidence of the approximated ratio. To demonstrate the effectiveness, we compare Q_{AER} with the following two intermediate versions:

(1). $Q_{AER,Ada}$: This method follows [14] and sets ϵ to different values, e.g., $\{0, 0.1, \dots, 1\}$ to obtain several sub-queries, and then refines the best query from them based on EER.

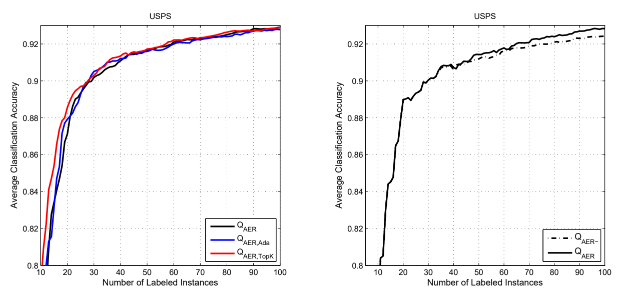
(2). $Q_{AER,TopK}$: This method first selects top $K(K{=}10)$ datapoints based on AER as sub-queries and then refines the best query from them based on EER.

We conduct experiments on the USPS dataset with 500-125-anchor-graphs. The average accuracy curves over 20 trias of these three methods are displayed in Fig.6(a).

From this figure, we observe that Q_{AER} can obtain comparable performance to $Q_{AER,Ada}$, which requires the extra time cost of query refining. The reason behind is that, although the setting of ϵ in $Q_{AER,Ada}$ is more flexible, most of them are useless. For example, at initial stages, the sub-queries with small expected impact values over all datapoints will not lead to rapid improvements. With the error reducing, the sub-queries with large impact values may not improve decision boundaries. In contrast, Q_{AER} prefers the datapoints with large impact values at first and those near decision boundaries with the increase of labels. That is, AER adaptively weighs the impact over all datapoints against the accumulated uncertainty over nearby datapoints. Meanwhile, we observe that although Q_{AER} performs slightly worse than $Q_{AER,TopK}$ at the early stages, it obtains comparable accuracies with the increase of labels. This result also implies that, we may further improve the performance of Q_{AER} with an extra query refining step, where the time cost is still much smaller than that of the direct EER-based query selection.

5.2.2 *On the Hierarchical Candidate Expansion*

We finally investigate the effectiveness of the candidate expansion based on hierarchical anchor graphs. We compare Q_{AER} with Q_{AER}-, which is a simplified Q_{AER} without the candidate expansion via Eq.8. The average accuracy curves of these two methods are shown in Fig.6(c).



(a) (b)

Figure 6: Average accuracy curves with respect to the number of labeled instances on USPS.

From this figure, we observe that the accuracies of $\mathrm{Q_{AER}}$ and $\mathrm{Q_{AER}}$- are similar at initial stages. It means that, although $\mathrm{Q_{AER}}$ expands the candidate set from the beginning, it will not immediately fall into finer candidates for local accuracy improvements. Once the performance becomes better with the increase of labeled data, $\mathrm{Q_{AER}}$ will pay attention to these finer candidates and bring higher classification accuracies. This result further demonstrates the effectiveness of introducing finer candidates for tuning decision boundaries.

6 CONCLUSIONS

This paper proposed a novel query selection criterion called approximated error reduction (AER). Different from other criteria, AER estimates the error reduction of a datapoint based on its expected impact over all datapoints and its approximated ratio between the error reduction and the impact over its nearby datapoints. Meanwhile, AER employs hierarchical anchor graphs to expand a small candidate set with the increase of labels, which further accelerates its estimation. Benefiting from AER, we can obtain an efficient estimation of the error reduction without incrementally re-inferring labels of massive datapoints. We introduced AER into an efficient semi-supervised classifier for scalable active learning. The experiments on publicly available datasets demonstrated both the efficiency and the effectiveness of our approach.

It is worthwhile to note that since the supervised and semi-supervised classifiers in the literature [8], [22], [24], [28] have similar solutions to HAGR in their model optimization, our future work includes the integration of AER with them, where the proposed fast impact estimation can be generalized. In these works, all AER-based approaches are supposed to be much fater than EER, as the repeatedly label re-inference is not required in AER.

APPENDIXES

A. Proof of Proposition 1

Let $f(\alpha,\beta)$ denote $f_1(\alpha)\times f_2(\beta)$, where $f_1(\alpha)$=α and $f_2(\beta)$= $(\beta)^{1-\epsilon}$ are two mapping functions on the variables α, β>0 with $\epsilon\in(0,1)$, respectively. Denote $r_\alpha=\frac{\alpha_1}{\alpha_2}$, and $r'_\alpha=\frac{f_1(\alpha_1)}{f_1(\alpha_2)}$ as the original ratio and the mapped ratio on α, respectively, and $r_\beta=\frac{\beta_1}{\beta_2}$, and $r'_\beta=\frac{f_2(\beta_1)}{f_2(\beta_2)}$ as the original ratio and the mapped ratio on β, respectively. We have (1). $r_\beta>r'_\beta>1(r_\beta<r'_\beta<1)$ if $r_\beta>1(r_\beta<1)$. (2). $r'_\alpha=r_\alpha$. That is, the mapped r'_β is closer to 1 than r_β, and the mapped r'_α is same to r_α, namely, the difference of two βs corresponding to two instances is reduced via the mapping function f_2, and that of two αs is kept via f_1. As a consequence, it leads to the reduction of the influence of β on the objective value, and relatively increases the influence of α. According to Eq.6, let α denote the expected impact $\mathcal{I}_q$, and β be the approximated ratio $\frac{\mathcal{E}r_{\langle q\rangle}}{\mathcal{I}_{\langle q\rangle}}$, which completes the proof.

B. Fast Impact Estimation

Below we presents the derivation of the fast impact estimation used in Section 4.1.3.

Let $\mathbf{z}_q^{\mathrm{H}}$ be the cascaded inter-set adjacency vector of $\mathbf{x}_q$, and $\hat{\mathbf{y}}_{qr}$ be its class indicator vector where only the r-th element is 1. To obtain Eq.10, the traces of the following terms are needed: $\mathbf{A}^{+\hat{y}_{qr}\,\mathrm{T}}\mathbf{\Delta}\mathbf{A}^{+\hat{y}_{qr}}$, $\mathbf{A}^{+\hat{y}_{qr}\,\mathrm{T}}\mathbf{\Delta}\mathbf{A}$, and $\mathbf{A}^{\mathrm{T}}\mathbf{\Delta}\mathbf{A}$, where

$$\mathbf{A}^{+\hat{y}_{qr}}=(\mathbf{z}_q^{\mathrm{H}\,\mathrm{T}}\mathbf{z}_q^{\mathrm{H}}+\mathbf{Z}_{\mathrm{L}}^{\mathrm{H}\,\mathrm{T}}\mathbf{Z}_{\mathrm{L}}^{\mathrm{H}}+\lambda\mathbf{rL})^{-1}(\mathbf{z}_q^{\mathrm{H}\,\mathrm{T}}\hat{\mathbf{y}}_{qr}+\mathbf{Z}_{\mathrm{L}}^{\mathrm{H}\,\mathrm{T}}\mathbf{Y}_{\mathrm{L}}).$$

Suppose $\mathbf{M}$=$(\mathbf{Z}_{\mathrm{L}}^{\mathrm{H}\,\mathrm{T}}\mathbf{Z}_{\mathrm{L}}^{\mathrm{H}}+\lambda\mathbf{rL})$, then $\mathbf{A}^{+\hat{y}_{qr}}$=$(\mathbf{z}_q^{\mathrm{H}\,\mathrm{T}}\mathbf{z}_q^{\mathrm{H}}+\mathbf{M})^{-1}$ $(\mathbf{z}_q^{\mathrm{H}\,\mathrm{T}}\hat{\mathbf{y}}_{qr}+\mathbf{Z}_{\mathrm{L}}^{\mathrm{H}\,\mathrm{T}}\mathbf{Y}_{\mathrm{L}})$. By matrix inversion, we can therefore calculate $\mathbf{A}^{+\hat{y}_{qr}}$ as

$$(\mathbf{M}^{-1}-\frac{\mathbf{M}^{-1}\mathbf{z}_q^{\mathrm{H}}\mathbf{z}_q^{\mathrm{H}\,\mathrm{T}}\mathbf{M}^{-1}}{\mathbf{z}_q^{\mathrm{H}\,\mathrm{T}}\mathbf{M}^{-1}\mathbf{z}_q^{\mathrm{H}}+1})(\mathbf{z}_q^{\mathrm{H}\,\mathrm{T}}\hat{\mathbf{y}}_{qr}+\mathbf{Z}_{\mathrm{L}}^{\mathrm{H}\,\mathrm{T}}\mathbf{Y}_{\mathrm{L}})$$

Let $\hat{\mathbf{M}}$=$\mathbf{M}^{-1}$, α_q=$\mathbf{z}_q^{\mathrm{H}}\hat{\mathbf{M}}\mathbf{z}_q^{\mathrm{H}\,\mathrm{T}}$, and β_q=$\frac{1}{1+\alpha_q}$. We can substitute the above results into the trace of the first term $(\mathbf{A}^{+\hat{y}_{qr}\,\mathrm{T}}\mathbf{\Delta}\mathbf{A}^{+\hat{y}_{qr}})$ and obtain it:

$$\begin{aligned}
&\mathrm{trace}[(\hat{\mathbf{y}}_{qr}^{\mathrm{T}}\mathbf{z}_q^{\mathrm{H}}+\mathbf{Y}_{\mathrm{L}}{}^{\mathrm{T}}\mathbf{Z}_{\mathrm{L}}^{\mathrm{H}})(\mathbf{I}-\beta_q\hat{\mathbf{M}}\mathbf{z}_q^{\mathrm{H}}\mathbf{z}_q^{\mathrm{H}\,\mathrm{T}})\hat{\mathbf{M}}\mathbf{\Delta}\\
&\hat{\mathbf{M}}(\mathbf{I}-\beta_q\mathbf{z}_q^{\mathrm{H}}\mathbf{z}_q^{\mathrm{H}\,\mathrm{T}}\hat{\mathbf{M}})(\mathbf{z}_q^{\mathrm{H}\,\mathrm{T}}\hat{\mathbf{y}}_{qr}+\mathbf{Z}_{\mathrm{L}}^{\mathrm{H}\,\mathrm{T}}\mathbf{Y}_{\mathrm{L}})]\\
=&\mathrm{trace}[\hat{\mathbf{y}}_{qr}^{\mathrm{T}}(\mathbf{z}_q^{\mathrm{H}}\hat{\mathbf{M}}\mathbf{\Delta}\hat{\mathbf{M}}\mathbf{z}_q^{\mathrm{H}\,\mathrm{T}})\hat{\mathbf{y}}_{qr}]+2\mathrm{trace}[\hat{\mathbf{y}}_{qr}^{\mathrm{T}}(\mathbf{z}_q^{\mathrm{H}}\hat{\mathbf{M}}\mathbf{\Delta}\hat{\mathbf{M}}\mathbf{Z}_{\mathrm{L}}^{\mathrm{H}\,\mathrm{T}}\mathbf{Y}_{\mathrm{L}})]\\
&-2\beta_q\mathrm{trace}[\hat{\mathbf{y}}_{qr}^{\mathrm{T}}(\mathbf{z}_q^{\mathrm{H}}\hat{\mathbf{M}}\mathbf{\Delta}\hat{\mathbf{M}}\mathbf{z}_q^{\mathrm{H}\,\mathrm{T}})(\mathbf{z}_q^{\mathrm{H}}\mathbf{M}\mathbf{z}_q^{\mathrm{H}\,\mathrm{T}})\hat{\mathbf{y}}_{qr}]\\
&-2\beta_q\mathrm{trace}[\hat{\mathbf{y}}_{qr}^{\mathrm{T}}(\mathbf{z}_q^{\mathrm{H}}\hat{\mathbf{M}}\mathbf{\Delta}\hat{\mathbf{M}}\mathbf{z}_q^{\mathrm{H}\,\mathrm{T}})(\mathbf{z}_q^{\mathrm{H}}\mathbf{M}\mathbf{Z}_{\mathrm{L}}^{\mathrm{H}\,\mathrm{T}}\mathbf{Y}_{\mathrm{L}})]\\
&-2\beta_q\mathrm{trace}[\hat{\mathbf{y}}_{qr}^{\mathrm{T}}(\mathbf{z}_q^{\mathrm{H}}\hat{\mathbf{M}}\mathbf{z}_q^{\mathrm{H}\,\mathrm{T}})(\mathbf{z}_q^{\mathrm{H}}\hat{\mathbf{M}}\mathbf{\Delta}\hat{\mathbf{M}}\mathbf{Z}_{\mathrm{L}}^{\mathrm{H}\,\mathrm{T}}\mathbf{Y}_{\mathrm{L}})]\\
&+\beta_q^2\mathrm{trace}[(\hat{\mathbf{y}}_{qr}^{\mathrm{T}}(\mathbf{z}_q^{\mathrm{H}}\hat{\mathbf{M}}\mathbf{z}_q^{\mathrm{H}\,\mathrm{T}})(\mathbf{z}_q^{\mathrm{H}}\hat{\mathbf{M}}\mathbf{\Delta}\hat{\mathbf{M}}\mathbf{z}_q^{\mathrm{H}\,\mathrm{T}})(\mathbf{z}_q^{\mathrm{H}}\mathbf{M}\mathbf{z}_q^{\mathrm{H}\,\mathrm{T}})\hat{\mathbf{y}}_{qr}]\\
&+2\beta_q^2\mathrm{trace}[\hat{\mathbf{y}}_{qr}^{\mathrm{T}}(\mathbf{z}_q^{\mathrm{H}}\hat{\mathbf{M}}\mathbf{z}_q^{\mathrm{H}\,\mathrm{T}})(\mathbf{z}_q^{\mathrm{H}}\hat{\mathbf{M}}\mathbf{\Delta}\hat{\mathbf{M}}\mathbf{z}_q^{\mathrm{H}\,\mathrm{T}})(\mathbf{z}_q^{\mathrm{H}}\mathbf{M}\mathbf{Z}_{\mathrm{L}}^{\mathrm{H}\,\mathrm{T}}\mathbf{Y}_{\mathrm{L}})]\\
&+\mathrm{trace}[(\mathbf{Y}_{\mathrm{L}}^{\mathrm{T}}\mathbf{Z}_{\mathrm{L}}^{\mathrm{H}}\hat{\mathbf{M}})\mathbf{\Delta}(\hat{\mathbf{M}}\mathbf{Z}_{\mathrm{L}}^{\mathrm{H}\,\mathrm{T}}\mathbf{Y}_{\mathrm{L}})]\\
&-2\beta_q\mathrm{trace}[(\mathbf{Y}_{\mathrm{L}}^{\mathrm{T}}\mathbf{Z}_{\mathrm{L}}^{\mathrm{H}}\hat{\mathbf{M}}\mathbf{\Delta}\hat{\mathbf{M}}\mathbf{z}_q^{\mathrm{H}\,\mathrm{T}})(\mathbf{z}_q^{\mathrm{H}}\mathbf{M}\mathbf{Z}_{\mathrm{L}}^{\mathrm{H}\,\mathrm{T}}\mathbf{Y}_{\mathrm{L}})]\\
&+\beta_q^2\mathrm{trace}[(\mathbf{Y}_{\mathrm{L}}^{\mathrm{T}}\mathbf{Z}_{\mathrm{L}}^{\mathrm{H}}\hat{\mathbf{M}}\mathbf{z}_q^{\mathrm{H}\,\mathrm{T}})(\mathbf{z}_q^{\mathrm{H}}\hat{\mathbf{M}}\mathbf{\Delta}\hat{\mathbf{M}}\mathbf{z}_q^{\mathrm{H}\,\mathrm{T}})(\mathbf{z}_q^{\mathrm{H}}\hat{\mathbf{M}}\mathbf{Z}_{\mathrm{L}}^{\mathrm{H}\,\mathrm{T}}\mathbf{Y}_{\mathrm{L}})].
\end{aligned}$$

Suppose γ_q=$\mathbf{z}_q^{\mathrm{H}}(\hat{\mathbf{M}}\mathbf{\Delta}\hat{\mathbf{M}})\mathbf{z}_q^{\mathrm{H}\,\mathrm{T}}$, $\boldsymbol{\varphi}_q$=$\mathbf{z}_q^{\mathrm{H}}\mathbf{A}$, $\boldsymbol{\phi}_q$=$\mathbf{z}_q^{\mathrm{H}}\hat{\mathbf{M}}\mathbf{\Delta}\mathbf{A}$, and δ=trace$(\mathbf{A}^{\mathrm{T}}\mathbf{\Delta}\mathbf{A})$, where $\mathbf{A}$=$\hat{\mathbf{M}}\mathbf{Z}_{\mathrm{L}}^{\mathrm{H}\,\mathrm{T}}\mathbf{Y}_{\mathrm{L}}$. Besides, we obtain trace$(\hat{\mathbf{y}}_{qr}^{\mathrm{T}}\gamma_q\hat{\mathbf{y}}_{qr})$=$\gamma_q$, and trace$(\hat{\mathbf{y}}_{qr}^{\mathrm{T}}\boldsymbol{\phi}_q)$=$(\boldsymbol{\phi}_q)_r$, where $(\cdot)_r$ is the r-th element of the inside vector. By substituting them into the above equation, we obtain the trace of the first term:

$$\delta + (1 - 2\alpha_q\beta_q + \alpha_q^2\beta_q^2)\gamma_q + (2\alpha_q\beta_q^2\gamma_q - 2\beta_q\gamma_q)(\boldsymbol{\varphi}_q)_r + (2 - 2\alpha_q\beta_q)(\boldsymbol{\phi}_q)_r - 2\beta_q{\boldsymbol{\phi}_q}^{\mathrm{T}}\boldsymbol{\varphi}_q + \beta_q^2\gamma_q{\boldsymbol{\varphi}_q}^{\mathrm{T}}\boldsymbol{\varphi}_q,$$

Similarly, the trace of the second term can be obtained:

$$\begin{aligned}
&\mathrm{trace}[\mathbf{A\Delta\hat{M}}(\mathbf{I} - \beta_q\mathbf{z}_q^{\mathrm{H}}{\mathbf{z}_q^{\mathrm{H}}}^{\mathrm{T}}\mathbf{\hat{M}})({\mathbf{z}_q^{\mathrm{H}}}^{\mathrm{T}}\hat{\mathbf{y}}_{qr} + {\mathbf{Z}_{\mathrm{L}}^{\mathrm{H}}}^{\mathrm{T}}\mathbf{Y}_{\mathrm{L}})]\\
=&\mathrm{trace}[(\mathbf{A\Delta\hat{M}}{\mathbf{z}_q^{\mathrm{H}}}^{\mathrm{T}})\hat{\mathbf{y}}_{qr}] + \mathrm{trace}(\mathbf{A\Delta A})\\
&- \mathrm{trace}[\beta_q(\mathbf{A\Delta\hat{M}}\mathbf{z}_q^{\mathrm{H}})({\mathbf{z}_q^{\mathrm{H}}}^{\mathrm{T}}\mathbf{\hat{M}}{\mathbf{z}_q^{\mathrm{H}}}^{\mathrm{T}})\hat{\mathbf{y}}_{qr}]\\
&- \mathrm{trace}[\beta_q(\mathbf{A\Delta\hat{M}}\mathbf{z}_q^{\mathrm{H}})({\mathbf{z}_q^{\mathrm{H}}}^{\mathrm{T}}\mathbf{\hat{M}}{\mathbf{Z}_{\mathrm{L}}^{\mathrm{H}}}^{\mathrm{T}}\hat{\mathbf{y}}_{\mathrm{L}})]\\
=&\delta + (1 - \alpha_q\beta_q)(\boldsymbol{\phi}_q)_r - \beta_q{\boldsymbol{\varphi}_q}^{\mathrm{T}}\boldsymbol{\phi}_q.
\end{aligned}$$

We briefly analyze the above time cost. First, for N_q candidates, the total time cost of α_qs is $O(N_h^2N_q)$, and the following cost for β_q is $O(N_q)$. Then for these candidates, the time cost of γ_qs is $O(N_h^3+N_h^2N_q)$, and that of all $\boldsymbol{\phi}_q$s, $\boldsymbol{\varphi}_q$s, ${\boldsymbol{\phi}_q}^{\mathrm{T}}\boldsymbol{\varphi}_q$s and ${\boldsymbol{\varphi}_q}^{\mathrm{T}}\boldsymbol{\varphi}_q$s is $O(N_hN_qC+N_h^2C+N_qC^2)$. Therefore, for N_q candidates, the time cost of their expected impact estimation is $O(N_h^2N_q+N_h^3+N_hN_qC+N_h^2C+N_qC^2)$. Since $N_q \geq N_h \gg C$, it can be simplified as $O(N_h^2N_q)$. Of note, these impact values can be computed via direct matrix operations rather than multiple iterations.

ACKNOWLEDGMENTS

This work was partially supported by the National Key Research and Development Program of China under grant 2016-YFB1000901, the National Nature Science Foundation of China under grant 61632007, 61772171, and 71490725, 91746-209, and the Program for Changjiang Scholars and Innovative Research Team in University of the Ministry of Education, China under grant IRT17R32.

REFERENCES

[1] Oisin Mac Aodha, Neill D. F. Campbell, Jan Kautz, and Gabriel J. Brostow. 2014. Hierarchical subquery evaluation for active learning on a graph. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*. 564–571.

[2] Deng Cai and Xiaofei He. 2012. Manifold adaptive experimental design for text categorization. *IEEE Transactions on Knowledge and Data Engineering* 24, 4 (2012), 707–719.

[3] Wenbin Cai, Ya Zhang, and Jun Zhou. 2013. Maximizing expected model change for active learning in regression. In *Proceedings of the IEEE International Conference on Data Mining*. 51–60.

[4] Xiaojun Chang, Yao-Liang Yu, and Yi Yang. 2017. Robust Top-k Multiclass SVM for Visual Category Recognition. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 75–83.

[5] Wei-Lin Chiang, Mu-Chu Lee, and Chih-Jen Lin. 2016. Parallel Dual Coordinate Descent Method for Large-scale Linear Classification in Multi-core Environments. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1485–1494.

[6] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based models for speech recognition. In *Proceedings of the Advances in Neural Information Processing Systems*. 577–585.

[7] Gautam Dasarathy, Robert Nowak, and Xiaojin Zhu. 2015. S2: An efficient graph based active learning algorithm with application to nonparametric classification. In *Proceedings of the Annual Conference on Learning Theory*. 503–522.

[8] Weijie Fu, Meng Wang, Shijie Hao, and Tingting Mu. 2017. FLAG: faster learning on anchor graph with label predictor optimization. *IEEE Transactions on Big Data* (2017). To appear.

[9] Sheng-Jun Huang, Rong Jin, and Zhi-Hua Zhou. 2014. Active learning by querying informative and representative examples. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 10, 36 (2014), 1936–1949.

[10] Ajay J Joshi, Fatih Porikli, and Nikolaos P Papanikolopoulos. 2012. Scalable active learning for multiclass image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34, 11 (2012), 2259–2273.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Proceedings of the Advances in Neural Information Processing Systems*. 1097–1105.

[12] David D Lewis and William A Gale. 1994. A sequential algorithm for training text classifiers. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*. 3–12.

[13] Xin Li and Yuhong Guo. 2013. Active Learning with Multi-Label SVM Classification. In *Proceedings of the International Joint Conferences on Artificial Intelligence*. 1479–1485.

[14] Xin Li and Yuhong Guo. 2013. Adaptive active learning for image classification. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*. 859–866.

[15] Christopher H Lin, M Mausam, and Daniel S Weld. 2016. Re-Active Learning: Active Learning with Relabeling. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 1845–1852.

[16] Wei Liu, Junfeng He, and Shih Fu Chang. 2010. Large graph construction for scalable semi-supervised learning. In *Proceedings of the International Conference on Machine Learning*. 679–686.

[17] Marius Muja and David G Lowe. 2014. Scalable nearest neighbor algorithms for high dimensional data. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36, 11 (2014), 2227–2240.

[18] Carlos Riquelme, Mohammad Ghavamzadeh, and Alessandro Lazaric. 2017. Active learning for accurate estimation of linear models. In *Proceedings of the International Conference on Machine Learning*.

[19] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision* 115, 3 (2015), 211–252.

[20] Burr Settles. 2010. Active learning literature survey. *University of Wisconsin, Madison* 52, 55-66 (2010), 11.

[21] Burr Settles and Mark Craven. 2008. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 1070–1079.

[22] Vikas Sindhwani, Partha Niyogi, Mikhail Belkin, and Sathiya Keerthi. 2005. Linear manifold regularization for large scale semi-supervised learning. In *Proceedings of the International Conference on Machine Learning*, Vol. 28.

[23] Meng Wang, Weijie Fu, Shijie Hao, Hengchang Liu, and Xindong Wu. 2017. Learning on big graph: Label inference and regularization with anchor hierarchy. *IEEE Transactions on Knowledge and Data Engineering* 29, 5 (2017), 1101–1114.

[24] Meng Wang, Weijie Fu, Shijie Hao, Dacheng Tao, and Xindong Wu. 2016. Scalable semi-supervised learning by efficient anchor graph regularization. *IEEE Transactions on Knowledge and Data Engineering* 28, 7 (2016), 1864–1877.

[25] Zheng Wang and Jieping Ye. 2015. Querying discriminative and representative samples for batch mode active learning. *ACM Transactions on Knowledge Discovery from Data* 9, 3 (2015), 17.

[26] Kai Yu, Jinbo Bi, and Volker Tresp. 2006. Active learning via transductive experimental design. In *Proceedings of the International Conference on Machine Learning*. 1081–1088.

[27] Raphael Yuster and Uri Zwick. 2005. Fast sparse matrix multiplication. *ACM Transactions on Algorithms* 1, 1 (2005), 2–13.

[28] Kai Zhang, Liang Lan, James T Kwok, Slobodan Vucetic, and Bahram Parvin. 2015. Scaling up graph-based semisupervised learning via prototype vector machines. *IEEE Transactions on Neural Networks and Learning Systems* 26, 3 (2015), 444–457.

[29] Xiaojin Zhu, John Lafferty, and Zoubin Ghahramani. 2003. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the International Conference on Machine Learning*, Vol. 3.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2757522, IEEE Transactions on Big Data

FLAG: Faster Learning on Anchor Graph with Label Predictor Optimization

Weijie Fu, Meng Wang, *Senior Member, IEEE*, Shijie Hao, and Tingting Mu, *Member, IEEE*

Abstract—Knowledge graphs have received intensive research interests. When the labels of most nodes or datapoints are missing, anchor graph and hierarchical anchor graph models can be employed. With an anchor graph or hierarchical anchor graph, we only need to optimize the labels of the coarsest anchors, and the labels of datapoints can be inferred from these anchors in a coarse-to-fine manner. The complexity of optimization is therefore reduced to a cubic cost with respect to the number of the coarsest anchors. However, to obtain a high accuracy when a data distribution is complex, the scale of this anchor set still needs to be large, which thus inevitably incurs an expensive computational burden. As such, a challenge in scaling up these models is how to efficiently estimate the labels of these anchors while keeping classification performance. To address this problem, we propose a novel approach that adds an anchor label predictor in the conventional anchor graph and hierarchical anchor graph models. In the proposed approach, the labels of the coarsest anchors are not directly optimized, and instead, we learn a label predictor which estimates the labels of these anchors with their spectral representations. The predictor is optimized with a regularization on all datapoints based on a hierarchical anchor graph, and we show that its solution only involves the inversion of a small-size matrix. Built upon the anchor hierarchy, we design a sparse intra-layer adjacency matrix over these anchors, which can simultaneously accelerate spectral embedding and enhance effectiveness. Our approach is named Faster Learning on Anchor Graph (FLAG) as it improves conventional anchor-graph-based methods in terms of efficiency. Experiments on a variety of publicly available datasets with sizes varying from thousands to millions of samples demonstrate the effectiveness of our approach.

Index Terms—Semi-supervised learning, graph-based learning, machine learning

1 Introduction

Knowledge graphs, which organize information in a structured way by describing the relationships among entities, have received much attention from both academia and industry in the past few years [16], [17], [36]. Well-known knowledge graph systems include Google Knowledge Graph, Probase, DBPedia, YAGO, and TrueKnowledge. In a knowledge graph, entities are denoted as nodes or datapoints, categories are their labels, and relationships are directed links between these datapoints [32]. However, in most data mining applications, the labels of many datapoints are missing, and it is often prohibitively labor-intensive and time-consuming to collect their labels. In contrast, the amount of unlabeled data can be huge in many domains. Semi-Supervised Learning (SSL), which exploits the prior knowledge from both unlabeled and labeled data, thus attracts considerable attention to estimate the missing labels. In recent years, various SSL methods have been developed, such as mixture methods [6], co-training [4], [48], semi-supervised support vector machines [7], [18], and graph-based methods [1], [35], [53].

In this paper, we focus on Graph-based SSL (GSSL), which is one of the most successful SSL approaches and shows the state-of-art performance in many areas [29], [45], [54]. In general, GSSL first constructs an adjacency graph to capture the data distribution for all datapoints, where the weight of the edge between two nodes represents their similarity. As a consequence, the labels for classification can be propagated from limited labeled data to remaining unlabeled data on the graph. This intuitive interpretation of the label propagation also offers GSSL more expansibility and many novel graph models have been recently proposed, such as hypergraphs for modeling higher-order relevances [51], and multigraphs for integrating multi-view features [43].

Despite the success in a variety of applications, the traditional GSSL approaches have a bottleneck in dealing with large-scale data. They face a quadratic complexity in graph construction. In addition, denoting N as the number of datapoints, the optimal solution of GSSL requires the inversion of a graph Laplacain matrix with the size $N \times N$ [53], which requires a computational cost of $O(N^3)$. The costs thus become impractical for large-scale datasets.

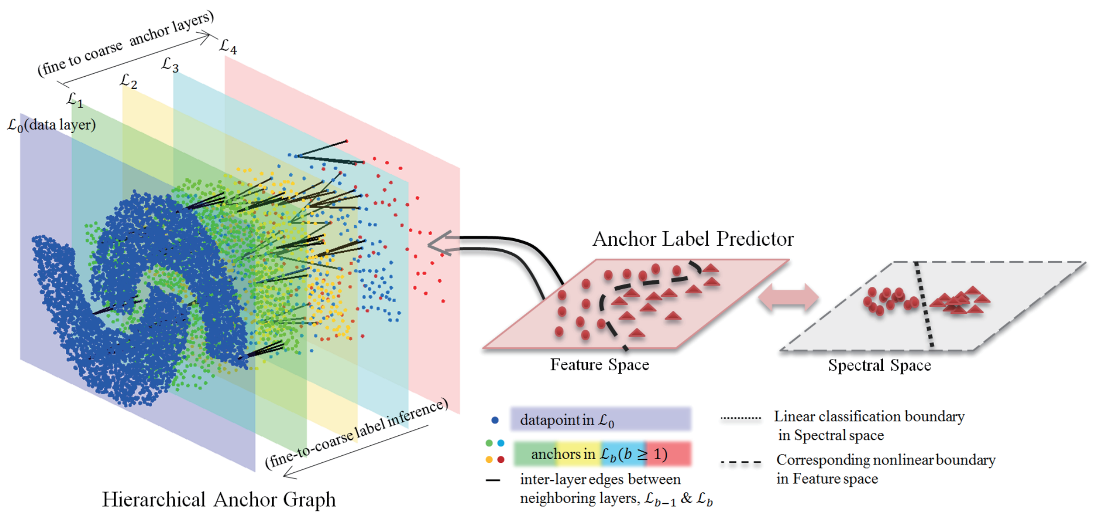
Recent works seek to employ anchors to build fast graph-based learning methods [23], [42], [47]. Here anchors refer to the points that roughly cover the data distribution in a feature space[1]. Specifically, these methods first estimate the inter-layer adjacency weights between datapoints and anchors, and then infer the labels of datapoints from these anchors. Consequently, they reduce the scale of the matrix inversion to the size of the anchor set and correspondingly speed up the optimization. However, to guarantee classification accuracies, anchors in these approaches need to be dense when the data distribution is complex, which leads to

- W. Fu, M. Wang, S. Hao are with the School of Computer and Information, Hefei University of Technology, 230009, China (E-mail: {fwj.edu, eric.mengwang, hfut.hsj}@gmail.com).
- T. Mu is with the School of Computer Science, University of Manchester, Manchester M13 9PL, U.K. (e-mail: tingting.mu@manchester.ac.uk).

[1]Throughout the manuscript, we call the space of the raw data representation as ”feature space”, and the one spanned by the eigenvectors of Graph Laplacian as ”spectral space” [31].

2332-7790 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2757522, IEEE Transactions on Big Data



(fine to coarse anchor layers)
$\mathcal{L}_1$
$\mathcal{L}_2$
$\mathcal{L}_3$
$\mathcal{L}_4$
$\mathcal{L}_0$(data layer)
(fine-to-coarse label inference)
Hierarchical Anchor Graph
Anchor Label Predictor
Feature Space
Spectral Space
datapoint in $\mathcal{L}_0$
anchors in $\mathcal{L}_b (b \geq 1)$
inter-layer edges between neighboring layers, $\mathcal{L}_{b-1}$ & $\mathcal{L}_b$
Linear classification boundary in Spectral space
Corresponding nonlinear boundary in Feature space

Fig. 1. An illustrative example of FLAG, where the labels of the coarsest anchors are first obtained based on their spectral representations with an anchor label predictor, and then spread to datapoints in a coarse-to-fine manner with the inter-layer adjacency matrices. Note that owing to the nonlinear spectral embedding, the linear predictor in FLAG actually corresponds to a nonlinear classification boundary in the feature space.

huge computational costs for large-scale datasets. Recently, a hierarchical anchor graph model with a pyramid-style structure is proposed in [41], which explores multiple-layer anchors to perform hierarchical label inference layer by layer. As a result, this approach can enlarge its finest anchor layer to improve classification performance. However, since the labels of all datapoints are essentially inferred from the coarsest anchors, to obtain a high accuracy, the number of these anchors still needs to be large. Like the pervious approaches, the computational complexity for optimizing their labels can be increasingly expensive. Therefore, these fast anchor-based methods are still insufficiently efficient to handle large-scale datasets with complex data distributions.

To address this issue, in this paper we develop a novel approach called Faster Learning on Anchor Graph (FLAG), which further scales up anchor-graph-based methods. As illustrated in Fig.1, instead of directly optimizing the labels of the coarsest anchors in hierarchical anchor graph models[2], we estimate their labels with a predictor and optimize the predictor in a hierarchical-anchor-graph-based regularization framework. To keep computational efficiency, we need to employ a linear predictor with a small hypothesis space. Therefore, we first design a sparse intra-layer adjacency matrix over the coarsest anchors with anchor hierarchy. Then we perform spectral embedding on these anchors and build their low-dimensional but discriminative spectral representations. In this way, the optimization can be computed with the matrix inversion, where the matrix size is just equivalent to the dimensionality of the spectral representation.

The rest of this paper is organized as follows. In Section 2, we briefly review related work on graph-based learning algorithms. In Section 3, we introduce the formulation of the anchor-based learning framework and analyze its dilemma. We propose our faster learning approach in Section 4. In Section 5, we make comparisons with other approaches. We finally conclude this paper in Section 6.

[2]Note that we may not further discriminate anchor graphs and hierarchical anchor graphs, as anchor graph is actually just a case of hierarchical anchor graph with an individual anchor layer.

2 Related Work

In this section, we focus on the related work on improving the efficiency of GSSL from two aspects, namely, graph construction and model optimization.

To reduce the time cost in the first aspect, many fast graph construction approaches have been developed. Chen et al. [8] first proposed to construct an approximate kNN graph for high-dimensional data via recursive Lanczos bisection. To deal with arbitrary similarity measures, Dong et al. [10] subsequently presented a simple but efficient solution based on local search algorithms. In [39], Wang et al. also proposed a multiple random divide-and-conquer approach for the graph construction and additionally presented a neighborhood propagation scheme to improve its effectiveness. To deal with the data distributed over commodity clusters, Goyal et al. [13] proposed a distributed online approaches based on sketching algorithms, and introduced it into natural language processing. Besides, Wang et al. [40] employed parallel auction algorithms to recover a sparse yet nearly balanced subgraph for social networks, which significantly reduces computational costs. As a powerful method, Approximate Nearest Neighbor Searching (ANNS) is also used to build big graphs. For example, Zhang et al. [49] employed Locality-Sensitive Hashing into the graph construction. Wang et al. [41] introduced a tree-based ANNS algorithm [30] to accelerate their weight estimation.

In spite of the progress in the fast graph construction, most GSSL methods remain challenging due to their cubic computational complexity in the optimization. To address this issue, Tsang et. al [37] introduced an ϵ-insensitive constraint into traditional LapSVM problems [27]. As a result, its optimization turns to a minimum enclosing ball problem and can be solved with core vector machines [38]. However, to obtain a satisfying accuracy, ϵ needs to be small, which makes this method close to the original LapSVM with a

2332-7790 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2757522, IEEE Transactions on Big Data

TABLE 1
Notations and definitions

Notation	Definition
$\mathcal{G} = \{\mathcal{X}, \mathcal{U}, \mathcal{Z}\}$	A hierarchical anchor graph model, where $\mathcal{X}$ and $\mathcal{U}$ indicate the sets of datapoints and anchors, respectively, and $\mathcal{Z}$ indicates the set of inter-layer adjacency edges between different sets of points.
N_0	The number of datapoints.
C	The number of classes.
l	The number of labeled datapoints.
h	The number of anchor layers.
$\mathcal{L}_b$	The bth layer in the pyramidal graph structure, where $\mathcal{L}_0$ is the layer of raw data and $\mathcal{L}_b (b \geq 1)$ denotes the b-th anchor layer.
N_b	The number of anchors in $\mathcal{L}_b (b \geq 1)$.
$\mathbf{Z}^{a,b}$	The inter-layer adjacency matrix between $\mathcal{L}_a$ and $\mathcal{L}_b$.
$Z_{is}^{a,b}$	The inter-layer adjacency weight between point i in $\mathcal{L}_a$ and point s in $\mathcal{L}_b$.
$\mathbf{W}$	The intra-layer adjacency matrix over all datapoints.
$\mathbf{\Lambda}$	The diagonal matrix of the degrees of the finest anchors.
$\mathbf{A}$	The soft label matrix of the coarsest anchor set.
$\mathbf{F}$	The soft label matrix of the datapoint set.
$\mathbf{Y}_{\mathrm{L}}$	The class indicator matrix on labeled datapoints.
$\tilde{\mathbf{L}}$	The reduced Laplacian matrix in regularization.
$\mathbf{Z}^{\mathrm{H}}$	The cascaded inter-layer adjacency matrix of a hierarchical anchor graph.
$\tilde{\mathbf{P}}$	The linear label predictor on the spectral representation.
$\tilde{\mathbf{W}}$	The intra-layer adjacency matrix over the coarsest anchors.
$\mathbf{\Sigma}$	The diagonal matrix of the degrees of the coarsest anchors.
$\tilde{\mathbf{U}}$	The the spectral representations of the coarsest anchors.

huge cost. Chen et al. [9] presented a method to combine an original kernel with the adjacency graph for scalable manifold regularization, whose cost is still larger than square in practice. Meanwhile, since the above algorithms are merely designed for binary classification, they can only learn individual classifiers for different classes. Benefitting from the development of parallel computation platforms, such as Mapreduce, many parallel algorithms are also proposed to accelerate the model optimization [2], [26], [33].

More recent works seek to employ anchors for scaling up graph-based learning, which can reduce the computational costs of both the graph construction and the model optimization. Zhang et al. [46], [47] first suggested employing a set of anchors to perform a low-rank approximation of a data distribution and span an effective model for label reconstruction. However, since it requires a dense weight matrix to build the relationships between each datapoint and all anchors, its storage requirement becomes impractical for large-scale datasets. Liu et al. [23], [24] then presented anchor graph models by constructing the inter-layer edges between datapoints and their nearby anchors. Besides, they also introduced a geometric reconstruction method for their weight estimation to improve its effectiveness. In [42], Wang et al. improved the efficiency of the reconstruction problem by introducing a new constrict, and alternatively proposed to build an intra-layer adjacency matrix over anchors rather than datapoints. Nevertheless, to obtain a reasonable accuracy, a large-size anchor set is required. As a consequence, the computational costs of the geometric reconstruction and the model optimization can be expensive. Wang et al. [41] therefore extended anchor graph models into a pyramid-style structure by exploring multiple anchor sets, which can improve the classification accuracy by introducing a finer anchor set while fixing the size of the coarsest anchor set. As a result, it carries out hierarchical label inference from the coarsest anchors in a coarse-to-fine manner, and its optimization only involves the inversion of a matrix with the size of the coarsest anchor set. Although the size of this coarsest anchor set can be forced to be small for a lower complexity, the classification performance will accordingly become worse. To obtain a high accuracy, the number of these anchors still need to be relatively large, and the time cost can be expensive.

3 Anchor-Graph-Based Learning

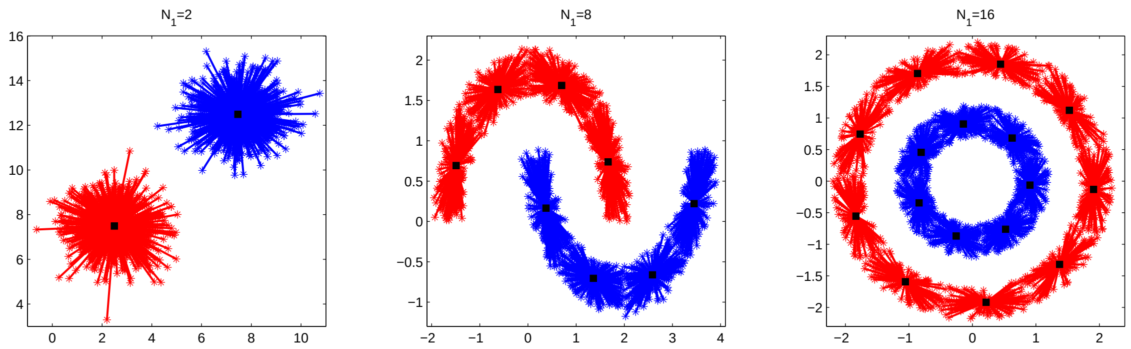
In this section, we first review the traditional fast learning approaches built upon anchor graph and hierarchical anchor graph models, and then make an analysis on their dilemma. For convenience, some important notations used throughout the paper and their explanations are listed in Table 1.

Given a dataset $\mathcal{X} = \{\mathbf{x}_1; \mathbf{x}_2; \ldots; \mathbf{x}_{N_0}\} \in \mathbb{R}^{N_0 \times D}$ with the first l samples being labeled from C distinct classes, these approaches aim at classifying the remaining unlabeled data according to their dependence on a hierarchical anchor graph. For this purpose, multiple sets of representative anchors $\mathcal{U}_b \in \mathbb{R}^{N_b \times D} (b = 1, \ldots, h)$ with fine-to-coarse sizes, namely $N_1 > \cdots > N_h$, are first generated. These anchors can be obtained by a clustering process or a random selection [23]. Then suppose the raw datapoint set locates at the bottom layer ($\mathcal{L}_0$) of the pyramid, and the remaining layers ($\mathcal{L}_b, b = 1, \ldots, h$) are all composed of multiple anchor sets. A hierarchical anchor graph can be constructed by linking all these layers up to a pyramid-style structure with the inter-layer adjacency matrices between neighboring layers, represented by $\{\mathbf{Z}^{0,1}, \ldots, \mathbf{Z}^{h-1,h}\}$.

For simplicity, we first consider the inter-layer adjacency matrix between the datapoint set in $\mathcal{L}_0$ and the finest anchor set in $\mathcal{L}_1$, namely $\mathbf{Z}^{0,1} \in \mathbb{R}^{N_0 \times N_1}$. The entries in the i-th row of this matrix are the weights between the i-th datapoint and

2332-7790 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2757522, IEEE Transactions on Big Data



(a) (b) (c)

Fig. 2. The toy examples with increasing complexities of data distributions. For each datapoint, we only show its inter-layer adjacency edge with the largest weight in the corresponding anchor graph ($h = 1$). Note that we increase N_1 by $2, 2^2, \dots,$ until all the edges are "reliable". As we can see, compared with Gaussian data (a), when data distributions become more complex (b-c), more anchors are needed to build the "reliable" inter-layer adjacency edges between datapoints and anchors in the same class.

its nearest anchors in $\mathcal{L}_1$, which can be determined by the kernel regression [12]:

$$Z_{is}^{0,1} = \frac{K_\delta(\mathbf{x}_i, \mathbf{u}_s)}{\sum_{s' \in \langle i \rangle} K_\delta(\mathbf{x}_i, \mathbf{u}_{s'})} \quad \forall s \in \langle i \rangle, \tag{1}$$

where δ is the bandwidth of the Gaussian kernel, and the notation $\langle i \rangle \subseteq [1 : N_1]$ denotes the indices of the k closest anchors of $\mathbf{x}_i$. Then we can conduct the similar kernel regression procedure to estimate the remaining inter-layer adjacency matrices. That is, to obtain $\mathbf{Z}^{b-1,b}$ between two neighboring anchor layers, we have:

$$Z_{is}^{b-1,b} = \frac{K_\delta(\mathbf{u}_i, \mathbf{u}_s)}{\sum_{s' \in \langle i \rangle} K_\delta(\mathbf{u}_i, \mathbf{u}_{s'})} \quad \forall s \in \langle i \rangle, 1 < b \leq h \tag{2}$$

where $\mathbf{u}_i$ is the i-th anchor in $\mathcal{L}_{b-1}$ and $\mathbf{u}_s$ denotes the s-th anchor in $\mathcal{L}_b$. In practice, for large-scale datasets, we can speed up the adjacency matrix estimation with ANNS techniques [30]. As such, the time cost of hierarchical anchor graph construction can be reduced to $O(N_0 \log N_1 D)$.

Let $\mathbf{A} = [\mathbf{a}_1; \mathbf{a}_2; \dots; \mathbf{a}_{N_h}] \in \mathbb{R}^{N_h \times C}$ denote the optimized label matrix of the coarsest anchor set in $\mathcal{L}_h$. With the anchor hierarchy, the label matrix of the datapoint set $\mathbf{F} \in \mathbb{R}^{N_0 \times C}$can be inferred from this anchor set in a coarse-to-fine manner:

$$\mathbf{F} = \mathbf{Z}^{0,1}(\dots(\mathbf{Z}^{h-1,h}\mathbf{A})) = \mathbf{Z}^{\mathrm{H}}\mathbf{A}, \tag{3}$$

where $\mathbf{Z}^{\mathrm{H}}$ denotes the cascaded inter-layer adjacency matrix between the data layer and the coarsest anchor layer. As pointed in [41], this adjacency matrix naturally introduces adaptive inter-layer adjacency relationships between each datapoint and its nearby coarsest anchors.

Meanwhile, an intra-layer adjacency matrix over datapoints, represented by $\mathbf{W}$, can be obtained based on the inter-layer adjacency weights between $\mathcal{L}_0$ and $\mathcal{L}_1$:

$$\mathbf{W} = \mathbf{Z}^{0,1}\mathbf{\Lambda}^{-1}\mathbf{Z}^{0,1^{\mathrm{T}}} \in \mathbb{R}^{N_0 \times N_0}, \tag{4}$$

where $\mathbf{\Lambda}$ is a diagonal matrix with $\Lambda_{ss} = \sum_{i=1}^{N_0} Z_{is}^{0,1}$. As we can see that, $W_{ij} > 0$ means two datapoints share at least one finest anchor.

Let $\mathbf{Y}_{\mathrm{L}} = [\mathbf{y}_1; \dots; \mathbf{y}_l] \in \mathbb{R}^{l \times C}$ denote the class indicator matrix on labeled datapoints, where $y_{ir} = 1$ if $\mathbf{x}_i$ belongs to class r, and $y_{ir} = 0$ otherwise. To deal with a standard multi-class SSL problem, Hierarchical Anchor Graph Regularization (HAGR) [41] is formulated by minimizing the following problem:

$$\begin{aligned}\mathcal{Q}_{\mathbf{A}} &= \sum_{i=1}^{l} \| \mathbf{Z}_{i\cdot}^{\mathrm{H}}\mathbf{A} - \mathbf{y}_i \|^2 + \frac{\lambda}{2}\sum_{i,j=1}^{N_0} W_{ij} \| \mathbf{Z}_{i\cdot}^{\mathrm{H}}\mathbf{A} - \mathbf{Z}_{j\cdot}^{\mathrm{H}}\mathbf{A} \|^2 \\ &= \|\mathbf{Z}_{\mathrm{L}}^{\mathrm{H}}\mathbf{A} - \mathbf{Y}_{\mathrm{L}}\|_{\mathrm{F}}^2 + \lambda \mathrm{tr}(\mathbf{A}^{\mathrm{T}}\mathbf{Z}^{\mathrm{H}^{\mathrm{T}}}(\mathbf{I} - \mathbf{W})\mathbf{Z}^{\mathrm{H}}\mathbf{A}) \\ &= \|\mathbf{Z}_{\mathrm{L}}^{\mathrm{H}}\mathbf{A} - \mathbf{Y}_{\mathrm{L}}\|_{\mathrm{F}}^2 + \lambda \mathrm{tr}(\mathbf{A}^{\mathrm{T}}\widetilde{\mathbf{L}}\mathbf{A}),\end{aligned} \tag{5}$$

where $\|\cdot\|_{\mathrm{F}}$ stands for the Frobenius norm, λ is the trade-off parameter balancing different terms, $\mathbf{Z}_{\mathrm{L}}^{\mathrm{H}}$ is the labeled part of $\mathbf{Z}^{\mathrm{H}}$, and

$$\widetilde{\mathbf{L}} = \mathbf{Z}^{\mathrm{H}^{\mathrm{T}}}\mathbf{Z}^{\mathrm{H}} - \mathbf{Z}^{\mathrm{H}^{\mathrm{T}}}\mathbf{Z}^{0,1}\mathbf{\Lambda}^{-1}\mathbf{Z}^{0,1^{\mathrm{T}}}\mathbf{Z}^{\mathrm{H}} \in \mathbb{R}^{N_h \times N_h}. \tag{6}$$

With the native spare matrix multiplication [44], the cost of Eq.6 scales as $O(N_0 N_h k)$, as all the inter-layer adjacency matrices in $\mathbf{Z}^{\mathrm{H}}$, namely $\mathbf{Z}^{b-1,b}$s, are k-sparse.

Differentiating $\mathcal{Q}_{\mathbf{A}}$ with respect to $\mathbf{A}$ and setting it to zero, we can obtain an optimal solution in the closed-form:

$$\mathbf{A} = (\mathbf{Z}_{\mathrm{L}}^{\mathrm{H}^{\mathrm{T}}}\mathbf{Z}_{\mathrm{L}}^{\mathrm{H}} + \lambda\widetilde{\mathbf{L}})^{-1}\mathbf{Z}_{\mathrm{L}}{}^{\mathrm{T}}\mathbf{Y}_{\mathrm{L}}. \tag{7}$$

Evidently, this matrix inversion takes a cost of $O({N_h}^3)$, namely, a cubic cost with respect to the number of the coarsest anchors.

Note that when the above hierarchical anchor graph model becomes an anchor graph with an individual anchor layer, namely $h = 1$, HAGR degrades to the Anchor Graph Regularization (AGR) method [23].

The above fast anchor-based approaches have shown their promising results for large-scale SSL. However, according to Eq.3, the effectiveness of the label inference highly depends on the distribution of the coarsest anchors. To obtain a high accuracy, the cascaded inter-layer relationships are supposed to be "reliable", namely, connections only exist between the datapoints and the coarsest anchors within the same class. For this purpose, when a data distribution becomes more complex, the required number of the coarsest anchors will increase dramatically. To make it intuitive,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2757522, IEEE Transactions on Big Data

Figs.2a-2c illustrate this observation by increasing the complexity of the data distribution (for simplicity, we construct anchor graphs with h=1). As we can see, to build "reliable" inter-layer edges, two anchors are sufficient for the example in Fig.2a. However, for the examples in Fig.2b-2c, more anchors are clearly needed. According to Eq.7, it can lead to a large computational burden in the model optimization [41]. In summary, since a large-size coarsest anchor set is required for obtaining a high accuracy, AGR and HAGR still face a dramatic increase of the computational cost for optimizing their labels.

4 Faster Learning on Anchor Graph(FLAG)

To address the above issue, we propose the FLAG model. It employs a label predictor on the spectral representations of the coarsest anchors to estimate their labels, which is optimized in a regularization framework over all datapoints. We first propose the formulation of learning with an anchor label predictor in Section 4.1. We introduce the efficient estimation of spectral representations by employing a sparse intra-layer adjacency matrix over anchors in Section 4.2. The complete FLAG model and its optimization are presented in Section 4.3, followed by a complexity analysis in Section 4.4.

4.1 Learning with Label Predictor Optimization

Instead of directly optimizing the labels of the coarsest anchors, we estimate the labels of these anchors with a label predictor. Let $\mathbf{U} = [\mathbf{u}_1; \ldots; \mathbf{u}_{N_h}] \in \mathbb{R}^{N_h \times D}$ denote the raw representation of the coarsest anchor set. Then based on the anchor label predictor $p : \mathcal{R}^D \to \mathcal{R}^C$, the soft label matrix of this anchor set can be obtained as $[p(\mathbf{u}_1); \ldots; p(\mathbf{u}_{N_h})] \in \mathbb{R}^{N_h \times C}$. Note that in HAGR, $p(\mathbf{U}) = \mathbf{A}$.

Recall a standard multi-class SSL problem, where a set of labeled datapoints $\mathbf{x}_i$ $(i = 1, \ldots, l)$ with the corresponding discrete labels $y_i \in \{1, \ldots, C\}$ is given and the goal is to estimate the labels of the remaining unlabeled datapoints. Let $\mathbf{Y}_l = [\mathbf{y}_1; \mathbf{y}_2; \ldots; \mathbf{y}_l] \in \mathbb{R}^{l \times C}$ denote the class indicator matrix of the labeled data with $Y_{ij} = 1$ if $\mathbf{x}_i$ belongs to class j and $Y_{ij} = 0$ otherwise. Meanwhile, similar to HAGR, we denote $\mathbf{W}$ as the intra-layer adjacency matrix over datapoints, and $\mathbf{Z}^{\mathrm{H}}$ as the cascaded inter-layer adjacency matrix of a hierarchial anchor graph. Then, the regularization framework of learning with an anchor label predictor can be formulated as the following minimization problem:

$$\begin{aligned}\mathrm{argmin}_p \mathcal{Q}_p = &\sum_{i=1}^{l} \|\mathbf{Z}^{\mathrm{H}}_{i\cdot} p(\mathbf{U}) - \mathbf{y}_i\|_{\mathrm{F}}^2 + \mu\Omega(p) \\ &+ \frac{\lambda}{2} \sum_{i,j=1}^{N_0} W_{ij} \|\mathbf{Z}^{\mathrm{H}}_{i\cdot} p(\mathbf{U}) - \mathbf{Z}^{\mathrm{H}}_{j\cdot} p(\mathbf{U})\|_{\mathrm{F}}^2.\end{aligned} \tag{8}$$

where $\Omega(\cdot)$ denotes a regularizer on the label predictor, and μ is the corresponding trade-off parameter. As a result, compared with HAGR where a large number of soft labels need to be learned, Eq.8 only needs to optimize a label predictor.

However, the raw representation in the feature space is usually insufficiently powerful to capture the similarity between anchors, and may lead to a poor performance for the following label prediction. Therefore, there are two issues left: (1). how to get effective representations of these coarsest anchors, and (2). how to optimize the corresponding predictor with a faster solution. We address them in Section 4.2 and 4.3, respectively.

4.2 Efficient Estimation of Spectral Representations

In this section, we introduce how to efficiently estimate a discriminative representation of the coarsest anchor set via spectral embedding.

For this purpose, we first consider the issue of constructing an intra-layer adjacency matrix $\tilde{\mathbf{W}}$ over the coarsest anchors, aiming at performing an efficient spectral embedding procedure on these anchors while improving effectiveness. Although there exists a similar method for building the adjacency relationships between these anchors based on an anchor graph [42], our intra-layer adjacency matrix is actually quite different.

First, we determine the relationships of the coarsest anchors with the anchor hierarchy if a hierarchical anchor graph consists of multiple anchor layers. In particular, we employ the cascaded inter-layer adjacency matrix $\mathbf{Z}^{\mathrm{H}}$ to estimate the intra-layer weight between the coarsest anchors:

$$\tilde{W}_{ij} = \sum_{s=1}^{N_0} Z^{\mathrm{H}}_{si} Z^{\mathrm{H}}_{sj}. \tag{9}$$

According to Eq.9, once two coarsest anchors share at least one common datapoint based on the cascaded inter-layer adjacency relationships, there will be an intra-layer adjacency edge between them. Meanwhile, if a hierarchical anchor graph only contains an individual anchor layer, this step degrades to the same situation in [42].

The above adjacency relationships can also be expressed in a matrix form:

$$\tilde{\mathbf{W}} = \mathbf{Z}^{\mathrm{H}^{\mathrm{T}}} \mathbf{Z}^{\mathrm{H}} \in \mathbb{R}^{N_h \times N_h}. \tag{10}$$

which can be efficiently computed in $O(N_0 N_h k)$.

Second, we impose a strict sparse constraint on the above intra-layer adjacency matrix. Since Eq.10 accumulates all the nonnegative intra-layer adjacency weights, the obtained adjacency matrix is usually dense, which will slow down the spectral embedding. Therefore, we further prune this intra-layer adjacency matrix by forcing it to be k-sparse:

$$\tilde{W}_{ij} = \begin{cases} \tilde{W}_{ij} & \text{if } \mathbf{u}_j \text{ and } \mathbf{u}_i \text{ are "close"}, \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

where we define the closeness according to their accumulated adjacency weights rather than the Euclidean distance. Practically, we first retain the adjacency edges with the k largest weights for each coarsest anchor, and then make them undirected. We can thus obtain a sparse and symmetric adjacency matrix, where only the elements corresponding to the anchors with large correlations are reserved.

When two coarsest anchors are close to a classification boundary but locate at different classes, the above pruning operation can remove their intra-layer adjacency relationship and accordingly improve the effectiveness of the intra-layer adjacency matrix. It is understandable that, as the data density between different classes is much lower

2332-7790 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2757522, IEEE Transactions on Big Data

than that within the same class, the accumulated intra-layer adjacency weight of this suspicious adjacency edge tends to be much smaller than those within the same class. In the experimental section, we will compare the above intra-layer adjacency matrix with the one built upon the RBF kernel under the different settings of graph structures.

Now we look back to the remaining issue of performing the spectral embedding on the coarsest anchors to obtain their representations in a spectral space. Donate $\boldsymbol{\Sigma}$ as a diagonal degree matrix with $\Sigma_{ii} = \sum_{j=1}^{N_h} \tilde{W}_{ij}$. Then the embedding can be formulated as

$$\text{argmin} \sum_{ij} \tilde{W}_{ij} || \frac{\tilde{\mathbf{u}}_i}{\sqrt{\Sigma_{ii}}} - \frac{\tilde{\mathbf{u}}_j}{\sqrt{\Sigma_{jj}}} ||^2, \tag{12}$$

where $\tilde{\mathbf{u}}_i$ is the spectral representation of the coarsest anchor $\mathbf{u}_i$. According to Eq.12, a pair of anchors with a small intra-layer adjacency weight will have dissimilar spectral representations, and otherwise, their representations tend to be similar.

Let $\tilde{\mathbf{U}} = [\tilde{\mathbf{u}}_1; \ldots; \tilde{\mathbf{u}}_{N_h}] \in \mathbb{R}^{N_h \times d}$ denote the optimized d-dimensional representation of the coarsest anchor set, where each row is the spectral representation of a coarsest anchor. Meanwhile, denote $\mathbf{S} = \boldsymbol{\Sigma}^{-\frac{1}{2}} \tilde{\mathbf{W}} \boldsymbol{\Sigma}^{-\frac{1}{2}}$ as the normalized intra-layer adjacency matrix over the coarsest anchors. Then the spectral-representation matrix $\tilde{\mathbf{U}}$ can be obtained according to the following optimization problem:

$$\begin{aligned} &\text{argmin}_{\tilde{\mathbf{U}}} \tilde{\mathbf{U}}^{\mathrm{T}} (\mathbf{I} - \mathbf{S}) \tilde{\mathbf{U}} \\ &\text{s.t.} \tilde{\mathbf{U}}^{\mathrm{T}} \tilde{\mathbf{U}} = \mathbf{I} \end{aligned} \tag{13}$$

which is solved by combing the d smallest eigenvectors of the matrix $(\mathbf{I} - \mathbf{S})$ [31]. Based on Arnoldi technique [21], the time cost of calculating the first d eigenvectors via Eq.13 scales as $O(N_h k d)$, where k is the average number of non-zero entries in each column of this matrix. Of note, the first eigenvectors reflect the main structure of data distributions, and the remaining ones indicate the small difference or the noise. As a result, we can obtain the discriminative spectral representation of this coarsest anchor set with a small number of eigenvectors, which insures its low dimensionality.

When the intra-layer adjacency matrix over the coarsest anchors is "ideal" (which means the coarsest anchors in different classes have zero weights with each other, and those within the same class have large weights with each other), we can employ their spectral representations with C eigenvectors for C-classes, namely, we have $d = C$. Even this adjacency matrix is noisy in most real-world applications and more eigenvectors are required, the dimensionality of the spectral representations can still be much smaller than the size of the anchor set. Later in the experimental section, we will empirically show both the effectiveness and efficiency of the low-dimensional spectral representations .

Note that one can also conduct spectral embedding on the datapoint set or another finer anchor set in a hierarchical anchor graph. Nevertheless, it will lead to a larger computational cost for constructing the corresponding intra-layer adjacency matrix and also slow down the spectral embedding. On the contrary, the spectral representations of the coarsest anchors can be efficiently computed. Besides, as these anchors can roughly cover the data distribution, based on the corresponding intra-layer adjacency matrix, their spectral representations can still be discriminative .

4.3 Faster Optimization with a Linear Predictor

So far we have described how to efficiently estimate the spectral representations of the coarsest anchors. Now we integrate these spectral representations into the regularization framework on a hierarchical anchor graph, and propose the complete FLAG model with a faster optimization.

Let $\tilde{p} : \mathcal{R}^d \to \mathcal{R}^C$ denote the label predictor on the spectral representation of the coarsest anchor set $\tilde{\mathbf{U}} \in \mathbb{R}^{N_h \times d}$. The label matrix of this anchor set can be obtained as $\tilde{p}(\tilde{\mathbf{U}}) \in \mathbb{R}^{N_h \times C}$. To keep computational efficiency, we only consider a linear predictor, represented by $\tilde{\mathbf{P}}$. As such, we can obtain the soft label matrix of the coarsest anchor set:

$$\mathbf{A} = \tilde{\mathbf{U}} \tilde{\mathbf{P}} \in \mathbb{R}^{N_h \times C}. \tag{14}$$

Of note, benefitting from the nonlinear embedding, this simple label predictor can still classify the coarsest anchors with a complex distribution in the feature space.

Then the label matrix on datapoints can be inferred by

$$\mathbf{F} = \mathbf{Z}^{\mathrm{H}} \tilde{\mathbf{U}} \tilde{\mathbf{P}}, \tag{15}$$

where $\mathbf{Z}^{\mathrm{H}}$ is the cascaded inter-layer adjacency matrix in hierarchical anchor graph models. Compared with pervious anchor-based methods [23], [42], [41], to eventually estimate the labels of unlabeled datapoints, we only need to optimize the above anchor label predictor, of which the size is proportional to the dimensionality of the spectral representation rather than the number of the coarsest anchors.

Now, by integrating the above linear label predictor into the previous regularization framework, Faster Learning on Anchor Graph (FLAG) can be finally formulated as the following optimization problem:

$$\begin{aligned} \text{argmin}_{\tilde{p}} \mathcal{Q}_p = & \sum_{i=1}^{l} \|\mathbf{Z}_{i\cdot}^{\mathrm{H}} \tilde{\mathbf{U}} \tilde{\mathbf{P}} - \mathbf{y}_i\|^2 + \mu ||\tilde{\mathbf{P}}||_{\mathrm{F}}^2 \\ & + \frac{\lambda}{2} \sum_{i,j=1}^{N_0} W_{ij} \|\mathbf{Z}_{i\cdot}^{\mathrm{H}} \tilde{\mathbf{U}} \tilde{\mathbf{P}} - \mathbf{Z}_{j\cdot}^{\mathrm{H}} \tilde{\mathbf{U}} \tilde{\mathbf{P}}\|^2. \end{aligned} \tag{16}$$

Denoting $\mathbf{Z}_{\mathrm{L}}^{\mathrm{H}}$ as the labeled part of $\mathbf{Z}^{\mathrm{H}}$, we can rewrite Eq.16 into the matrix form:

$$\begin{aligned} \mathcal{Q}_{\tilde{\mathbf{P}}} &= \|\mathbf{Z}_{\mathrm{L}}^{\mathrm{H}} \tilde{\mathbf{U}} \tilde{\mathbf{P}} - \mathbf{Y}_{\mathrm{L}}\|_{\mathrm{F}}^2 + \mu \mathrm{tr}(\tilde{\mathbf{P}}^{\mathrm{T}} \tilde{\mathbf{P}}) \\ &\quad + \lambda \mathrm{tr}(\tilde{\mathbf{P}}^{\mathrm{T}} \tilde{\mathbf{U}}^{\mathrm{T}} \mathbf{Z}^{\mathrm{H}^{\mathrm{T}}} (\mathbf{I} - \mathbf{W}) \mathbf{Z}^{\mathrm{H}} \tilde{\mathbf{U}} \tilde{\mathbf{P}}) \\ &= \|\mathbf{Z}_{\mathrm{L}}^{\mathrm{H}} \tilde{\mathbf{U}} \tilde{\mathbf{P}} - \mathbf{Y}_{\mathrm{L}}\|_{\mathrm{F}}^2 + \mu \mathrm{tr}(\tilde{\mathbf{P}}^{\mathrm{T}} \tilde{\mathbf{P}}) + \lambda \mathrm{tr}(\tilde{\mathbf{P}}^{\mathrm{T}} \tilde{\mathbf{L}} \tilde{\mathbf{P}}), \end{aligned} \tag{17}$$

where $\tilde{\mathbf{L}}$ is the reduced Laplacian over the spectral representation:

$$\begin{aligned} \tilde{\mathbf{L}} &= \tilde{\mathbf{U}}^{\mathrm{T}} \mathbf{Z}^{\mathrm{H}^{\mathrm{T}}} (\mathbf{I} - \mathbf{Z}^{0,1} \boldsymbol{\Lambda}^{-1} \mathbf{Z}^{0,1^{\mathrm{T}}}) \mathbf{Z}^{\mathrm{H}} \tilde{\mathbf{U}} \in \mathbb{R}^{d \times d} \\ &= \tilde{\mathbf{U}}^{\mathrm{T}} \mathbf{Z}^{\mathrm{H}^{\mathrm{T}}} \mathbf{Z}^{\mathrm{H}} \tilde{\mathbf{U}} - \tilde{\mathbf{U}}^{\mathrm{T}} \mathbf{Z}^{\mathrm{H}^{\mathrm{T}}} \mathbf{Z}^{0,1} \boldsymbol{\Lambda}^{-1} \mathbf{Z}^{0,1^{\mathrm{T}}} \mathbf{Z}^{\mathrm{H}} \tilde{\mathbf{U}}. \end{aligned} \tag{18}$$

With simple algebra, the optimization of the label predictor can be computed with a matrix inversion procedure:

$$\tilde{\mathbf{P}} = (\tilde{\mathbf{U}}^{\mathrm{T}} \mathbf{Z}_{\mathrm{L}}^{\mathrm{H}^{\mathrm{T}}} \mathbf{Z}_{\mathrm{L}}^{\mathrm{H}} \tilde{\mathbf{U}} + \mu \mathbf{I} + \lambda \tilde{\mathbf{L}})^{-1} \tilde{\mathbf{U}}^{\mathrm{T}} \mathbf{Z}_{\mathrm{L}}^{\mathrm{H}^{\mathrm{T}}} \mathbf{Y}_{\mathrm{L}}, \tag{19}$$

where the matrix size is equivalent to the dimensionality of the spectral representation d. Compared with HAGR which

2332-7790 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2757522, IEEE Transactions on Big Data

TABLE 2
Comparison of time complexity of five graph-based methods.

Methods	Graph Construction	Model Optimization
LLGC	$O(N_0 \log N_0 D)$	$O(N_0^3)$
AGR, EAGR	$O(N_0 \log N_1 D)$	$O(N_0 N_1 k + N_1^3)$
HAGR	$O(N_0 \log N_1 D)$	$O(N_0 N_h k + N_h^3)$
FLAG	$O(N_0 \log N_1 D)$	$O(N_0 N_h k + N_h dk + N_h d^2)$
Note that for each method, we have $N_0 > N_1 > N_h \gg d$.		

optimizes all the labels of the coarsest anchors via Eq.7, the computation of Eq.19 can be more efficient, as d is practically much smaller than the size of the coarsest anchor set N_h.

Based on the optimized predictor, the soft label matrix on datapoints $\mathbf{F}$ can be estimated by Eqs.14-15. Finally, we can obtain a hard label for any unlabeled datapoint:

$$\widehat{y}_i = \mathrm{argmax}_{r\in\{1,\ldots,c\}} \frac{F_{i,r}}{\beta_r}, i = l+1, \ldots, N_x, \tag{20}$$

where $\beta_r = \mathbf{1}^{\mathrm{T}}\mathbf{F}_{\cdot r}$ is a normalization factor [53], and $\mathbf{F}_{\cdot r}$ is the rth column of $\mathbf{F}$.

4.4 Complexity Analysis

In this section, we summarize the steps of the proposed approach and analyze their time complexities.

(1). Compute inter-layer adjacency matrices to build a hierarchical anchor graph with Eqs.1-2, and construct a sparse intra-layer matrix over the coarsest anchors with Eqs.9-11. The computational cost of the former scales as $O(N_0 \log N_1 D)$, and the latter is $O(N_0 N_h k)$.

(2). Estimate the spectral representations of the coarsest anchors with d eigenvectors, whose cost scales as $O(N_h dk)$.

(3). Calculate the reduced Laplacian in Eq.18 with $O(N_0 kd + N_h d^2)$, and carry out the graph regularization via Eq.19 with $O(d^3)$.

(4). Predict the labels of the coarsest anchors with Eq.14 in $O(N_h dC)$, and infer the labels of datapoints based on Eq.15 in a coarse-to-fine manner, which scales as $O(N_0 kC)$.

Since we usually have $N_h \gg d \gg C$, the total time complexity of FLAG can be simplified to

$$O[N_0(\log N_1 D + N_h k) + N_h d(k+d)].$$

Table 2 lists the time complexities of LLGC (Learning with Local and Global Consistency [50], a typical graph-based SSL method), AGR, EAGR ([42], an improved AGR method), HAGR, and FLAG, in which the ANNS-based kernel regression [41] is applied into all these methods for a fair comparison. From the table we can observe that, although LLGC reduces its graph construction to a linear complexity with respect to the data size, it still faces a cubic cost for its matrix inversion during the optimization. To obtain a high accuracy with more anchors, AGR and EAGR also face a dramatic increase of the computational cost in their model optimization. Then we focus on HAGR and our FLAG. As both of them have the same procedures of computing a series of inter-layer matrices and an intra-layer adjacency matrix (in HAGR means the first term in Eq.6), the comparison of the complexities comes from their rest parts. As we can see from Eq.18 and Eq.6, the remaining costs of calculating the reduced Laplacian in FLAG and HAGR are $O(N_0 kd + N_h d^2)$ and $O(N_0 N_h k)$, respectively. Meanwhile, rather than conducting the matrix inversion with a cost of $O(N_h^3)$, FLAG implements a spectral embedding procedure with a linear complexity of $O(N_h dk)$ on the pruned adjacency matrix, and its optimization only involves the matrix inversion on a $d \times d$ matrix with a cubic cost of $O(d^3)$. Since d and k are much smaller than N_h, we have $O(N_0 kd + N_h kd + N_h d^2 + d^3) \ll O(N_0 N_h k + N_h^3)$. As a result, FLAG has a much less cost than HAGR and other conventional fast learning approaches, and can efficiently deal with large-scale datasets with more coarsest anchors.

5 Experiment

Now we investigate both the effectiveness and efficiency of FLAG on real-world datasets. All the experiments are implemented on a PC with E5-2620 v2 @2.10 GHz and 64G RAM. Here we use the following five datasets with sizes varying from thousands to millions. The descriptions of these datasets are given in below, and some statistics of them are listed in Table 3.

(1) *FaceMIT:* The FaceMIT data set [3] contains 6,977 training samples (2,429 positives, 4,548 negatives) and 24,045 testing samples (472 positives, 23,573 negatives). In our experiment, we only employ the training set for binary classification.

(2) *Newsgroup:* A tiny version of the 20newsgroups data, with binary occurance data for 100 words across 16,242 postings. It is tagged the postings by the highest level domain in the array 'newsgroups' [34].

(3) *Extended MNIST:* This dataset is widely used in many large-scale graph-based works [14], [19], [24]. The original MNIST dataset contains 70,000 samples of handwritten digits from '0' to '9' [20]. Each of the ten classes contains about 7,000 samples, which are images centered in a 28×28 field by computing the center of mass of the pixels. This extended dataset is constructed by translating the original images in MNIST one pixel in each direction. As a result, there are 630,000 samples in 900 dimensions by using the normalized grayscale values as features.

(4) *Extended USPS:* The USPS dataset is for hand-written digits recognition and contains 7,291 training samples and 2,007 test samples of digit images. All the digits in the training set of USPS are extended by shifting the 16 × 16 images in all directions for up to five pixels [38]. There are 882,211 samples in 676 dimensions in total. We directly use the normalized grayscale values as features.

2332-7790 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2757522, IEEE Transactions on Big Data

TABLE 3
Details of the five databases used in our experiments.

	FaceMIT	Newsgroup	Extended MNIST	Extended USPS	MNIST8M
# of instances	6,977	16,242	630,000	882,211	8,100,000
# of categories	2	4	10	10	10
# of dimensions	361	100	900	676	784

(5) *MNIST8M:* This dataset has been used in [15], [22], [25] to verify the efficiency of large-scale learning algorithms. It contains totally 8,100,000 samples in 784 dimensions.

For convenience, these datasets are categorized into small, medium and large sizes. Specifically, we regard FaceMIT, and Newsgroup as small-size datasets, Extended MNIST, and Extended USPS as medium-size datasets, and MNIST8M as a large-size dataset.

5.1 Comparison to State-of-the-art Approaches

We compare FLAG with several state-of-the-art fast learning approaches, such as AGR, EAGR and HAGR, to demonstrate its efficiency and effectiveness. We also report the performances of two baseline methods including 1NN, and linear SVM. The methods for comparison are described in below.

(1) 1NN: It determines the label of an instance by referring to its nearest datapoint in the labeled set.

(2) LSVM [11]: We use the Linear SVM implementation from LIBLINEAR, which is a library for large-scale linear classification.

(3) LLGC [50]: This typical graph-based method directly optimizes the labels of datapoints. In our experiments, we employ a kNN strategy for its graph construction.

(4) AGR-N_1 [23]: It is the original anchor-graph-based learning method with an individual anchor layer ($h = 1$), where N_1 is the number of anchors. Of note, this method can be viewed as a reduced version of HAGR.

(5) EAGR-N_1 [42]: Compared with AGR-N_1, it remodifies the regularizer by constructing a smoothness constraint on the labels of anchors.

(6) FLAG-N_1: Compared with AGR-N_1, it first imposes a strict sparse constraint on the intra-layer adjacency matrix over anchors to obtain their spectral representations, and then introduces a linear predictor to estimate the labels of these anchors. As a simplest version of the FLAG method, it is proposed to verify the efficiency of learning an anchor label predictor for small-size datasets and show the effectiveness of adding a finer anchor layer for larger-size datasets.

(7) HAGR-N_1-. . .-N_h [41]: This HAGR approach is built upon a hierarchical anchor graph with $h(h > 1)$ anchor layers, where $N_b(b = 1, \ldots, h)$ denotes the scale of the b-th anchor layer. Compared with method (4), it infers the labels of datapoints in a coarse-to-fine manner.

(8) FLAG-N_1-. . .-N_h: Different from method (7), it integrates an anchor label predictor with the hierarchial label inference to estimate the labels of datapoints. Meanwhile, compared with method (6), the spectral representations of the coarsest anchors here are obtained based on their sparse intra-layer adjacency matrix, which is built upon the anchor hierarchy. This approach is designed for classification on medium-size and large-size datasets.

For a fair comparison, the kernel widths (δ) in above approaches are set by cross validation, and the trade-off parameters (λ and μ) are tuned to their optimal values. Besides, we empirically choose sparse parameters (k) from 2 to 6 for all approaches.

5.1.1 *Small-Size Datasets*

We first conduct experiments on FaceMIT and Newsgroup. As the sizes of these datasets are small, we only build anchor graphs with an individual anchor set, where N_1=2,000 and N_1=3,000 are used for FaceMIT and Newsgroup, respectively. We perform FLAG on these anchor graphs with the optimized d chosen from 1 to 30. We vary the number of labeled samples in $\{2, 4, \ldots, 10\}$ per class. The average classification accuracies over 20 trials are reported in Table 4 and 5, where the time costs with 10 labels per class are listed at the last column.

From the tables, we obtain the following observations. **First**, the accuracies of all graph-based SSL approaches stay at a higher level than those of LSVM and 1NN, especially when the number of labeled datapoints is small, which demonstrates the importance of leveraging unlabeled data in SSL. **Second**, benefitting from anchor-based label inference, both AGR-N_1 and EAGR-N_1 reduce the time cost to a much lower level than LLGC. However, compared with LLGC, their classification accuracies can be worse in some cases. **Third**, FLAG-N_1 performs SSL with the smallest time cost, and consistently obtains higher accuracies than other three approaches. These results demonstrate the superiority of FLAG-N_1 for scaling up graph-based learning.

5.1.2 *Medium-Size Datasets*

For two medium-size datasets, we first follow [23] and perform PCA to reduce the original image dimensions to 86. Then for each of them, we construct two anchor graphs with N_1 =5,000 and N_1=20,000, and one hierarchical anchor graph with N_1=200,000, N_2=20,000. We perform FLAG on these graphs with the optimized d chosen from 10 to 300. The number of labeled samples varies from $\{2, 4, \ldots, 10\}$ and $\{20, 40, \ldots, 100\}$ per class for Extended MNIST and Extended USPS, respectively. The average classification accuracies over 20 trials are shown in Table 6 and 7, and the time costs are reported at the last column.

From the results in these tables, the following observations can be made. **First**, compared with other approaches under the same anchor configurations, FLAG consistently obtains better performances with less time costs. This result demonstrates both the efficiency and effectiveness of our label predictor optimization. **Second**, anchor-graph-based approaches with N_1 =20,000 almost obtain higher

2332-7790 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2757522, IEEE Transactions on Big Data

TABLE 4
The comparison of different approaches on FaceMIT.

Num of labels per class	2	4	6	8	10	Time cost (in second)
1NN	55.36±10.66	64.91±10.26	68.29±7.45	68.92±5.59	72.36±5.45	0.03
LSVM	50.21±16.32	56.73±15.82	64.44±14.44	69.35±10.18	73.72±9.40	0.11
LLGC	71.43±12.23	83.37±8.85	85.36±4.59	88.11±4.56	89.62±3.73	8.96
AGR-2,000	68.06±14.18	77.65±10,81	82.18±5.86	86.86±5.18	87.60±5.88	0.85
EAGR-2,000	66.11±14.23	79.18±9.58	83.37±6.50	85.31±5.97	88.66±4.31	0.76
FLAG-2,000	**80.43±14.41**	**86.06±10.19**	**90.28±3.78**	**91.33±1.68**	**91.96±1.90**	0.54

The best results are shown in **bold**. The last column shows time costs with 10 labeled data per class.

TABLE 5
The comparison of different approaches on Newsgroup.

Num of labels per class	2	4	6	8	10	Time cost (in second)
1NN	33.30±6.12	37.26±5.69	39.46±5.65	40.57±6.03	40.91±5.61	0.04
LSVM	40.42±5.77	48.07±3.86	53.32±4.29	57.26±4.44	60.43±3.09	0.01
LLGC	52.12±6.67	57.48±3.75	59.85±2.92	60.02±3.23	61.41±2.42	85.73
AGR-3,000	54.55±3.70	59.93±5.48	61.79±3.72	62.82±3.21	63.91±2.85	1.98
EAGR-3,000	54.36±6.86	59.39±6.55	61.67±4.39	62.69±3.00	63.21±2.69	1.85
FLAG-3,000	**60.28±7.06**	**66.43±3.01**	**68.80±2.01**	**69.01±1.62**	**69.67±1.67**	1.33

The best results are shown in **bold**. The last column shows time costs with 10 labeled data per class.

TABLE 6
The comparison of different approaches on Extended MNIST.

Num of labels per class	2	4	6	8	10	Time cost (in second)
1NN	42.21±3.45	49.11±3.26	54.55±2.85	57.56±2.16	60.04±1.70	2.56
LSVM	45.74±2.49	51.96±3.13	56.44±2.65	59.78±2.77	61.79±1.96	3.15
AGR-5,000	79.47±2.20	83.11±1.89	85.77±2.20	87.53±0.95	88.53±0.82	7.98
EAGR-5,000	79.76±1.86	83.82±1.91	86.92±1.37	88.72±0.60	89.48±0.73	7.24
FLAG-5,000	85.15±2.70	89.27±3.00	91.24±1.59	92.40±0.39	92.57±0.29	5.81
AGR-20,000	79.15±2.49	84.05±2.60	86.90±1.82	89.27±1.35	90.17±1.31	151.62
EAGR-20,000	79.18±2.51	84.41±2.26	87.52±1.64	89.98±1.10	90.86±1.10	150.04
FLAG-20,000	81.66±3.21	89.48±2.61	91.36±1.67	93.17±0.93	94.05±0.24	7.91
HAGR-200,000-20,000	83.01±2.18	87.46±1.93	90.04±1.42	91.66±0.92	92.62±0.46	156.83
FLAG-200,000-20,000	**87.78±1.97**	**92.42±1.83**	**94.01±1.38**	**95.02±0.39**	**95.07±0.34**	14.98

The best results are shown in **bold**. The last column shows time costs with 10 labeled data per class.

TABLE 7
The comparison of different approaches on Extended USPS.

Num of labels per class	20	40	60	80	100	Time cost (in second)
1NN	45.94±1.38	56.58±0.52	62.03±0.58	66.40±0.56	69.25±0.51	3.65
LSVM	22.57±1.40	24.73±1.01	25.55±0.52	26.94±0.59	27.52±1.19	4.75
AGR-5,000	73.13±1.37	78.98±0.82	81.48±0.83	83.61±0.45	84.63±0.32	9.49
EAGR-5,000	76.19±1.38	82.29±0.59	83.63±0.69	85.90±0.48	86.58±0.40	8.65
FLAG-5,000	80.40±1.83	84.84±0.23	86.12±0.63	87.01±0.48	87.70±0.24	7.35
AGR-20,000	75.69±1.01	81.57±1.42	83.92±1.61	86.35±0.75	87.20±0.48	159.90
EAGR-20,000	77.61±1.27	84.09±0.95	86.11±0.67	88.18±0.44	88.89±0.27	158.48
FLAG-20,000	80.55±1.55	86.39±0.84	88.22±0.80	89.87±0.65	90.71±0.44	21.87
HAGR-200,000-20,000	79.92±1.08	85.87±0.96	87.85±0.59	89.78±0.66	90.65±0.71	167.68
FLAG-200,000-20,000	**84.71±1.50**	**89.52±0.41**	**90.68±0.40**	**91.53±0.33**	**92.07±0.48**	28.91

The best results are shown in **bold**. The last column shows time costs with 100 labeled data per class.

2332-7790 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2757522, IEEE Transactions on Big Data

TABLE 8
The comparison of different approaches on MNIST8M.

Num of labeled per class	2	4	6	8	10	Time cost (in second)
1NN	41.80±3.44	51.89±3.58	56.15±2.41	59.57±2.14	61.87±1.28	35.13
LSVM	45.08±4.19	54.69±2.45	56.23±2.52	59.50±1.45	61.66±1.19	39.82
AGR-5,000	76.53±4.15	81.99±2.98	83.22±1.66	83.94±1.94	85.02±1.48	81.26
EAGR-5,000	79.46±4.02	84.84±4.02	86.63±2.39	87.24±2.87	88.48±1.59	80.09
FLAG-5,000	81.25±4.40	86.92±1.36	89.14±1.63	89.77±1.25	90.72±0.90	76.25
AGR-30,000	79.50±3.57	86.83±3.14	89.26±2.39	89.92±1.96	90.54±0.97	663.75
EAGR-30,000	79.77±3.19	87.17±2.45	89.64±1.93	90.27±1.25	90.91±0.89	661.31
FLAG-30,000	81.55±4.11	87.51±2.63	89.84±1.44	90.46±1.47	91.85±0.89	102.41
HAGR-300,000-30,000	83.24±3.89	88.29±1.86	89.87±1.79	90.73±1.45	91.96±1.09	705.68
HAGR-300,000-30,000-5,000	82.27±3.74	87.50±2.07	89.39±2.27	90.32±1.78	91.49±1.16	137.15
FLAG-300,000-30,000	**85.84±4.21**	**91.12±1.92**	**92.24±1.19**	**92.77±0.69**	**93.40±0.20**	134.58

The best results are shown in **bold**. The last column shows time costs with 10 labeled data per class.

accuracies than those with N_1=5,000, which shows the importance of employing a large anchor set. **Third**, by adding a larger anchor layer, HAGR-200,000-20,000 achieves higher classification accuracies than AGR-20,000. However, its performances are only comparable or even worse than those of FLAG-5,000 on Extended MNIST and FLAG-20,000 on Extended USPS. **Forth**, based on the anchor hierarchy, FLAG-200,000-20,000 improves its adjacency relationships and consistently outperforms all other methods with the efficient implementation.

5.1.3 *Large-Size Dataset*

We further test the scalability of FLAG on a large-size dataset, namely MNIST8M. In particular, we build two anchor graphs with N_1=5,000 and N_1=30,000. Besides, we construct two hierarchical anchor graphs with N_1=300,000, N_2=30,000 and N_1=300,000, N_2=30,000, N_3=5,000. By repeating the similar evaluation process, we report the average classification accuracies over 10 trials in Table 8.

Similar to the results of the above experiments, we can see that, based on the same anchor configuration, FLAG consistently outperforms other fast learning approaches with different numbers of labeled samples in terms of both the efficiency and effectiveness. We also observe that, by adding a small anchor layer, the time cost of HAGR-300,000-30,000-5,000 is much smaller than that of HAGR-300,000-30,000. However, the former sacrifices the accuracy at the same time, and even results in slightly worse performances than FLAG-30,000. In contrast, FLAG-300,000-30,000 obtains much higher accuracies with less time costs.

It is worthwhile to note that in all the experiments above, the accuracy of FLAG is much higher than that of HAGR with the same graph structure, especially when the number of the labeled data is small. The main reason is that, the HAGR classifier employs the coarsest-anchor-based coding as its input feature and requires the size of this anchor set N_h to be large for well capturing the data distribution. In contrast, our FLAG classifier employs the spectral representation with the dimensionality d as its input feature, which can keep its hypothesis space in a lower level while enlarging the scale of the coarsest anchor set. As both of them actually assign labels based on linear decision surfaces with the corresponding feature, their VC dimensions can be easily obtained [28], namely $N_h + 1$ for HAGR and $d + 1$ for FLAG. According to the computational learning theory [5], to approximately learn a target function in a hypothesis space, the number of required labeled datapoints is linear with its VC dimension. Therefore, compared with HAGR built upon the same hierarchical anchor graph, FLAG requires less labeled data to well train the model itself. As our work aims at the setting of semi-supervised learning where only a few data are labeled, FLAG is a more powerful approach to handle the large scale classification in terms of both the effectiveness and efficiency.

5.2 On the Improvements of FLAG

So far, we can see that FLAG improves both the efficiency and effectiveness of anchor-graph-based learning by optimizing an anchor label predictor on the spectral representation. In this section, we further investigate how these improvements are obtained based on the following aspects: (1). the construction of the intra-layer adjacency matrix over the coarsest anchors, and (2). the regularizer on the label predictor.

For simplicity, we follow the settings in [37] and use the dataset MNIST [20] for binary classification under two settings: (1). the first 5 versus the last 5 digits in MNIST1, and (2). the odd digits versus the even digits in MNIST2.

5.2.1 *On the Intra-layer Adjacency Matrix Construction*

In order to verify the first aspect, we define two intermediate methods for a better comparison:

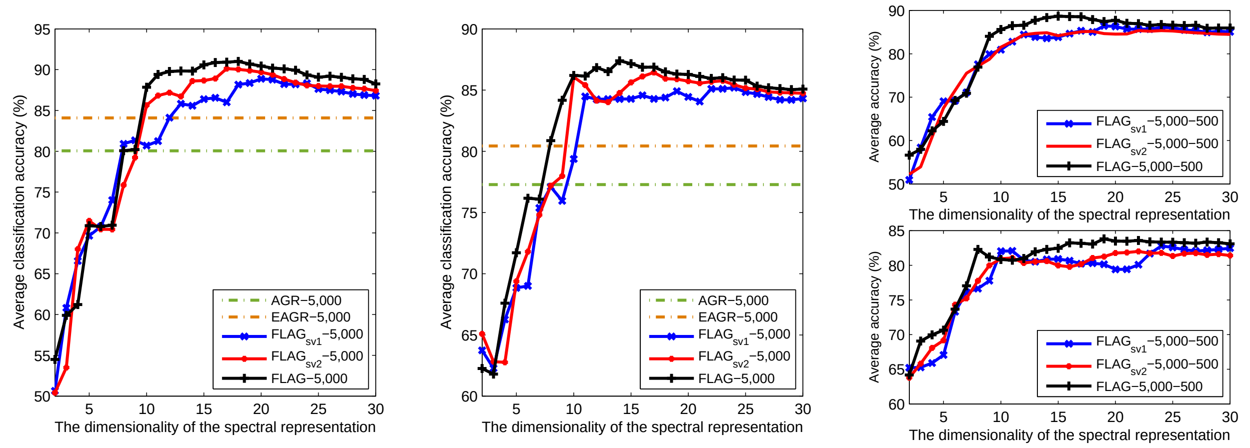
FLAG$_{\mathrm{sv1}}$: Compared with FLAG, this simplified version does not implement the pruning operation on the accumulated intra-layer matrix over the coarsest anchors. In other words, it directly obtains their spectral representations by performing spectral embedding with the original accumulated intra-layer matrix.

FLAG$_{\mathrm{sv2}}$: Compared with FLAG, this simplified version first constructs an intra-layer adjacency matrix over the coarsest anchors with the RBF kernel and keeps the top k values for each anchor. Then, it estimates the spectral representations of these anchors by performing spectral embedding with the adjacency matrix.

We fix μ=0 and optimize λ for a fair comparison. For all the compared anchor-graph-based methods, we first

2332-7790 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2757522, IEEE Transactions on Big Data



(a) MNIST1 (b) MNIST2 (c) MNIST1(top) and MNIST2(bottom)

Fig. 3. The accuracy variation with respect to the dimensionality of the spectral representation. Note the approaches in (a) and (b) are built upon a hierarchical anchor graph ($h = 1$) with 5,000 anchors, and those in (c) are built upon a hierarchical anchor graph ($h = 2$) with 5,000 and 500 anchors.

TABLE 9
The comparison of time costs (in seconds) of different approaches (without the time cost of graph construction). Note that the cost of calculating an intra-layer adjacency matrix over anchors, i.e., $\mathbf{Z}^{\mathrm{T}}\mathbf{Z}$, is nearly 0.05s.

Procedure	step.1	step.2	step.3	step.4	
N_1=5,000	spectral representation estimation	reduced Laplacian calculation	matrix inversion	label inference	total time cost
AGR-5,000	-	0.30	3.06	0.01	3.37
EAGR-5,000	-	0.08	3.06	0.01	3.15
FLAG$_{\text{sv1}}$-5,000 (d=18)	1.37	0.07	0.01	0.01	1.46
FLAG$_{\text{sv2}}$-5,000 (d=18)	0.20	0.07	0.01	0.01	0.29
FLAG-5,000 (d=18)	0.18	0.07	0.01	0.01	0.27

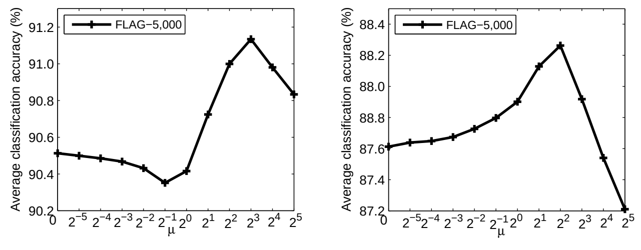
employ the same anchor graph model with 5,000 anchors. We randomly sample 20 datapoints as the labeled part and keep the rest unlabeled for the setting of SSL. The average accuracy curves of AGR-5,000, EAGR-5,000, and FLAG$_{\text{sv1}}$-5,000, FLAG$_{\text{sv2}}$-5,000, FLAG-5,000 with the varying dimensionality of the spectral representation are shown in Fig.3, and the time costs of different approaches (without the time cost of graph construction) are listed in Table 9.

From the Fig.3, we can obtain the following observations. **First**, by introducing a linear label predictor on the spectral representations of anchors, FLAG-5,000 and FLAG$_{\text{sv1}}$-5,000, FLAG$_{\text{sv2}}$-5,000 can obtain better performances than AGR-5,000. **Second**, benefitting from the strict sparse constraint on the intra-layer adjacency matrix over anchors, FLAG-5,000 obtains higher classification accuracies than FLAG$_{\text{sv1}}$-5,000. This result shows that the pruning operation can remove most of suspicious edges and improve the effectiveness of the intra-layer adjacency relationships, which is also consistent with the observation that sparse graphs perform better than dense graphs [52]. **Third**, although FLAG$_{\text{sv2}}$-5,000 employs a sparse intra-layer matrix over the coarsest anchors for estimating their spectral representations as well, its best accuracy is still worse than that of FLAG-5,000. The main reason is that, when two anchors are close to a classification boundary but locate at different classes, the pruning operation in FLAG can remove this kind of intra-layer adjacency edges as we mentioned in Section 4.2, which accordingly improves effectiveness. In contrast, the intra-layer adjacency relationships built upon the RBF kernel in FLAG$_{\text{sv2}}$ completely depend on the Euclidean distance, and the above noisy adjacency edges are hard to be discovered and filtered from the intra-layer adjacency matrix.

Taking into account the time cost in Table 9, the following observations can be obtained. **First**, the time cost of EAGR-5,000 is slightly smaller than that of AGR-5,000 (HAGR with $h = 1$). It is understandable that when calculating the reduced Laplacian matrix, AGR involves multiple times sparse matrix multiplication and EAGR only involves the operation one time [42]. **Second**, FLAG-5,000 and FLAG$_{\text{sv2}}$-5,000 are faster than FLAG$_{\text{sv1}}$-5,000, as the time complexity of the spectral embedding with a sparse matrix is much smaller than the one with a dense matrix. **Third**, all three FLAG-based versions carry out the inversion of a smaller-size matrix, and are therefore more efficient than the traditional fast learning approaches, especially AGR (HAGR). It is worthwhile to note that when the number of the coarsest anchors increases for better capturing the data distribution, the improvement of the efficiency will be more significant.

Moreover, for three FLAG-based versions, namely FLAG, FLAG$_{\text{sv1}}$, and FLAG$_{\text{sv2}}$, we additionally compare their performances based on a hierarchical anchor graph, in order to further demonstrate the effectiveness of our intra-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2757522, IEEE Transactions on Big Data



(a) MNIST1 (b) MNIST2

Fig. 4. The accuracy variation with respect to the parameter μ.

layer adjacency matrix construction. For this purpose, we repeatedly construct two-anchor-layer graphs for 10 times with the sizes of 5,000 and 500, respectively. By 10 times randomly sampling for building labeled data, the average accuracies of three approaches built upon these graphs are shown in Fig.3(c). As we can see, the best accuracies of FLAG-5,000-500 are still higher than those of FLAG$_{\mathrm{sv1}}$-5,000-500 and FLAG$_{\mathrm{sv2}}$-5,000-500 under two settings.

5.2.2 On the Predictor Regularizer

Finally we test the sensitivity of the weighting parameter μ on the predictor regularizer. We set other parameters to the optimized values according to the experiments above. Then, we vary μ and the average classification accuracies over 10 trials are shown in Fig.4. As we can see, compared with the accuracy where μ=0, this proposed regularizer can further improve the performance of the anchor label predictor and the classification accuracy stays at a high level over a wide range of the parameter variation.

6 Conclusion and future work

This work introduces a novel approach called Faster Learning on Anchor Graph (FLAG), which further scales up anchor-graph-based models and meanwhile improves their effectiveness. In FLAG, the labels of the coarsest anchors are obtained by learning a linear predictor on their low-dimensional spectral representations, which can be efficiently estimated based on a proposed sparse intra-layer adjacency matrix over these anchors. To optimize the anchor label predictor, we also develop a novel regularization framework based on a hierarchical anchor graph. In this way, the optimization can be computed with a faster matrix inversion procedure, where the matrix size is only equivalent to the dimensionality of the spectral representation. Furthermore, owing to the flexible structure of hierarchical anchor graph models, FLAG can be scaled to different scales of datasets, including large-scale ones. The experiments on publicly available datasets of various sizes have demonstrated this superiority over the conventional fast learning models.

Finally, we discuss the possible research direction in the future. In this work, we only employ a linear label predictor on the spectral representation to keep computational efficiency. Differently, we may use more complex label prediction models, such as deep neural networks. In this way, we can build a hierarchical anchor graph model upon a deep neural network, which leads to a semi-supervised deep neural network training approach for large-scale data.

Acknowledgments

The authors sincerely appreciate the useful comments and suggestions from the anonymous reviewers. This work was partially supported by the National 973 Program of China under grant 2014CB347600, and the National Nature Science Foundation of China under grant 61432019, and 61772171.

References

[1] R. K. Ando and T. Zhang, "Learning on graph with laplacian regularization," in *Proceedings of the Conference on Advances in Neural Information Processing Systems (NIPS)*, 2007, pp. 25–32.

[2] R. Bekkerman, M. Bilenko, and J. Langford, *Scaling up machine learning: Parallel and distributed approaches*. Cambridge University Press, 2011.

[3] M. C. F. Biological and C. Learning, "Cbcl face database #1," http://www.ai.mit.edu/projects/cbcl.

[4] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proceedings of the conference on Computational Learning Theory*. ACM, 1998, pp. 92–100.

[5] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth, "Learnability and the vapnik-chervonenkis dimension," *Journal of the ACM*, vol. 36, no. 4, pp. 929–965, 1989.

[6] V. Castelli and T. M. Cover, "On the exponential value of labeled samples," *Pattern Recognition Letters*, vol. 16, no. 1, pp. 105–111, 1995.

[7] O. Chapelle, V. Sindhwani, and S. S. Keerthi, "Optimization techniques for semi-supervised support vector machines," *Journal of Machine Learning Research*, vol. 9, no. Feb, pp. 203–233, 2008.

[8] J. Chen, H.-r. Fang, and Y. Saad, "Fast approximate knn graph construction for high dimensional data via recursive lanczos bisection," *Journal of Machine Learning Research*, vol. 10, no. Sep, pp. 1989–2012, 2009.

[9] L. Chen, I. W. Tsang, and D. Xu, "Laplacian embedded regression for scalable manifold regularization," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 6, pp. 902–915, 2012.

[10] W. Dong, C. Moses, and K. Li, "Efficient k-nearest neighbor graph construction for generic similarity measures," in *Proceedings of the International Conference on World Wide Web (WWW)*. ACM, 2011, pp. 577–586.

[11] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *The Journal of Machine Learning Research*, vol. 9, no. 9, pp. 1871–1874, 2008.

[12] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer series in statistics Springer, 2001, vol. 1.

[13] A. Goyal, H. Daumé III, and R. Guerra, "Fast large-scale approximate graph construction for nlp," in *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, 2012, pp. 1069–1080.

[14] M. Hein and S. Setzer, "Beyond spectral clustering-tight relaxations of balanced graph cuts," in *Proceedings of the Conference on Advances in Neural Information Processing Systems (NIPS)*, 2011, pp. 2366–2374.

[15] C.-J. Hsieh, S. Si, and I. S. Dhillon, "A divide-and-conquer solver for kernel support vector machines." in *Proceedings of the International Conference on Machine learning (ICML)*, 2014, pp. 566–574.

[16] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, "Knowledge graph embedding via dynamic mapping matrix." in *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2015, pp. 687–696.

[17] L. Jiang, P. Luo, J. Wang, Y. Xiong, B. Lin, M. Wang, and N. An, "Grias: an entity-relation graph based framework for discovering entity aliases," in *Proceedings of the IEEE International Conference on Data Mining (ICDM)*. IEEE, 2013, pp. 310–319.

[18] T. Joachims, "Transductive inference for text classification using support vector machines," in *Proceedings of the International Conference on Machine learning (ICML)*, vol. 99, 1999, pp. 200–209.

[19] M. Karlen, J. Weston, A. Erkan, and R. Collobert, "Large scale manifold transduction," in *Proceedings of the International Conference on Machine learning (ICML)*. ACM, 2008, pp. 448–455.

[20] Y. Lecun, "Mnist," http://yann.lecun.com/exdb/mnist/.

[21] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*. Society for Industrial and Applied Mathematics (SIAM), 1998.

2332-7790 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2757522, IEEE Transactions on Big Data

[22] M. Li, J. T.-Y. Kwok, and B. Lü, "Making large-scale nyström approximation possible," in *Proceedings of the International Conference on Machine learning (ICML)*, 2010, pp. 631–638.

[23] W. Liu, J. He, and S.-F. Chang, "Large graph construction for scalable semi-supervised learning," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2010, pp. 679–686.

[24] W. Liu, J. Wang, and S.-F. Chang, "Robust and scalable graph-based semisupervised learning," *Proceedings of the IEEE*, vol. 100, no. 9, pp. 2624–2638, 2012.

[25] G. Loosli, S. Canu, and L. Bottou, "Training invariant support vector machines using selective sampling," *Large Scale Kernel Machines*, pp. 301–320, 2007.

[26] Y. Low, J. E. Gonzalez, A. Kyrola, D. Bickson, C. E. Guestrin, and J. Hellerstein, "Graphlab: A new framework for parallel machine learning," *arXiv preprint arXiv:1408.2041*, 2014.

[27] S. Melacci and M. Belkin, "Laplacian Support Vector Machines Trained in the Primal," *Journal of Machine Learning Research*, vol. 12, pp. 1149–1184, 2011.

[28] T. M. Mitchell, *Machine learning*. McGraw-Hill Science, 1997.

[29] T. Mu, J. Y. Goulermas, J. Tsujii, and S. Ananiadou, "Proximity-based frameworks for generating embeddings from multi-output data," *IEEE transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2216–2232, 2012.

[30] M. Muja and D. G. Lowe, "Scalable nearest neighbor algorithms for high dimensional data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 11, pp. 2227–2240, 2014.

[31] A. Y. Ng, M. I. Jordan, Y. Weiss *et al.*, "On spectral clustering: Analysis and an algorithm," in *Proceedings of the Conference on Advances in Neural Information Processing Systems (NIPS)*, vol. 14, no. 2, 2001, pp. 849–856.

[32] J. Pujara, H. Miao, L. Getoor, and W. Cohen, "Knowledge graph identification," in *Proceedings of the International Semantic Web Conference (ISWC)*, 2013, pp. 542–557.

[33] D. Rao and D. Yarowsky, "Ranking and semi-supervised classification on large scale graphs using map-reduce," in *Proceedings of the Workshop on Graph-based Methods for Natural Language Processing*. Association for Computational Linguistics, 2009, pp. 58–65.

[34] S. Roweis, "Newsgroup," http://www.cs.nyu.edu/~roweis/data.html.

[35] W. Shen, J. Wang, P. Luo, and M. Wang, "A graph-based approach for ontology population with named entities," in *Proceedings of the ACM International Conference on Information and knowledge management*. ACM, 2012, pp. 345–354.

[36] A. Singhal, "Introducing the knowledge graph: things, not strings," *Official google blog*, 2012.

[37] I. W. Tsang and J. T. Kwok, "Large-scale sparsified manifold regularization," in *Proceedings of the Conference on Advances in Neural Information Processing Systems (NIPS)*, 2006, pp. 1401–1408.

[38] I. W. Tsang, J. T. Kwok, and P.-M. Cheung, "Core vector machines: Fast svm training on very large data sets," *Journal of Machine Learning Research*, vol. 6, no. 1, pp. 363–392, 2005.

[39] J. Wang, J. Wang, G. Zeng, Z. Tu, R. Gan, and S. Li, "Scalable k-nn graph construction for visual descriptors," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2012, pp. 1106–1113.

[40] J. Wang and Y. Xia, "Fast graph construction using auction algorithm," in *Proceedings of the Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2012, pp. 873–882.

[41] M. Wang, W. Fu, S. Hao, H. Liu, and X. Wu, "Learning on big graph: Label inference and regularization with anchor hierarchy," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 5, pp. 1101–1114, 2017.

[42] M. Wang, W. Fu, S. Hao, D. Tao, and X. Wu, "Scalable semi-supervised learning by efficient anchor graph regularization," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 7, pp. 1864–1877, 2016.

[43] M. Wang, X.-S. Hua, R. Hong, J. Tang, G.-J. Qi, and Y. Song, "Unified video annotation via multigraph learning," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 5, pp. 733–746, 2009.

[44] R. Yuster and U. Zwick, "Fast sparse matrix multiplication," *ACM Transactions on Algorithms*, vol. 1, no. 1, pp. 2–13, 2005.

[45] Z.-J. Zha, T. Mei, J. Wang, Z. Wang, and X.-S. Hua, "Graph-based semi-supervised learning with multiple labels," *Journal of Visual Communication and Image Representation*, vol. 20, no. 2, pp. 97–103, 2009.

[46] K. Zhang, J. T. Kwok, and B. Parvin, "Prototype vector machine for large scale semi-supervised learning," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2009, pp. 1233–1240.

[47] K. Zhang, L. Liang, J. T. Kwok, and S. Vucetic, "Scaling up graph-based semi-supervised learning via prototype vector machines," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 3, pp. 444–457, 2015.

[48] M. Zhang, J. Tang, X. Zhang, and X. Xue, "Addressing cold start in recommender systems: A semi-supervised co-training algorithm," in *Proceedings of the international ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 2014, pp. 73–82.

[49] Y.-M. Zhang, K. Huang, G. Geng, and C.-L. Liu, "Fast knn graph construction with locality sensitive hashing," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2013, pp. 660–674.

[50] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Proceedings of the Conference on Advances in Neural Information Processing Systems (NIPS)*, 2004, pp. 321–328.

[51] D. Zhou, J. Huang, and B. Schölkopf, "Learning with hypergraphs: Clustering, classification, and embedding," in *Proceedings of the Conference on Advances in Neural Information Processing Systems (NIPS)*, 2006, pp. 1601–1608.

[52] X. Zhu, "Semi-supervised learning literature survey," *Technical report, University of Wisconsin Madison*, 2005.

[53] X. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2003, pp. 912–919.

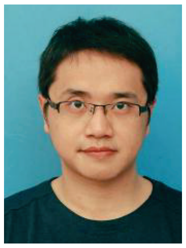
[54] X. Zhu and A. B. Goldberg, "Introduction to semi-supervised learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 3, no. 1, pp. 1–130, 2009.

Weijie Fu is currently working toward the PhD degree in the School of Computer and Information, Hefei University of Technology(HFUT). His current research interest focuses on machine learning and data mining.

Meng Wang is a professor at the Hefei University of Technology, China. He received his B.E. degree and Ph.D. degree in the Special Class for the Gifted Young and the Department of Electronic Engineering and Information Science from the University of Science and Technology of China (USTC), Hefei, China, in 2003 and 2008, respectively. His current research interests include multimedia content analysis, computer vision, and pattern recognition. He has authored more than 200 book chapters, journal and conference papers in these areas. He is the recipient of the ACM SIGMM Rising Star Award 2014. He is an associate editor of IEEE Transactions on Knowledge and Data Engineering (IEEE TKDE), IEEE Transactions on Circuits and Systems for Video Technology (IEEE TCSVT), and IEEE Transactions on Neural Networks and Learning Systems (IEEE TNNLS).

2332-7790 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2757522, IEEE Transactions on Big Data



Shijie Hao is an associate professor at the Hefei University of Technology(HFUT), China. He received his B.E., M.S. and Ph.D. Degree in the School of Computer and Information from HFUT. His current research interests include machine learning and image processing.

Tingting Mu received the B.Eng. degree in electronic engineering and information science from the University of Science and Technology of China, Hefei, China, in 2004, and the Ph.D. degree in electrical engineering and electronics from the University of Liverpool, Liverpool, U.K., in 2008. She is currently a Lecturer with the School of Computer Science, University of Manchester, Manchester, U.K. Her current research interests include machine learning, data visualization, and mathematical modeling, with applications to information retrieval, text mining, and bioinformatics.

2332-7790 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

Learning on Big Graph: Label Inference and Regularization with Anchor Hierarchy

Meng Wang, *Member, IEEE*, Weijie Fu, Shijie Hao, Hengchang Liu, and Xindong Wu, *Fellow, IEEE*

Abstract—Several models have been proposed to cope with the rapidly increasing size of data, such as Anchor Graph Regularization (AGR). The AGR approach significantly accelerates graph-based learning by exploring a set of anchors. However, when a dataset becomes much larger, AGR still faces a big graph which brings dramatically increasing computational costs. To overcome this issue, we propose a novel Hierarchical Anchor Graph Regularization (HAGR) approach by exploring multiple-layer anchors with a pyramid-style structure. In HAGR, the labels of datapoints are inferred from the coarsest anchors layer by layer in a coarse-to-fine manner. The label smoothness regularization is performed on all datapoints, and we demonstrate that the optimization process only involves a small-size reduced Laplacian matrix. We also introduce a fast approach to construct our hierarchical anchor graph based on an approximate nearest neighbor search technique. Experiments on million-scale datasets demonstrate the effectiveness and efficiency of the proposed HAGR approach over existing methods. Results show that the HAGR approach is even able to achieve a good performance within 3 minutes in an 8-million-example classification task.

Index Terms—Semi-supervised learning, graph-based learning, label smoothness regularization, label inference

1 Introduction

Semi-supervised learning (SSL) methods [51], which exploit the prior knowledge from unlabeled data to improve classification performance, have been widely used to handle datasets where only a portion of data are labeled. Most of these methods are developed based on the cluster assumption [47] or the manifold assumption [1]. The former supposes that nearby points are likely to have the same label, while the latter assumes that each class lies on a separate low-dimensional manifold embedded in a higher dimensional space. In recent years, various semi-supervised learning methods have been developed under these assumptions, including mixture methods [4], co-training [2], semi-supervised support vector machines [18], and graph-based methods [50].

In this paper, we focus on the family of graph-based semi-supervised learning (GSSL) methods, where the label dependencies among datapoints are captured by a weighted graph. These methods first construct adjacency relationships between all datapoints and then propagate labels from labeled data to unlabeled data with the above adjacency edges. Since many forms of real-world data, such as handwritten digits, faces, medical data, and speech data, exhibit such a kind of intrinsic graph structure, GSSL has been applied to many applications, and achieves satisfying performance [10], [41], [42]. Meanwhile, this roadmap can be extended to building other advanced graph models, such as hypergraph [17], [48] and multi-graph [7], [37], to describe more complex relationships among real-world entities like multimodal media contents [12], [13], [27].

In spite of the progress made in recent years, most GSSL methods remain challenging mainly due to their cubic complexity in optimization. Facing the ever increasing data size, these approaches tend to be inefficient in dealing with large-scale datasets. To address this issue, recent works seek to employ anchors in scaling up graph-based learning models, such as Anchor Graph Regularization (AGR) [23], and Efficient Anchor Graph Regularization (EAGR) [36] (for simplicity, we call both of them AGR without differentiation except in comparative experiments). In these models, anchors refer to the points that roughly cover the data distribution. AGR then builds an anchor graph to model the inter-layer adjacency between the data layer and the anchor layer. For clarity, a few inter-layer edges in the anchor graph built on a two-moon dataset are shown in Fig. 1a. The efficiency of these approaches lies in two steps: 1) they build the intra-layer adjacency relationships based on the anchors, instead of computing all pair-wise adjacencies between the datapoints in an exhaustive way; and 2) they infer the labels of datapoints from the anchors based on their inter-layer adjacency relationships. As the number of anchors can be much smaller than datapoints, both the graph construction and the learning process become much faster than those in traditional graph-based approaches. However, to obtain a reasonable accuracy, anchors need to be sufficiently dense in order to build effective adjacency relationships.

- *M. Wang, W. Fu, and S. Hao are with the School of Computer Science and Information Engineering, Hefei University of Technology, Hefei 230009, China. E-mail: {eric.mengwang, fwj.edu, hfut.hsj}@gmail.com.*
- *H. Liu is with the Department of Computer Science, University of Science and Technology of China, Hefei, Shi 230022, China. E-mail: hcliu@ustc.edu.cn.*
- *X. Wu is with the School of Computer Science and Information Engineering, Hefei University of Technology, Hefei 230009, China, and the School of Computing and Informatics, University of Louisiana at Lafayette, Lafayette, LA 70504. E-mail: xwu@louisiana.edu.*

*Manuscript received 4 Sept. 2016; revised 8 Nov. 2016; accepted 7 Jan. 2017. Date of publication 17 Jan. 2017; date of current version 30 Mar. 2017. Recommended for acceptance by H.T. Shen.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2017.2654445*

1041-4347 © 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

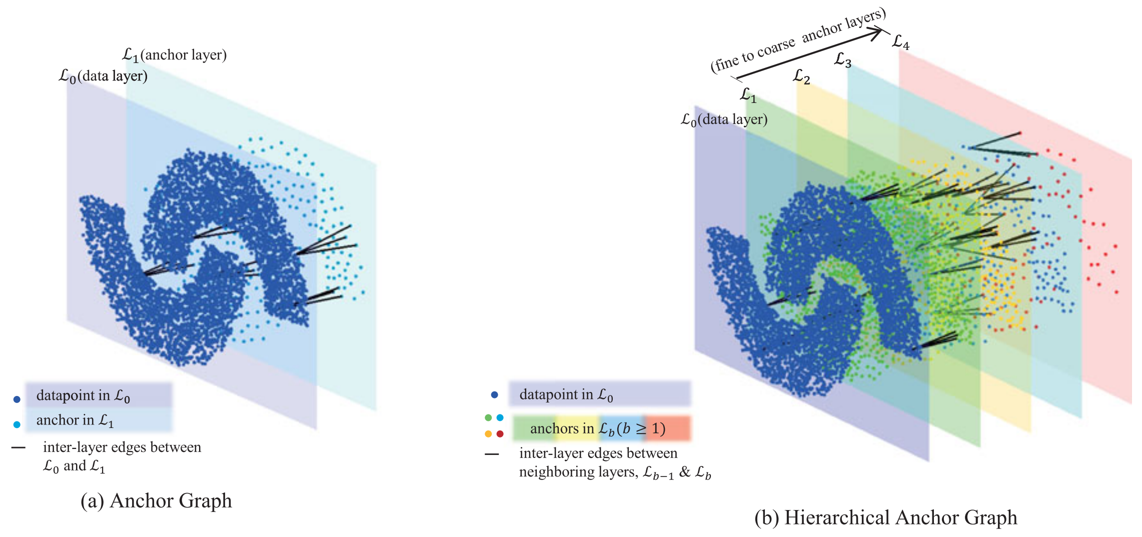

$\mathcal{L}_1$(anchor layer)
$\mathcal{L}_0$(data layer)
datapoint in $\mathcal{L}_0$
anchor in $\mathcal{L}_1$
inter-layer edges between $\mathcal{L}_0$ and $\mathcal{L}_1$
(a) Anchor Graph
(fine to coarse anchor layers)
$\mathcal{L}_4$
$\mathcal{L}_3$
$\mathcal{L}_2$
$\mathcal{L}_1$
$\mathcal{L}_0$(data layer)
datapoint in $\mathcal{L}_0$
anchors in $\mathcal{L}_b (b \geq 1)$
inter-layer edges between neighboring layers, $\mathcal{L}_{b-1}$ & $\mathcal{L}_b$
(b) Hierarchical Anchor Graph

Fig. 1. An illustrative example of anchor graph (consisting of 5,000 datapoints and an anchor layer with 250 anchors) and hierarchical anchor graph (consisting of 5,000 datapoints, and multiple anchor layers with 1,000, 500, 250, and 100 anchors, respectively). For simplify, only a tiny fraction of inter-layer edges are shown.

Therefore, when dealing with extremely large-scale datasets, the computational costs of existing anchor-graph-based approaches will dramatically increase and become even practically intractable. One possible way is to use only a small number of anchors, but too sparse anchors will degrade performance as the label inference and the label smoothness regularization cannot be performed reliably.

To address this issue, in this paper, we introduce a novel hierarchical anchor graph and develop a scalable SSL approach named Hierarchical Anchor Graph Regularization (HAGR). Different from the existing graphs, our proposed graph contains multiple layers of anchors in a pyramid-style structure. It consists of a layer of original datapoints and multiple layers of anchors that describe the original datapoints from fine to coarse, as illustrated in Fig. 1b. Based on the proposed graph model, we infer the labels of datapoints from the coarsest anchors layer by layer based on inter-layer adjacency relationships. Although the label smoothness regularization is performed on all datapoints, we demonstrate that the optimization only involves a reduced Laplacian matrix with the size of the coarsest anchor layer. Therefore, the HAGR approach overcomes the limitation of anchor-graph-based approaches and well compromises classification performance and computational efficiency. The HAGR approach is quite flexible, as we can set different layers of anchors according to the scales of classification tasks. We show that it will degrade to AGR when there is only one anchor layer. In order to further improve the efficiency of the construction of this hierarchical anchor graph, we also investigate an Approximate Nearest Neighbor Search (ANNS) technique to build inter-layer adjacency relationships fastly.

The main contributions of our work are as follows.

1) We make a deep analysis on the existing anchor graph and point out its limitations in dealing with large-scale datasets. That is, AGR faces either an intractable computational cost with dense anchors or a degraded performance with sparse anchors.
2) We propose to build hierarchical anchor graph with a pyramid structure, and develop a scalable classifier based on it. The labels of datapoints are inferred from the coarsest anchors layer by layer and the optimization only involves a small-size reduced Laplacian matrix. The proposed approach overcomes the limitations of AGR and is able to efficiently accomplish large-scale classification with a good performance (detailed computational costs will be shown in Section 4.3).
3) We introduce a fast hierarchical anchor graph construction process, in which the ANNS technique is employed to build inter-layer adjacency relationships.

The rest of this paper is organized as follows. In Section 2, we briefly introduce related work on the graph-based learning. In Section 3, we analyze the traditional AGR approach and its limitations. The proposed approach is described in Section 4. In Section 5, we validate our method and make comparisons with other approaches on large-scale datasets. We also evaluate different graph structures of HAGR to test its flexibility as well as robustness. We finally conclude this paper in Section 6.

2 Related Work

Zhu et al. [50] first introduced the formulation of learning problem based on a Gaussian random field, and analyzed its intimate connections with random walks and spectral graph theory. Zhou et al. [47] subsequently suggested an effective algorithm to obtain the solution of a classification function, which is sufficiently smooth with respect to the intrinsic structure collectively revealed by known labeled and unlabeled datapoints. Later, Zhu et al. [52] developed an improved nonparametric kernel approach by incorporating order constraints during the convex optimization in learning. Zelnik et al. [44] introduced a local scale in computing the affinity between each pair of datapoints for the weighted edge. Meanwhile, inspired by locally linear embedding [31], many works that focus on improving the weight estimation of the graph via sparse representation are proposed. For example, Wang et al. [34] presented a linear neighborhood model for label propagation, which assumes

TABLE 1
Notations and Definitions

Notation	Definition
$\mathcal{G} = \{\mathcal{X}, \mathcal{U}, \mathcal{E}\}$	An anchor graph or hierarchical anchor graph, where $\mathcal{X}$ and $\mathcal{U}$ indicate datapoints and anchors, respectively, and $\mathcal{E}$ indicates the sets of inter-layer adjacency edges between different sets of points.
h	The number of anchor layers.
$\mathcal{L}_b$	The bth layer in the pyramidal graph structure, where $\mathcal{L}_0$ is the layer of original data and $\mathcal{L}_b (b \geq 1)$ denotes the bth anchor layer.
m_b	The number of points in $\mathcal{L}_b$.
$\mathbf{Z}^{a,b}$	The inter-layer adjacency matrix between $\mathcal{L}_a$ and $\mathcal{L}_b$. By default, we have $b = a + 1$ for estimating the adjacencies between neighboring layers.
$Z_{is}^{a,b}$	The inter-layer adjacency weight between point i in $\mathcal{L}_a$ and point s in $\mathcal{L}_b$.
$\mathbf{W}$	The intra-layer adjacency matrix used in label smoothness regularization.
Λ^b	The diagonal matrix of the degrees of the anchors in $\mathcal{L}_b$.
$\mathbf{A}$	The soft label matrix of anchors.
$\mathbf{F}$	The soft label matrix of datapoints.
$\mathbf{Y}_{\mathrm{L}}$	The class indicator matrix on labeled datapoints.
$\mathbf{L}$	The reduced Laplacian matrix in the anchor graph or hierarchical anchor graph.
n	The number of datapoints.
c	The number of classes in the dataset.
l	The number of labeled datapoints in the dataset.
$\mathbf{Z}^{\mathrm{H}}$	The accumulated inter-layer adjacency matrix in the hierarchical anchor graph.

that each datapoint can be linearly reconstructed from its neighborhoods with l_2 minimization. Similarly, Cheng et al. [6] proposed a weight estimation method which optimizes the sparse reconstruction coefficients on a l_1 graph. Since selecting local neighbors may lead to disjoint components and incorrect neighbors in graph, Tian et al. [32] advocated learning a nonnegative low-rank graph to capture global linear neighborhoods, under the assumption that each datapoint can be linearly reconstructed from weighted combinations of its direct neighbors and reachable indirect neighbors.

The above graph-based approaches show impressive performances in various applications. However, they are not sufficiently scalable, which imposes limitations in handling larger datasets. With the rapid increase in data size, researchers have paid more attention to designing novel approaches to reduce the computational cost of graph-based learning. Tsang et al. [33] formulated a sparsified manifold regularizer as a center-constrained minimum enclosing ball problem to produce sparse solutions with lower time and space complexities. Wang et al. [35] proposed a multiple random divide-and-conquer approach to construct an approximated neighborhood graph and presented a neighborhood propagation scheme to further enhance the accuracy. Chen et al. [5] presented a method to combine both the original kernel and the graph kernel for scalable manifold regularization.

More recent works seek to employ anchors in scaling up the graph model. Different from the conventional graphs, the anchor-based approaches build the adjacency relationships between original datapoints based on anchors. Zhang et al. [45], [46] first suggested using a set of anchors to perform an effective low-rank approximation of the data manifold, and to span a model suffering the minimum information loss. Liu et al. [23] first presented the anchor graph model, and introduced it into the graph-based learning tasks. Wang et al. [36] subsequently proposed an improved algorithm, which shows better performance and computational efficiency. Compared with the conventional graphs, these anchor-graph-based approaches can largely reduce the complexity in graph construction, and have been widely used in many applications [3], [20], [25], [38]. However, the two-layer anchor graph structure is still limited for processing large-scale learning tasks, which will be analyzed in detail in the following.

3 Anchor-Graph-Based Learning

In this section, we first present a brief description of the anchor-graph-based approach and then give a detailed analysis on its limitations. For convenience, some important notations used throughout the paper and their explanations are listed in Table 1.

3.1 Formulations of AGR

We consider a standard multiclass SSL problem. Given a dataset $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \in \mathbf{R}^{d \times n}$ with the first l samples being labeled from c distinct classes, anchor-graph-based methods start with clustering a set of representative anchors $\mathcal{U} = \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{m_1}\} \in \mathbf{R}^{d \times m_1} (m_1 \leq n)$, which share the same feature space with original datapoints.

To be consistent with the notations in HAGR, here we let $\mathcal{L}_0$ denote the layer of datapoints, and $\mathcal{L}_1$ denote the layer of anchors, as illustrated in Fig. 1a. Different from the conventional graph denoted by an $n \times n$ adjacency matrix, an anchor graph $\mathcal{G}$ is represented by an $n \times m_1$ nonnegative matrix $\mathbf{Z}^{0,1}$, which models inter-layer adjacency relationships between points in $\mathcal{L}_0$ and $\mathcal{L}_1$. Specifically, the entries in each row of $\mathbf{Z}^{0,1}$ are the weights between datapoint $\mathbf{x}_i$ and its k nearest anchors, which can be defined by Nadaraya-Natson kernel regression [14]

$$Z_{is}^{0,1} = \frac{K_\sigma(\mathbf{x}_i, \mathbf{u}_s)}{\sum_{s' \in \langle i \rangle} K_\sigma(\mathbf{x}_i, \mathbf{u}_{s'})} \forall s \in \langle i \rangle, \tag{1}$$

where the notation $\langle i \rangle \subseteq [1 : m_1]$ is the indices of the k closest anchors of $\mathbf{x}_i$.

Given the labels of anchors and the above inter-layer adjacency, the label of each datapoint can be estimated as a weighted average of them, i.e.,

$$f(\mathbf{x}_i) = \sum_{s=1}^{m_1} Z_{is}^{0,1} f(\mathbf{u}_s), \tag{2}$$

where f is a prediction function that assigns each point a soft label.

Then, to smooth these inferred labels, one can also construct an intra-layer adjacency matrix of datapoints based on inter-layer adjacency relationships

$$\mathbf{W} = \mathbf{Z}^{0,1} (\Lambda^1)^{-1} \mathbf{Z}^{0,1^{\mathrm{T}}} \in \mathbf{R}^{n \times n}, \tag{3}$$

where the diagonal matrix Λ^1 is defined as $\Lambda^1_{ss} = \sum_{i=1}^n Z_{is}^{0,1}$. From Eq. (3), we can see that, $W_{ij} > 0$ means the two datapoints share at least one common anchor, and otherwise $W_{ij} = 0$. It is likely that datapoints sharing common anchors would have similar labels.

Let $\mathbf{Y}_{\mathrm{L}} = [\mathbf{y}_1^{\mathrm{T}}, \ldots, \mathbf{y}_l^{\mathrm{T}}]^{\mathrm{T}} \in \mathbf{R}^{l \times c}$ denote the class indicator matrix on labeled datapoints, where $y_{ir} = 1$ if $\mathbf{x}_i$ belongs to class r, and $y_{ir} = 0$ otherwise. Let $\mathbf{A} = [\mathbf{a}_1{}^{\mathrm{T}}, \mathbf{a}_2{}^{\mathrm{T}}, \ldots, \mathbf{a}_{m_1}{}^{\mathrm{T}}]^{\mathrm{T}} \in \mathbf{R}^{m_1 \times c}$ denote the soft label matrix of the anchors in $\mathcal{L}_1$. To deal with the standard multi-class SSL problem, Anchor Graph Regularization [23] is formulated by minimizing $\mathcal{Q}_{\mathbf{A}}$ as

$$\mathcal{Q}_{\mathbf{A}} = \sum_{i=1}^{l} \| \mathbf{Z}_{i\cdot}^{0,1}\mathbf{A} - \mathbf{y}_i \|^2 + \frac{\lambda}{2} \sum_{i,j=1}^{n} W_{ij} \| \mathbf{Z}_{i\cdot}^{0,1}\mathbf{A} - \mathbf{Z}_{j\cdot}^{0,1}\mathbf{A} \|^2, \tag{4}$$

where $\lambda > 0$ is the trade-off parameter balancing different terms, and $\mathbf{Z}_{i\cdot}^{0,1}$ is the ith row of $\mathbf{Z}^{0,1}$. From the above equation, we can see that, the labels of datapoints in both the fitting and smoothness terms are inferred from the anchors. Note that there are other alternative methods for manifold regularization, such as [1], [5]. As it is not the main point of this paper, we simply follow the idea of AGR.

Meanwhile, for the degree of each datapoint, we have $D_{ii} = \sum_j W_{ij} = \sum_{sj} Z_{is}^{0,1} (\Lambda_{ss}^1)^{-1} Z_{js}^{0,1} = \sum_s Z_{is}^{0,1} = 1.$ Therefore, we obtain the diagonal matrix $\mathbf{D} = \mathbf{I}$, and Eq. (4) is reformulated into a matrix form as

$$\begin{aligned} \mathcal{Q}_{\mathbf{A}} &= \|\mathbf{Z}_{\mathrm{L}}^{0,1}\mathbf{A} - \mathbf{Y}_{\mathrm{L}}\|_{\mathrm{F}}^2 + \lambda \mathrm{tr}(\mathbf{A}^{\mathrm{T}} \mathbf{Z}^{0,1^{\mathrm{T}}} (\mathbf{I} - \mathbf{W}) \mathbf{Z}^{0,1} \mathbf{A}) \\ &= \|\mathbf{Z}_{\mathrm{L}}^{0,1}\mathbf{A} - \mathbf{Y}_{\mathrm{L}}\|_{\mathrm{F}}^2 + \lambda \mathrm{tr}(\mathbf{A}^{\mathrm{T}} \widetilde{\mathbf{L}} \mathbf{A}), \end{aligned} \tag{5}$$

where $\mathbf{Z}_{\mathrm{L}}^{0,1}$ is the labeled part of $\mathbf{Z}^{0,1}$, and $\widetilde{\mathbf{L}} = \mathbf{Z}^{0,1^{\mathrm{T}}}\mathbf{Z}^{0,1} - (\mathbf{Z}^{0,1^{\mathrm{T}}}\mathbf{Z}^{0,1})(\Lambda^1)^{-1}(\mathbf{Z}^{0,1^{\mathrm{T}}}\mathbf{Z}^{0,1}) \in \mathbf{R}^{m_1 \times m_1}$ is the reduced Laplacian matrix in AGR.

Differentiating $\mathcal{Q}_{\mathbf{A}}$ with respect to $\mathbf{A}$ and setting it to zero, we can obtain an optimal solution in the closed-form

$$\mathbf{A} = (\mathbf{Z}_{\mathrm{L}}^{0,1^{\mathrm{T}}}\mathbf{Z}_{\mathrm{L}}^{0,1} + \lambda \widetilde{\mathbf{L}})^{-1} \mathbf{Z}_{\mathrm{L}}^{0,1^{\mathrm{T}}} \mathbf{Y}_{\mathrm{L}}. \tag{6}$$

Clearly, this matrix inversion takes a time cost of $O(m_1^3)$.

Finally, AGR employs the solved labels associated with the anchors in $\mathcal{L}_1$ to infer the hard label of any unlabeled datapoint in $\mathcal{L}_0$

$$\widehat{y}_i = \mathrm{argmax}_{r \in \{1,\ldots,c\}} \frac{\mathbf{Z}_{i\cdot}^{0,1} \times \mathbf{A}_{\cdot r}}{\beta_r}, i = l+1, \ldots, n, \tag{7}$$

where $\mathbf{A}_{\cdot r}$ is the rth column of $\mathbf{A}$, and $\beta_r = \mathbf{1}^{\mathrm{T}}\mathbf{Z}^{0,1}\mathbf{A}_{\cdot r}$ is the normalization factor, which balances skewed class distributions [50].

3.2 Limitation of AGR: A Dilemma

Compared with the traditional graph models, anchor graph additionally introduces an anchor set into the graph construction. As the number of these anchors can be much smaller than datapoints, both the graph construction and the optimization (especially the inverse computation in Eq. (6)) become much faster. AGR thus becomes a popular tool to handle relatively large datasets.

However, AGR still has limitation in dealing with extremely large-scale datasets. Specifically, it faces a dilemma between performance and computational cost. If we only employ a relatively small number of anchors, the performance of the AGR approaches will degrade as label smoothness regularization and label inference cannot be performed effectively. For the label smoothness regularization, it will introduce many noisy intra-layer edges between dissimilar datapoints by Eq. (3), as they tend to share a distant anchor. For the label inference, it will lead to unreliable integration of label information from k nearest anchors, as inter-layer adjacencies are estimated with too sparse anchors that can be far away from the datapoint. Therefore, to obtain a reasonable accuracy, anchors in the AGR approaches need to be sufficiently dense to build effective adjacency relationships. According to Eq. (6), it results in a dramatically increase of computational cost, which makes the approach practically intractable.

4 Hierarchical Anchor Graph Regularization

We first introduce the definition of hierarchical anchor graph and how we use this graph to build a scalable GSSL approach. Then, we present the efficient graph construction based on ANNS, followed with the analysis on time complexity and other discussions.

4.1 Label Inference and Regularization in HAGR

For graph-based SSL, to obtain good performances in large-scale classification tasks, it always requires an effective smoothness term for regularization, and an efficient solution for model optimization. To build such a scalable graph-based classifier, we extend the anchor graph to a pyramid-like structure and propose a novel graph model called hierarchical anchor graph. For clarity, an illustrative example of the hierarchical anchor graph is shown in Fig. 1b.

Definition 1 (Hierarchical Anchor Graph). $\mathcal{G} = \{\mathcal{X}, \mathcal{U}, \mathcal{E}\}$ *is a multiple-layer pyramidal graph, where* $\mathcal{X}$ *indicates the data set,* $\mathcal{U}$ *indicates the collection of anchor sets, and* $\mathcal{E}$ *indicates the collection of the adjacency matrices of inter-layer edges between neighboring layers. Suppose the original datapoints* $\mathcal{X} \in \mathbf{R}^{d \times n}$ *locate in the bottom layer* ($\mathcal{L}_0$) *of the pyramid. The remaining layers* ($\mathcal{L}_b, b = 1, \ldots, h$) *are all composed of multiple anchor sets* $\mathcal{U}_i$*s from fine to coarse, where the size of*

$\mathcal{U}_b \in \mathbf{R}^{d\times m_b}(b=1,\ldots,h)$ is gradually reduced, namely, $m_1 > \cdots > m_h$. All the layers are linked up to a complete graph with h sets of inter-layer adjacency edges, represented by $\mathcal{E} = \{\mathbf{Z}^{0,1},\ldots,\mathbf{Z}^{h-1,h}\} \in \mathbf{R}^{\{n\times m_1,\ldots,m_{h-1}\times m_h\}}$, in which $\mathbf{Z}^{b-1,b}$ denotes the adjacencies between points in $\mathcal{L}_{b-1}$ and $\mathcal{L}_b$.

Based on the hierarchical anchor graph, we can construct a scalable graph-based classifier for multi-class classification tasks, of which the following two key parts are presented in detail: 1) inter-layer adjacency relationships for label inference, which is designed to reduce the number of parameters and make the learning more efficient; and 2) intra-layer adjacency relationships for label smoothness, which is to build an effective regularization and ensure the learning accuracy.

We first pay attention to the former one. Based on the collection of the inter-layer adjacency relationships, we propose to infer the labels of datapoints from $\mathcal{L}_h$ layer by layer throughout the whole graph. As a result, we only need to learn the labels of the coarsest anchors in $\mathcal{L}_h$. Denote $\mathbf{Z}^{\mathrm{H}}$ as the adjacency matrix that estimates the accumulated inter-layer relationships from $\mathcal{L}_0$ to $\mathcal{L}_h$, and we can compute $\mathbf{Z}^{\mathrm{H}}$ as

$$\mathbf{Z}^{\mathrm{H}} = \mathbf{Z}^{0,1}\ldots\mathbf{Z}^{h-1,h} \in \mathbf{R}^{n\times m_h}. \tag{8}$$

Let $\mathbf{A}$ denote the soft label matrix of the anchor set in $\mathcal{L}_h$, and $\mathbf{F}$ denote the inferred label matrix of datapoints in $\mathcal{L}_0$. With the above accumulated matrix, we can conduct the label inference from $\mathcal{L}_h$ to $\mathcal{L}_0$ in a coarse-to-fine manner

$$\begin{aligned}\mathbf{F} &= \mathbf{Z}^{0,1}\mathbf{Z}^{h-1,h}\mathbf{A}\\ &= \mathbf{Z}^{\mathrm{H}}\mathbf{A}.\end{aligned} \tag{9}$$

Next, we consider the label smoothness regularization. In traditional graph-based learning, we prefer sparse intra-layer adjacency matrix because a sparse graph has much less spurious connections between dissimilar points and tends to exhibit high quality. Zhu [49] also pointed out that fully-connected dense graphs perform worse than sparse graphs empirically. Denoting $\mathbf{W}$ as the intra-layer adjacency matrix used in label smoothness regularization, we therefore formulate $\mathbf{W}$ only based on the inter-layer adjacencies between the data layer $\mathcal{L}_0$ and the finest anchor layer $\mathcal{L}_1$ in the hierarchical anchor graph

$$\mathbf{W} = \mathbf{Z}^{0,1}(\Lambda^1)^{-1}\mathbf{Z}^{0,1\mathrm{T}} \in \mathbf{R}^{n\times n}, \tag{10}$$

where the diagonal matrix Λ^1 is defined as $\Lambda^1_{ss} = \sum_{j=1}^n Z^{0,1}_{js}$.

Based on the inferred label matrix $\mathbf{F}$ and the intra-layer adjacency matrix $\mathbf{W}$, we finally obtain Hierarchical Anchor Graph Regularization

$$\mathrm{argmin}_{\mathbf{A}} \parallel \mathbf{Z}^{\mathrm{H}}_{\mathrm{L}}\mathbf{A} - \mathbf{Y}_{\mathrm{L}} \parallel^2_{\mathrm{F}} + \frac{\lambda}{2}\sum_{i,j=1}^{n} W_{ij}\parallel \mathbf{Z}^{\mathrm{H}}_{i\cdot}\mathbf{A} - \mathbf{Z}^{\mathrm{H}}_{j\cdot}\mathbf{A} \parallel^2, \tag{11}$$

where $\mathbf{Z}^{\mathrm{H}}_{\mathrm{L}}$ is the labeled part of $\mathbf{Z}^{\mathrm{H}}$. Similar to Eq. (4), we have $D_{ii} = \sum_j W_{ij} = 1$, and the above expression can be written in the matrix form

$$\mathrm{argmin}_{\mathbf{A}}\|\mathbf{Z}^{\mathrm{H}}_{\mathrm{L}}\mathbf{A} - \mathbf{Y}_{\mathrm{L}}\|^2_{\mathrm{F}} + \lambda\mathrm{tr}(\mathbf{A}^{\mathrm{T}}\mathbf{Z}^{\mathrm{H}^{\mathrm{T}}}(\mathbf{I}-\mathbf{W})\mathbf{Z}^{\mathrm{H}}\mathbf{A}),$$

or

$$\mathrm{argmin}_{\mathbf{A}}\|\mathbf{Z}^{\mathrm{H}}_{\mathrm{L}}\mathbf{A} - \mathbf{Y}_{\mathrm{L}}\|^2_{\mathrm{F}} + \lambda\mathrm{tr}(\mathbf{A}^{\mathrm{T}}\hat{\mathbf{L}}\mathbf{A}), \tag{12}$$

where $\hat{\mathbf{L}}$ is the reduced Laplacian matrix in HAGR, computed by

$$\begin{aligned}\hat{\mathbf{L}} &= \mathbf{Z}^{\mathrm{H}^{\mathrm{T}}}(\mathbf{I}-\mathbf{W})\mathbf{Z}^{\mathrm{H}}\\ &= \mathbf{Z}^{\mathrm{H}^{\mathrm{T}}}\mathbf{Z}^{\mathrm{H}} - (\mathbf{Z}^{\mathrm{H}^{\mathrm{T}}}\mathbf{Z}^{0,1})(\Lambda^1)^{-1}(\mathbf{Z}^{0,1^{\mathrm{T}}}\mathbf{Z}^{\mathrm{H}}) \in \mathbf{R}^{m_h\times m_h}.\end{aligned} \tag{13}$$

As we can see, although our label smoothness regularization is first performed on the labels of all datapoints with the finest anchor layer, the optimization only involves a reduced Laplacian matrix with the size of the coarsest anchor layer. Therefore, HAGR can overcome the limitation of AGR and improve the computation in matrix inversion. Note that since $\mathbf{Z}^{\mathrm{H}}$ is the product of a series of k-sparse adjacency matrices, we will show that the computation of $\hat{\mathbf{L}}$ is also efficient. The detailed computational costs of HAGR will be analyzed later.

With simple derivations, we obtain a global optimal solution for the soft label matrix of the anchor set in $\mathcal{L}_h$ as

$$\mathbf{A} = (\mathbf{Z}^{\mathrm{H}^{\mathrm{T}}}_{\mathrm{L}}\mathbf{Z}^{\mathrm{H}}_{\mathrm{L}} + \lambda\hat{\mathbf{L}})^{-1}\mathbf{Z}^{\mathrm{H}^{\mathrm{T}}}_{\mathrm{L}}\mathbf{Y}_{\mathrm{L}}. \tag{14}$$

Based on the learnt labels of the coarsest anchors and the inter-layer adjacency matrix $\mathbf{Z}^{\mathrm{H}}$, we can finally infer the hard label for any unlabeled datapoint

$$\widehat{y}_i = \mathrm{argmax}_{r\in\{1,\ldots,c\}}\frac{\mathbf{Z}^{\mathrm{H}}_{i\cdot}\times\mathbf{A}_{\cdot r}}{\beta_r}\pi_r, i = l+1,\ldots,n, \tag{15}$$

where $\beta_r = \mathbf{1}^{\mathrm{T}}\mathbf{Z}^{\mathrm{H}}\mathbf{A}_{\cdot r}$ is the normalization factor, and π_r is the desirable proportion for class r [50].

From the definition of hierarchical anchor graph, we can see its flexibility. We can vary the number of anchor layers and the number of anchors in each layer. We leave the specific analysis on the parameter settings in the experimental section. In particular, we find that, if our hierarchical anchor graph only contains one anchor layer $(h=1)$, it degrades to the anchor graph, and correspondingly HAGR becomes equivalent to AGR.

4.2 Efficient Graph Construction

Like anchor graph, the construction of a hierarchical anchor graph involves two issues, i.e., the generation of anchor sets and the inter-layer adjacency estimation between neighboring layers.

For the first issue, we can simply follow the anchor graph models in [24], [38] to use a fast clustering algorithms to handle it. As for the issue of the weight estimation, besides the standard kernel regression method, formulating it as a geometric reconstruction problem is an alternative choice [23], [36]. However, the kernel regression takes $O(dnm_1)$ time complexity, while the geometric based methods need more time in solving an optimization problem. For extremely large datasets, both of them can bring intractable computational costs.

To improve the efficiency of the graph construction, we investigate an ANNS technique to accelerate the weight estimation. To obtain adjacency relationships between points in $\mathcal{L}_{b-1}$ and $\mathcal{L}_b$, we first build a *Kmeans tree* $\mathcal{T}$ upon points in $\mathcal{L}_b$. Then, for each point in $\mathcal{L}_{b-1}$, we find its k nearest points in $\mathcal{L}_b$ with $\mathcal{T}$, and compute a set of l_1 normalized weights between them. With this operation, for example, estimating the adjacencies between n datapoints and m_1 anchors can be efficiently implemented in $O(dn\log m_1)$ [29]. The whole process is summarized in Algorithm 1. Note that aiming at further efficiency, other state-of-the-art techniques, such as Hashing [30], [39], [40], can be considered.

Algorithm 1. *Kmeans-Tree*-Based Inter-Layer Weight Estimation

```
Input: points in $\mathcal{L}_{b-1}$, points in $\mathcal{L}_b$, number of nearest neighbors $k$.
1: Employ Kmeans tree building algorithm on points in $\mathcal{L}_b$,
       and obtain a Kmeans tree $\mathcal{T}$.
For each point $\mathbf{v}_i$ in $\mathcal{L}_{b-1}$
2: Employ Kmeans tree searching algorithm for $\mathbf{v}_i$ on tree
       $\mathcal{T}$, and obtain indices of its $k$ approximate nearest
       neighbors $\langle i \rangle$, with the corresponding distances $\mathbf{d}_{\langle i \rangle}$.
3: Compute the $l_1$ normalized inter-layer weights
       $\mathbf{z}_{\langle i \rangle} = \exp(-\frac{\mathbf{d}_{\langle i \rangle}}{\sigma})/\hat{z}$, where $\hat{z} = \sum \exp(\frac{-\mathbf{d}_{\langle i \rangle}}{\sigma})$.
End for
4: Construct a sparse inter-layer adjacency matrix $\mathbf{Z}^{b-1,b}$
       with the above indices $\langle i \rangle$s and weights $\mathbf{z}_{\langle i \rangle}$s.
Output: $\mathbf{Z}^{b-1,b}$.
Note: The details about Kmeans tree building and searching algorithms can be found in [29].
```

4.3 Computational Cost of HAGR

We now analyze the computational cost of HAGR. As inter-layer adjacency matrices in HAGR are all k-sparse, we first introduce the following theorem for the fast sparse matrix multiplication. According to it, for example, the time cost of the accumulated inter-layer matrix $\mathbf{Z}^{\mathrm{H}}$ scales as $O(knm_h)$.

Theorem 1. *Let $\mathbf{P}$ and $\mathbf{Q}$ be two $a \times b$ matrices. If $\mathbf{Q}$ contains at most c non-zero entries, the naive algorithm can obtain product $\mathbf{O} = \mathbf{P}\mathbf{Q}^{\mathrm{T}}$ with ac multiplications. The similar bound is obtained when $\mathbf{P}$ contains at most c non-zero entries. The number of additions required is also bounded by the required number of multiplications.*

The proof of the above theorem can be found in [43].

Then, the steps of HAGR and the corresponding time costs are summarized as follows.

1) Construct a hierarchical anchor graph with Algorithm 1. The computational cost of computing the adjacency matrices is $O(\sum_{b=1}^{h} dm_{b-1}\log m_b)$, where $m_0 = n$. Since practically we usually have $n \gg m_b$, this cost can be approximated as $O(dn\log m_1)$.
2) Calculate the reduced Laplacian matrix $\hat{\mathbf{L}}$ via Eq. (13). As the main cost of this step is the sparse matrix multiplication, based on Theorem 1, the total cost here scales as $O(knm_h)$.
3) Carry out the graph regularization via Eq. (14). The complexity of the matrix inversion is $O(m_h^3)$.
4) Predict the hard labels of unlabeled datapoints via Eq. (15). As we have obtained $\mathbf{Z}^{\mathrm{H}}$ in step 2, it can be conducted efficiently in $O(nm_hc)$.

To sum up, the time complexity of HAGR scales as

$$O(dn\log m_1 + knm_h + m_h^3 + nm_hc),$$

TABLE 2
Comparison of Computational Complexities of Three Graph-Based Methods

Methods	LLGC	AGR	HAGR
Graph construction	$O(dn\log n)$	$O(dn\log m_1)$	$O(dn\log m_1)$
Regularization	$O(n^3)$	$O(knm_1 + m_1^3)$	$O(knm_h + m_h^3)$
Inference	-	$O(nm_1c)$	$O(nm_hc)$

where d is the number of feature dimensions, m_b is the number of anchors in the bth layer, k is the number of nearest neighbors in adjacency estimation, and c is the number of classes.

Here we also summarize the computational costs of Learning with Local and Global Consistency (LLGC [47], a typical graph-based SSL method), AGR and HAGR in Table 2, in which Algorithm 1 is applied into all these methods for a fair comparison. From the table we can observe that, although their complexities in graph construction become linear with respect to the data size and can be comparable, LLGC still has a cubic-time complexity in graph regularization. AGR faces a dramatically increase of computational cost when anchors need to be sufficiently dense for a reasonable accuracy. However, as the scale of the coarsest anchors can be much smaller than the finest anchors, i.e., $m_1 \gg m_h$, HAGR has a much less computational cost and is able to deal with large-scale datasets.

4.4 Discussion on Adjacency Designs

In Section 4.1, we suggest to build the inter-layer adjacency matrix $\mathbf{Z}^{\mathrm{H}}$ with anchors layer by layer, and the intra-layer adjacency matrix $\mathbf{W}$ only based on points in $\mathcal{L}_0$ and $\mathcal{L}_1$. Now we present an in-depth analysis on these two aspects.

4.4.1 On the Inter-Layer Adjacency

In HAGR, we model inter-layer adjacency relationships from $\mathcal{L}_0$ to $\mathcal{L}_h$ layer by layer, and then infer the labels of datapoints in a coarse-to-fine manner. According to Eq. (7), it leads to the adaptive relationships between datapoints and the coarsest anchors. That is, when the datapoint is inside the convex envelope of its k nearest anchors in $\mathcal{L}_h$, this datapoint only has connection with these k anchors. When the datapoint is close to the convex envelope's margin, it can build extra inter-layer edges with other nearest anchors in $\mathcal{L}_h$. Otherwise, if we build adjacencies between points in $\mathcal{L}_0$ and $\mathcal{L}_h$ in one step, we are only able to obtain the inflexible relationships between datapoints and their fixed k nearest anchors in $\mathcal{L}_h$.

In the label inference, the above adaptive relationships lead to more reliable integration of label information from the coarsest anchors. Without loss of generality, we demonstrate this by a toy example in Fig. 2, where we have $k = 3$ and $h = 2$. In this example, we want to infer the labels of datapoints ($\mathbf{x}_i, i = 1, 2$) assisted with the labels of the nearby anchors ($\mathbf{u}_s, s = 1, 2, 3, 4$) in $\mathcal{L}_2$. In our coarse-to-fine manner, the datapoint $\mathbf{x}_1$, which is inside the convex envelope of its 3 nearest anchors in $\mathcal{L}_2$, receives labels from these 3 anchors. Meanwhile, the datapoint $\mathbf{x}_2$, which is near to a margin of its convex envelope, can receive label information from both $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3$ and $\mathbf{u}_4$, due to the transitional anchor $\mathbf{u}'_4$

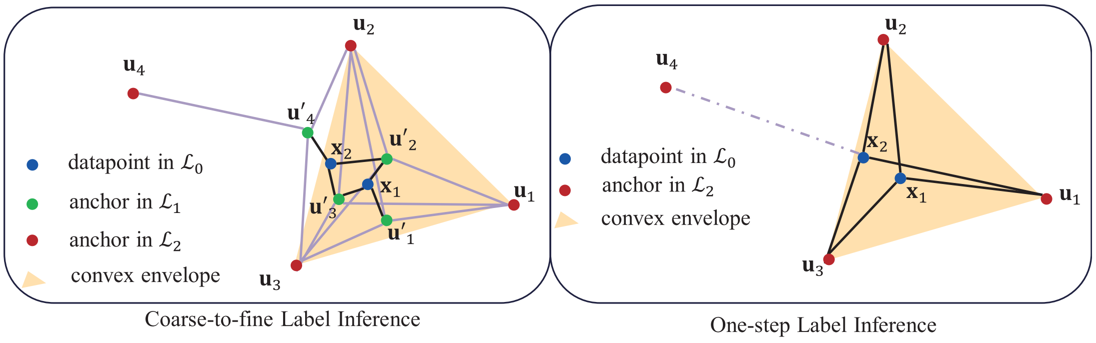


Fig. 2. An illustration of anchor-based label inference in a hierarchical anchor graph. Note that we ignore other points as they have no influence on the label inference here.

in $\mathcal{L}_1$. On the contrary, suppose we conduct the one-step label inference between datapoints and their fixed k coarsest anchors. When $k = 3$, the label information on $\mathbf{u}_4$ is ignored in predicting $\mathbf{y}_2$. When $k = 4$, the noisy labels of the coarsest anchors far away are introduced while inferring $\mathbf{y}_1$. In Section 5.1, we will empirically demonstrate that the classification accuracy can be obviously improved due to this characteristic of $\mathbf{Z}^{\mathrm{H}}$, although the labels of datapoints are still inferred from sparse anchors, namely, the coarsest anchors.

4.4.2 On the Intra-Layer Adjacency

Note that we compute the intra-layer adjacency matrix as $\mathbf{W} = \mathbf{Z}^{0,1}\Lambda_1^{-1}\mathbf{Z}^{0,1^{\mathrm{T}}}$. That means, we build $\mathbf{W}$ only based on the anchors in $\mathcal{L}_1$ to force the intra-layer edges stay between similar datapoints. We can also build $\mathbf{W}$ based on anchors in coarser layers, such as using the hth layer, i.e., $\mathbf{W} = \mathbf{Z}^{0,h}\Lambda_h^{-1}\mathbf{Z}^{0,h^{\mathrm{T}}} \in \mathbf{R}^{n\times n}$, $h > 1$, But note that using different anchor layers to build $\mathbf{W}$ leads to nearly the same computational costs of HAGR. Meanwhile, too sparse anchors will arise many incorrect intra-layer edges between dissimilar datapoints since they possibly share a common anchor. That is why we compute $\mathbf{W}$ only using the anchors in $\mathcal{L}_1$.

5 Experiment

In this section, we investigate both the effectiveness and efficiency of our proposed HAGR on real-world datasets. All the experiments are implemented on a PC with E5-2620 v2 @2.10 GHz and 64G RAM. Here we use the following five datasets with scales varying from 20,000 to 8,100,000. The descriptions of these datasets are given in below, and some statistics of them are listed in Table 3.

1) *Letter*. The dataset contains 20,000 samples of capital letters from 'A' to 'Z' in the English alphabet [11]. Each sample is converted into 16 primitive numerical attributes (statistical moments and edge counts).
2) *MNIST*. It contains 70,000 samples of handwritten digits from '0' to '9' [21]. Each of the ten classes contains about 7,000 samples, which are images centered in a 28×28 field by computing the center of mass of the pixels. We directly use the normalized grayscale value as the feature.
3) *Extended MNIST*. The extended MNIST is widely used in many large-scale graph-based works [15], [19], [24]. The dataset is constructed by translating the original images in MNIST one pixel in each direction. As a result, there are 630,000 samples in 900 dimensions by using the normalized grayscale values as features.
4) *Extended USPS*. The original USPS dataset contains 7,291 training samples of handwritten digits in ten classes [23]. All the digits from '0' to '9' are extended by shifting the 16×16 images in all directions for up to five pixels [33]. There are 882,211 samples in 676 dimensions in total.
5) *MNIST8M*. The MNIST8M dataset has been used in [16], [22], [26] to verify the effectiveness of large-scale learning algorithms. It contains totally 8,100,000 samples in 784 dimensions. In this dataset, the first 70,000 samples belong to the standard MNIST dataset, and each remaining example is generated by applying a pseudo-random transformation to the MNIST training example.

Similar to [23] and [36], the above datasets are categorized into small, medium and large sizes. Specifically, in our experiments, we regard Letter and MNIST as small-size datasets, Extended MNIST and Extended USPS as medium-size datasets, and MNIST8M as a large-size dataset.

5.1 On the Effectiveness of Intra-Layer and Inter-Layer Adjacency Matrices

We conduct experiments on small-size datasets, i.e., Letter and MNIST, to validate the effectiveness of two adjacency designs discussed in Section 4.4.

We first construct a hierarchical anchor graph with two anchor layers, where the size of $\mathcal{L}_1$ is empirically set to

TABLE 3
Details of the Five Databases Used in Our Experiments

	Letter	MNIST	Extended MNIST	Extended USPS	MNIST8M
# of instances	20,000	70,000	630,000	882,211	8,100,000
# of categories	26	10	10	10	10
# of dimensions	16	784	900	676	784

TABLE 4
The Differences of HAGR_{base}, HAGR_W, HAGR_Z, and HAGR in Terms of Label Inference and Label Smoothness Regularization

Approaches	Adjacency Matrix	
	Label Inference Term	Label Smoothness Regularization Term
HAGR_{base}	$\mathbf{Z}^{0,2}$	$\mathbf{Z}^{0,2}(\Lambda^2)^{-1}\mathbf{Z}^{0,2^{\mathrm{T}}}$
HAGR_W	$\mathbf{Z}^{0,2}$	$\mathbf{Z}^{0,1}(\Lambda^1)^{-1}\mathbf{Z}^{0,1^{\mathrm{T}}}$
HAGR_Z	$\mathbf{Z}^{0,1}\mathbf{Z}^{1,2}$	$\mathbf{Z}^{0,2}(\Lambda^2)^{-1}\mathbf{Z}^{0,2^{\mathrm{T}}}$
HAGR	$\mathbf{Z}^{0,1}\mathbf{Z}^{1,2}$	$\mathbf{Z}^{0,1}(\Lambda^1)^{-1}\mathbf{Z}^{0,1^{\mathrm{T}}}$

$n/10$, and the size of $\mathcal{L}_2$ varies from 100 to 500. For a clear comparison, we change the formulation of AGR to

$$\mathrm{argmin}_{\mathbf{A}}\|\mathbf{Z}_{\mathrm{L}}\mathbf{A}-\mathbf{Y}_{\mathrm{L}}\|_{\mathrm{F}}^2+\frac{\lambda}{2}\sum_{i,j=1}^{n}W_{ij}\|\,\mathbf{Z}_{i\cdot}\mathbf{A}-\mathbf{Z}_{j\cdot}\mathbf{A}\,\|^2, \quad (16)$$

where $\mathbf{W}$ is the intra-layer adjacency matrix for label smoothness regularization and $\mathbf{Z}$ is the inter-layer adjacency for label inference.

Then, based on the above formulation and the hierarchical anchor graph, we define three intermediate versions for HAGR:

1) HAGR_{base}, which has the same structure to AGR and is a baseline for comparison. It only employs anchors in $\mathcal{L}_2$ and datapoints in $\mathcal{L}_0$ to build adjacency matrices for both the label smoothness regularization and label inference.
2) HAGR_W, which is an improved version of HAGR_{base} with a change on label smoothness regularization. This method is compared in order to validate the intra-layer adjacency design. Compared with HAGR_{base}, the only difference is that, for the label smoothness regularization, it builds $\mathbf{W}$ as $\mathbf{Z}^{0,1}(\Lambda^1)^{-1}\mathbf{Z}^{0,1^{\mathrm{T}}}$ in Eq. (16)
3) HAGR_Z, which is an improved version of HAGR_{base} with a change on label inference. This method is compared in order to validate the inter-layer adjacency design with a coarse-to fine manner. Compared with HAGR_{base}, it builds an accumulated inter-layer matrix $\mathbf{Z}$ as $\mathbf{Z}^{0,1}\mathbf{Z}^{1,2}$ in Eq. (16).

The differences of HAGR and the above methods are summarized in Table 4. For the other parameters, we set k to 3 to make the graph sparse, and tune λ to its optimal values. In this way, we can provide a fair comparison for these algorithms to validate different adjacency designs. We randomly select 260 and 100 labeled samples for Letter and MNIST, respectively, and leave the remaining ones unlabeled for SSL models.

Table 5 shows the classification accuracies of the above methods. From this table, we have three observations. *First*, by comparing HAGR_W with HAGR_{base}, we can see that, although two methods build the same inter-layer relationships between points in $\mathcal{L}_0$ and $\mathcal{L}_2$, HAGR_W obtains higher accuracies than HAGR_{base}. The main reason is that, by introducing a much sparser intra-layer adjacency matrix, HAGR_W can better smooth the labels of datapoints. *Second*, by comparing HAGR_Z with HAGR_{base}, we can see that, the former obtains better classification performances than the latter, which shows the effectiveness of our adaptive inter-layer adjacency relationships in label inference. *Third*, when the size of $\mathcal{L}_2$ increases, the accuracies of all these approaches increase, and HAGR consistently outperforms the other three methods.

5.2 Comparison with Existing Methods

To demonstrate both the efficiency and effectiveness of the proposed HAGR, we compare it with several state-of-the-art anchor-based SSL models, such as AGR and EAGR. We also report the performance of several baseline methods including 1NN, linear SVM. *For clarity, here we use 'HAGR-m_1-m_2-$\cdots$-m_h' to denote the HAGR method built upon a hierarchical anchor graph with h anchor layers.* For example, 'HAGR-5000-500' means there are two anchor layers in its graph structure, which contain 5,000 and 500 anchors, respectively. The methods for comparison are described in below.

1) The nearest neighbor method, which determines the label of a sample by referring to its closest sample in the labeled set. The method is denoted as '1NN'.
2) Linear SVM [8]. We use the SVM implementation from LIBLINEAR, which is a library for large-scale linear classification. The method is denoted as 'LSVM'.
3) Anchor graph Regularization [23], which is built upon an anchor graph with single anchor layer. It is the prime counterpart in our experiments, and we denote it as 'AGR'.

TABLE 5
Accuracy (%) Comparison of HAGR_{base}, HAGR_W, HAGR_Z, and HAGR on the Letter and MINST Datasets

Dataset	m_1	m_2	HAGR_{base}	HAGR_W	HAGR_Z	HAGR
Letter	2,000	100	45.19 ± 1.53	45.61 ± 1.59	49.19 ± 0.70	$\mathbf{49.57 \pm 1.02}$
($l = 260$)		200	50.91 ± 1.40	51.30 ± 1.31	54.75 ± 1.21	$\mathbf{54.93 \pm 1.05}$
		300	53.72 ± 1.54	54.76 ± 1.73	57.00 ± 1.43	$\mathbf{57.78 \pm 1.48}$
		400	55.58 ± 1.66	56.99 ± 1.57	58.30 ± 1.34	$\mathbf{59.88 \pm 1.42}$
		500	56.80 ± 1.65	58.00 ± 1.69	59.51 ± 1.18	$\mathbf{60.86 \pm 1.36}$
MNIST	7,000	100	79.17 ± 1.39	80.33 ± 1.30	82.33 ± 1.15	$\mathbf{84.59 \pm 0.89}$
($l = 100$)		200	83.28 ± 1.21	83.92 ± 1.17	85.01 ± 0.87	$\mathbf{86.79 \pm 0.66}$
		300	84.44 ± 0.98	85.03 ± 1.02	85.89 ± 0.97	$\mathbf{88.01 \pm 1.00}$
		400	85.30 ± 1.31	85.92 ± 1.26	86.21 ± 1.15	$\mathbf{88.32 \pm 1.15}$
		500	86.35 ± 1.80	86.82 ± 1.74	86.84 ± 1.18	$\mathbf{88.66 \pm 1.23}$

TABLE 6
Classification Accuracies (%) with Different Number of Labeled Samples on the Extended MNIST Dataset

# of labeled samples	1NN	LSVM	AGR	EAGR	HAGR-20,000-5,000	HAGR-200,000-20,000-5,000
100	60.95 ± 0.59	58.58 ± 2.11	88.42 ± 1.36	88.48 ± 1.29	87.97 ± 1.28	$\mathbf{90.75 \pm 1.03}$
200	69.33 ± 0.99	64.07 ± 1.58	90.23 ± 0.44	90.80 ± 0.42	89.81 ± 0.56	$\mathbf{92.21 \pm 0.42}$
300	73.51 ± 0.89	66.99 ± 0.53	91.10 ± 0.38	91.84 ± 0.35	90.65 ± 0.43	$\mathbf{93.13 \pm 0.31}$
400	75.80 ± 0.60	69.50 ± 0.86	91.35 ± 0.34	92.15 ± 0.29	91.15 ± 0.33	$\mathbf{93.39 \pm 0.18}$
500	77.77 ± 0.41	71.47 ± 1.21	92.12 ± 0.18	92.66 ± 0.17	91.96 ± 0.12	$\mathbf{93.73 \pm 0.19}$
600	79.09 ± 0.53	72.69 ± 1.30	92.47 ± 0.10	92.99 ± 0.13	92.27 ± 0.11	$\mathbf{93.95 \pm 0.11}$
700	80.17 ± 0.29	73.69 ± 1.96	92.54 ± 0.11	93.11 ± 0.13	92.43 ± 0.12	$\mathbf{94.05 \pm 0.11}$
800	81.10 ± 0.41	75.22 ± 1.40	92.79 ± 0.13	93.39 ± 0.09	92.61 ± 0.10	$\mathbf{94.15 \pm 0.06}$
900	81.94 ± 0.45	75.93 ± 1.11	93.09 ± 0.09	93.63 ± 0.10	92.93 ± 0.09	$\mathbf{94.23 \pm 0.06}$
1,000	82.60 ± 0.38	76.69 ± 0.95	93.23 ± 0.11	93.77 ± 0.08	93.09 ± 0.10	$\mathbf{94.28 \pm 0.09}$

4) Efficient Anchor graph Regularization [36], which is an improved version of AGR. The method is denoted as 'EAGR'.
5) HAGR-m_1-m_2, which denotes a two-anchor-layer HAGR method. We build the corresponding hierarchical anchor graph by adding a coarser anchor layer above the anchor graph in methods (3-4). The purpose of comparing our approach with this method is to demonstrate the efficiency by introducing a smaller anchor layer.
6) HAGR-m_1-m_2-m_3, which denotes a three-anchor-layer HAGR method. Based on the graph structure in method (5), we add a finer anchor layer with sampled points between its data layer and two anchor layers. This three-anchor-layer HAGR is our proposed approach for the classification on the medium-size and large-size datasets.

Note that here we do not further compare our approach with several other SSL or large-scale classification methods such as conventional graph-based SSL [47], the Eigenfunctions method introduced in [9], the Laplacian SVM method introduced in [28] and the Prototype Vector Machines method introduced in [45], [46], due to the following two facts. First, several methods can hardly be implemented on very large datasets. Second, existing studies have already demonstrated the performance superiority of AGR and EAGR over these methods [23], [36], and thus the superiority of HAGR will be greater if it outperforms AGR and EAGR.

For fair comparisons, we consistently apply the proposed ANNS-based weight estimation algorithm for methods (3-6) in their graph construction, and corresponding kernel widths are set by cross validation. For the above methods, we tune λ to the optimal values.

5.2.1 Medium-Size Datasets

We first conduct experiments on the Extended MNIST and Extended USPS datasets. To accelerate the running speed, we follow [23] and perform PCA to reduce the feature dimension to 86. For the two medium-size datasets, we consistently construct the anchor graph with 20,000 anchors to build AGR and EAGR. Then, by further adding anchor layers, we build two HAGR methods as 'HAGR-20,000-5,000' and 'HAGR-200,000-20,000-5,000', respectively. As for the setting of semi-supervised learning, we vary the number of labeled samples $l = \{100, 200, \ldots, 1,000\}$, while the rest samples remain as unlabeled data.

Averaged over 20 trials, the classification accuracies of the Extend MNIST and Extend USPS datasets are shown in Tables 6 and 7, respectively. The time costs of SSL methods are listed in Table 8.

From these tables, the following observations can be made. *First*, the performances of all graph-based SSL approaches stay at a higher level than Linear SVM and 1NN. This demonstrates the usefulness of unlabeled data in SSL. *Second*, compared with AGR and EAGR, the accuracies of HAGR-20,000-5,000 are slightly lower. However, the performance gap is quite limited-compared with AGR, the accuracy loss of this two-anchor-layer HAGR is smaller than 0.5 percent in most cases. It means that, by building an intra-layer adjacency matrix based on a fixed-size anchor set for label smoothness regularization, the effectiveness of the anchor-based learning can be almost maintained even

TABLE 7
Classification Accuracies (%) with Different Number of Labeled Samples on the Extended USPS Dataset

# of labeled samples	1NN	AGR	EAGR	HAGR-20000-5000	HAGR-200000-20000-5000
100	38.37 ± 0.71	63.82 ± 1.33	64.09 ± 0.85	63.43 ± 1.11	$\mathbf{68.06 \pm 1.55}$
200	47.96 ± 1.14	73.60 ± 0.90	73.75 ± 0.96	73.48 ± 0.96	$\mathbf{77.90 \pm 1.33}$
300	53.48 ± 1.06	78.47 ± 0.91	78.88 ± 0.78	78.28 ± 0.67	$\mathbf{82.28 \pm 0.92}$
400	57.61 ± 1.00	81.41 ± 0.60	81.75 ± 0.45	81.00 ± 0.57	$\mathbf{84.50 \pm 0.81}$
500	61.19 ± 0.80	83.51 ± 0.94	84.09 ± 0.75	84.05 ± 0.78	$\mathbf{86.35 \pm 0.94}$
600	63.94 ± 0.64	84.50 ± 0.88	85.28 ± 0.76	84.09 ± 0.65	$\mathbf{87.11 \pm 0.84}$
700	65.90 ± 0.44	85.80 ± 0.70	86.52 ± 0.56	85.31 ± 0.42	$\mathbf{88.10 \pm 0.49}$
800	67.79 ± 0.38	86.31 ± 0.93	87.59 ± 0.74	86.29 ± 0.69	$\mathbf{88.91 \pm 0.65}$
900	69.18 ± 0.38	86.95 ± 0.68	88.23 ± 0.59	86.95 ± 0.50	$\mathbf{89.44 \pm 0.32}$
1,000	70.71 ± 0.60	87.87 ± 0.55	88.82 ± 0.35	87.55 ± 0.47	$\mathbf{90.01 \pm 0.31}$

TABLE 8
The Comparison of Time Costs (in Seconds) of AGR, EAGR, and HAGR Methods on Medium-Size Datasets

Dataset	AGR	EAGR	HAGR-20,000-5,000	HAGR-200,000-20,000-5,000
Extended MNIST	152.81	151.58	12.12	17.06
Extended USPS	163.31	162.75	17.55	24.18

TABLE 9
Classification Accuracies (%) with Different Number of Labeled Samples on the MNIST8M Dataset

# of labeled samples	1NN	LSVM	AGR	EAGR	HAGR-30,000-5,000	HAGR-300,000-30,000-5,000
100	60.16 ± 1.96	59.67 ± 2.19	89.87 ± 1.78	90.27 ± 0.18	89.46 ± 1.24	$\mathbf{91.36 \pm 0.70}$
200	68.66 ± 1.29	64.46 ± 2.37	91.15 ± 0.59	91.76 ± 0.57	90.85 ± 0.50	$\mathbf{92.46 \pm 0.42}$
300	72.78 ± 0.81	66.79 ± 2.25	92.21 ± 0.51	92.37 ± 0.51	91.66 ± 0.42	$\mathbf{93.05 \pm 0.37}$
400	75.33 ± 0.60	68.33 ± 1.97	92.47 ± 0.44	92.73 ± 0.38	92.16 ± 0.36	$\mathbf{93.43 \pm 0.37}$
500	77.24 ± 0.55	70.65 ± 1.49	92.70 ± 0.41	93.05 ± 0.29	92.50 ± 0.29	$\mathbf{93.78 \pm 0.24}$
600	78.58 ± 0.54	72.64 ± 1.36	92.80 ± 0.34	93.17 ± 0.27	92.64 ± 0.26	$\mathbf{93.90 \pm 0.27}$
700	79.87 ± 0.70	73.80 ± 1.27	93.12 ± 0.31	93.41 ± 0.30	92.92 ± 0.28	$\mathbf{94.10 \pm 0.25}$
800	81.02 ± 0.50	73.87 ± 1.18	93.19 ± 0.23	93.51 ± 0.15	93.06 ± 0.16	$\mathbf{94.21 \pm 0.15}$
900	81.76 ± 0.49	73.97 ± 0.96	93.29 ± 0.36	93.63 ± 0.21	93.18 ± 0.26	$\mathbf{94.28 \pm 0.16}$
1,000	82.51 ± 0.42	76.95 ± 1.13	93.49 ± 0.22	93.79 ± 0.15	93.37 ± 0.16	$\mathbf{94.39 \pm 0.12}$

TABLE 10
The Comparison of Time Costs (in Seconds) of AGR, EAGR, and HAGR Methods on the MNIST8M Dataset

Dataset	AGR	EAGR	HAGR-30,000-5,000	HAGR-300,000-30,000-5,000
MNIST8M	665.07	662.60	104.97	137.54

we significantly reduce the size of to-be-learned anchor set (from 20,000 to 5,000). When taking into account the running time of the learning process (shown in Table 8), this accuracy loss becomes acceptable in real applications. *Third*, by increasing the size of $\mathcal{L}_1$ over 20,000, the performances of the anchor-based approaches can be further improved due to better adjacency relationships. Based on this, HAGR-200,000-20,000-5,000 consistently outperforms AGR and EAGR. When the number of labeled samples is small, this advantage is more obvious, e.g., improvements of about 2 and 4 percent on Extended MNIST and Extended USPS, respectively. It verifies the effectiveness of HAGR by introducing a finer anchor layer to improve the graph regularization. *Fourth*, although the ANNS-based graph construction can be applied to all graph-based approaches, it is particularly suitable for our HAGR. Due to this efficient graph construction and the fast optimization, HAGRs overcome the limitation of AGR and EAGR, and achieve good performances with less computational costs.

5.2.2 Large-Size Dataset

To demonstrate the scalability of HAGR, we conduct experiments on the MNIST8M dataset, where the dimension of examples is also reduce to 86 by PCA. We first construct an anchor graph with 30,000 anchors to build AGR and EAGR. Then, we build two HAGR methods, i.e., 'HAGR-30000-5,000' and 'HAGR-300,000-30,000-5,000', respectively. By repeating the similar evaluation process, we display the classification accuracies over 20 trials in Table 9. The time costs of SSL approaches are listed in Table 10.

From these tables, we have the following observations. *First*, compared with AGR and EAGR, the accuracy loss of HAGR-30,000-5,000 is acceptable while its time cost is much less. *Second*, as the number of labeled data varies from 100 to 1,000, the performances of all methods increase, and our HAGR-300,000-30,000-5,000 consistently outperforms the other methods.

It is also worth noting that, in our experimental results, we actually have put more emphasis on the performance superiority of the three-anchor-layer HAGR. One may argue that we can increase the number of anchors in AGR and EAGR, say, setting 300,000 anchors. But this will dramatically increase the time costs of these methods (increased by about 10^3 times), which are practically intractable.

5.3 On the Structure of Hierarchical Anchor Graph

We can see that HAGR is a quite flexible approach, and thus designing the structure of the hierarchical anchor graph can be important in classification tasks. Therefore, we conduct additional experiments by varying the number of anchor layers (h) and the size of each anchor layer (m_b) in the proposed graph model, trying to investigate the impact of these parameters. For convenience, Extended USPS dataset is used in this experiment. We build three anchor graphs with 5,000, 10,000, and 20,000 anchors for implementing AGR and EAGR as comparisons.

For clarity, in the pictures illustrating results, we use black/blue/red lines to show the results of HAGR built with 2/3/4 anchor layers, respectively. Among them, each dot-curve denotes HAGRs with the varying anchor size m_b, and the straight line without dots means the size of each anchor layer of this HAGR is fixed. The specific size is displayed at the bottom of each subfigure.

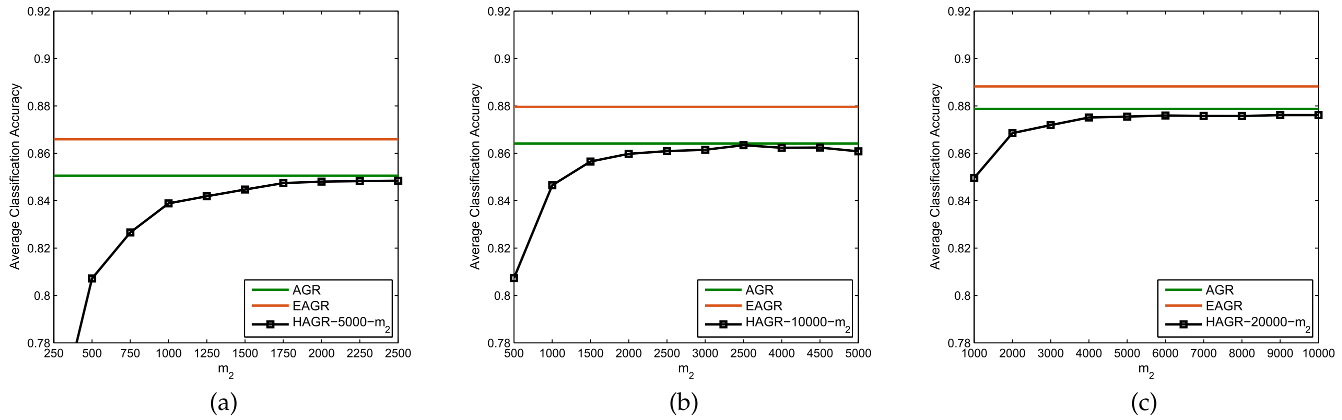


Fig. 3. Average performance curves with respect to the variation of m_2 in HAGR-m_1-m_2.

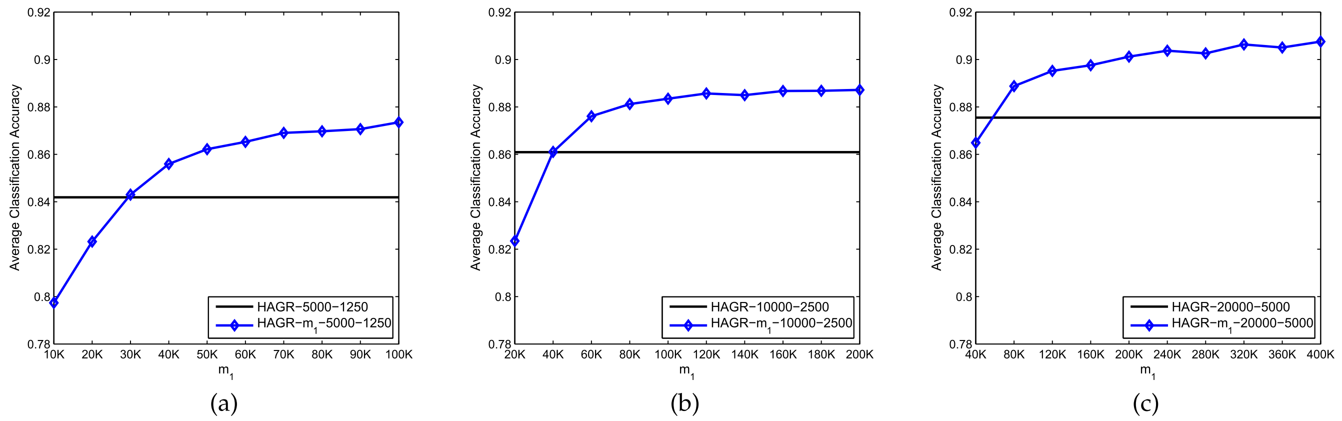


Fig. 4. Average performance curves with respect to the variation of m_1 in HAGR-m_1-m_2-m_3.

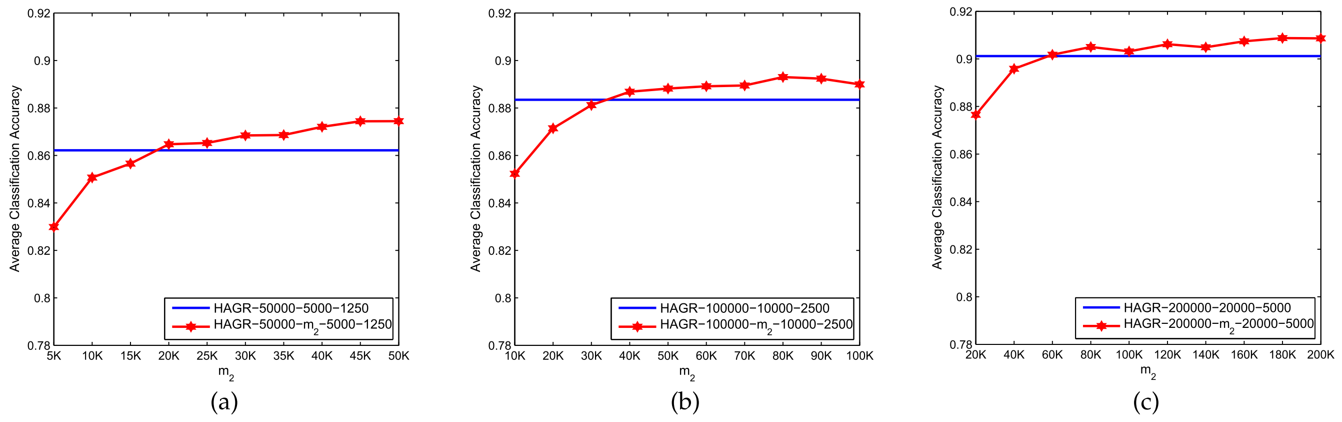


Fig. 5. Average performance curves with respect to the variation of m_2 in HAGR-m_1-m_2-m_3-m_4.

We first test the two-anchor-layer HAGR method. Similar to the process in Section 5.2, for each of the three anchor graphs, we add a coarser anchor layer $\mathcal{L}_2$ above the original $\mathcal{L}_1$ to construct our hierarchical anchor graph ($h = 2$). The performance curves of HAGR-m_1-m_2 with respect to the size of $\mathcal{L}_2$ are shown in Fig. 3. As we can see, the accuracy of HAGR increases rapidly at the first stage. It means that, although we can build a large-scale finest anchor layer in HAGR to improve the performance, the size of to-be-learned anchors cannot be too small. Otherwise, it tends to restrict the performance of HAGR. When m_2 reaches about $m_1/4$, the accuracy of the HAGR becomes stable. We note that this number of to-be-learned anchors m_h can still be much smaller than the number of anchors m_1 used in label smoothness regularization.

Then, we investigate the three-anchor-layer HAGR method. Based on each two-anchor-layer graph above, we add a finer anchor layer $\mathcal{L}_1$ with m_1 anchors between the original data layer and two anchor layers to construct new graph ($h = 3$). The number of the coarsest anchors is set to the empirical value, and the specific settings can be found in Fig. 4. The performance curve of HAGR-m_1-m_2-m_3 with respect to m_1 is shown in Fig. 4. As we can see, larger m_1 brings higher accuracy. The reason is that, by increasing the size of the finest anchor layer, the representation power of these anchors becomes stronger. Then, we can obtain more

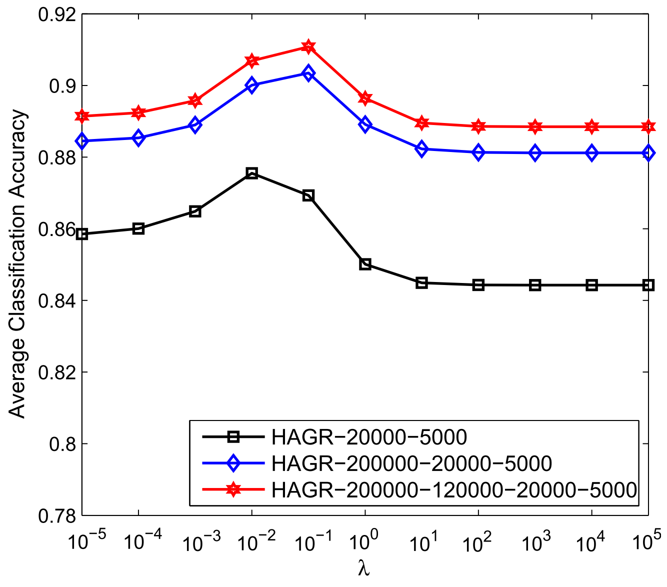


Fig. 6. Average performance curves of HAGR with respect to the variation of λ.

effective adjacency relationships for both the label smoothness regularization and label inference.

We also build the four-anchor-layer HAGR method ($h = 4$) by further adding an anchor layer between original $\mathcal{L}_1$ and $\mathcal{L}_2$ in each three-anchor-layer graph above. The performance curve of HAGR-m_1-m_2-m_3-m_4 with respect to current m_2 is shown in Fig. 5. As we can see, further increasing the number of anchor layers can improve the accuracy, which demonstrates the effectiveness of exploring multiple-layer anchors to model the coarse-to-fine label inference.

To summarize, when fixing the number of anchor layers in hierarchical anchor graph, the performance of HAGR is fairly robust to the number of anchors in each layer over a large range (as we can see in each subfigure), and the to-be-learned anchors can be much fewer than the anchors used in label smoothness regularization. Meanwhile, increasing the number of anchor layers can improve accuracy (under a comparison from Figs. 3 to 5), and in our experiments, we empirically find that three or four anchor layers are already able to well handle million-scale datasets.

5.4 On the Trade-Off Parameter λ

We also test the sensitivity of parameter λ in the proposed approach. For simplicity, we only illustrate the results of HAGR with $m_h = 5,000$ on the Extended USPS dataset. But similar observations can also be obtained in other cases. As we can see in Fig. 6, the performance of HAGR will not severely degrade when λ varies in a wide range, and increasing the number of anchor layers does not change the robustness of λ.

6 Conclusion

This work proposes a novel Hierarchical Anchor Graph Regularization approach by exploring multiple-layer anchors in a pyramid-style structure. It generalizes the conventional graph-based and anchor-graph-based SSL methods to a hierarchical approach. In HAGR, we perform label smoothness regularization on all datapoints based on the finest anchors. By inferring the labels of datapoints started from the coarsest anchors, we obtain an efficient optimization which only involves a small-size reduced Laplacian matrix. It overcomes the limitations of existing AGR approach in dealing with extremely large datasets. We also investigate ANNS to improve the efficiency of the construction of the hierarchical anchor graph. Experiments on publicly available large-scale datasets of various sizes have demonstrated the effectiveness of our approach in terms of computational speed and classification accuracy.

Acknowledgments

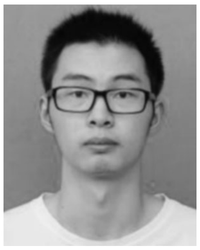
This work is partially supported by the National 973 Program of China under grants 2014CB347600 and 2013CB329604, the Program for Changjiang Scholars and Innovative Research Team in University (PCSIRT) of the Ministry of Education, China, under grant IRT13059, and the National Nature Science Foundation of China under grant 61432019.

References

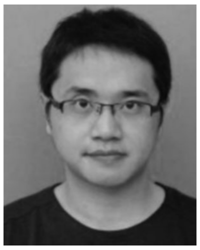
[1] M. Belkin, P. Niyogi, and V. Sindhwani, “Manifold regularization: A geometric framework for learning from labeled and unlabeled examples,” *J. Mach. Learn. Res.*, vol. 7, no. 11, pp. 2399–2434, 2006.
[2] A. Blum and T. Mitchell, “Combining labeled and unlabeled data with co-training,” in *Proc. Conf. Comput. Learn. Theory*, 1998, pp. 92–100.
[3] D. Cai and X. Chen, “Large scale spectral clustering with landmark-based representation,” *IEEE Trans. Cybern.*, vol. 45, no. 8, pp. 1669–1680, Aug. 2015.
[4] V. Castelli and T. M. Cover, “On the exponential value of labeled samples,” *Pattern Recog. Lett.*, vol. 16, no. 1, pp. 105–111, 1995.
[5] L. Chen, I. W. Tsang, and D. Xu, “Laplacian embedded regression for scalable manifold regularization,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 6, pp. 902–915, Jun. 2012.
[6] B. Cheng, J. Yang, S. Yan, Y. Fu, and T. S. Huang, “Learning with $l1$-graph for image analysis,” *IEEE Trans. Image Process.*, vol. 19, no. 4, pp. 858–866, Apr. 2010.
[7] C. Deng, R. Ji, W. Liu, D. Tao, and X. Gao, “Visual reranking through weakly supervised multi-graph learning,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 2013, pp. 2600–2607.
[8] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, “Liblinear: A library for large linear classification,” *J. Mach. Learn. Res.*, vol. 9, no. 9, pp. 1871–1874, 2008.
[9] R. Fergus, Y. Weiss, and A. Torralba, “Semi-supervised learning in gigantic image collections,” in *Proc. Conf. Advances Neural Inf. Proc. Syst.*, 2009, pp. 522–530.
[10] P. Foggia, G. Percannella, and M. Vento, “Graph matching and learning in pattern recognition in the last 10 years,” *Int. J. Pattern Recog. Artif. Intell.*, vol. 28, no. 1, pp. 178–215, 2014.
[11] P. W. Frey and D. J. Slate, “Letter recognition using holland-style adaptive classifiers,” *Mach. Learn.*, vol. 6, no. 2, pp. 161–182, 1991.
[12] L. Gao, J. Song, F. Nie, Y. Yan, N. Sebe, and H. T. Shen, “Optimal graph learning with partial tags and multiple features for image and video annotation,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 4371–4379.
[13] M. Guillaumin, J. Verbeek, and C. Schmid, “Multimodal semi-supervised learning for image classification,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2010, pp. 902–909.
[14] T. J. Hastie, R. J. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Berlin, Germany: Springer, 2010.
[15] M. Hein and S. Setzer, “Beyond spectral clustering-tight relaxations of balanced graph cuts,” in *Proc. Conf. Advances Neural Inf. Process. Syst.*, 2011, pp. 2366–2374.
[16] C.-J. Hsieh, S. Si, and I. S. Dhillon, “A divide-and-conquer solver for kernel support vector machines.” in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 566–574.
[17] S. Huang, M. Elhoseiny, A. Elgammal, and D. Yang, “Learning hypergraph-regularized attribute predictors,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 409–417.

[18] T. Joachims, "Transductive inference for text classification using support vector machines," in *Proc. Int. Conf. Mach. Learn.*, 1999, pp. 200–209.
[19] M. Karlen, J. Weston, A. Erkan, and R. Collobert, "Large scale manifold transduction," in *Proc. Int. Conf. Mach. Learn.*, 2008, pp. 448–455.
[20] S. Kim and S. Choi, "Multi-view anchor graph hashing," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2013, pp. 3123–3127.
[21] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[22] M. Li, J. T.-Y. Kwok, and B. Lü, "Making large-scale nyström approximation possible," in *Proc. Int. Conf. Mach. Learn.*, 2010, pp. 631–638.
[23] W. Liu, J. He, and S.-F. Chang, "Large graph construction for scalable semi-supervised learning," in *Proc. Int. Conf. Mach. Learn.*, 2010, pp. 679–686.
[24] W. Liu, J. Wang, and S.-F. Chang, "Robust and scalable graph-based semisupervised learning," *Proc. IEEE*, vol. 100, no. 9, pp. 2624–2638, Sep. 2012.
[25] W. Liu, J. Wang, S. Kumar, and S.-F. Chang, "Hashing with graphs," in *Proc. Int. Conf. Mach. Learn.*, 2011, pp. 1–8.
[26] L. Bottou, "Large-scale kernel machines," MIT press, 2007.
[27] D. Machin, *Introduction to Multimodal Analysis*. London, U.K.: Bloomsbury Publishing, 2016.
[28] S. Melacci and M. Belkin, "Laplacian support vector machines trained in the primal," *J. Mach. Learn. Res.*, vol. 12, no. 5, pp. 1149–1184, 2009.
[29] M. Muja and D. G. Lowe, "Scalable nearest neighbor algorithms for high dimensional data," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 11, pp. 2227–2240, Nov. 2014.
[30] M. Norouzi, A. Punjani, and D. J. Fleet, "Fast exact search in hamming space with multi-index hashing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 6, pp. 1107–1119, Jun. 2014.
[31] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.
[32] Z. Tian and R. Kuang, "Global linear neighborhoods for efficient label propagation," in *Proc. SIAM Int. Conf. Data Mining*, 2012, pp. 863–872.
[33] I. W. Tsang, J. T. Kwok, and P.-M. Cheung, "Core vector machines: Fast SVM training on very large data sets," *J. Mach. Learn. Res.*, vol. 6, no. 1, pp. 363–392, 2005.
[34] F. Wang and C. Zhang, "Label propagation through linear neighborhoods," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 1, pp. 55–67, Jan. 2008.
[35] J. Wang, J. Wang, G. Zeng, Z. Tu, R. Gan, and S. Li, "Scalable *k*-NN graph construction for visual descriptors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2012, pp. 1106–1113.
[36] M. Wang, W. Fu, S. Hao, D. Tao, and X. Wu, "Scalable semi-supervised learning by efficient anchor graph regularization," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 7, pp. 1864–1877, Jul. 2016.
[37] M. Wang, X.-S. Hua, R. Hong, J. Tang, G.-J. Qi, and Y. Song, "Unified video annotation via multigraph learning," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 5, pp. 733–746, May 2009.
[38] B. Xu, J. Bu, C. Chen, C. Wang, D. Cai, and X. He, "EMR: A scalable graph-based ranking model for content-based image retrieval," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 1, pp. 102–114, Jan. 2015.
[39] Y. Yang, Y. Luo, W. Chen, F. Shen, J. Shao, and H. T. Shen, "Zero-shot hashing via transferring supervised knowledge," in *Proc. ACM Conf. Multimedia*, 2016, pp. 1286–1295.
[40] Y. Yang, F. Shen, H. T. Shen, H. Li, and X. Li, "Robust discrete spectral hashing for large-scale image semantic indexing," *IEEE Trans. Big Data*, vol. 1, no. 4, pp. 162–171, Dec. 2015.
[41] Y. Yang, Y. Yang, H. T. Shen, Y. Zhang, X. Du, and X. Zhou, "Discriminative nonnegative spectral clustering with out-of-sample extension," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 8, pp. 1760–1771, Aug. 2013.
[42] G. Yu, G. Zhang, Z. Zhang, Z. Yu, and L. Deng, "Semi-supervised classification based on subspace sparse representation," *Knowl. Inf. Syst.*, vol. 43, no. 1, pp. 81–101, 2015.
[43] R. Yuster and U. Zwick, "Fast sparse matrix multiplication," *ACM Trans. Algorithms*, vol. 1, no. 1, pp. 2–13, 2005.
[44] L. Zelnik-Manor and P. Perona, "Self-tuning spectral clustering," in *Proc. Conf. Advances Neural Inf. Process. Syst.*, 2005, pp. 1601–1608.
[45] K. Zhang, J. T. Kwok, and B. Parvin, "Prototype vector machine for large scale semi-supervised learning," in *Proc. Int. Conf. Mach. Learn.*, 2009, pp. 1233–1240.
[46] K. Zhang, L. Lan, J. T. Kwok, S. Vucetic, and B. Parvin, "Scaling up graph-based semi-supervised learning via prototype vector machines," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 3, pp. 444–457, Mar. 2015.
[47] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Proc. Conf. Advances Neural Inf. Process. Syst.*, 2004, pp. 321–328.
[48] D. Zhou, J. Huang, and B. Schölkopf, "Learning with hypergraphs: Clustering, classification, and embedding," in *Proc. Conf. Advances Neural Inf. Process. Syst.*, 2006, pp. 1601–1608.
[49] X. Zhu, "Semi-supervised learning literature survey," Comput. Sci., Univ. Wisconsin-Madison, Madison, WI, USA, Tech. Rep. 1530, 2005.
[50] X. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *Proc. Int. Conf. Mach. Learn.*, 2003, pp. 912–919.
[51] X. Zhu and A. B. Goldberg, "Introduction to semi-supervised learning," *Synthesis Lectures Artif. Intell. Mach. Learn.*, vol. 3, no. 1, pp. 1–130, 2009.
[52] X. Zhu, J. Kandola, Z. Ghahramani, and J. D. Lafferty, "Nonparametric transforms of graph kernels for semi-supervised learning," in *Proc. Conf. Advances Neural Inf. Process. Syst.*, 2004, pp. 1641–1648.

Meng Wang received the BE and PhD degrees from the Special Class for the Gifted Young, Department of Electronic Engineering and Information Science, University of Science and Technology of China (USTC), Hefei, China, in 2003 and 2008, respectively. He is a professor with the Hefei University of Technology, China. His current research interests include multimedia content analysis, computer vision, and pattern recognition. He has authored more than 200 book chapters, and journal and conference papers in these areas. He received the ACM SIGMM Rising Star Award 2014. He is an associate editor of the *IEEE Transactions on Knowledge and Data Engineering* and the *IEEE Transactions on Circuits and Systems for Video Technology*. He is a member of the IEEE.



Weijie Fu is currently working toward the PhD degree in the School of Computer and Information, Hefei University of Technology (HFUT). His current research interest focuses on machine learning and data mining.



Shijie Hao received the BE, MS, and PhD degrees from the School of Computer and Information, Hefei University of Technology (HFUT). He is an associate professor with HFUT, China. His current research interests include machine learning and image processing.

Hengchang Liu received the PhD degree from the University of Virginia, in 2011, under the supervision of Professor John Stankovic. He is currently an assistant professor with USTC. His research interests mainly includes big data mining, cyber physical systems, and mobile systems.

Xindong Wu received the bachelor's and master's degrees in computer science from the Hefei University of Technology, China, and the PhD degree in artificial intelligence from the University of Edinburgh, Britain. He is a Yangtze River scholar in the School of Computer Science and Information Engineering, Hefei University of Technology, China, and a professor of computer science with the University of Louisiana at Lafayette. His research interests include data mining, knowledge-based systems, and Web information exploration. He is the steering committee chair of the IEEE International Conference on Data Mining (ICDM), the editor-in-chief of *Knowledge and Information Systems* (*KAIS*, by Springer), and a series editor of the *Springer Book Series on Advanced Information and Knowledge Processing* (*AI&KP*). He was the editor-in-chief of the *IEEE Transactions on Knowledge and Data Engineering* (*TKDE*, by the IEEE Computer Society) between 2005 and 2008. He served as program committee chair/co-chair for ICDM '03 (the 2003 IEEE International Conference on Data Mining), KDD-07 (the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining), and CIKM 2010 (the 19th ACM Conference on Information and Knowledge Management). He is a fellow of the IEEE.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.

Scalable Semi-Supervised Learning by Efficient Anchor Graph Regularization

Meng Wang, *Member, IEEE*, Weijie Fu, Shijie Hao,
Dacheng Tao, *Fellow, IEEE*, and Xindong Wu, *Fellow, IEEE*

Abstract—Many graph-based semi-supervised learning methods for large datasets have been proposed to cope with the rapidly increasing size of data, such as Anchor Graph Regularization (AGR). This model builds a regularization framework by exploring the underlying structure of the whole dataset with both datapoints and anchors. Nevertheless, AGR still has limitations in its two components: (1) in anchor graph construction, the estimation of the local weights between each datapoint and its neighboring anchors could be biased and relatively slow; and (2) in anchor graph regularization, the adjacency matrix that estimates the relationship between datapoints, is not sufficiently effective. In this paper, we develop an Efficient Anchor Graph Regularization (EAGR) by tackling these issues. First, we propose a fast local anchor embedding method, which reformulates the optimization of local weights and obtains an analytical solution. We show that this method better reconstructs datapoints with anchors and speeds up the optimizing process. Second, we propose a new adjacency matrix among anchors by considering the commonly linked datapoints, which leads to a more effective normalized graph Laplacian over anchors. We show that, with the novel local weight estimation and normalized graph Laplacian, EAGR is able to achieve better classification accuracy with much less computational costs. Experimental results on several publicly available datasets demonstrate the effectiveness of our approach.

Index Terms—Semi-supervised learning, anchor graph, local weight estimation

1 Introduction

In many real-world classification tasks, we are usually faced with datasets in which only a small portion of samples are labeled while the rest are unlabeled. A learning mechanism called semi-supervised learning (SSL), which is capable of fully leveraging unlabeled data and labeled data to achieve better classification, is therefore proposed to deal with this situation. In recent years, various semi-supervised learning methods [55] have been developed to adapt different kinds of data, including mixture models [6], co-training [4], semi-supervised support vector machines [18], and graph-based SSL [54]. This learning mechanism is broadly used in many real-world applications such as data mining [30], [34], [35], [36], [52], [53] and multimedia content analysis [14], [15], [24], [25], [49].

In this paper, we focus on the family of graph-based semi-supervised learning methods. These methods are built based on a cluster assumption [51]: nearby points are likely to have the same label. A typical model of these algorithms consists of two main parts: a fitting constraint and a smoothness constraint, both of which have clear geometric meanings. The former means that a good classification function should not change too much from the initial label assignment, while the latter means that this function should have similar semantic labels among nearby points. Based on the above formulation, these algorithms generally produce satisfying classification results in a manifold space [12], [13], [20], [40], [45], [50]. Meanwhile, graph-based SSL can be intuitively explained in a label propagation perspective [54], i.e., the label information from labeled vertices is gradually propagated through graph edges to all unlabeled vertices.

Most traditional graph-based learning methods, however, focus on classification accuracy while ignoring the underlying computational complexity, which is of great importance for the classification of a large dataset, especially given the recent explosive increase in Internet data. The complexity mainly arises from two aspects. The first is the kNN strategy for graph construction, and the second is the inverse calculation of the normalized Laplacian matrix in optimization. Both of them are time-consuming and have a large storage requirement.

To reduce the cubic-time complexity, recent studies seek to speed up the intensive computation of the graph Laplacian manipulation. Anchor graph regularization (AGR) [21], [22], a recently proposed graph-based learning model for large datasets, constructs a novel graph with datapoints and anchors. It reduces the computational cost via subtle matrix factorization by utilizing the intrinsic structure of data distribution. It is exactly in linear time with data size. The anchor graph model has been widely applied to many applications [8], [19], [41], [42], [43] and achieves satisfactory performance.

- *M. Wang, W. Fu, and S. Hao are with the School of Computer and Information Science, Hefei University of Technology, Hefei 230009, China. E-mail: {eric.wangmeng, fwj.edu, hfut.hsj}@gmail.com.*
- *D. Tao is with the Centre for Quantum Computation & Intelligent Systems, and the Faculty of Engineering & Information Technology, University of Technology, Ultimo, NSW 2007, Australia. E-mail: dacheng.tao@uts.edu.au.*
- *X. Wu is with the School of Computer Science and Information Engineering, Hefei University of Technology, Hefei 230009, China, and the Department of Computer Science, University of Vermont, Burlington, VT 05405, USA. E-mail: xwu@cs.uvm.edu.*

Manuscript received 6 Oct. 2015; revised 2 Feb. 2016; accepted 16 Feb. 2016. Date of publication 26 Feb. 2016; date of current version 1 June 2016. Recommended for acceptance by J. Bailey. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TKDE.2016.2535367

1041-4347 © 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

There are two key parts in AGR. The first part is anchor graph construction, in which a local weight matrix that measures the relationship between datapoints and anchors needs to be estimated. The second part is anchor graph regularization, in which an adjacency matrix that measures the strength of graph edges needs to be designed. For local weight estimation, a method called Local Anchor Embedding (LAE) is employed in [21] to replace the conventional kernel-defined weight computation. However, we will demonstrate the limitation of the optimization objective of LAE. Moreover, the LAE process is expensive as it involves a gradient descent solver. For adjacency matrix design, Liu et al. [21] introduce a method that constructs the adjacency matrix based on local weight matrix. But actually it has been shown in [21] that the regularization framework can be equivalent to a regularization on anchors with a ”reduced” graph Laplacian. Therefore, in this work, we propose a novel approach named Efficient Anchor Graph Regularization (EAGR) with a novel local weight estimation method and a more effective normalized graph Laplacian over anchors. In comparison with AGR, EAGR obtains comparable or better accuracy in a shorter time in SSL based classification tasks.

The main contributions of our work are as follows.

(1) We introduce a novel graph-based SSL approach that is able to deal with large datasets. By improving the conventional AGR method, we show that the proposed EAGR is able to achieve better classification accuracy with even much less implementation time. The EAGR also empirically shows its advantages over many existing SSL methods developed for large datasets, such as the methods in [26], [47], [48].

(2) We point out the limitation of the conventional local anchor embedding method and propose a novel approach for local weight estimation. We reformulate the objective function of LAE by replacing the inequality constraint with an absolute operation and obtain an efficient and effective analytical solution. In addition, we incorporate the locality constraints into the objective function to further improve the performance.

(3) Instead of designing an adjacency matrix for all datapoints, we directly compute an adjacency matrix for anchors by exploring their commonly connected datapoints. We show that the derived normalized graph Laplacian over anchors is more effective than the ”reduced” graph Laplacian in [21]. Graph-based learning is performed with the corresponding regularization on anchors.

The rest of this paper is organized as follows. In Section 2, we survey related work. In Section 3, we briefly review the AGR algorithm and conduct an in-depth analysis of its limitations. The proposed approach EAGR is described in Section 4. In Section 5, we conduct experiments on several publicly available datasets to validate our model. Finally, we conclude in Section 6.

2 Related Work

In this section, we focus on the related work of graph-based semi-supervised learning. Once we have constructed a graph for all datapoints, the labels for classification can be propagated from limited labeled data to remaining unlabeled data [17].

For many years, researchers have focused on improving the classification accuracy of graph-based SSL via designing more appropriate label propagation models with simple kNN graph. Zhu et al. [54] advocated the formulation of the learning problem based on Gaussian random field and gave intuitive interpretations for their model. Belkin et al. [2] proposed a classification function which is defined only on the sub-manifold rather than the whole ambient space. Zhou et al. [51] subsequently suggested the design of a classification model which is sufficiently smooth with respect to the intrinsic structure collectively revealed by the known labeled and unlabeled points. There are additionally many works that focus on graph construction to improve classification accuracy. For instance, Zelnik et al. [46] first stated that it would be helpful to consider a local scale in computing the affinity between each pair of points for the edge. Wang et al. [37] developed a graph-based SSL approach based on a linear neighborhood model which assumes that each datapoint can be linearly reconstructed from its neighborhood. Similarly, Tian et al. [33] proposed learning a nonnegative low-rank graph to capture global linear neighborhoods. Although these methods show promising performance in various applications, they are not sufficiently scalable in terms of computing and storage costs, which imposes limitations in handling large datasets.

With the rapid increase in data size, researchers have paid more attention to designing novel approaches to reduce the computational cost of graph-based learning. Wang et al. [38] proposed a multiple random divide-and-conquer approach to construct an approximate neighborhood graph and presented a neighborhood propagation scheme to further enhance the accuracy. Huang et al. [16] proposed a novel label propagation algorithm in which the label information is first propagated from labeled instances to unlabeled instances, and then labels spread among the unlabeled instances until a steady state is reached. These algorithms simplify either the graph construction or the label propagation, and so the computational cost is reduced to some extent. Additionally, hashing strategies [7], [32], [29] can also be applied to facilitate large-scale classification.

Recently, Liu et al. [21] proposed an anchor graph regularization approach. The graph is constructed with datapoints and anchors, which simultaneously reduces computational cost and storage cost. As a result, the anchor graph model has been applied to clustering [5], [44], hashing [23], manifold ranking [43], multi-graph learning [9], and tracking [41]. For example, Yang et al. [44] proposed a low-rank learning method to improve the clustering performance for large-scale manifold data by a two-step bipartite random walk through cluster nodes. Cai et al. [5] proposed an efficient computation of the spectral embedding of data with an anchor-based representation to improve spectral clustering. Liu et al. [23] proposed an anchor graph based hashing method to learn appropriate compact codes with a feasible computational cost. Xu et al. [43] designed a new adjacency matrix with the anchor graph to speed up manifold ranking for image retrieval. Wu et al. [41] presented a local landmark approximation (LLA) model, which iteratively solves its target function based on gradient projection. The model is applied to visual tracking and achieves state-of-the-art performance. In spite of these, AGR still has some limitations, which are analyzed and addressed in the following sections.

3 Anchor Graph Regularization

In this section, we first review the anchor graph regularization model and then give a detailed analysis of its limitations.

3.1 Anchor Graph Regularization Formulation

Given a dataset $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \ldots, (\mathbf{x}_l, \mathbf{y}_l), \mathbf{x}_{l+1}, \ldots, \mathbf{x}_n\}$ with n samples in d dimensions, we can obtain a set of representative anchors $\mathbf{U} = \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m\} \in \mathbb{R}^{d\times m}$. Typically, these anchors are selected by a clustering method such as K-means. These anchors clearly share the same feature space with the original datapoints. Let $f : \mathbf{x} \rightarrow \mathbf{R}$ be a real value function which assigns each point a label from c distinct classes. Then, once we obtain a local weight matrix $\mathbf{Z}$ that measures the potential relationships between datapoints and anchors, we can estimate $f(\mathbf{x})$ for each datapoint as a weighted average of the labels of the anchor set

$$f(\mathbf{x}_i) = \sum_{k=1}^{m} Z_{ik} f(\mathbf{u}_k), \tag{1}$$

with the constraints $\sum_{k=1}^{m} Z_{ik} = 1$ and $Z_{ik} \geq 0$. Note that the element Z_{ik} represents the local weight between datapoint $\mathbf{x}_i$ and anchor $\mathbf{u}_k$.

Kernel-defined weights are usually sensitive to hyperparamters and lack a meaningful interpretation. To measure the local weights more robustly, one can adopt a LLE[31] like objective function with a clear geometric meaning:

$$\text{argmin}_{\mathbf{z}_i} \|\mathbf{x}_i - \mathbf{U}_{\langle i\rangle}\mathbf{z}_{\langle i\rangle}\|^2 \tag{2}$$

$$\text{s.t.} \mathbf{1}^{\mathrm{T}}\mathbf{z}_{\langle i\rangle} = 1, \mathbf{z}_{\langle i\rangle} \succeq 0,$$

where $\langle i\rangle$ is a index set of s closest anchors of $\mathbf{x}_i$. A standard quadratic problem (QP) solver can be used to solve Eq. (2). To achieve a faster optimization, Liu et al. suggested a new algorithm named local anchor embedding, which employs Nesterovs method [28] to accelerate the gradient descent. More details can be found in [21].

Based on the local weights $\mathbf{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_n\}^{\mathrm{T}} \in \mathbb{R}^{n\times m}$, the adjacency matrix between datapoints can be designed as

$$\mathbf{W} = \mathbf{Z}\Lambda^{-1}\mathbf{Z}^{\mathrm{T}}, \tag{3}$$

where the diagonal matrix Λ is defined as $\Lambda_{kk} = \sum_{i=1}^{n} Z_{ik}$, $k = 1, \ldots, m$. From Eq. (3), we can see that if two points are correlative ($W_{ij} > 0$), they share at least one common anchor, and otherwise $W_{ij} = 0$. It is likely that datapoints sharing common anchors have similar semantic concepts.

Let $\mathbf{Y}_l = [\mathbf{y}_1{}^{\mathrm{T}}, \mathbf{y}_2{}^{\mathrm{T}}, \ldots, \mathbf{y}_l{}^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{l\times c}$ denote a class indicator matrix on labeled samples, with $Y_{ij} = 1$ if $\mathbf{x}_i$ belongs to class j and $Y_{ij} = 0$ otherwise. Let $\mathbf{A} = [\mathbf{a}_1{}^{\mathrm{T}}, \mathbf{a}_2{}^{\mathrm{T}}, \ldots, \mathbf{a}_m{}^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{m\times c}$ denote the prediction label matrix on the anchor set. Anchor graph regularization can be naturally formulated to deal with the standard multi-class SSL problem as follows:

$$\begin{aligned} Q(\mathbf{A}) &= \sum_{i=1}^{l} \|\mathbf{z}_i^{\mathrm{T}}\mathbf{A} - \mathbf{y}_i\|^2 + \frac{\gamma}{2}\sum_{i,j=1}^{n} W_{ij}\|\mathbf{z}_i^{\mathrm{T}}\mathbf{A} - \mathbf{z}_j^{\mathrm{T}}\mathbf{A}\|^2 \\ &= \|\mathbf{Z}_l\mathbf{A} - \mathbf{Y}_l\|_{\mathrm{F}}^2 + \gamma\mathrm{tr}(\mathbf{A}^{\mathrm{T}}\mathbf{Z}^{\mathrm{T}}(\mathbf{I} - \mathbf{W})\mathbf{Z}\mathbf{A}) \\ &= \|\mathbf{Z}_l\mathbf{A} - \mathbf{Y}_l\|_{\mathrm{F}}^2 + \gamma\mathrm{tr}(\mathbf{A}^{\mathrm{T}}\tilde{\mathbf{L}}\mathbf{A}), \end{aligned} \tag{4}$$

where $\tilde{\mathbf{L}} = \mathbf{Z}^{\mathrm{T}}(\mathbf{I} - \mathbf{W})\mathbf{Z} \in \mathbb{R}^{m\times m}$ is the reduced Laplacian, $\mathbf{Z}_l \in \mathbb{R}^{l\times m}$ is the sub-matrix according to the labeled part of local weight matrix $\mathbf{Z}$, and $\gamma > 0$ is a trade-off parameter.

From the above equation, we can see that, although AGR is performed with a regularization on all datapoints, it is equivalent to a regularization on anchors with a graph Laplacian $\tilde{\mathbf{L}}$. This is understandable, as the labels of other datapoints are actually inferred from anchors.

Then, the optimal $\mathbf{A}$ can be computed as follows:

$$\mathbf{A} = (\mathbf{Z}_l^{\mathrm{T}}\mathbf{Z}_l + \gamma\tilde{\mathbf{L}})^{-1}\mathbf{Z}_l^{\mathrm{T}}\mathbf{Y}_l. \tag{5}$$

Finally, we can employ the solved labels associated with anchors to predict the hard label for unlabeled samples as

$$\widehat{y}_i = \text{argmax}_{j\in\{1,\ldots,c\}} \frac{\mathbf{Z}_{i\cdot} \times \mathbf{A}_{\cdot j}}{\lambda_j}, i = l+1, \ldots, n, \tag{6}$$

where $\mathbf{Z}_{i\cdot} \in \mathbb{R}^{1\times m}$ denotes the ith row of $\mathbf{Z}$, $\mathbf{A}_{\cdot j} \in \mathbb{R}^{m\times 1}$ is the jth column of $\mathbf{A}$, and the normalization factor $\lambda_j = \mathbf{1}^{\mathrm{T}}\mathbf{Z}\mathbf{A}_{\cdot j}$, suggested as a useful normalization strategy in [54], balances skewed class distributions.

3.2 Analysis of Anchor Graph Regularization

The AGR model has been widely used in many applications for its capability in dealing with relatively large datasets. However, the approach has limitations in the local weight estimation and adjacency matrix design, which are analyzed below.

The first limitation comes from the LAE method for local weight estimation. We demonstrate this fact by a toy example. Fig. 1 illustrates a toy example in a 3D space, in which the LAE method attempts to reconstruct the datapoint $\mathbf{x}_1$ by anchors $\mathbf{u}_1$, $\mathbf{u}_2$, $\mathbf{u}_3$ and minimizes $\|\mathbf{x}_1' - \mathbf{x}_1\|_2$, where $\mathbf{x}_1'$ is a reconstructed datapoint. Usually, to follow the shift-invariant and nonnegative weight requirements, we introduce two constraints $\mathbf{1}^{\mathrm{T}}\mathbf{z}_{\langle i\rangle} = 1$ and $\mathbf{z}_{\langle i\rangle} \succeq 0$ into the geometry reconstruction problem, as in Eq. (2). We put $\mathbf{x}_1$ at the origin of the coordinate according to the shift-invariant constraint. Under these constraints, all feasible reconstructed datapoints are limited in the region enclosed by these anchors, such as simplex $\boldsymbol{\alpha}$, as shown in Fig. 1a. It means that for $\mathbf{x}_1$, the value of $\|\mathbf{x}_1' - \mathbf{x}_1\|_2$ is at least the distance from $\mathbf{x}_1$ to $\boldsymbol{\alpha}$, i.e., the length of the blue line segment. Therefore, the best reconstructed point $\mathbf{x}_1^*$, following Eq. (2), is the crossover point as shown in this figure. This point $\mathbf{x}_1^*$ is on the boundary of the closed region $\boldsymbol{\alpha}$ and is linearly reconstructed by merely $\mathbf{u}_1$ and $\mathbf{u}_2$, e.g., $0.5 \times \mathbf{u}_1 + 0.5 \times \mathbf{u}_2$. This is to say, the local weight between $\mathbf{x}_1$ and $\mathbf{u}_3$ is zero. In addition, if we set up a plane along the nearest boundary and make it perpendicular to $\boldsymbol{\alpha}$, namely $\boldsymbol{\beta}$, and then change the positions of these anchors like Fig. 1b, we can see that this zero weight will not change as long as $\mathbf{u}_3$ and $\mathbf{x}_1$ are at different sides of the plane $\boldsymbol{\beta}$, even if $\mathbf{u}_3$ is closer to $\mathbf{x}_1$ than $\mathbf{u}_1$ and $\mathbf{u}_2$, because the best reconstructed

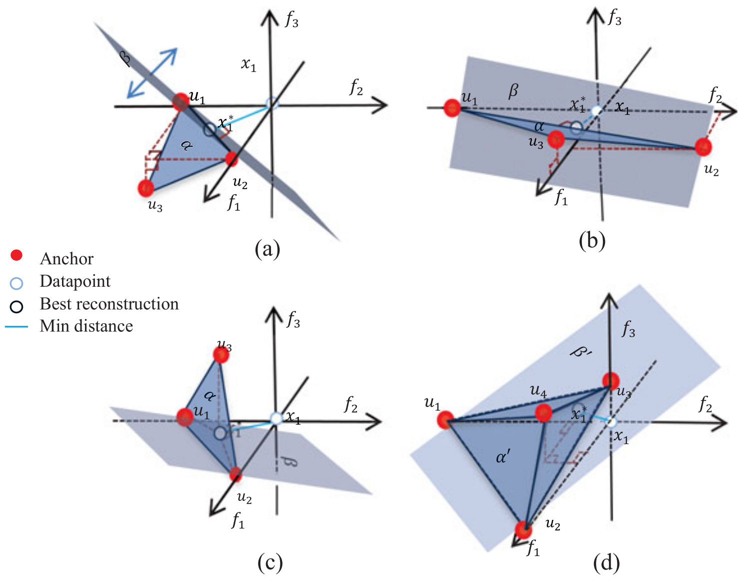


Fig. 1. 3-D toy example for datapoint reconstruction with anchors.

point $\mathbf{x}_1^*$ is still on the boundary. As shown in Fig. 1c, only when $\mathbf{u}_3$ and $\mathbf{x}_1$ are at the same side of $\boldsymbol{\beta}$, $\mathbf{x}_1^*$ will move inside the simplex $\boldsymbol{\alpha}$, and the previous zero weight can change to a positive value. We have discussed the situation above where $s = 3$. However, this geometric interpretation can also be easily extended to the case where $s > 3$. From Fig. 1d, we can see the difference is that the region enclosed by anchors now becomes a closed space, i.e., $\boldsymbol{\alpha}'$, and $\boldsymbol{\beta}'$ now lies along the boundary simplex of $\boldsymbol{\alpha}'$ which is closest to $\mathbf{x}_1$. Similarly, from this figure, we can see that the local weight between $\mathbf{x}_1$ and the anchor, which is on the opposite space of $\boldsymbol{\beta}'$, namely $\mathbf{u}_4$, is always zero. In addition to the above problem, computational cost can also be a disadvantage of LAE, despite several strategies have been applied in [21] to speed up the process.

The second limitation is the adjacency matrix design. After describing the local connection between datapoints and their neighboring anchors, we consider building the adjacency relationship in the whole data space for graph regularization. For instance, we obtain a part of local weights as listed in Fig. 2a. To view these values graphically, we also use a toy example in Fig. 2b to show the relationship between these points, where the length of the edge represents the Euclidean distance between the datapoints and anchors. Now, if we calculate the adjacency weight according to Eq. (3) with these local weights, we have $W_{12} = \sum_{k=1}^{m} \frac{Z_{1k}Z_{2k}}{\Lambda_{kk}} = \frac{0.3\times0.4}{\Lambda_{11}} + \frac{0.4\times0.3}{\Lambda_{22}} + \frac{0.3\times0.3}{\Lambda_{33}}$ and $W_{34} = \sum_{k=1}^{m} \frac{Z_{3k}Z_{4k}}{\Lambda_{kk}} = \frac{0.1\times0.1}{\Lambda_{11}} + \frac{0.1\times0.1}{\Lambda_{22}} + \frac{0.8\times0.8}{\Lambda_{33}}$. Further, if we suppose that Λ_{kk} is nearly the same in homogeneous regions, such as in Fig. 2b, we can obtain $W_{12} = \frac{0.33}{\Lambda_{kk}}$ and $W_{34} = \frac{0.66}{\Lambda_{kk}}$. In this context, the adjacency weights of $\mathbf{x}_1\&\mathbf{x}_2$ and $\mathbf{x}_3\&\mathbf{x}_4$ can be numerically quite different, although the Euclidean distances between them are nearly the same. Therefore, this issue is likely to introduce mistakes in the remaining steps of graph-based SSL tasks.

4 Efficient Anchor Graph Regularization

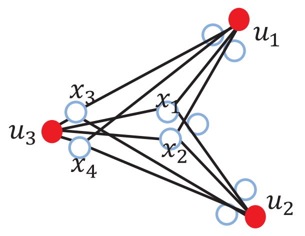
To address the above issues in AGR, we accordingly propose two improvements. First, we introduce a fast local anchor embedding method in anchor graph construction, which reformulates the local weight estimation problem to better measure $\mathbf{Z}$ and speeds up optimization. Second, we directly design a normalized graph Laplacian over anchors and show that it is more effective than the reduced graph Laplacian in AGR. More details are given in the following.

4.1 Fast Local Anchor Embedding (FLAE)

Apart from LAE described above, there exist other similar methods for local weight estimation like LLE [31], LLC [39], and LLA [41]. However, these methods either do not enforce weights to be non-negative or impose the non-negative constraint into objective function via inequality. In the former cases, non-negative similarity measures cannot be guaranteed. In the latter cases, limitation still exists when datapoints are outside of the convex envelope of anchors, according to the analysis in Section 3.2.

$\mathbf{Z}$	$\mathbf{u}_1$	$\mathbf{u}_2$	$\mathbf{u}_3$
$\mathbf{x}_1$	0.4	0.3	0.3
$\mathbf{x}_2$	0.3	0.4	0.3
$\mathbf{x}_3$	0.1	0.1	0.8
$\mathbf{x}_4$	0.1	0.1	0.8

(a) a part of local weight matrix



(b) local edges for these points

Fig. 2. Example of the adjacency modeling.

Therefore, here we aim to design a better local weight matrix $\mathbf{Z}$ to tackle these problems. We call our new weight estimation method Fast Local Anchor Embedding as we will demonstrate that the optimization problem has an analytical solution and can be implemented fast. It is worth mentioning here that, although we use a close name, the formulation of FLAE and its solution are actually quite different from the conventional LAE method. In fact, we have made two changes.

Change 1. Instead of the inequality constraint, we use absolute constraint in geometric reconstruction. Since the non-negative property in similar measurement is important to guarantee the global optimum of graph-based SSL [21], inequality constraint has been employed in most local weight estimation methods, e.g., LAE and LLA. However, as illustrated in Section 3.2, it would introduce additional mistakes when the datapoint is outside of convex envelope of anchors. Therefore, we integrate the non-negative property from another perspective: absolute operator. To this end, we set $\mathbf{z}_{\langle i\rangle} = |\mathbf{c}_{\langle i\rangle}|$ and the constraint $\mathbf{1}^{\mathrm{T}}|\mathbf{c}_{\langle i\rangle}| = 1$ is imposed to follow the shift-invariant requirement in our geometric reconstruction problem. Then, we can obtain the local weight vector $\mathbf{z}_{\langle i\rangle}$ for each datapoint $\mathbf{x}_i$, corresponding to the following problem:

$$\underset{\mathbf{c}_i}{\operatorname{argmin}} \|\mathbf{x}_i - \mathbf{U}_{\langle i\rangle}\mathbf{c}_{\langle i\rangle}\|^2 \qquad (7)$$

$$\text{s.t.} \mathbf{1}^{\mathrm{T}}|\mathbf{c}_{\langle i\rangle}| = 1.$$

Compared with LAE, Eq. (7) can be a more direct model to handle the non-negative property in similarity measure. However, since there lacks a straightforward solution of the optimization problem, here we obtain a solution via shrinking the domain of the above problem.

Specifically, we first drop the absolute constraint in our model, which reduces the problem to a simple coding problem as:

$$\underset{\hat{\mathbf{c}}_{\langle i\rangle}}{\operatorname{argmin}} \|\mathbf{x}_i - \mathbf{U}_{\langle i\rangle}\hat{\mathbf{c}}_{\langle i\rangle}\|^2 \qquad (8)$$

$$\text{s.t.} \mathbf{1}^{\mathrm{T}}\hat{\mathbf{c}}_{\langle i\rangle} = 1.$$

And the solution can be derived analytically by:

$$\tilde{\mathbf{c}}_{\langle i\rangle} = \mathbf{C}_i^{-1}\mathbf{1}, \qquad (9)$$

$$\hat{\mathbf{c}}_{\langle i\rangle} = \tilde{\mathbf{c}}_{\langle i\rangle}/\mathbf{1}^{\mathrm{T}}\tilde{\mathbf{c}}_{\langle i\rangle}, \qquad (10)$$

where $\mathbf{C}_i = (\mathbf{U}_{\langle i\rangle}^{\mathrm{T}} - \mathbf{1}\mathbf{x}_i{}^{\mathrm{T}})(\mathbf{U}_{\langle i\rangle}^{\mathrm{T}} - \mathbf{1}\mathbf{x}_i{}^{\mathrm{T}})^{\mathrm{T}} \in \mathbb{R}^{s\times s}$ is a data covariance matrix.

Then, we compute our local weight vector $\mathbf{z}_{\langle i\rangle}$ after obtaining the code $\hat{\mathbf{c}}_{\langle i\rangle}$ as follows:

$$\rho = \mathbf{1}^{\mathrm{T}}|\hat{\mathbf{c}}_{\langle i\rangle}| \qquad (11)$$

$$\mathbf{c}_{\langle i\rangle} = \hat{\mathbf{c}}_{\langle i\rangle}/\rho, \qquad (12)$$

$$\mathbf{z}_{\langle i\rangle} = |\mathbf{c}_{\langle i\rangle}|. \qquad (13)$$

As we can see, the above solution $\mathbf{c}_{\langle i\rangle}$ satisfies the constraint $\mathbf{1}^{\mathrm{T}}|\mathbf{c}_{\langle i\rangle}| = 1$, it means this $\mathbf{c}_{\langle i\rangle}$ is a feasible solution of Eq. (7). Meanwhile, we can have the following conclusion.

Proposition 1. *The minimum of Eq. (8) will not greater than the minimum of Eq. (2).*

We leave the proof of the proposition to appendix.

Then, our task is to demonstrate that, with the above solution, the value of the objective function in Eq. (7) is not greater than the value of Eq. (8) with its optimal solution $\hat{\mathbf{c}}_{\langle i\rangle}$. If this conclusion can be drawn, it means that our method can lead to a smaller reconstruction error than LAE, and the effectiveness of our approach can be validated. The details are as follows.

Recall that, by solving Eq. (8), we obtain the optimal solution $\hat{\mathbf{c}}_{\langle i\rangle}$. Clearly, there are two possible cases regarding the obtained codes $\hat{c}_{ij} \in \hat{\mathbf{c}}_{\langle i\rangle}$: (1) $\forall\, \hat{c}_{ij} \geq 0$; and (2) $\exists\, \hat{c}_{ij} < 0$. Then, we follow Eqs. (11), (12) to yield codes $\mathbf{c}_{\langle i\rangle}$.

For the first case, we obtain $\mathbf{c}_{\langle i\rangle} = \hat{\mathbf{c}}_{\langle i\rangle}$. It means that the value of the objective function in Eq. (7) with our feasible solution is the same with the value of Eq. (8) with its optimal solution.

We then mainly focus on the second case. Since the obtained code $\hat{\mathbf{c}}_{\langle i\rangle}$ satisfies the constraint $\mathbf{1}^{\mathrm{T}}\hat{\mathbf{c}}_{\langle i\rangle} = 1$, we first substitute it into the objective function in Eq. (8) to yield the minimum of Eq. (8) as

$$\|\mathbf{x}_{\langle i\rangle}\mathbf{1}^{\mathrm{T}}\hat{\mathbf{c}}_{\langle i\rangle} - \mathbf{U}_{\langle i\rangle}\hat{\mathbf{c}}_{\langle i\rangle}\|^2 = \|\tilde{\mathbf{U}}_{\langle i\rangle}\hat{\mathbf{c}}_{\langle i\rangle}\|^2, \qquad (14)$$

where $\tilde{\mathbf{U}}_{\langle i\rangle} = [\mathbf{U}_{\langle i\rangle}(:,1) - \mathbf{x}_i, \ldots, \mathbf{U}_{\langle i\rangle}(:,s) - \mathbf{x}_i]$ can be viewed as the new coordinates of s closest anchors of $\mathbf{x}_i$ in the datapoint-centered coordinate system.

Then, according to Eqs. (11), (12), we scale the code $\hat{\mathbf{c}}_{\langle i\rangle}$ in Eq. (14) by a constant ρ and obtain the value of the objective function in Eq. (7) with our feasible solution $\mathbf{c}_{\langle i\rangle}$,

$$\|\tilde{\mathbf{U}}_{\langle i\rangle}\mathbf{c}_{\langle i\rangle}\|^2 = \left\|\frac{1}{\rho}\tilde{\mathbf{U}}_{\langle i\rangle}\hat{\mathbf{c}}_{\langle i\rangle}\right\|^2 = \frac{1}{\rho^2}\|\tilde{\mathbf{U}}_{\langle i\rangle}\hat{\mathbf{c}}_{\langle i\rangle}\|^2. \qquad (15)$$

Since $\exists\, \hat{c}_{ij} < 0$, we have $\rho > 1$. Therefore, we obtain the inequality relation that the value of Eq. (15) is smaller than Eq. (14).

Up to now, we have demonstrated that the value of the objective function in Eq. (7) with our feasible solution is not greater than the value of Eq. (8) with its optimal solution, which means we can obtain a smaller reconstruction error than LAE. Moreover, our analytical solution based on Eqs. (9)-(13) is much faster than the iterative solution obtained by LAE.

Change 2. We further incorporate the locality constraint into local weight estimation. To enhance coding efficiency, the locality constraint functions in other similar methods [21], [41] are replaced by using s closest anchors (landmarks), like approximated LLC [39] does. However, this manipulation is insufficiently suitable, because it ignores the real distance between local points (an negative example is shown in Fig. 1b). We therefore suggest keeping the locality constraint while using s closest anchors in local weight estimation simultaneously.

We obtain local weight vector $\mathbf{z}_{\langle i\rangle} = |\mathbf{c}_{\langle i\rangle}|$ according to the following objective:

$$\underset{\mathbf{c}_i}{\operatorname{argmin}} \|\mathbf{x}_i - \mathbf{U}_{\langle i\rangle}\mathbf{c}_{\langle i\rangle}\|^2 + \lambda\|\mathbf{d}_{\langle i\rangle} \odot \mathbf{c}_{\langle i\rangle}\|^2 \qquad (16)$$

$$\text{s.t.}\mathbf{1}^{\mathrm{T}}|\mathbf{c}_{\langle i\rangle}| = 1,$$

where $\odot$ denotes the element-wise multiplication. $\mathbf{d}_{\langle i\rangle}$ is the locality adaptor that allows a different freedom for each anchor proportional to its distance to the datapoint $\mathbf{x}_i$. Specifically,

$$\mathbf{d}_{\langle i\rangle}(k) = \exp\left(\frac{\|\mathbf{x}_i - \mathbf{U}_{\langle i\rangle}(:,k)\|^2}{\sigma}\right), \quad k = 1,\ldots,s \qquad (17)$$

where the parameter σ is used to adjust the weight decay speed for the locality adaptor.

Similarly, we can obtain a feasible solution as before. The only change compared to the pervious processes is that we replace Eq. (9) with

$$\tilde{\mathbf{c}}_{\langle i\rangle} = (\mathbf{C}_i + \lambda\,\mathrm{diag}(\mathbf{d}_{\langle i\rangle}))^{-1}\mathbf{1}. \qquad (18)$$

To summarize, we demonstrate the steps of FLAE in Algorithm 1.

Algorithm 1. Fast Local Anchor Embedding (FLAE)

```
Input: datapoints $\{\mathbf{x}_i\}_{i=1}^n \in \mathbb{R}^{d\times 1}$, anchor set obtained from K-means $\mathbf{U} \in \mathbb{R}^{d\times m}$, parameters $s$ and $\lambda$.
for $i = 1$ to $n$ do
    1. For $\mathbf{x}_i$, find $s$ nearest neighbors in $\mathbf{U}$ and record the index set $\langle i\rangle$.
    2. Measure the locality adaptor $\mathbf{d}_{\langle i\rangle}$ for datapoint $\mathbf{x}_i$ with its nearest neighbors $\mathbf{U}_{\langle i\rangle}$ via Eq. (17).
    3. Compute the code $\mathbf{c}_{\langle i\rangle}$ via Eq. (18), and Eqs. (10)-(12).
    4. Obtain the final solution $\mathbf{z}_{\langle i\rangle}$ via Eq. (13).
    5. $Z_{i,\langle i\rangle} = \mathbf{z}_{\langle i\rangle}^{\mathrm{T}}$.
end for
Output: FLAE matrix $\mathbf{Z}$ .
```

4.2 Normalized Graph Laplacian over Anchors

We consider a graph as a stable one if it satisfies the cluster assumption, which means nearby datapoints are likely to have the same labels. Obviously, it is important for building a well performed anchor-graph-based SSL classification model. In Section 3.2, we have observed the limitation of the anchor graph built based on the conventional adjacency between datapoints. As demonstrated in Section 3.1, in anchor graph regularization, the regularization on all datapoints is actually equivalent to regularization on anchors with a reduced graph Laplacian, which is built over only anchors. Therefore, here we directly design an adjacency matrix among anchors and then derive a normalized graph Laplacian over anchors. For clarity, we use the subscripts i, j, k and s, t, r to denote the indices of datapoints and anchors respectively in this section. The details are as follows.

Recall that the label for each datapoint is estimated as a weighted average of the labels of the anchor set via Eq. (1), which only involves the local weight vector of the datapoint

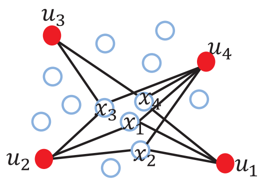


Fig. 3. Toy example of the nearest anchors of datapoints.

and the labels of its nearest anchors. Given local weights, the label vectors of the nearest anchors are crucial to the final label prediction for the datapoint. If two nearby datapoints share a lot of nearest anchors, their labels are likely to be similar. However, nearby datapoints are not guaranteed to have identical nearest anchors quite often. Fig. 3 illustrates a toy example, where the nearest anchors of nearby pairwise datapoints x_1 and x_2 are the same while those of x_1 and x_3 are not. We prefer the Laplacian matrix has the following characteristics: 1) the elements corresponding to the nearby pairwise anchors should be negative, which means these anchors tend to have similar labels; 2) the elements corresponding to dissimilar pairwise anchors should be zero, which means their labels are irrelevant [27]. To this end, we design the adjacency matrix between anchors as

$$\mathbf{W} = \mathbf{Z}^{\mathrm{T}}\mathbf{Z}, \qquad (19)$$

where $W_{st} = \sum_{k=1}^{n} Z_{ks}Z_{kt}$. Accordingly, its normalized Laplacian matrix is

$$\bar{\mathbf{L}} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}, \qquad (20)$$

where the diagonal matrix $\mathbf{D}$ is defined as $D_{ss} = \sum_{t=1}^{m} W_{st}$. Note that our adjacency matrix actually explores all datapoints as the transitional points rather than anchors alone. Thus, our model keeps the computational efficiency of the anchor graph regularization and its effectiveness in regularization.

As for AGR, it constructs an $n \times n$ adjacency matrix as $\mathbf{W} = \mathbf{Z}\Lambda^{-1}\mathbf{Z}^{\mathrm{T}}$. Based on this, AGR employs the following reduced Laplacian matrix $\tilde{\mathbf{L}}$ over anchors for its regularization function

$$\begin{aligned}\tilde{\mathbf{L}} &= \mathbf{Z}^{\mathrm{T}}(\mathbf{I} - \mathbf{Z}\Lambda^{-1}\mathbf{Z}^{\mathrm{T}})\mathbf{Z}, \qquad (21)\\ &= \mathbf{Z}^{\mathrm{T}}\mathbf{Z} - \mathbf{Z}^{\mathrm{T}}(\mathbf{Z}\Lambda^{-1}\mathbf{Z}^{\mathrm{T}})\mathbf{Z}.\end{aligned}$$

Now we compare the two $m \times m$ Laplacian matrices over anchors, according to the previously mentioned characteristics. For convenience, we take a non-diagonal element L_{st} for example, which is computed as

$$L_{st} = \begin{cases} -\alpha\mathbf{Z}_{\cdot s}^{\mathrm{T}}\mathbf{Z}_{\cdot t} & \text{for EAGR,} \\ \mathbf{Z}_{\cdot s}^{\mathrm{T}}\mathbf{Z}_{\cdot t} - \mathbf{Z}_{\cdot s}^{\mathrm{T}}(\mathbf{Z}\Lambda^{-1}\mathbf{Z}^{\mathrm{T}})\mathbf{Z}_{\cdot t} & \text{for AGR ,} \end{cases} \qquad (22)$$

where $\alpha = (D_{ss}D_{tt})^{-\frac{1}{2}}$, $\mathbf{Z}_{\cdot s} \in \mathbb{R}^{n\times 1}$ denotes the sth col of $\mathbf{Z}$.

We discuss the sign of L_{st} in the following two cases according to the relations of the pairwise anchors $\mathbf{u}_s\&\mathbf{u}_t$.

(1) The first case is that $\mathbf{u}_s\&\mathbf{u}_t$ share at least one common datapoint $\mathbf{x}_i$, namely, they are nearby. For EAGR, we clearly have $L_{st} = -\alpha\mathbf{Z}_{\cdot s}^{\mathrm{T}}\mathbf{Z}_{\cdot t} \leq -\alpha Z_{is}Z_{it} < 0$.

However, for AGR, the sign of the element L_{st} depends on the values of $\mathbf{Z}_{\cdot s}^{\mathrm{T}}\mathbf{Z}_{\cdot t}$ and $\mathbf{Z}_{\cdot s}^{\mathrm{T}}(\mathbf{Z}\Lambda^{-1}\mathbf{Z}^{\mathrm{T}})\mathbf{Z}_{\cdot t}$. Therefore, the element L_{st} can be positive, zero, or negative. If $L_{st} > 0$, these nearby anchors tend to have different labels. If $L_{st} = 0$, these nearby anchors tend to be irrelevant. When $L_{st} < 0$, we can obtain the expected result, that is, nearby anchors tend to have similar labels. In short, the labels of nearby pairwise anchors in EAGR are enforced to be similar, while this is uncertain for AGR.

(2) The second case is that $\mathbf{u}_s \& \mathbf{u}_t$ does not share any common datapoint, namely, they are dissimilar. For EAGR, we have $L_{st} = -\alpha\mathbf{Z}_{\cdot s}^{\mathrm{T}}\mathbf{Z}_{\cdot t} = 0$. However, for AGR, although $\mathbf{Z}_{\cdot s}^{\mathrm{T}}\mathbf{Z}_{\cdot t}$ is zero, $\mathbf{Z}_{\cdot s}^{\mathrm{T}}(\mathbf{Z}\Lambda^{-1}\mathbf{Z}^{\mathrm{T}})\mathbf{Z}_{\cdot t}$ may still be positive, which makes L_{st} negative. For example, anchors $\mathbf{u}_s \& \mathbf{u}_t$ connect with two different datapoints $\mathbf{x}_i \& \mathbf{x}_j$ respectively, and these datapoints share another anchor $\mathbf{u}_r$ simultaneously. Then, we obtain $L_{st} < 0$, since $L_{st} = 0 - \mathbf{Z}_{\cdot s}^{\mathrm{T}}(\mathbf{Z}\Lambda^{-1}\mathbf{Z}^{\mathrm{T}})\mathbf{Z}_{\cdot t} < 0 - Z_{is} \quad (\mathbf{Z}\Lambda^{-1}\mathbf{Z}^{\mathrm{T}})_{ij} Z_{jt}$, where $(\mathbf{Z}\Lambda^{-1}\mathbf{Z}^{\mathrm{T}})_{ij} \geq Z_{ir}\Lambda_{rr}^{-1}Z_{jr} > 0$ and $Z_{is} > 0$, $Z_{jt} > 0$. Therefore, unexpected negative Laplacian weights could exist between dissimilar anchors in AGR model while our EAGR is free from this situation.

Later in Section 5.2, based on real-world datasets, we will show several examples by comparing the number of non-zero elements at the non-diagonal positions of the above $m \times m$ Laplacian matrices. In conclusion, based on the proposed $\mathbf{W}$, we can better describe the adjacency between anchors and make the smoothness constraint more effective for the stable anchor graph construction. As the common linked datapoints are used as transitional points in building $\mathbf{W}$, we note that our adjacency matrix $\mathbf{W}$ is totally different from the simple kNN graph, which only depends on the sparse anchors themselves.

4.3 Learning on Anchor Graph

Now we consider a standard multi-class SSL task. Given a set of labeled datapoints $\mathbf{x}_i$ $(i = 1, \ldots, l)$ with the corresponding discrete label $y_i \in \{1, \ldots, c\}$, our goal is to predict the labels on the remaining unlabeled real datapoints associated with anchors. Let $\mathbf{Y}_l = [\mathbf{y}_1{}^{\mathrm{T}}, \mathbf{y}_2{}^{\mathrm{T}}, \ldots, \mathbf{y}_l{}^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{l\times c}$ denote a class indicator matrix on labeled datapoints with $Y_{ij} = 1$ if $\mathbf{x}_i$ belongs to class j and $Y_{ij} = 0$ otherwise. Suppose that, for each datapoint, we obtain a label prediction function $f_i = \mathbf{z}_i^{\mathrm{T}}\mathbf{A}$ where $\mathbf{A} = [\mathbf{a}_1{}^{\mathrm{T}}, \mathbf{a}_2{}^{\mathrm{T}}, \ldots, \mathbf{a}_m{}^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{m\times c}$ denotes the prediction label matrix on the anchor set. Specifically, $\mathbf{a}_i \in \mathbb{R}^{1\times c}$ is the label vector of the anchor $\mathbf{u}_i$. Then, we combine anchors with datapoints to build a model called Efficient Anchor-Graph Regularization (EAGR) for SSL as follows:

$$\underset{\mathbf{A}}{\operatorname{argmin}} \sum_{i=1}^{l} \|\mathbf{z}_i^{\mathrm{T}}\mathbf{A} - \mathbf{y}_i\|^2 + \frac{\gamma}{2}\sum_{i,j=1}^{m} W_{ij}\left\|\frac{\mathbf{a}_i}{\sqrt{D_{ii}}} - \frac{\mathbf{a}_j}{\sqrt{D_{jj}}}\right\|^2. \quad (23)$$

We can also present this optimization problem in the following matrix form:

$$\underset{\mathbf{A}}{\operatorname{argmin}} \|\mathbf{Z}_l\mathbf{A} - \mathbf{Y}_l\|_{\mathrm{F}}^2 + \gamma \operatorname{tr}(\mathbf{A}^{\mathrm{T}}\bar{\mathbf{L}}\mathbf{A}), \quad (24)$$

where $\bar{\mathbf{L}} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2} \in \mathbb{R}^{m\times m}$ is the previously introduced normalized Laplacian matrix, $\|\cdot\|_{\mathrm{F}}$ stands for the Frobenius norm, and $\gamma > 0$ is the trade-off parameter.

With simple algebra, we can obtain a globally optimal solution for the anchors' label matrix as follows:

$$\mathbf{A} = (\mathbf{Z}_l^{\mathrm{T}}\mathbf{Z}_l + \gamma\bar{\mathbf{L}})^{-1}\mathbf{Z}_l^{\mathrm{T}}\mathbf{Y}_l. \quad (25)$$

This yields a closed-form solution for handling large scale SSL tasks. Lastly, like AGR, we utilize the local weight matrix with anchors' soft scores to predict the hard labels for unlabeled datapoints as Eq. (6).

To summarize, our EAGR consists of the following steps: (1) finding anchors by K-means; (2) estimating the local weight matrix $\mathbf{Z}$ by FLAE, as illustrated in Algorithm 1; (3) computing the normalized Laplacian matrix $\bar{\mathbf{L}}$ over anchors via Eq. (20); (4) carrying out the graph regularization via Eq. (25); and (5) predicting the hard labels on unlabeled datapoints via Eq. (6). As we can see, our EAGR keeps the simplicity of AGR and tends to be much faster, since it efficiently solves the local weight estimation with FLAE. To be clear, Tables 1 and 2 show the storage costs and time complexity of several semi-supervised learning algorithms, where n is the number of datapoints, m is the number of anchors, s is the number of the closest anchors to a datapoint, d is the dimension of features, and t is the number of iterations in the corresponding iterative optimization process.

TABLE 1
Comparison of Storage Costs of Three Graph-Based Methods

Approach	Storage
Learning with Local and Global Consistency (LLGC) [51]	$O(n^2)$
Anchor Graph Regularization (AGR) [21]	$O(mn)$
Efficient Anchor Graph Regularization (EAGR)	$O(mn)$

TABLE 2
Comparison of Computational Complexities of Three Graph-Based Methods

Approach	Find anchors	Design $\mathbf{Z}$	(reduced) graph Laplacian $\mathbf{L}$	Graph Regularization
LLGC	_	_	$O(dn^2)$	$O(n^3)$
AGR	$O(mndt)$	$O(mns + s^2dnt)$	$O(m^2n)$	$O(m^2n + m^3)$
EAGR	$O(mndt)$	$O(mns + s^2dn)$	$O(m^2n)$	$O(m^2n + m^3)$

TABLE 3
Details of the Five Datasets

	Alphadigits	Semeion	USPS	Letter	MNIST
# of instances	1,404	1,593	11,000	20,000	60,000
# of categories	36	10	10	26	10
# of features	320	256	256	16	784

5 Experiments

5.1 Data Settings

To evaluate the performance of our EAGR, we conduct experiments on five widely-used handwritten datasets: Binary Alphadigits[1] (Alphadigits for short), USPS[1], MNIST[2], Semeion,[2] and Letter Recognition[2] (Letter for short). Following the settings of [21], we divide them into three groups based on their sizes, that is, number of samples. It is worthwhile to note that the images in Letter have already been converted into 16 primitive numerical attributes (statistical moments and edge counts) to represent each sample. In other datasets, samples are still pixel images of different size. For these datasets, we directly use a high-dimensional vector of normalized grayscale values to represent each instance. The attributes of these datasets are listed in Table 3. All the experiments are implemented on a 2.4 GHz CPU, 32 GB RAM PC.

5.2 On the Two Improvements in EAGR

Before comparing our EAGR with other methods, we conduct two experiments to validate the effectiveness of the two improvements described in Sections 4.1 and 4.2 respectively. For a fair comparison, we define two intermediate versions between AGR and EAGR as:

(1) AGR$+$, which employs FLAE for estimating $\mathbf{Z}$ and use the reduced Laplacian matrix in Eq. (21).

(2) EAGR$-$, which employs LAE for estimating $\mathbf{Z}$ and use the normalized Laplacian matrix in Eq. (20).

The differences of the methods for comparison are also summarized in Table 4. For the involved parameters, we simply set anchor number to 500 and empirically set s to 3 to make the anchor graph sparse. We tune other parameters, i.e., λ and γ, to their optimal values. In this way, we can provide a fair comparison for these algorithms. We randomly select 10 labeled samples per class and leave the remaining ones unlabeled for SSL models in this section.

Table 5 shows the classification accuracies of the four different methods. Table 6 presents the average CPU time (in seconds) of two local estimation methods on different datasets. Meanwhile, Table 7 illustrates the number of nonzero elements at the non-diagonal positions in different Laplacian matrices as mentioned in Sectioin 4.2. From the results, we have two observations as follows.

First, by comparing AGR+ with AGR, we see that the former has comparable or better performances than the latter. In addition, Table 6 reveals that our proposed FLAE in AGR+ is much faster than LAE in AGR. These performances demonstrate the efficiency of our improvement on the local weight estimation.

1. available at http://cs.nyu.edu/~roweis/data.html
2. available at http://archive.ics.uci.edu/ml/

Second, by comparing EAGR- with AGR, we see that, although two methods use the same optimized $\mathbf{Z}$, the EAGR- has better classification performance than AGR on all five datasets. We can see from Table 7 that the normalized graph Laplacian of EAGR- has less non-zero elements than the reduced graph Laplacian of AGR. This means that, with a sparse normalized graph Laplacian, several incorrect links have been removed.

5.3 Comparison with Other Methods

Here we further compare the proposed EAGR approach with the following methods.

(1) FLAE-based label inference, which constructs the local matrix between unlabeled datapoints and the labeled ones with FLAE, and then predicts labels for unlabeled datapoints. This method actually introduces FLAE into the kernel regression approach [3], since each row vector $\mathbf{z}_i$ of $\mathbf{Z}$ is non-negative and normalized. The method is denoted as "FLAE-LI".

(2) The Eigenfunctions method introduced in [11], which solves the semi-supervised problem in a dimension-reduced feature space by only working with a small number of eigenvectors of the Laplacian. The method is denoted as "Eigenfucntion".

(3) Laplacian Support Vector Machines Trained in the Primal with preconditioned conjugate gradient, which is introduced in [26]. We first decompose a c-class classification problem into c one-versus-rest binary classification problems. Prediction is then made by assigning the sample to the class with the highest decision function value. The method is denoted as "LapSVMp".

(4) Learning with local and global consistency in [51], which is a typical graph-based learning method. It directly computes the affinity matrix with Gaussian kernel. In our experiments, we use the 6NN strategy in graph construction and use Gaussian kernel for weighting the edges. The method is denoted as "LLGC".

(5) Prototype Vector Machines with the Square Loss in [47], [48], which is a scale-up graph-based semi-supervised learning for multi-label classification tasks using a set of sparse prototypes derived from the data and the Gaussian kernel is used to define the graph affinity. The method is denoted as "PVM".

(6) Anchor Graph Regularization, which employs clustering centers as anchors with LAE, is proposed in [21]. It is the prime counterpart in our experiments, and we aim to improve its performance. The method is denoted as "AGR".

Since the last two methods and EAGR are based on either the anchors or the prototypes, we group them into "landmark-based learning" methods and perform K-means to obtain these cluster centers as the landmarks. Aiming at a fair comparison, the radius parameters of the Gaussian kernel in the methods (2-4) are set by five-fold cross-validation. The trade-off parameters in regularization of the above methods are empirically tuned to their optimal values.

5.3.1 Small Size Datasets

We first conduct experiments on two small datasets: Alphadigits and Semeion. For the three landmark-based methods,

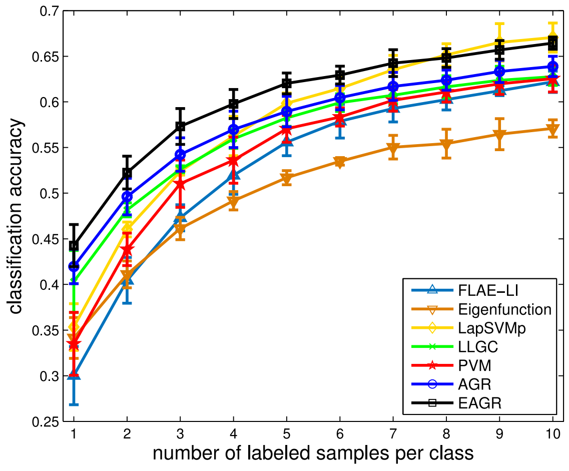
TABLE 4
Comparison of AGR, EAGR, AGR+, and EAGR-

	AGR	AGR+	EAGR-	EAGR
local weight matrix $\mathbf{Z}$	LAE	FLAE	LAE	FLAE
(reduced) graph Laplacian	$\mathbf{Z}^T\mathbf{Z} - \mathbf{Z}^T(\mathbf{Z}\Lambda^{-1}\mathbf{Z}^T)\mathbf{Z}$	$\mathbf{Z}^T\mathbf{Z} - \mathbf{Z}^T(\mathbf{Z}\Lambda^{-1}\mathbf{Z}^T)\mathbf{Z}$	$\mathbf{I} - \mathbf{D}^{-1/2}\mathbf{Z}^T\mathbf{Z}\mathbf{D}^{-1/2}$	$\mathbf{I} - \mathbf{D}^{-1/2}\mathbf{Z}^T\mathbf{Z}\mathbf{D}^{-1/2}$

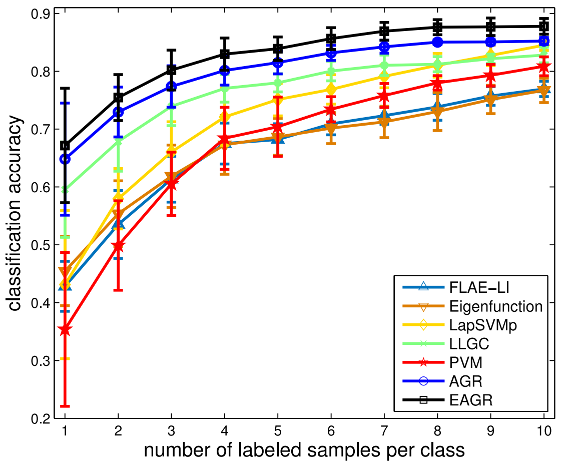
For AGR and AGR+, we demonstrate the reduced graph Laplacian over anchors.

TABLE 5
Comparison on Classification Accuracy (Mean$\pm$std) of Different Anchor-Graph-Based Approaches

	AGR	AGR+	EAGR-	EAGR
Alphadigits	64.43 ± 1.06	65.21 ± 1.32	65.27 ± 0.76	65.92 ± 0.71
Semeion	85.16 ± 1.25	87.05 ± 1.10	85.31 ± 0.92	87.17 ± 1.33
USPS	86.53 ± 1.34	86.21 ± 1.46	87.62 ± 1.20	87.21 ± 1.33
Letter	57.35 ± 0.81	60.40 ± 0.97	58.28 ± 0.97	61.31 ± 1.04
MNIST	86.66 ± 1.14	86.51 ± 1.19	88.54 ± 1.06	88.45 ± 0.94



(a) Alphadigits



(b) Semeion

Fig. 4. Classification accuracy versus the number of labeled samples on small-size datasets.

we empirically set the landmark number to 500 and $l = \{1, 2, \ldots, 10\}$ labeled samples per class. The average classification accuracies with standard deviations over 10 trials are illustrated in Fig. 4. As a general trend, it can be seen that, as the number of labeled data increases, the performances of all methods become better. In addition, we observe that the performances of the LapSVMp, LLGC, PVM, AGR, and EAGR methods stay at a higher level than Eigenfunction. The reason is that the former produce a good complete graph to model the data distribution, while the latter builds a backbone graph only. Specifically, the proposed EAGR method has the similar standard deviations with AGR, and it achieves the best accuracy among all landmark-based SSL methods in most cases, which demonstrates the effectiveness of the improved relationship modeling.

Besides classification accuracies, we record the implementation time costs of the above algorithms with 10 labeled samples per class in Table 8. It is noted that the time costs of K-means clustering in the three landmark-based SSL methods are listed separately in the table and the time excluding clustering is listed at the right side. As can be seen, excluding the time costs of K-means, our EAGR method is faster than AGR and PVM, which demonstrates the efficiency of our improvement among landmark-based methods. Although the total time costs of these landmark-

TABLE 6
Comparison on the Time Costs (Seconds) of Different Local Weight Estimation Approaches

	LAE	FLAE
Alphadigits	2.03	0.23
Semeion	2.34	0.28
USPS	15.81	1.80
Letter	27.35	2.80
MNIST	79.06	11.01

TABLE 7
Number of Non-Zero Elements at the Non-Diagonal Positions in Different Graph Laplacian Matrices

	Alphadigits	Semeion	USPS	Letter	MNIST
AGR	19,580	21,540	189,330	152,556	697,658
EAGR-	3,340	3,540	25,252	24,276	68,172

For AGR, we count the numbers in reduced graph Laplacian matrices over anchors. We can see that the normalized graph Laplacian of EAGR- is more sparse.

TABLE 8
Time Costs (Seconds) of the Compared Learning Algorithms on Small-Size Datasets

	Landmark-based Learning				Others			
	K-means	AGR	EAGR	PVM	FLAE-LI	Eigenfunction	LapSVMp	LLGC
Alphadigits	2.10	+2.17	+0.28	+0.32	0.14	0.95	7.25	0.73
Semeion	2.01	+2.48	+0.30	+0.34	0.13	0.73	1.13	0.83

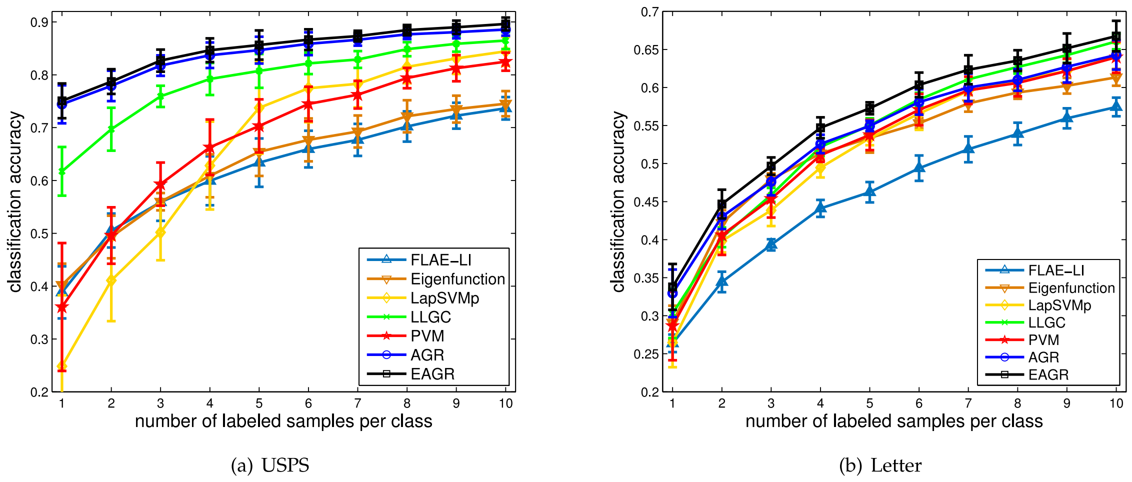


Fig. 5. Classification accuracy versus the number of labeled samples on medium-size datasets.

TABLE 9
Time Costs (Seconds) of the Compared Learning Algorithms on Medium-Size Datasets

	Landmark-Based Learning				Others			
	K-means	AGR	EAGR	PVM	FLAE-LI	Eigenfunction	LapSVMp	LLGC
USPS	59.00	+20.72	+4.45	+16.43	0.69	4.95	56.68	120.67
Letter	11.15	+35.42	+7.16	+25.04	1.14	1.09	139.39	432.73

based methods are larger than LLGC on these small datasets, we will see that LLGC will be quite slow on large datasets. Here we have simply used K-means to obtain cluster centers like [21], as clustering is not the focus of this work. In fact, many fast clustering algorithms can be explored to improve the speed of this process [1], [10].

5.3.2 *Medium Size Datasets*

For the USPS and Letter datasets, we empirically set the landmark number to 2,000 by taking both effectiveness and efficiency into consideration. Averaged over 10 trials, we calculate the classification accuracies with standard deviations for the referred methods. The results of USPS and Letter are shown in Fig. 5. Here we have the following three observations. First, although FLAE-LI produces reasonable results, it has lower accuracy than semi-supervised methods in most cases. This demonstrates the importance of graph construction which utilizes unlabeled samples in regularization. Second, although all the methods perform poorly when the number of labeled samples is small, the two anchor-graph-based methods are clearly superior to LLGC. A possible reason is that the fitting constraints of both AGR and EAGR are built on the labeled samples while the constraints of LLGC are built on the whole set. The fitting effects of AGR and EAGR, therefore, are more biased towards labeled information than that of LLGC. Third, when the number of labeled samples increases, the performances of all methods become better and the accuracy of EAGR improves more significantly than AGR. The reason is that, as the number of labeled samples increases, these classifiers model the characteristics of different classes better. However, this increase also makes the classifiers tend to be overfitted, which needs to be handled by an effective smoothness constraint. Owning to the better relationship modeling in the data space to meet this requirement, EAGR addresses the limitation of AGR as expected.

We also demonstrate the implementation time costs of the above algorithms with 10 labeled samples per class in Table 9. We can observe that, although LapSVMp is faster than LLGC, it is still computationally expensive on the larger data set, e.g., Letter, as its time complexity is quadratic with respect to n. The landmark-based learning algorithms, especially EAGR, need much less implementation costs than LLGC. In addition, the K-means clustering process accounts for the main portion of the EAGR's time cost. Therefore, if we can reduce the clustering time by employing other fast clustering algorithms [1], [10] or just selecting a part of the database samples to run K-means like [43], the advantage of our EAGR over the other two in terms of speed is expected to be greater.

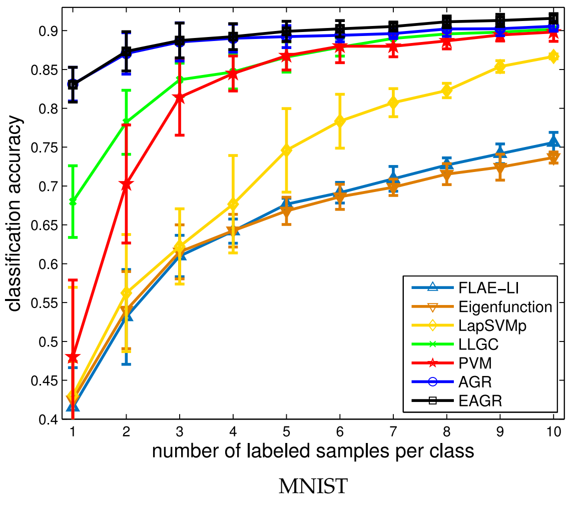


Fig. 6. Classification accuracy versus the number of labeled samples on the large-size dataset.

5.3.3 Large Size Dataset

For the MNIST dataset, we empirically set the landmark number to 2,000 in the three landmark-based methods. Fig. 6 shows the results of the algorithms on MNIST. We can observe that the two anchor-graph-based methods are clearly superior to LLGC when the number of labeled samples is small. Similar to the results in the previous experiments, we see that our method achieves better performance than AGR and PVM. Meanwhile, Table 10 also demonstrates that our EAGR is more advantageous than the AGR and PVM methods in terms of speed for the large dataset.

5.4 On the Parameters λ and γ

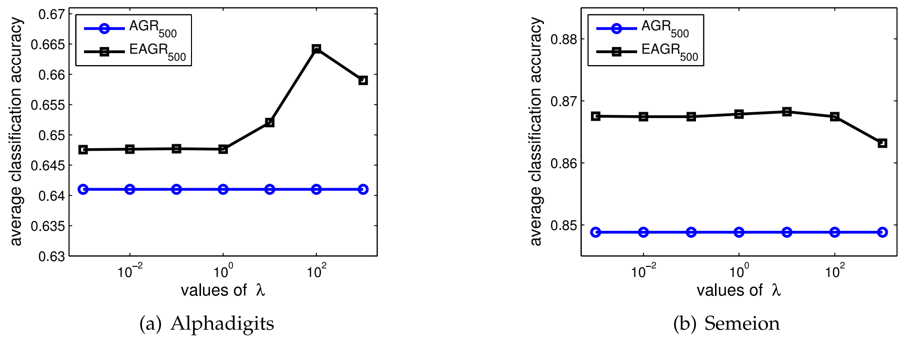
Finally, we test the sensitivity of the two parameters λ and γ involved in the proposed algorithm. For convenience, we simply set anchor number to 500 and empirically set $s = 3$. We first set γ to 1 and vary λ from 0.001 to 1,000. Fig. 7 shows the performance curve with respect to the variation of λ. From the figure, we observe that our method consistently outperforms AGR, and the performances stay at a stable level over a wide range of parameter variation. We then vary the value of γ from 0.001 to 1,000 for both EAGR and AGR ($\lambda = 10$). Fig. 8 demonstrates the performance variation. We can see that EAGR is superior to AGR when γ varies in a wide range and the curve of EAGR has a peak when $\gamma = 1$ in most cases. These observations demonstrate the robustness of the parameter selection in applying our method to different datasets.

6 Conclusion

This work introduces a novel scalable graph-based semi-supervised learning algorithm named Efficient Anchor Graph Regularization (EAGR). It improves the AGR approach in the following two aspects. First, in anchor graph

TABLE 10
Time Costs (Seconds) of the Compared Learning Algorithms on Large-Size Dataset

	Landmark-Based Learning				Others			
	K-means	AGR	EAGR	PVM	FLAE-LI	Eigenfunction	LapSVMp	LLGC
MNIST	168.43	+130.07	+35.15	+87.74	5.59	27.40	3296.71	9521.15



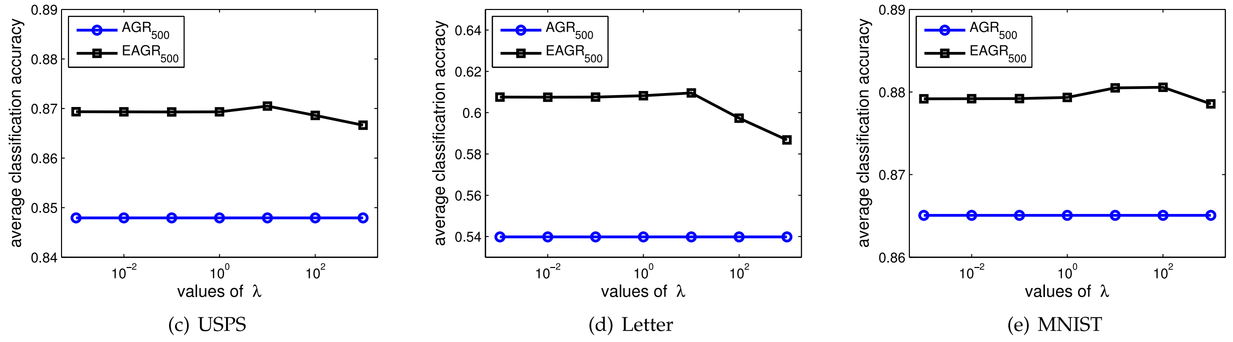
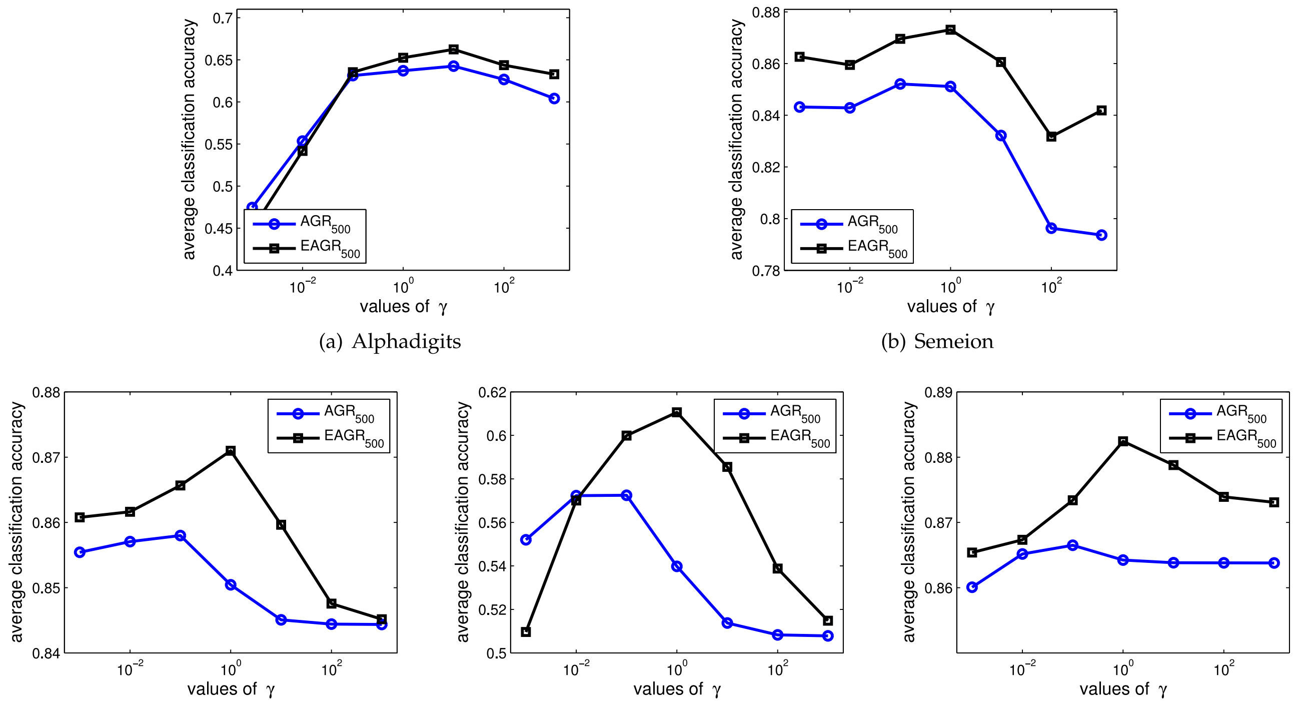


Fig. 7. Average performance curves of EAGR with respect to the variation of λ. Here, the number of labeled samples is set to 10 per class.



(a) Alphadigits (b) Semeion (c) USPS (d) Letter (e) MNIST

Fig. 8. Average performance curves of EAGR and AGR with respect to the variation of γ. Here, the number of labeled samples is set to 10 per class.

construction, it employs a novel fast local anchor embedding method to better measure the local weights between datapoints and neighboring anchors. Second, in anchor graph regularization, it employs a novel normalized graph Laplacian over anchors, which works better than the reduced graph Laplacian in AGR. For each improvement, we have provided an in-depth analysis on the limitation of the conventional method and the advantages of the new method. Experiments on publicly available datasets of various sizes have validated our EAGR in terms of classification accuracy and computational speed.

Appendix

Proof of Proposition 1

First, we present the following lemma.

Lemma 1. *Suppose f_1 is the minimization problem with objective function g in domain A, f_2 is the minimization problem with the same objective function g in domain B, and $B \subset A$, then for the optimal solution to f_1, e.g., x_A, and the optimal solution to f_2, e.g., x_B, we have $g(x_A) \leq g(x_B)$.*

The proof of the above lemma is clear. Since x_B is the optimal solution to f_2 in domain B, we have $x_B \in B$. Note $B \subset A$, so $x_B \in A$. Therefore, if $g(x_A) > g(x_B)$, x_A is not the optimal solution to f_1 in domain A.

Now we prove Proposition 1. Clearly, both Eq. (8) and Eq. (2) have the same objective function. In addition, if A denotes the domain of Eq. (8), i.e., $\mathbf{1}^{\mathrm{T}}\hat{\mathbf{c}}_{\langle i\rangle} = 1$ and B denotes the domain of Eq. (2), i.e., $\mathbf{1}^{\mathrm{T}}\mathbf{z}_{\langle i\rangle} = 1, \mathbf{z}_{\langle i\rangle} \succeq 0$, we have $B \subset A$. According to Lemma 1, the minimum of Eq. (8) is not greater than the minimum of Eq. (2), which completes the proof.

Acknowledgments

This work is partially supported by the National 973 Program of China under grants 2014CB347600 and 2013CB329604, the Program for Changjiang Scholars and Innovative Research Team in University (PCSIRT) of the Ministry of Education, China, under grant IRT13059, the National Nature Science Foundation of China under grants 61272393, 61322201 and 61432019, and the Australian Research Council Projects under grants DP-140102164 and FT-130101457.

References

[1] J. S. Beis and D. G. Lowe, "Shape indexing using approximate nearest-neighbour search in high-dimensional spaces," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 1997, pp. 1000–1006.
[2] M. Belkin and P. Niyogi, "Semi-supervised learning on riemannian manifolds," *Mach. Learning*, vol. 56, nos. 1–3, pp. 209–239, 2004.
[3] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer, 2006.
[4] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proc. 11th Annu. Conf. Comput. Learning Theory*, 1998, pp. 92–100.
[5] D. Cai and X. Chen, "Large scale spectral clustering with landmark-based representation," *IEEE Trans. Cybern.*, vol. 45, no. 8, pp. 1669–1680, Aug. 2015.
[6] V. Castelli and T. M. Cover, "On the exponential value of labeled samples," *Pattern Recog. Lett.*, vol. 16, no. 1, pp. 105–111, 1995.
[7] J. Cheng, C. Leng, P. Li, M. Wang, and H. Lu, "Semi-supervised multi-graph hashing for scalable similarity search," *Comput. Vis. Image Understanding*, vol. 124, pp. 12–21, 2014.
[8] C. Deng, R. Ji, W. Liu, D. Tao, and X. Gao, "Visual reranking through weakly supervised multi-graph learning," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 2600–2607.
[9] C. Deng, R. Ji, D. Tao, X. Gao, and X. Li, "Weakly supervised multi-graph learning for robust image reranking," *IEEE Trans. Multimedia*, vol. 16, no. 3, pp. 785–795, Apr. 2014.

[10] C. Elkan, "Using the triangle inequality to accelerate k-means," in *Proc. 20th Int. Conf. Mach. Learning*, 2003, vol. 3, pp. 147–153.

[11] R. Fergus, Y. Weiss, and A. Torralba, "Semi-supervised learning in gigantic image collections," in *Proc. Conf. Adv. Neural Inform. Process. Syst.*, 2009, pp. 522–530.

[12] P. Foggia, G. Percannella, and M. Vento, "Graph matching and learning in pattern recognition in the last 10 years," *Int. J. Pattern Recog. Artif. Intell.*, vol. 28, no. 1, pp. 1–40, 2014.

[13] L. Gao, J. Song, F. Nie, Y. Yan, N. Sebe, and H. T. Shen, "Optimal graph learning with partial tags and multiple features for image and video annotation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2015, pp. 4371–4379.

[14] M. Guillaumin, J. Verbeek, and C. Schmid, "Multimodal semi-supervised learning for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2010, pp. 902–909.

[15] S. Gupta, J. Kim, K. Grauman, and R. Mooney, "Watch, listen & learn: Co-training on captioned images and videos," in *Proc. Eur. Conf. Mach. Learning Knowl. Discovery Databases*, 2008, pp. 457–472.

[16] L. Huang, X. Liu, B. Ma, and B. Lang, "Online semi-supervised annotation via proxy-based local consistency propagation," *Neurocomputing*, vol. 149, pp. 1573–1586, 2015.

[17] M. S. T. Jaakkola and M. Szummer, "Partially labeled classification with Markov random walks," in *Proc. Conf. Adv. Neural Inform. Process. Syst.*, 2002, vol. 14, pp. 945–952.

[18] T. Joachims, "Transductive inference for text classification using support vector machines," in *Proc. 16th Int. Conf. Mach. Learning*, 1999, pp. 200–209.

[19] S. Kim and S. Choi, "Multi-view anchor graph hashing," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 3123–3127.

[20] P. Li, J. Bu, L. Zhang, and C. Chen, "Graph-based local concept coordinate factorization," *Knowl. Inform. Syst.*, vol. 43, no. 1, pp. 103–126, 2015.

[21] W. Liu, J. He, and S.-F. Chang, "Large graph construction for scalable semi-supervised learning," in *Proc. 27th Int. Conf. Mach. Learning*, 2010, pp. 679–686.

[22] W. Liu, J. Wang, and S.-F. Chang, "Robust and scalable graph-based semisupervised learning," *Proc. IEEE*, vol. 100, no. 9, pp. 2624–2638, Sep. 2012.

[23] W. Liu, J. Wang, S. Kumar, and S.-F. Chang, "Hashing with graphs," in *Proc. 28th Int. Conf. Mach. Learning*, 2011, pp. 1–8.

[24] X. Liu and B. Huet, "Concept detector refinement using social videos," in *Proc. Int. Workshop Very-Large-Scale Multimedia Corpus, Mining Retrieval*, 2010, pp. 19–24.

[25] F. D. Malliaros, V. Megalooikonomou, and C. Faloutsos, "Estimating robustness in large social graphs," *Knowl. Inform. Syst.*, vol. 45, no. 3, pp. 645–678, 2015.

[26] S. Melacci and M. Belkin, "Laplacian support vector machines trained in the primal," *J. Mach. Learning Res.*, vol. 12, pp. 1149–1184, 2011.

[27] B. Mohar, Y. Alavi, G. Chartrand, and O. Oellermann, "The laplacian spectrum of graphs," *Graph Theory, Combinatorics, Appl.*, vol. 2, pp. 871–898, 1991.

[28] Y. Nesterov, *Introductory Lectures on Convex Optimization*. New York, NY, USA: Springer, 2004, vol. 87.

[29] M. Norouzi, A. Punjani, and D. J. Fleet, "Fast exact search in hamming space with multi-index hashing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 6, pp. 1107–1119, Jun. 2014.

[30] M. Okabe and S. Yamada, "Semisupervised query expansion with minimal feedback," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 11, pp. 1585–1589, Nov. 2007.

[31] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[32] J. Song, Y. Yang, Z. Huang, H. T. Shen, and J. Luo, "Effective multiple feature hashing for large-scale near-duplicate video retrieval," *IEEE Trans. Multimedia*, vol. 15, no. 8, pp. 1997–2008, Dec. 2013.

[33] Z. Tian and R. Kuang, "Global linear neighborhoods for efficient label propagation," in *Proc. SIAM Int. Conf. Data Mining*, 2012, pp. 863–872.

[34] I. Triguero, S. García, and F. Herrera, "Self-labeled techniques for semi-supervised learning: Taxonomy, software and empirical study," *Knowl. Inform. Syst.*, vol. 42, no. 2, pp. 245–284, 2015.

[35] R. R. Vatsavai, S. Shekhar, and T. E. Burk, "A semi-supervised learning method for remote sensing data mining," in *Proc. 17th IEEE Int. Conf. Tools Artif. Intell.*, Nov. 2005, pp. 207–211.

[36] D. Wang, F. Nie, and H. Huang, "Large-scale adaptive semi-supervised learning via unified inductive and transductive model," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2014, pp. 482–491.

[37] F. Wang and C. Zhang, "Label propagation through linear neighborhoods," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 1, pp. 55–67, Jan. 2008.

[38] J. Wang, J. Wang, G. Zeng, Z. Tu, R. Gan, and S. Li, "Scalable k-NN graph construction for visual descriptors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2012, pp. 1106–1113.

[39] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong, "Locality-constrained linear coding for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2010, pp. 3360–3367.

[40] M. Wang, X.-S. Hua, R. Hong, J. Tang, G.-J. Qi, and Y. Song, "Unified video annotation via multigraph learning," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 5, pp. 733–746, May 2009.

[41] Y. Wu, M. Pei, M. Yang, J. Yuan, and Y. Jia, "Robust discriminative tracking via landmark-based label propagation," *IEEE Trans. Image Process.*, vol. 24, no. 5, pp. 1510–1523, May 2015.

[42] Y. Xiong, W. Liu, D. Zhao, and X. Tang, "Face recognition via archetype hull ranking," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 585–592.

[43] B. Xu, J. Bu, C. Chen, C. Wang, D. Cai, and X. He, "EMR: A scalable graph-based ranking model for content-based image retrieval," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 1, pp. 102–114, Jan. 2015.

[44] Z. Yang and E. Oja, "Clustering by low-rank doubly stochastic matrix decomposition," in *Proc. 29th Int. Conf. Mach. Learning*, 2012, pp. 831–838.

[45] G. Yu, G. Zhang, Z. Zhang, Z. Yu, and L. Deng, "Semi-supervised classification based on subspace sparse representation," *Knowl. Inform. Syst.*, vol. 43, no. 1, pp. 81–101, 2015.

[46] L. Zelnik-Manor and P. Perona, "Self-tuning spectral clustering," in *Proc. Conf. Adv. Neural Inform. Process. Syst.*, 2004, pp. 1601–1608.

[47] K. Zhang, J. T. Kwok, and B. Parvin, "Prototype vector machine for large scale semi-supervised learning," in *Proc. 26th Annu. Int. Conf. Mach. Learning*, 2009, pp. 1233–1240.

[48] K. Zhang, L. Lan, J. T. Kwok, S. Vucetic, and B. Parvin, "Scaling up graph-based semi-supervised learning via prototype vector machines," *IEEE Trans. Neural Netw. Learning Syst.*, vol. 26, no. 3, pp. 444–457, Mar. 2015.

[49] L. Zhang, Y. Yang, M. Wang, R. Hong, L. Nie, and X. Li, "Detecting densely distributed graph patterns for fine-grained image categorization," *IEEE Trans. Image Process.*, vol. 25, no. 2, pp. 553–565, Feb. 2016.

[50] S. Zhao, H. Yao, Y. Yang, and Y. Zhang, "Affective image retrieval via multi-graph learning," in *Proc. 22nd ACM Int. Conf. Multimedia*, 2014, pp. 1025–1028.

[51] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Proc. Conf. Adv. Neural Inform. Process. Syst.*, 2004, vol. 16, no. 16, pp. 321–328.

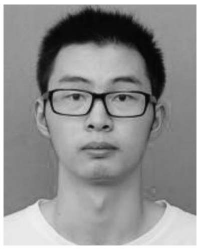
[52] Z.-H. Zhou and M. Li, "Semi-supervised regression with cotraining-style algorithms," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 11, pp. 1479–1493, Nov. 2007.

[53] Z. Zhou and M. Li, "Tri-training: Exploiting unlabeled data using three classifiers," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 11, pp. 1529–1541, Nov. 2005.

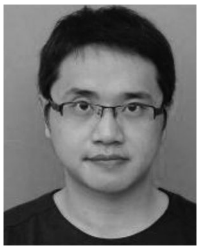
[54] X. Zhu, Z. Ghahramani, J. Lafferty et al.,"Semi-supervised learning using Gaussian fields and harmonic functions," in *Proc. 20th Int. Conf. Mach. Learning*, 2003, vol. 3, pp. 912–919.

[55] X. Zhu and A. B. Goldberg, "Introduction to semi-supervised learning," *Synthesis Lectures Artif. Intell. Mach. Learning*, vol. 3, no. 1, pp. 1–130, 2009.

Meng Wang received the BE and PhD degrees in the special class for the gifted young and the Department of Electronic Engineering and Information Science from the University of Science and Technology of China, Hefei, China, respectively. He is currently a professor at the Hefei University of Technology, Hefei. His current research interests include multimedia content analysis, search, mining, recommendation, and large-scale computing. He received the Best Paper Awards successively from the 17th and 18th ACM International Conference on Multimedia, the Best Paper Award from the 16th International Multimedia Modeling Conference, the Best Paper Award from the 4th International Conference on Internet Multimedia Computing and Service, and the Best Demo Award from the 20th ACM International Conference on Multimedia. He is a member of the IEEE.



Weijie Fu is currently working toward the PhD degree in the School of Computer and Information, Hefei University of Technology, Hefei, China. His current research interest includes machine learning and data mining.



Shijie Hao received the BE, MS, and PhD degrees from the School of Computer and Information, Hefei University of Technology (HFUT), Hefei, China. He is currently a lecturer at HFUT. His current research interests include machine learning and image processing.

Dacheng Tao is currently a professor of computer science with the Centre for Quantum Computation and Intelligent Systems, and the Faculty of Engineering and Information Technology , University of Technology Sydney, Ultimo, NSW, Australia, respectively. He mainly applies statistics and mathematics to data analytics problems and his research interests include computer vision, data science, image processing, machine learning, and video surveillance. His research results have expounded in one monograph and more than 200 publications at prestigious journals and prominent conferences, such as *IEEE T-PAMI, T-NNLS, T-IP, JMLR, IJCV*, NIPS, ICML, CVPR, ICCV, ECCV, AISTATS, ICDM; and ACM SIGKDD, with several Best Paper Awards, such as the Best theory/algorithm Paper Runner Up Award in IEEE ICDM07, the Best Student Paper Award in IEEE ICDM13, and the 2014 ICDM 10-year Highest-Impact Paper Award. He received the 2015 Australian Scopus-Eureka Prize, the 2015 ACS Gold Disruptor Award, and the 2015 UTS Vice-Chancellors Medal for Exceptional Research. He is a fellow of the IEEE, OSA, IAPR, and SPIE.

Xindong Wu received the bachelor's and master's degrees in computer science from the Hefei University of Technology, Hefei (HFUT), Hefei, China, and the PhD degree in artificial intelligence from the University of Edinburgh, Edinburgh, United Kingdom. He is currently a Yangtze River scholar in the School of Computer Science and Information Engineering, HFUT, and a professor of computer science at the University of Vermont, Burlington, VT, USA. His research interests include data mining, knowledge-based systems, and Web information exploration. He is the steering committee chair of the IEEE International Conference on Data Mining, the editor-in-chief of *Knowledge and Information Systems* (*KAIS*, by Springer), and a series editor of the Springer Book Series on Advanced Information and Knowledge Processing (AI&KP). He was the editor-in-chief of the *IEEE Transactions on Knowledge and Data Engineering* (*TKDE*, by the IEEE Computer Society) between 2005 and 2008. He served as the program committee chair for ICDM '03 (the 2003 IEEE International Conference on Data Mining), KDD-07 (the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining), and CIKM 2010 (the 19th ACM Conference on Information and Knowledge Management). He is a fellow of the IEEE.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Dimensionality reduction on Anchorgraph with an efficient Locality Preserving Projection

Rui Jiang, Weijie Fu, Li Wen, Shijie Hao*, Richang Hong

Hefei University of Technology, Hefei 230009, China

ARTICLE INFO

Article history:
Received 30 April 2015
Received in revised form
28 July 2015
Accepted 29 July 2015
Available online 3 December 2015

Keywords:
Dimensionality reduction
Anchorgraph
Locality Preserving Projection

ABSTRACT

Manifold learning based dimensionality reduction methods have been successfully applied in many pattern recognition tasks, due to their ability to well capture the underlying relationship between data points. These methods, however, meet some challenges in terms of the storage cost and the computation complexity with the rapidly increasing data size. We propose an improved dimensionality reduction algorithm called Anchorgraph-based Locality Preserving Projection (AgLPP), trying to cope with the limitations via a novel estimation of the relationship between data points. We extend AgLPP into a kernel version, and reformulate it into a novel sparse representation. The experiments on several real-world datasets have demonstrated the effectiveness and efficiency of our methods.

© 2015 Elsevier B.V. All rights reserved.

1. Introduction

In many real world applications, vectors of data representation often lie in high dimensional spaces, which makes large deviation to the similarity measure of features and brings high computational costs for classification and retrieval tasks [1–6] in various multimedia applications. Reducing the dimensionality of feature while capturing the discriminative information, therefore, becomes an important role in data preprocessing.

One of the classical dimensionality reduction algorithms is Principal Component Analysis (PCA) [7], which searches a set of orthogonal basis functions to capture the direction of maximum variance of data distribution. However, PCA has its limitations in addressing nonlinear data in many real applications, because it is based on the assumption that data can be embedded in a linear subspace of lower dimensionality.

To cope with the nonlinear data, dimensionality reduction methods based on manifold learning have been proposed and produced impressive results. For example, Isomap [8] is an approach which preserves certain inter-point relationships via the underlying global geometry. Locally Linear Embedding (LLE) [9] and Laplacian Eigenmap (LE) [10] are the methods that preserve the local geometry of the manifold with different measuring principles respectively. These approaches deal with fixed training sets well. However, as they do not produce an explicit mapping function between high and low dimensional spaces, they cannot be applied to new data points.

Later, He et al. [11] proposed an approach called Locality Preserving Projection (LPP), which incorporates the linear embedding assumption into manifold learning. LPP can effectively deal with both fixed training data and new data points. This method can be conducted in the reproducing kernel Hilbert space (RKHS) via kernel tricks as well.

These manifold-based dimensionality reduction methods, however, meet some challenges now because of the rapidly increasing data size. First, they are supposed to construct a graph with size $O(N^2)$ to measure the adjacency relationship. Second, they need to solve a generalized eigenvalue problem for an $N \times N$ matrix with $O(N^3)$ complexity when conducted in Hilbert space via kernel tricks. Both of the above processes introduce huge temporal and storage cost for the data preprocessing.

In this paper, we propose a novel manifold-based linear dimensionality reduction method, called Anchorgraph-based Locality Preserving Projection (AgLPP). Given a large set of data points, we first adopt the clustering algorithm to obtain a small number of clustering centers as virtual anchors, which have been validated to have a stronger representation power to adequately cover the vast point cloud in [12–15]. Then by regarding these anchors as transformation points, we measure the point-to-point relationship between real data points in two steps. Finally, based on the adjacency matrix of this novel relationship, the time and storage cost of AgLPP can be linear with respect to data size. The contributions of this paper are highlighted as follows. First, as AgLPP keeps the linear embedding assumption of LPP, this method can be applied to any new data point to locate the mapped position in the reduced representation space. We additionally extend AgLPP to a

* Corresponding author.

http://dx.doi.org/10.1016/j.neucom.2015.07.128
0925-2312/© 2015 Elsevier B.V. All rights reserved.

kernel version and reformulate it into a novel sparse representation. Second, our AgLPP and its two extended variants compute the mapping functions for nonlinear data with Anchorgraph. As a result, the dimensionality reduction on a large scale database can be implemented with fewer temporal and storage costs. Third, we conduct experiments to empirically validate our methods on five datasets. The experimental results demonstrate better effectiveness and efficiency of our methods.

The rest of this paper is organized as follows. In Section 2, we briefly discuss some related work. Our AgLPP is introduced in Section 3. In Section 4, we present the experiment results on several real world databases. Finally we conclude the paper in Section 5.

2. Related work

We focus on the dimensionality reduction models which are designed to cope with nonlinear data, because many kinds of real-world data lie on or near the manifold of the high dimensional space.

One popular family of these models is the manifold-based dimensionality reduction method, which is designed to keep the inter-point similarity from the original high dimensional space to the low one. For many years, researchers have focused on preserving the global property of the whole point set. Kruskal et al. [16] advocated expressing the quality of mapping by a stress function, which is based on the error between the pairwise distances in the low dimensional and high dimensional representation of the data. Tenenbaum et al. [8] proposed a method named Isomap, which aims at preserving the pairwise geodesic distances between data points by utilizing shortest-path algorithms. Weinberger et al. [17] suggested the formulation which maximizes the Euclidean distances between the data points while retaining the distances in the neighborhood graph. Apart from global properties, researchers have also been focusing on methods that preserve local properties of neighborhood points, because many experiments have vindicated that via preservation of local properties of the data, the global layout of the data manifold is retained as well. Roweis et al. [9] proposed the formulation that attempts to discover nonlinear structure in high dimensional data by exploiting the local symmetries of the linear reconstruction and retain these reconstruction weights in the space of lower dimensionality as good as possible. Similarly, Belkin et al. [10] proposed Laplacian Eigenmap, which is conducted in a weighted manner where the distance in the low dimensional data representation between a data point and its first nearest neighbor contributes more to the cost function than the distance between the data point and its second nearest neighbor. Donoho et al. [18] subsequently suggested the design of a mapping model that minimizes the ‘curviness’ of the high dimensional manifold when embedding it into a low dimensional space, with the constraint that the low dimensional data representation is locally isometric. Although these approaches deal with the fixed training set well, they have difficulties in being directly applied to new data points, due to the lack of an explicit mapping function between high and low dimensional spaces. To address this issue, He et al. suggested incorporating the linear embedding assumption into these manifold learning methods and proposed the linear approximations for these original nonlinear methods [11,19,20]. As these manifold learning methods have a good ability to capture the structure of nonlinear data, they are broadly adopted in many real-world applications, such as [21,22].

Another important family of models is kernel based dimensionality reduction methods. By applying different kernels, these methods implicitly map data into higher dimensional spaces, where the nonlinear data become linear or near linear. For instance, Schölkopf et al. [23] proposed a reformulation of traditional linear PCA in a high dimensional space, which computes the principal eigenvectors of the kernel matrix rather than the covariance matrix. Based on this work, Ham et al. [24] stated the kernel view of some manifold based dimensionality reduction methods. However, the choice of the kernel types plays an important role in kernel PCA. Weinberger et al. [25] suggested learning a kernel matrix instead, where the model is constructed by maximizing the variance in feature space while preserving the local angle and distance between nearest neighbors. Similarly, Zimmer et al. [26] presented a method to automatically choose a kernel function and its associated parameters from a pool of kernel candidates to generate the most optimal manifold embedding. To address the lack of an explicit out-of-sample extension of non-parametric dimensionality reduction techniques, Gisbrecht et al. [27] extended the *t*-distributed stochastic neighbor embedding to a kernel version.

Meanwhile, sparse dimensionality reduction is also becoming an emerging important research direction, which clearly maps the data into a meaningful space with learnt basis instead of implicitly mapping into the higher one as kernel methods. For instance, Han et al. [28] formulated the construction of the low-dimensional consensus representation to approximate the matrix of patterns by means of a low-dimensional consensus base matrix and a loading matrix, and proposed a sparse unsupervised dimensionality reduction for multiple view data. Zhu et al. [29] proposed the self-taught dimensionality reduction on the high-dimensional small-sized data via mapping the target data into the learnt basis space which is learnt from sufficient external data. Kandel et al. [30] proposed a sparse dimensionality reduction method for multi-modal medical image analysis via clustering sparse eigenvectors and selecting a subset of the eigenvectors. Shi et al. [31] presented a framework by combining the objective functions of graph embedding and sparse regression to explore the significant features of data for dimensionality reduction.

Apart from the traditional unsupervised dimensionality reduction methods, some methods with labeled information used recently are also proposed to cope with specific situations. For instance, Cheng et al. [32] extended LPP to a supervised one in which the geometric relation is preserved according to prior class-label information. Yan et al. [33] proposed a supervised dimensionality reduction algorithm called Marginal Fisher Analysis (MFA), where the intrinsic graph characterizes the intra-class compactness and connects each data point with its neighboring points of the same class, while the penalty graph characterizes the inter-class separability and connects the marginal points. Gui et al. [34] proposed a method called Discriminant Sparse Neighborhood Preserving Embedding (DSNPE), which not only preserves the sparse reconstructive relationship of SNPE, but also sufficiently utilizes the global discriminant structures. Li et al. [35] defined a novel non-parametric manifold-to-manifold distance to model separability between manifolds, where both labels and local structure information of manifolds are considered. Hashing methods [36–38] can be also an alternative for the task of dimension reduction.

3. Dimensionality reduction on Anchorgraph

In order to incorporate scalable relationships of large databases into the manifold-based dimensionality reduction methods, a natural way is to perform learning tasks on Anchorgraph [12–13]. In this section, we will describe our AgLPP which is based on Locality Preserving Projection and Anchorgraph construction (A flowchart is shown in Fig. 1). We begin with the description of Locality Preserving Projection [11].

3.1. Locality Preserving Projection

Given a set $\mathbf{X} = \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \mathbb{R}^{D \times n}$, the generic problem of the linear dimensionality reduction method is to find a transformation

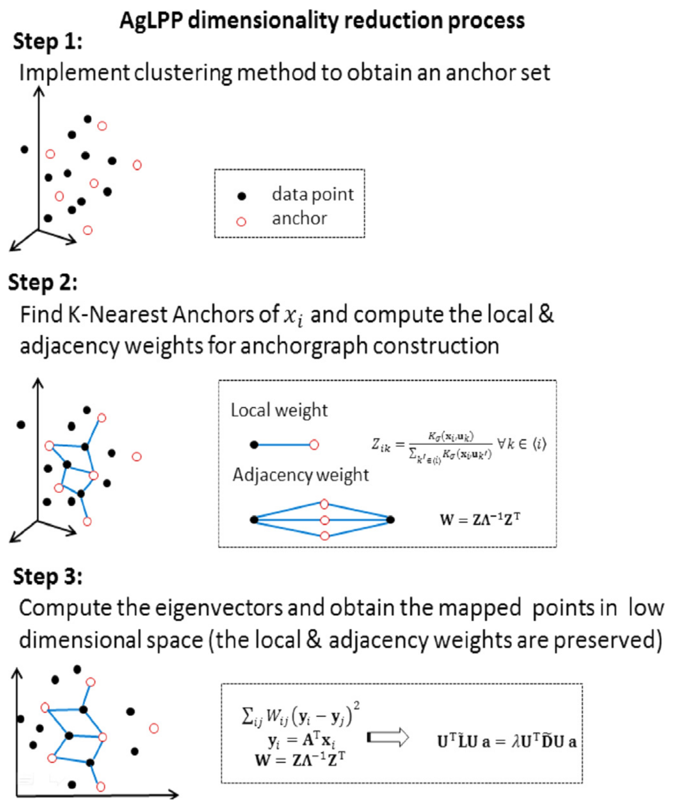

AgLPP dimensionality reduction process
Step 1:
Implement clustering method to obtain an anchor set
data point
anchor
Step 2:
Find K-Nearest Anchors of x_i and compute the local & adjacency weights for anchorgraph construction
Local weight
$Z_{ik} = \frac{K_\sigma(\mathbf{x}_i,\mathbf{u}_k)}{\sum_{k' \in \langle i \rangle} K_\sigma(\mathbf{x}_i,\mathbf{u}_{k'})} \forall k \in \langle i \rangle$
Adjacency weight
$\mathbf{W} = \mathbf{Z}\Lambda^{-1}\mathbf{Z}^{\mathrm{T}}$
Step 3:
Compute the eigenvectors and obtain the mapped points in low dimensional space (the local & adjacency weights are preserved)
$\sum_{ij} W_{ij}(\mathbf{y}_i - \mathbf{y}_j)^2$
$\mathbf{y}_i = \mathbf{A}^{\mathrm{T}}\mathbf{x}_i$
$\mathbf{W} = \mathbf{Z}\Lambda^{-1}\mathbf{Z}^{\mathrm{T}}$
$\mathbf{U}^{\mathrm{T}}\tilde{\mathbf{L}}\mathbf{U}\,\mathbf{a} = \lambda\mathbf{U}^{\mathrm{T}}\tilde{\mathbf{D}}\mathbf{U}\,\mathbf{a}$

Fig. 1. The proposed AgLPP dimensionality reduction process.

matrix $\mathbf{A} = \mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_d \in \mathbb{R}^{D \times d}$ that maps the original high dimensional set $\mathbf{X}$ to a low dimensional set $\mathbf{Y} = \mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n \in \mathbb{R}^{d \times n} (d < D)$, where $\mathbf{y}_i = \mathbf{A}^{\mathrm{T}}\mathbf{x}_i$.

Let $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ be the weighted graph with edges which connect nearby points to each other. LPP considers mapping the weighted graph $\mathbf{G}$ into a low dimensional map where the connected points stay as close as before. If $\mathbf{Y}$ is such a map, a reasonable criterion for obtaining a good map is to minimize the following objective function:

$$\sum_{ij} W_{ij}\left(\mathbf{y}_i - \mathbf{y}_j\right)^2 \tag{1}$$

This clearly makes a strong penalty if that neighboring points $\mathbf{x}_i$ and $\mathbf{x}_j$ are mapped far apart. That is to say, minimizing the above function is equal to keeping a close similarity between the pair of $\mathbf{x}_i \& \mathbf{x}_j$ and the pair of $\mathbf{y}_i \& \mathbf{y}_j$.

Suppose $\mathbf{a}$ is a transformation vector in $\mathbf{A}$, with simple algebra derivation, Eq. (1) can be reduced to

$$\begin{aligned}\tfrac{1}{2}\textstyle\sum_{ij}\left(\mathbf{y}_i - \mathbf{y}_j\right)^2 W_{ij} &= \tfrac{1}{2}\textstyle\sum_{ij}(\mathbf{a}^{\mathrm{T}}\mathbf{x}_i - \mathbf{a}^{\mathrm{T}}\mathbf{x}_j)^2 W_{ij} \\ &= \sum_i (\mathbf{a}^{\mathrm{T}}\mathbf{x}_i D_{ii}\mathbf{x}_i^{\mathrm{T}}\mathbf{a}) - \sum_{ij}\left(\mathbf{a}^{\mathrm{T}}\mathbf{x}_i W_{ij}\mathbf{x}_j^{\mathrm{T}}\mathbf{a}\right) \\ &= \mathbf{a}^{\mathrm{T}}\mathbf{X}(\mathbf{D}-\mathbf{W})\mathbf{X}^{\mathrm{T}}\mathbf{a} = \mathbf{a}^{\mathrm{T}}\mathbf{X}\mathbf{L}\mathbf{X}^{\mathrm{T}}\mathbf{a}.\end{aligned} \tag{2}$$

where $\mathbf{D}$ is a diagonal matrix with $D_{ii} = \sum_j W_{ij}$, and $\mathbf{L} = \mathbf{D} - \mathbf{W}$ is the graph Laplacian matrix. Here the bigger D_{ii} is, the more important $\mathbf{y}_i$ is. Thus, we empirically set the constraint of $\mathbf{y}\mathbf{D}\mathbf{y}^{\mathrm{T}} = 1$, which can be expressed as

$$\mathbf{a}^{\mathrm{T}}\mathbf{X}\mathbf{D}\mathbf{X}^{\mathrm{T}}\mathbf{a} = 1 \tag{3}$$

Finally, the minimization problem is reduced to solve the problem as

$$\begin{aligned}&\operatorname{argmin}_{\mathrm{a}}\ \mathbf{a}^{\mathrm{T}}\mathbf{X}\mathbf{L}\mathbf{X}^{\mathrm{T}}\mathbf{a} \\ &\mathrm{s.t.}\,\mathbf{a}^{\mathrm{T}}\mathbf{X}\mathbf{D}\mathbf{X}^{\mathrm{T}}\mathbf{a} = 1\end{aligned} \tag{4}$$

and it can be transformed to a generalized eigenvalue problem as:

$$\mathbf{X}\mathbf{L}\mathbf{X}^{\mathrm{T}}\mathbf{a} = \lambda\mathbf{X}\mathbf{D}\mathbf{X}^{\mathrm{T}}\mathbf{a} \tag{5}$$

where λ is the minimum eigenvalue solution of Eq. (5) and $\mathbf{a}$ is the corresponding eigenvector. We additionally consider conducting LPP in RKHS. Let $\phi(\mathbf{X})$ denote the data expression in Hilbert space, the eigenvalue problem then can be written as

$$\left[\phi(\mathbf{X})\mathbf{L}\phi(\mathbf{X})^{\mathrm{T}}\right]\mathbf{v} = \lambda\left[\phi(\mathbf{X})\mathbf{D}\phi(\mathbf{X})^{\mathrm{T}}\right]\mathbf{v} \tag{6}$$

Since eigenvectors of Eq. (6) are linear combinations of $\phi(\mathbf{x}_i), i = 1, \ldots, n$, there exist coefficients $\alpha_i s, i = 1, 2, \ldots, n$ such that

$$\mathbf{v} = \sum_{i=1}^{n} \alpha_i\phi(\mathbf{x}_i) = \phi(\mathbf{X})\boldsymbol{\alpha} \tag{7}$$

Let $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j) = (\phi(\mathbf{x}_i)\cdot\phi(\mathbf{x}_j)) = \phi(\mathbf{x}_j)^{\mathrm{T}}\phi(\mathbf{x}_i)$, we can obtain the solution of Kernel Locality Preserving Projection (KLPP) via the following eigenvalue problem

$$\mathbf{K}\mathbf{L}\mathbf{K}\boldsymbol{\alpha} = \lambda\mathbf{K}\mathbf{D}\mathbf{K}\boldsymbol{\alpha} \tag{8}$$

where λ is the minimum eigenvalue solution of Eq. (8) and $\mathbf{a}$ is the corresponding eigenvector. Particularly, if we use normalized Laplacian $\overline{\mathbf{L}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}$ instead, Eq. (8) can be written as

$$\mathbf{K}\overline{\mathbf{L}}\mathbf{K}\boldsymbol{\alpha} = \lambda\mathbf{K}\mathbf{K}\boldsymbol{\alpha} \tag{9}$$

LPP and its extended version KLPP have been widely used in many applications [39–40]. However, they have their own limitations to handle large scale databases.

The main limitation of LPP comes from its graph construction. KNN graph is a feasible approximation to model the manifold data and can well capture the local structure of data points. Nevertheless, the computational complexity of this graph construction is $O(n^2)$, and the storage cost of the adjacency matrix is $O(n^2)$ as well. When the size of dataset becomes larger, it would pose a challenge to the dimensionality reduction task due to the rapid growth of the temporal and spatial costs.

The limitation of KLPP mainly comes from the consumption of its eigenvector computation. It is a significant cost for the large dataset, because the time complexity of the mapping vectors computation via Eq. (9) is $O(n^3)$.

3.2. *Scalable graph construction*

To alleviate this issue, the cost of graph construction is expected to be linear with the data size in terms of both time and storage consumption. We therefore construct the Anchorgraph [12–13] instead of the original KNN graph to meet this requirement.

Given a dataset $\mathbf{X} = \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \mathbb{R}^{D \times n}$ with n samples in D dimensions, we are supposed to measure the adjacency weight between these samples to model the distribution of manifold data. Unlike the KNN strategy, we first adopt a clustering method to obtain a set of clustering centers as virtual anchors $= \{\mathbf{u}_1, \mathbf{u}_{2,} \ldots \mathbf{u}_m \in \mathbb{R}^{D \times m}$, i.e., Step 1 in Fig. 1. As these anchors share the same feature space with the original data points, we can construct the weighted graph in two steps as follows, i.e., Step 2 in Fig. 1.

First, we compute the local weight matrix, which measures the relationship between data points and its s closest anchors. A simple way to measure this underlying relationship is the Nadataya–Waston kernel regression [41], which defines such Z_{ik} based on a kernel function K_σ as

$$Z_{ik} = \frac{K_\sigma(\mathbf{x}_i, \mathbf{u}_k)}{\sum_{k' \in \langle i \rangle} K_\sigma(\mathbf{x}_i, \mathbf{u}_{k'})} \forall k \in \langle i \rangle \tag{10}$$

where the notation $\langle i \rangle \in [1: m]$ is the set which saves the indexes of the s closest anchors of $\mathbf{x}_i$.We typically adopt Gaussian kernel $K_\sigma(\mathbf{x}_i, \mathbf{u}_k) = \exp\left[-\frac{\|\mathbf{x}_i - \mathbf{u}_k\|^2}{2\sigma^2}\right]$ for this kernel regression. The smoothing parameter σ determines the size of the local region in which the anchor can affect the target point.

Then, we obtain the adjacency weight measurement, which estimates the relationship between real data points. Based on the local weight matrix, Liu et al. [12] suggested computing the adjacency matrix as

$$\mathbf{W} = \mathbf{Z}\boldsymbol{\Lambda}^{-1}\mathbf{Z}^{\mathrm{T}} \tag{11}$$

where $\boldsymbol{\Lambda}$ is a diagonal matrix defined as $\Lambda_{kk} = \sum Z_{ik}$. We can see that if two points are correlative ($W_{ij} > 0$), they share at least one common anchor, otherwise $W_{ij} = 0$.

3.3. LPP on Anchorgraph

After Anchorgraph construction, we introduce how to reduce the computational cost of original LPP via employing this adjacency matrix $\mathbf{W}$. We first approximate each data point $\mathbf{x}_i$ with $\hat{\mathbf{x}}_i$ by $\hat{\mathbf{x}}_i = \mathbf{U}\mathbf{Z}_i^{\mathrm{T}}$. Then Eq. (2) can be rewritten as

$$\frac{1}{2}\sum_{ij}(\mathbf{a}^{\mathrm{T}}\mathbf{x}_i - \mathbf{a}^{\mathrm{T}}\mathbf{x}_j)^2 W_{ij} = \mathbf{a}^{\mathrm{T}}\mathbf{U}\mathbf{Z}^{\mathrm{T}}\mathbf{L}\mathbf{Z}\mathbf{U}^{\mathrm{T}}\mathbf{a} \tag{12}$$

where $\mathbf{L} = \mathbf{D} - \mathbf{W} = \mathbf{I} - \mathbf{Z}\boldsymbol{\Lambda}^{-1}\mathbf{Z}^{\mathrm{T}}$.

We similarly impose the constraint of $\mathbf{y}\mathbf{y}^{\mathrm{T}} = 1$, which can be written as

$$\boldsymbol{a}^{\mathrm{T}}\mathbf{U}\mathbf{Z}^{\mathrm{T}}\mathbf{Z}\mathbf{U}^{\mathrm{T}}\mathbf{a} = 1 \tag{13}$$

The minimization problem then reduces to

$$\begin{aligned} &\operatorname{argmin}_{\mathbf{a}} \ \mathbf{a}^{\mathrm{T}}\mathbf{U}\tilde{\mathbf{L}}\mathbf{U}^{\mathrm{T}}\mathbf{a} \\ &\quad \text{s.t.} \mathbf{a}^{\mathrm{T}}\mathbf{U}\tilde{\mathbf{D}}\mathbf{U}^{\mathrm{T}}\mathbf{a} = 1 \end{aligned} \tag{14}$$

where $\tilde{\mathbf{D}} = \mathbf{Z}^{\mathrm{T}}\mathbf{Z}$ and $\tilde{\mathbf{L}} = \mathbf{Z}^{\mathrm{T}}\mathbf{Z} - \mathbf{Z}^{\mathrm{T}}\mathbf{Z}\boldsymbol{\Lambda}^{-1}\mathbf{Z}^{\mathrm{T}}\mathbf{Z}$ makes a new Laplacian matrix with much smaller size. Lastly, the above problem can be transformed to a generalized eigenvalue problem with minimum eigenvalue solution, i.e., Step 3 in Fig. (1),

$$\mathbf{U}^{\mathrm{T}}\tilde{\mathbf{L}}\mathbf{U}\,\mathbf{a} = \lambda\mathbf{U}^{\mathrm{T}}\tilde{\mathbf{D}}\mathbf{U}\,\mathbf{a} \tag{15}$$

For each new data point $\mathbf{x}_i$, the corresponding mapped point in lower dimensional representation can be obtained by

$$\mathbf{y}_i = \boldsymbol{a}^{\mathrm{T}}\mathbf{U}\mathbf{Z}_i^{\mathrm{T}} \tag{16}$$

where $\mathbf{Z}_i$ is the local weight between $\mathbf{x}_i$ and anchor set $\mathbf{U}$.

So far, we have built the reduced LPP with Anchorgraph, i.e., AgLPP, which is described in Algorithm 1. As a result, AgLPP has a computational cost of $O(TDnm + D^3)$ (T is the iteration times in k-means) and a storage cost of $O(nm)$, while LPP has a computational cost of $O\left(Dn^2 + D^3\right)$ and a storage cost of $O(n^2)$. When $n \gg D$, AgLPP clearly needs lower cost than LPP in terms of the implement time and storage space.

Algorithm 1. Anchorgraph-based Locality Preserving Projection (AgLPP)

```
Input: data points $\{\mathbf{x}_i\}_{i=1}^n \in \mathbb{R}^{D\times n}$, anchor number $m$, integer $s$,
  low dimensionality $d$.
1. Implement clustering method to produce $m$ cluster centers as
   anchors.
2. Measure the local weight matrix according to Eq. (10).
3. Compute the eigenvalues and eigenvectors of the generalized
   eigenvalue problem in Eq. (15).
4. Map the original data points into low dimensional space
   according to Eq. (16)
Output: mapped data points $\{\mathbf{y}_i\}_{i=1}^n \in \mathbb{R}^{d\times n}$, anchor set $\mathbf{U}$,
  mapping eigenvectors $\mathbf{A}$.
```

3.4. Kernel AgLPP and its reformulation of sparse representation

Now we consider the kernel version of AgLPP in RKHS. Suppose $\phi(\mathbf{U})$ denotes the anchor matrix in the Hilbert space, then Kernel AgLPP is given below:

$$\phi(\mathbf{U})\tilde{\mathbf{L}}\phi(\mathbf{U})^{\mathrm{T}}\mathbf{A} = \lambda\phi(\mathbf{U})\tilde{\mathbf{D}}\phi(\mathbf{U})^{\mathrm{T}}\mathbf{A} \tag{17}$$

As the eigenvectors of Eq. (17) are linear combinations of $\phi(\mathbf{U})$, i.e., $\mathbf{A} = \phi(\mathbf{U})\boldsymbol{\alpha}$, we have

$$\phi(\mathbf{U})^{\mathrm{T}}\phi(\mathbf{U})\tilde{\mathbf{L}}\phi(\mathbf{U})^{\mathrm{T}}\phi(\mathbf{U})\boldsymbol{\alpha} = \lambda\phi(\mathbf{U})^{\mathrm{T}}\phi(\mathbf{U})\tilde{\mathbf{D}}\phi(\mathbf{U})^{\mathrm{T}}\phi(\mathbf{U})\boldsymbol{\alpha} \tag{18}$$

Let $\tilde{\mathbf{K}} = \phi(\mathbf{U})^{\mathrm{T}}\phi(\mathbf{U}) \in \mathbb{R}^{m\times m}$ denote the kernel matrix of the anchors, then Eq. (18) is abbreviated by

$$\tilde{\mathbf{K}}\tilde{\mathbf{L}}\tilde{\mathbf{K}}\boldsymbol{\alpha} = \lambda\tilde{\mathbf{K}}\tilde{\mathbf{D}}\tilde{\mathbf{K}}\boldsymbol{\alpha} \tag{19}$$

where $\tilde{\mathbf{L}}$ and $\tilde{\mathbf{D}}$ are the same matrixes as those in Eq. (14). For each new data points, its low dimensional mapped point can be obtained by

$$\mathbf{y}_i = \mathbf{A}^{\mathrm{T}}\phi(\mathbf{U})\mathbf{Z}_i^{\mathrm{T}} = \boldsymbol{\alpha}^{\mathrm{T}}\mathbf{K}\mathbf{Z}_i^{\mathrm{T}} \tag{20}$$

Clearly, the computational cost of Kernel AgLPP is $O(TDnm + m^3)$ and a storage cost is $O(nm)$, while the original Kernel LPP has a computational cost of $O\left(Dn^2 + n^3\right)$ and a storage cost of $O(n^2)$.

So far, we have extended the dimensionality reduction into RKHS. We additionally reformulate it into a novel sparse representation in below.

Instead of approximating data point $\mathbf{x}_i$ by the local weight matrix $\mathbf{Z}_i$ with the fixed anchor set $\mathbf{U}$, we redefine each data point via its anchor based sparse representation as $\tilde{\mathbf{x}}_i := \mathbf{Z}_i^{\mathrm{T}}$. Then Eq. (2) can be rewritten as the following objective function:

$$\mathbf{a}^{\mathrm{T}}\mathbf{Z}^{\mathrm{T}}\mathbf{L}\mathbf{Z}\mathbf{a} = \mathbf{a}^{\mathrm{T}}\tilde{\mathbf{L}}\mathbf{a} \tag{21}$$

with the constraint of $\mathbf{y}\mathbf{y}^{\mathrm{T}} = 1$.

It can be transformed into the eigenvalue problem given below:

$$\mathbf{a}^{\mathrm{T}}\tilde{\mathbf{L}}\mathbf{a} = \lambda\mathbf{a}^{\mathrm{T}}\tilde{\mathbf{D}}\mathbf{a} \tag{22}$$

By letting $\mathbf{a} = \tilde{\mathbf{K}}\boldsymbol{\alpha}$, Eq. (22) becomes identical to the eigenvalue problem of Kernel AgLPP, i.e., Eq. (19). This validates that LPP on the new redefined data points yields similar results as Kernnel AgLPP on the data points. Later in our experiments, we named these two variants as KAgLPP (Kernel version) and SR AgLPP (Sparse Representation version) respectively.

For clarity, we demonstrate the relationship between the proposed AgLPP, KAgLPP and SR AgLPP in Fig. 2. Let D be the dimensionality of raw data feature, and m be the number of anchors. As we can see, AgLPP reduces the dimensionality of data in raw feature space with original dimensionality D. In contrast, KAgLPP and SR AgLPP reduce the dimensionality in an indirect way: for the former, it first maps data into Hilbert space, and then reduces the dimensionality via kernel trick to obtain m dimensional mapping vectors; for the later, it first maps data into the new space spanned by anchors, and then reduces the dimensionality of the sparse representation. Since the underlying data expressions of KAgLPP and SR AgLPP are transformed into either higher but effective or more meaningful feature spaces, their performances are expected to be better than AgLPP.

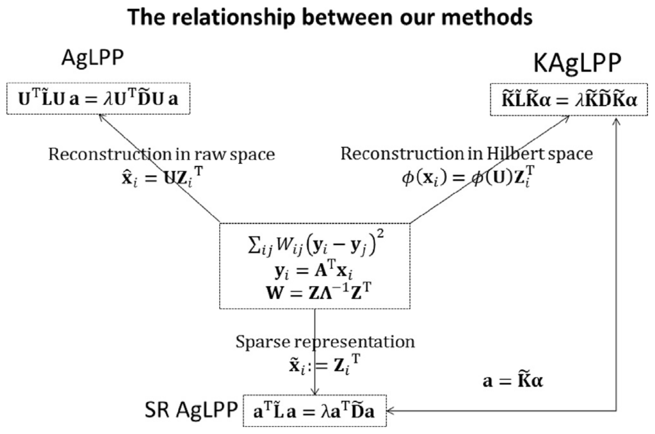

The relationship between our methods
AgLPP
$\mathbf{U}^{\mathrm{T}}\tilde{\mathbf{L}}\mathbf{U}\,\mathbf{a} = \lambda\mathbf{U}^{\mathrm{T}}\tilde{\mathbf{D}}\mathbf{U}\,\mathbf{a}$
KAgLPP
$\tilde{\mathbf{K}}\tilde{\mathbf{L}}\tilde{\mathbf{K}}\alpha = \lambda\tilde{\mathbf{K}}\tilde{\mathbf{D}}\tilde{\mathbf{K}}\alpha$
Reconstruction in raw space
$\hat{\mathbf{x}}_i = \mathbf{U}\mathbf{Z}_i^{\mathrm{T}}$
Reconstruction in Hilbert space
$\phi(\mathbf{x}_i) = \phi(\mathbf{U})\mathbf{Z}_i^{\mathrm{T}}$
$\Sigma_{ij} W_{ij}(\mathbf{y}_i - \mathbf{y}_j)^2$
$\mathbf{y}_i = \mathbf{A}^{\mathrm{T}}\mathbf{x}_i$
$\mathbf{W} = \mathbf{Z}\Lambda^{-1}\mathbf{Z}^{\mathrm{T}}$
Sparse representation
$\tilde{\mathbf{x}}_i := \mathbf{Z}_i^{\mathrm{T}}$
$\mathbf{a} = \tilde{\mathbf{K}}\alpha$
SR AgLPP
$\mathbf{a}^{\mathrm{T}}\tilde{\mathbf{L}}\mathbf{a} = \lambda\mathbf{a}^{\mathrm{T}}\tilde{\mathbf{D}}\mathbf{a}$

Fig.2. An illustration of the relationship between our methods.

Table 1
Details of the five datasets used in the experiments

Datasets	Sample size	Dimension	# of classes	# of labeled instances
Digits1	1500	241	2	100
USPS	1500	241	2	100
Newsgroups	16,242	100	4	100
USPS-All	11,000	256	10	100
MNIST-Train	60,000	784	10	100

4. Experiment study

4.1. Experiments setup

In this section, we show several experimental results and comparisons to evaluate the effectiveness and efficiency of our methods. The adopted datasets include two small scale datasets Digit1, and USPS, two middle scale datasets USPS-All and Newsgroups, and one large scale dataset MNIST-Train [42,43]. The details of these datasets are listed in Table 1. We here briefly describe the five datasets used in our experiments.

For Digit1 and USPS datasets, their instances were both divided into two different classes, and each of these samples has been preprocessed into a 241-demensional vector as feature.

The USPS-All dataset contains 11,000 instances which are 8-bit gray scale images (16×16) of “0” through “9”. The MNIST-Train dataset consists of 60,000 examples, and contains ten classes of 28×28 pixel digital images (0–9) with approximately 6000 instances in each class. Since the raw hand written image patch is relative small, we directly use the normalized grayscale value for each pixel as image features.

The Newsgroups here is a tiny version of the 20 news groups data, with binary occurance data for 100 words across 16,242 postings. It has also tagged the postings by the highest level domain in the array "newsgroups". In our experiments, we validate our methods via a simple 1NN classification task, which is conducted in the following steps. First, we construct the Anchorgraph as described in Section 3.2 and then calculate the eigenvectors of the generalized eigenvalue problem, for e.g., Eq. (15). Second, we project data points into the subspace based on the obtained mapping eigenvectors. Third, we randomly label a number of samples and identify other unlabeled points by the nearest neighbor classifier under the Euclidean metric for distance measure. Finally, we repeat the third process for several times and calculate the average classification accuracies. Except the baseline which employs the original data feature in the nearest neighborhood classification tasks, the following five linear methods, i.e., PCA [7], NPE [19], LPP [11], OLPP [44], AgLPP, are compared, which are conducted in the original data space. We also compare four methods, i.e., KPCA [23], KLPP [11], KAgLPP, SR AgLPP, which are conducted in the new transformed feature space.

It is noted that both the radius and the kernel bandwidth σ in the above algorithms are turned to the optimal one from $\sigma_0 \times 2^{-3}, 2^{-2}, 2^{-1}, 1, 2^1, 2^2, 2^3$ where σ_0 is the averaged distance between the points in the dataset. Since the number of anchors plays an important role in the quality of Anchorgraph construction [14], we set it to empirical values in different scale of experiments. As there is no explicit way to determine the optimal value for the target dimensionality d, we therefore test it in a wide range for all methods. Our experiments are implemented in MATLAB2013a on a computer with 2.6 GHz CPU, 32 GB RAM.

4.2. Performances

4.2.1. Performances on small datasets

To evaluate the effectiveness of our proposed AgLPP, we first conduct experiments on small datasets, where the anchor number is set to 500. Via selecting labeled samples for 10 repeated times, the average classification accuracies versus different number of target dimensionality of the methods conducted in original space are displayed in Fig. 3, and the accuracies in the new feature space are shown in Fig. 4.

From the results, we have the following observations. First, compared with the baseline method, we can see that all these methods improve their accuracies by representing data in lower dimensionality in most cases. This validates that although the feature of the original data is high dimensional, its underlying structure and discriminative information can be characterized and retained in a subspace with much lower dimensionality. Second, our AgLPP outperforms other methods consistently in the original feature space, which attributes to the effective approximation of the data point via the anchor based representation. Third, KAgLPP and SR AgLPP have similar performances as we expected, and they are generally better than AgLPP. Fourth, as the eigenvectors are the linear combination of the transformed data expression, i.e., $\mathbf{v} = \phi(\mathbf{X})\boldsymbol{\alpha}$ for KLPP, $\mathbf{A} = \phi(\mathbf{U})\boldsymbol{\alpha}$ for KAgLPP, and $\phi(\mathbf{X})$ is clearly denser than$\phi(\mathbf{U})$, the projecting performance of the former can be comparable or better than the latter.

4.2.2. Performance on middle datasets

In this experiment, we validate our method on two medium sized datasets, i.e. Newsgroups and USPS-All, where the anchor number is empirically set to 1000. To obtain the anchors, we particularly employ a two-layer hierarchical k-means, such as [21], to accelerate the clustering process, where the max numbers of the clusters are empirically set as 50 and 20. Via selecting labeled samples for 10 repeated times, the average classification accuracies versus different target dimensionality of the method conducted in original space and those in the new feature space are shown in Figs. 5 and 6. In our consideration, the issue of performance should include both efficiency and effectiveness. We therefore report the computational times of the reduced dimensionality with the best classification accuracies in Tables 2 and 3. Note for our methods, the total temporal cost consists of two parts, including the temporal cost for k-means based anchors selection and the remaining for real dimensionality reduction process. We list them separately in the tables as $\text{time}_{k-\text{means}} + \text{time}_{\text{rest}}$.

Similar with the results on small datasets, we can see that compared with the baseline method, the dimensionality reduction methods generally improve the classification precisions with lower dimensionality and our AgLPP outperforms other methods in the entire scope in the original feature space. And KAgLPP and SR AgLPP have similar performances as we expected, and they are generally better than AgLPP. In addition, as we can see in Fig. 5(b), features with larger dimensionality do not necessarily bring better classification results but probably introduce unnecessary noise in certain applications, because the redundant features will make the large deviation to the similarity measure.

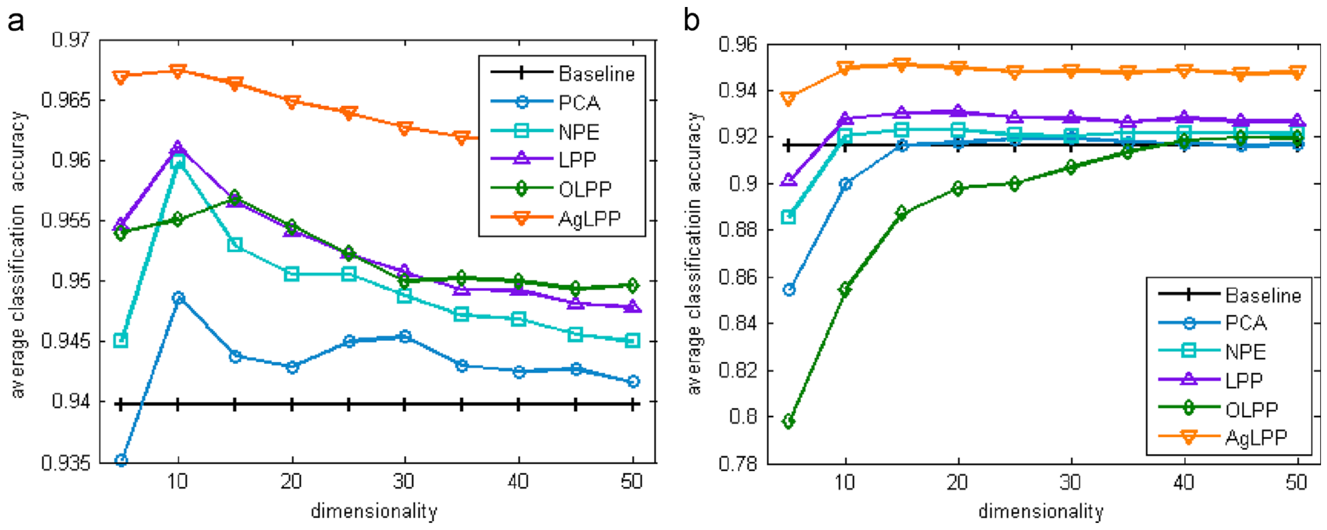


Fig. 3. Recognition rate versus dimensionality reduction on small databases in the original feature space. (a) Digit1 database and (b) USPS database.

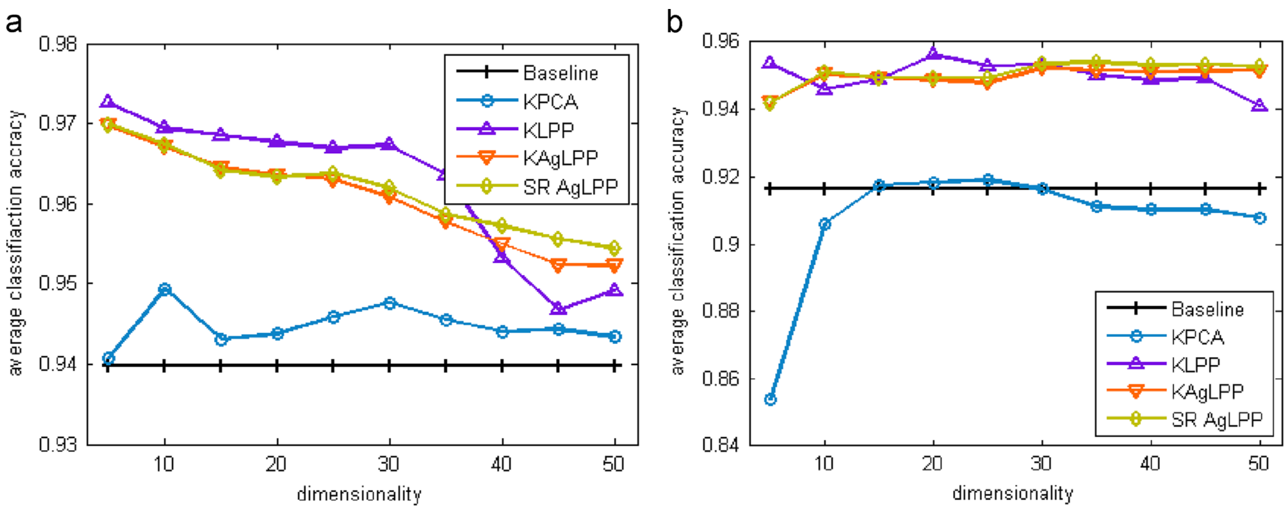


Fig. 4. Recognition rate versus dimensionality reduction on small databases in the new feature space. (a) Digit1 database and (b) USPS database.

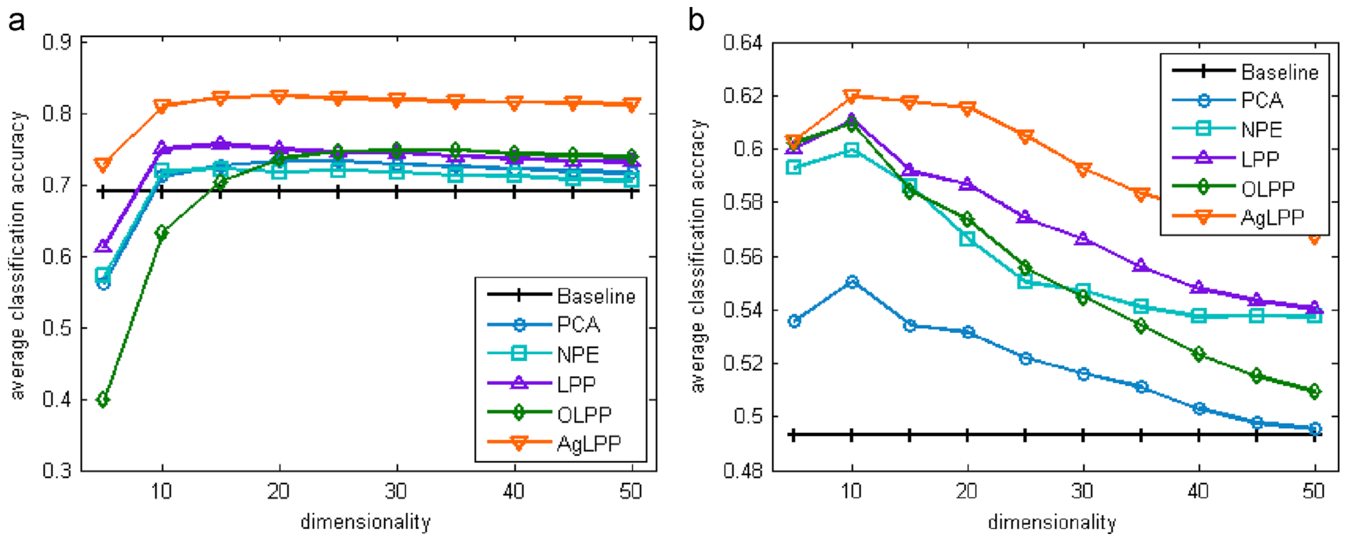


Fig.5. Recognition rate versus dimensionality reduction on middle databases in the original feature space. (a) USPS-All database and (b) Newsgroup database.

Besides the accuracies, we also pay attention to the temporal costs for implementing dimensionality reduction. From Tables 2 and 3, we find that PCA has the least time and NPE has the largest time, because PCA only needs to calculate the covariance matrix without graph construction while NPE needs to solve a minimization problem for weight estimation. We additionally see that,

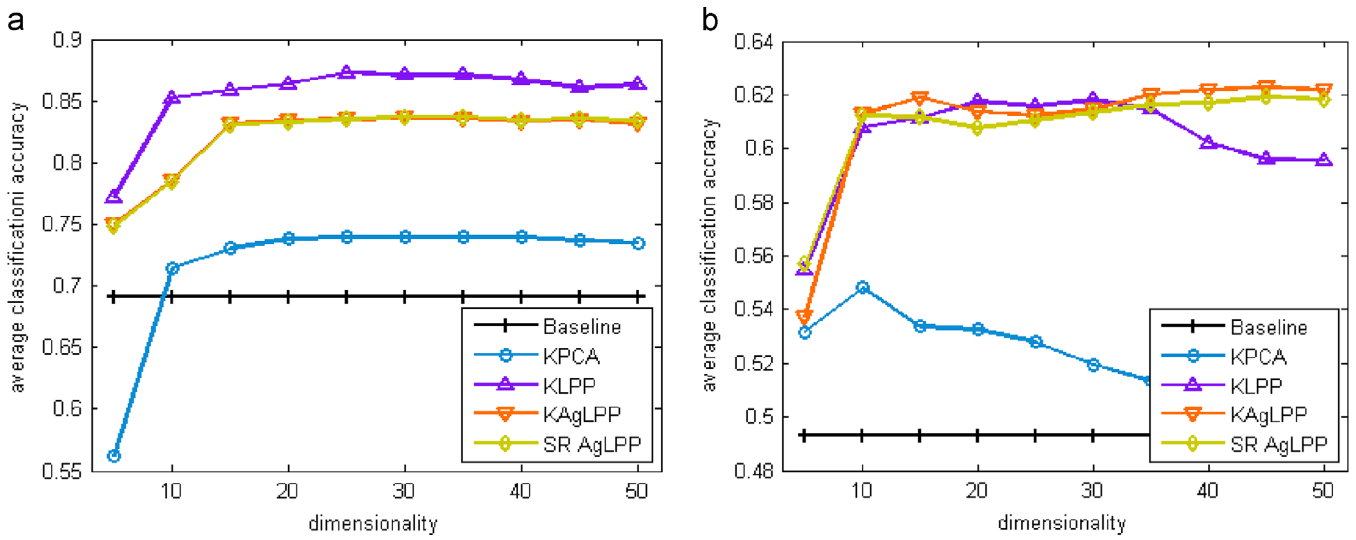


Fig. 6. Recognition rate versus dimensionality reduction on middle databases in the new feature space. (a) USPS-All database and (b) Newsgroup database.

Table 2
Implementation time (s) for dimensionality reduction in the original feature space

Method	PCA	NPE	LPP	OLPP	AgLPP
USPS-All	0.14	10.13	5.68	6.625	3.88 + 0.81
Newsgroup	0.04	25.84	11.08	10.92	2.90 + 0.85

Table 3
Implementation time (s) for dimensionality reduction in the new feature space

Method	KPCA	KLPP	KAgLPP	SR AgLPP
USPS-All	18.41	159.62	3.88 + 4.20	3.88 + 2.21
Newsgroup	34.29	496.87	2.90 + 3.29	2.90 + 1.91

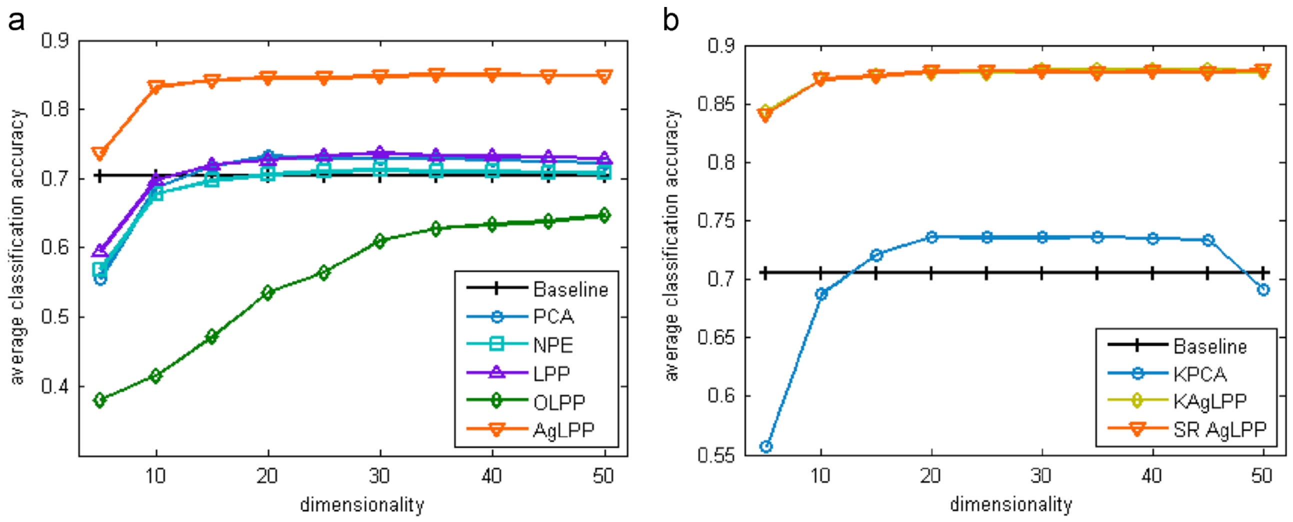


Fig.7. Recognition rate versus dimensionality reduction on MNIST-Train database. (a) In the original feature space and (b) in the new feature space.

via the hierarchical strategy of *k*-means, our methods need much less implementing time compared with the other manifold based learning methods, especially in the new transformed feature space. Therefore, although the performance of KLPP is comparable or better than KAgLPP, our method still highlights itself in largely reducing the computational cost, with an acceptable sacrifice of accuracy. We note that despite the adopted simple clustering strategy may bring the accuracy loss to our methods, we still obtain comparable or better performances than others. As an alternative, some other clustering methods can be considered to improve the performance, such as random forest clustering [45].

4.2.3. Performance on a large dataset

Now we conduct experiments on the large-scale MNIST-Train dataset. The required time for KLPP increases very fast and for this dataset, its procedures are out of memory due to the operation of

Table 4
Mean implementation time (s) for dimensionality reduction in the original feature space

Method	PCA	NPE	LPP	OLPP	AgLPP
MNIST-Train	2.09	430.43	257.00	260.93	62.18+17.08

Table 5
Implementation time (s) for dimensionality reduction in the new feature space

Method	KPCA	KAgLPP	SR AgLPP
MNIST-Train	1820.52	62.18+23.47	62.18+19.96

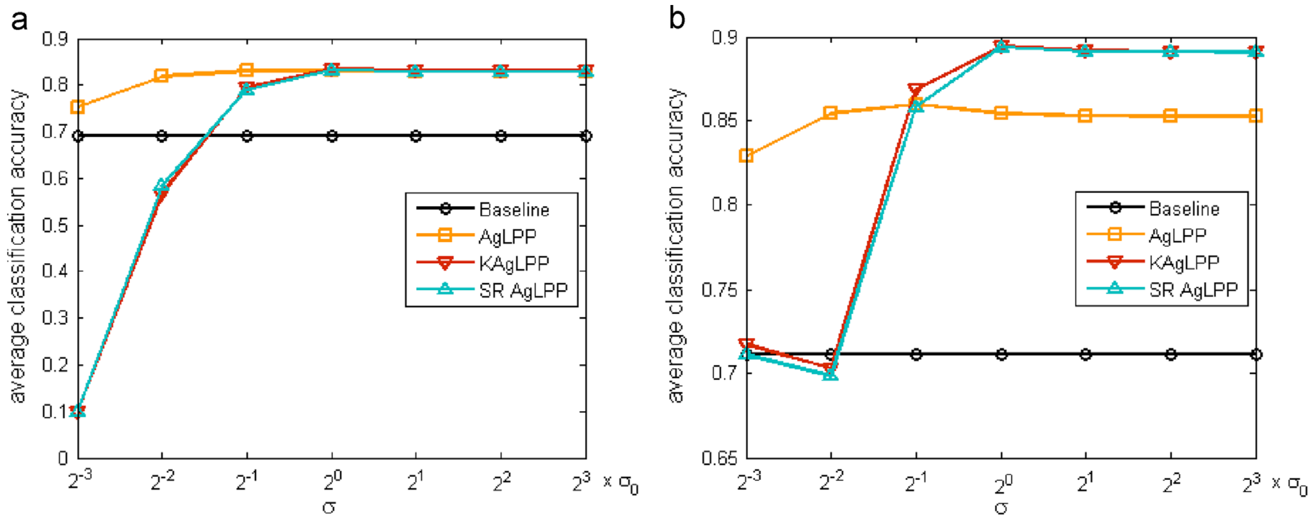


Fig. 8. Average performance curves of our methods with respect to the variation of σ. Results on the (a) USPS-All and (b) MNIST-Train.

computing **KLK** in Eq. (8). Like [14], we do not report the result of KLPP here. We empirically set the anchor number to 2000, and also employ a k-means clustering method in two layers to accelerate our methods, where the max number of the cluster is set to 50 and 40 respectively. Via selecting labeled samples for 10 repeated times, the average accuracies for increase dimensionality of the method conducted in original space are displayed in Fig. 7 (a), and those in the new feature space are shown in Fig. 7 (b) respectively. And the computational costs with the best classification accuracies are recorded in Tables 4 and 5.

As can be seen, our AgLPP algorithm outperforms the other methods in the original feature space, which again validates the effectiveness of the approximation of the data point via the anchor based representation. And we can also observe that KAgLPP and SR AgLPP have very similar performances, and they are both superior to KPCA. We additionally compare the temporal costs for the involved methods. In both Tables 4 and 5, we observe that our methods have the best performances among all the manifold based methods in terms of the implementing time. In general, our methods demonstrate better effectiveness and efficiency than other closely related methods.

4.3. Performance on varying the parameter σ

Finally, we test the sensitivity of the parameter σ, which is used in the proposed algorithms including AgLPP, KAgLPP, and SR AgLPP. For convenience, we implement experiment on the USPS-All and MNIST-Train datasets and simply set anchor number m and reduced dimensionality d to proper values, e.g., $m=1000, d=20$ for the former and $m=2000, d=20$ for the latter. We vary σ from $\sigma_0 \times 2^{-3}, 2^{-2}, 2^{-1}, 1, 2^1, 2^2, 2^3$ where σ_0 is the averaged distance between two points in the data set. Fig. 8 demonstrates the performance curve with respect to the variation of σ. From the figure, we observe that the performance of AgLPP is robust to this parameter, and the performances of other two versions stay at a stable level when the parameter becomes large.

5. Conclusions

In this paper, we propose AgLPP by addressing the challenges of the traditional dimensionality reduction methods in terms of computational complexity and storage cost. In addition, we extend AgLPP into two novel variants which are implemented in the new transformed feature space. As both the time and storage costs of the AgLPP grow linearly with the data size, this method is potentially useful in dealing with much larger datasets. Furthermore, our strategy that employs the Anchorgraph construction for describing the pairwise relationship between data points is universal and can be easily leveraged in many retrieval and classification tasks [46–50].

Acknowledgments

This work was supported by National Natural Science Foundation of China (Grant no. 61301222), China Postdoctoral Science Foundation (Grant no. 2013M541821), Fundamental Research Funds for the Central Universities (Grant nos. 2013HGQC0018, 2013HGBH0027, 2013HGBZ0166)

References

[1] M. Guillaumin, J. Verbeek and C. Schmid, Multimodal semi-supervised learning for image classification, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2010.
[2] J. Yu, Y. Rui, Y. Tang, D. Tao, High-order distance-based multiview stochastic learning in image classification, IEEE Trans. Cybern. 44 (12) (2014) 2431–2442.
[3] R. Hong, M. Wang, Y. Gao, D. Tao, Image annotation by multiple-instance learning with discriminative feature mapping and selection, IEEE Trans. Cybern. 44 (5) (2014) 669–680.
[4] M. Wang, H. Li, D. Tao, K. Lu, X. Wu, Multimodal graph-based reranking for web image search, IEEE Trans. Image Process. 21 (11) (2012) 4649–4661.
[5] X. Liu, and B. Huet, Concept detector refinement using social videos, in: Proceedings of the International Workshop on Very-Large-Scale Multimedia Corpus, Mining and Retrieval, 2010.
[6] J. Yu, D. Tao, M. Wang, Adaptive hypergraph learning and its application in image classification, IEEE Trans. Image Process. 21 (7) (2012) 3262–3272.
[7] H. Hotelling, Analysis of a complex of statistical variables into principal components, J. Educ. Psychol. 24 (6) (1933) 417.
[8] J.B. Tenenbaum, Mapping a manifold of perceptual observations, in: Proceedings of the Conference on Advances in Neural Information Processing Systems, 1998, pp. 682–688.
[9] S.T. Roweis, L.K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323–2326.
[10] M. Belkin, P. Niyogi, Laplacian eigenmaps and spectral techniques for embedding and clustering, in: Proceedings of the Conference on Advances in Neural Information Processing Systems, vol. 14, 2001, pp. 585–591.
[11] X. He, and P. Niyogi, Locality preserving projections, in: Proceedings of the Conference on Advances in Neural Information Processing Systems, vol. 16, 2004, p. 153.
[12] W. Liu, J. He, S.F. Chang, Large graph construction for scalable semi-supervised learning, in: Proceedings of the 27th International Conference on Machine Learning, 2010, pp. 679–686.
[13] W. Liu, J. Wang, S.F. Chang, Robust and scalable graph-based semisupervised learning, Proceedings of the IEEE, 100(9), 2012, pp. 2624–2638.
[14] B. Xu, J. Bu, C. Chen, C. Wang, D. Cai, X. He, EMR: a scalable graph-based ranking model for content-based image retrieval, IEEE Trans. Knowl. Data Eng. 27 (1) (2013) 102–114.
[15] W. Fu, S. Hao, M. Wang, Active learning on anchorgraph with an improved transductive experimental design, Neurocomputing 171 (1) (2016) 452–462.
[16] J.B. Kruskal, Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis, Psychometrika 29 (1) (1964) 1–27.
[17] K.Q. Weinberger, L.K. Saul, An introduction to nonlinear dimensionality reduction by maximum variance unfolding, In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 6, 2006, pp. 1683–1686.
[18] D.L. Donoho, C. Grimes, Hessian eigenmaps: locally linear embedding techniques for high-dimensional data, in: Proceedings of the National Academy of Sciences, vol.100, no. 10, 2003, pp. 5591–5596.
[19] X. He, D. Cai, S. Yan, H.J. Zhang, Neighborhood preserving embedding, in: Proceedings of the Tenth IEEE International Conference on Computer Vision, vol. 2, 2005, pp. 1208–1213.
[20] D. Cai, X. He, J. Han, Isometric projection, in: Proceedings of the National Conference on Artificial Intelligence, vol. 22, no. 1, 2007, p. 528.
[21] B. Cheng, L. Zhuo, J. Zhang, Comparative study on dimensionality reduction in large-scale image retrieval, Neurocomputing 141 (2014) 202–210.
[22] J.P. Cunningham, M.Y. Byron, Dimensionality reduction for large-scale neural recordings, Nat. Neurosci. 17 (11) (2014) 1500–1509.
[23] B. Schölkopf, A. Smola, K.R. Müller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Comput. 10 (5) (1998) 1299–1319.
[24] J. Ham, D.D. Lee, S. Mika, B. Schölkopf, A kernel view of the dimensionality reduction of manifolds, in: Proceedings of the ACM Twenty-first International Conference on Machine Learning, 2004, p. 47.
[25] K.Q. Weinberger, F. Sha, L.K. Saul, Learning a kernel matrix for nonlinear dimensionality reduction, in: Proceedings of the ACM Twenty-first International Conference on Machine Learning, 2004, p. 106.
[26] V.A. Zimmer, K. Lekadir, C. Hoogendoorn, A.F. Frangi, G. Piella, A framework for optimal kernel-based manifold embedding of medical image data, Comput. Med. Imaging Graph. 41 (2015) 93–107.
[27] A. Gisbrecht, A. Schulz, B. Hammer, Parametric nonlinear dimensionality reduction using kernel t-sne, Neurocomputing 147 (2015) 71–82.
[28] Y. Han, F. Wu, D. Tao, J. Shao, Y. Zhuang, J. Jiang, Sparse unsupervised dimensionality reduction for multiple view data, IEEE Trans. Circuits Syst. Video Technol. 22 (10) (2012) 1485–1496.
[29] X. Zhu, Z. Huang, Y. Yang, H. Shen, C. Xu, J. Luo, Self-taught dimensionality reduction on the high-dimensional small-sized data, Pattern Recognit. 46 (1) (2013) 215–229.
[30] B.M. Kandel, D.J. Wang, J.C. Gee, B.B. Avants, Eigenanatomy: sparse dimensionality reduction for multi-modal medical image analysis, Methods 73 (2015) 43–53.
[31] X. Shi, Z. Guo, Z. Lai, Y. Yang, Z. Bao, D. Zhang, A framework of joint graph embedding and sparse regression for dimensionality reduction, IEEE Trans. Image Process. 24 (4) (2015) 1341–1355.
[32] J. Cheng, Q. Liu, H. Lu, Y.W. Chen, Supervised kernel locality preserving projections for face recognition, Neurocomputing 67 (2005) 443–449.
[33] S. Yan, D. Xu, B. Zhang, H.J. Zhang, Q. Yang, S. Lin, Graph embedding and extensions: a general framework for dimensionality reduction, IEEE Trans. Pattern Anal. Mach. Intell. 29 (1) (2007) 40–51.
[34] J. Gui, Z. Sun, W. Jia, R. Hu, Y. Lei, S. Ji, Discriminant sparse neighborhood preserving embedding for face recognition, Pattern Recognit. 45 (8) (2012) 2884–2893.
[35] B. Li, J. Li, X.P. Zhang, Nonparametric discriminant multi-manifold learning for dimensionality reduction, Neurocomputing 152 (2015) 121–126.
[36] P. Li, M. Wang, J. Cheng, C. Xu, H. Lu., Spectral hashing with semantically consistent graph for image indexing, IEEE Trans. Multimed. 15 (1) (2013) 141–152.
[37] J. Chen, C. Leng, P. Li, M. Wang, H. Lu., Semi-supervised multi-graph hashing for scalable similarity search, Comput. Vis. Image Underst. 124 (2014) 12–21.
[38] M. Norouzi, A. Punjani, D.J. Fleet, Fast exact search in hamming space with multi-index hashing, IEEE Trans. Pattern Anal. Mach. Intell. 36 (6) (2014) 1107–1119.
[39] S.K. Biswas, P. Milanfar, Laplacian object: one-shot object detection by locality preserving projection, in: Proceedings of the 2014 IEEE International Conference on Image Processing, 2014, pp. 4062–4066.
[40] L. Shu, T. Ma, L.J. Latecki, Locality preserving projection for domain adaptation with multi-objective learning, In: Proceedings of the 28th AAAI Conference on Artificial Intelligence, 2014.
[41] T. Hastie, R. Tibshirani, J. Friedman, T. Hastie, J. Friedman, R. Tibshirani, The Elements Of Statistical Learning, Springer, New York, 2009.
[42] ⟨http://olivier.chapelle.cc/ssl-book/benchmarks.html⟩.
[43] ⟨http://www.cs.nyu.edu/~roweis/data.html⟩.
[44] D. Cai, X. He, J. Han, H.J. Zhang, Orthogonal laplacianfaces for face recognition, IEEE Trans. Image Process. 15 (11) (2006) 3608–3614.
[45] T. Shi, S. Horvath, Unsupervised learning with random forest predictors, J. Comput. Graph. Stat. 15 (1) (2006).
[46] J. Wang, C. Lu, M. Wang, P. Li, S. Yan, X. Hu, Robust face recognition via adaptive sparse representation, IEEE Trans. Cybern. 44 (12) (2014) 2368–2378.
[47] M. Wang, G. Li, Z. Lu, Y. Gao, T. Chua, When amazon meets google: product visualization by exploring multiple information sources, ACM Trans. Internet Technol. 12 (4) (2013) 1–17.
[48] M. Wang, Y. Gao, K. Lu, Y. Rui., View-based discriminative probabilistic modeling for 3D object retrieval and recognition, IEEE Trans. Image Process. 22 (4) (2013) 1395–1407.
[49] L. Zhang, Y. Yang, R. Zimmermann, Discriminative cellets discovery for fine-grained image categories retrieval, in: Proceedings of International Conference on Multimedia Retrieval, 2014.
[50] T. Jin, J. Yu, J. You, K. Zeng, C. Li, Z. Yu, Low-rank matrix factorization with multiple hypergraph regularizer, Pattern Recognit. 48 (3) (2015) 1011–1022.

Rui Jiang is pursing the master degree in School of Computer and Information, Hefei University of Technology (HFUT). His research interests include pattern recognition and image processing.

Weijie Fu is pursing the bachelor degree in School of Computer and Information, Hefei University of Technology (HFUT). His research focuses on machine learning.

Li Wen is pursing the bachelor degree in School of Computer and Information, Hefei University of Technology (HFUT). Her research interests include information retrieval and pattern recognition.

Shijie Hao received his Ph.D. from Hefei University of Technology (HFUT) in 2012. He is currently an assistant professor in School of Computer and Information, HFUT. His research interests are machine learning, image processing and multimedia content analysis.

Richang Hong received his Ph.D. from University of Science and Technology of China (USTC) in 2008. He is currently a professor in School of Computer and Information, HFUT. His research interests are multimedia content analysis, pattern recognition and data mining.

ExAD: An Ensemble Approach for Explanation-based Adversarial Detection

Raj Vardhan
Texas A&M University
raj_vardhan@tamu.edu

Ninghao Liu
Texas A&M University
nhliu43@tamu.edu

Phakpoom Chinprutthiwong
Texas A&M University
cpx0rpc@tamu.edu

Weijie Fu
Hefei University of Technology
fwj.edu@gmail.com

Zhenyu Hu
Texas A&M University
johnhu@tamu.edu

Xia Ben Hu
Texas A&M University
hu@cse.tamu.edu

Guofei Gu
Texas A&M University
guofei@cse.tamu.edu

ABSTRACT

Recent research has shown Deep Neural Networks (DNNs) to be vulnerable to adversarial examples that induce desired misclassifications in the models. Such risks impede the application of machine learning in security-sensitive domains. Several defense methods have been proposed against adversarial attacks to detect adversarial examples at test time or to make machine learning models more robust. However, while existing methods are quite effective under blackbox threat model, where the attacker is not aware of the defense, they are relatively ineffective under whitebox threat model, where the attacker has full knowledge of the defense.

In this paper, we propose ExAD, a framework to detect adversarial examples using an ensemble of explanation techniques. Each explanation technique in ExAD produces an explanation map identifying the relevance of input variables for the model's classification. For every class in a dataset, the system includes a detector network, corresponding to each explanation technique, which is trained to distinguish between normal and abnormal explanation maps. At test time, if the explanation map of an input is detected as abnormal by any detector model of the classified class, then we consider the input to be an adversarial example. We evaluate our approach using six state-of-the-art adversarial attacks on three image datasets. Our extensive evaluation shows that our mechanism can effectively detect these attacks under blackbox threat model with limited false-positives. Furthermore, we find that our approach achieves promising results in limiting the success rate of whitebox attacks.

1 INTRODUCTION

In recent years, Deep Neural Networks (DNNs) are being increasingly adopted in a wide range of tasks such as face-recognition [25], natural language processing [19], and malware classification [9]. This trend can be attributed to the superior performance achieved by DNNs in solving computational tasks that rely on high-dimensional data. However, increasing adoption of DNNs to security-critical applications, such as self-driving cars and malware classification, is hindered by the vulnerability of DNNs to adversarial attacks [15, 31, 39, 47]. Specifically, minor yet carefully computed perturbations to natural inputs can cause DNNs to misclassify.

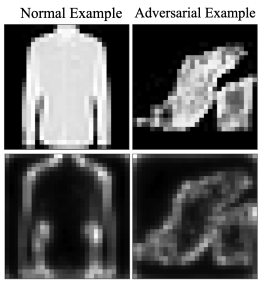


Figure 1: Intuition behind the proposed ExAD framework.

Several methods have been proposed for defending against adversarial examples. One direction of research is to improve the robustness of neural networks, such as through adversarial training [15] or gradient masking [17]. However, subsequent works have shown that neural network architectures modified with such techniques can still be attacked [7]. Another research direction is adversarial detection, where the goal is to detect if an input is an adversarial example or a normal example. Early works in this area either used a second neural network [14, 16, 36], or statistical tests [3, 18, 28] to classify between normal and adversarial examples. However, Carlini et al. [6] showed that while most of these mechanisms are successful against blackbox attacks, they lack robustness to whitebox attacks, where the adversary has knowledge of the defense. Although many recent methods have enhanced the detection of blackbox adversarial attacks [33–35, 50], improving the robustness to whitebox attacks remains an open problem.

One way of uncovering the reasons for the resulting misclassification of an adversarial example can be understanding why the model predicts what it predicts through explanation techniques [1, 2, 23, 37, 41, 42, 44, 46]. For an image input, the result from an explanation technique encodes the relevance of each pixel for the prediction result and is commonly referred to as an *explanation map*. Our hypothesis is that the explanation map of an adversarial example being misclassified as the target class may not be consistent with explanation maps generated for correctly classified normal examples of that class. We term the former type as *abnormal explanations* and the latter as *normal explanations* throughout this paper. Figure 1 shows an intuitive example where we can observe that the explanation map of an adversarial example classified into the shirt class (bottom-right) is quite distinguishable from that of a normal example of the targeted class (bottom-left). Overall, the distinguishability between normal and abnormal explanation maps guides us in exploring the effectiveness of using explainability as a tool for detecting adversarial examples.

However, a defense method that relies on a single explanation technique may still not be robust under whitebox setting. An adaptive adversary can leverage recent findings which show that explanations can be unreliable [22] and can be manipulated to produce a target explanation map [10, 52]. Such an adaptive adversary can generate adversarial examples that not only fool the target model into producing desired misclassifications, but also fool the targeted explanation technique into producing normal explanation maps. Towards building a mitigation strategy, we take motivation from previous work on N-variant systems [8]. To provide higher resistance against attacks on software vulnerabilities, these systems combine multiple variants with disjoint exploitation sets into a single system. In context of our work, we propose to use an ensemble of multiple kinds of explanation techniques. The benefit of this approach is that it requires an adaptive adversary to construct an adversarial example that fools the target model and simultaneously fools all explanation techniques. By incorporating diverse explanation techniques, we can reduce the probability that an attacker will achieve this goal.

In this work, using the above insights, we propose ExAD, an Ensemble approach for Explanation-based Adversarial Detection. ExAD uses an ensemble of explanation techniques wherein each technique provides an explanation map for every classification decision by a target model. To introduce explanation diversity, we include both gradient-based [2, 41–43, 46] and propagation-based [1, 37, 41] explanation techniques in ExAD. Furthermore, for any class in a dataset, the system includes a detector model associated with each explanation technique. The detector model determines if an explanation map produced by the respective explanation technique is normal or not for that class. The key idea here is to use the distinguishability between normal and abnormal explanations for any class. Finally, for a test input classified into a particular class, if the explanation map produced by any technique is detected as abnormal by the corresponding detector model, then we classify the input as an adversarial example.

We evaluate ExAD using six state-of-the-art adversarial attacks on three image datasets, namely MNIST [27], Fashion-MNIST (FM-NIST) [49] and CIFAR-10 [24]. We first perform the evaluation under the blackbox threat model. Our experimental results show that we can effectively detect all attacks, achieving a detection rate above 98% (many having 100% detection rate) across the three datasets with a low false-positive rate of under 1.1%.

More importantly, we further evaluate ExAD under whitebox threat model. We build on previous research [10, 52], and create a strong adaptive adversary to generate adversarial examples that fool the target model as well as a target explanation technique. Through experimental results, we make an interesting finding on the transferability of adaptive attacks on explanation-based detector models. We observe that on targeting a propagation-based technique, the resulting adversarial examples are more successful in fooling detector models of other propagation-based techniques (into misclassifying an explanation map as normal) as compared to fooling detector models of gradient-based techniques. Likewise, we find that targeting a gradient-based technique transfers better to the detector model of the other gradient-based technique compared to those of propagation-based techniques. Using an ensemble of detector models corresponding to diverse techniques, ExAD achieves a mean detection rate of over 88% for this whitebox attack across the three datasets. The results indicate that our proposed defense can significantly limit the success rate of such whitebox attacks. Additionally, we find that our ensemble approach makes it considerably harder for attackers to perform more advanced whitebox attacks, such as simultaneously targeting all explanation techniques.

We summarize our main contributions as follows.

- We develop a novel framework called ExAD to detect adversarial examples. ExAD uses an ensemble of diverse explanation techniques to improve the robustness against whitebox attacks.
- We evaluate ExAD on six state-of-the-art adversarial attacks and three image datasets under blackbox threat model. The results show that the proposed system can consistently achieve high detection rates with a low false-positive rate.
- We extensively evaluate ExAD under whitebox threat model by creating a strong adaptive adversary which targets the classification model as well as an explanation technique. Our findings show that ExAD achieves promising results in limiting the success rate of whitebox attacks.

The rest of the paper is organized as follows. In Section 2, we review background and related work. In Section 3, we introduce our proposed framework in light of two applicable threat models. Subsequently, we report experimental results and comparison with state-of-the-art detection methods in Section 4. Then, we discuss aspects such as the fragility of explanations and limitations of our work in Section 5. Finally, we conclude the paper in Section 6.

2 BACKGROUND AND RELATED WORK

2.1 Neural networks

A DNN is a computational graph of elementary computing units, called neurons, organized into layers that represent the extraction of successive representations from the input. We use notations consistent with previous work [6, 33] to denote an m-class DNN as a function $f : \mathbb{R}^d \to \mathbb{R}^m$. The i-th layer of the network computes

$$f^i(x) = ReLU(W^i f^{i-1}(x) + b^i)$$

where W^i is a weight matrix, b^i is a vector of bias values, and ReLU is a non-linear activation function. Let $Z(x)$ denote a vector of m elements representing the output of the last layer (before softmax), known as *logits*, i.e., $Z(x) = f^n(x)$. A softmax function is used to obtain the normalized output of the network given by $y = f(x) = \text{softmax}(Z(x))$ where $x \in \mathbb{R}^d$ and $y \in \mathbb{R}^m$ with y_i representing the probability of the input being recognized as class i. Then, we represent the classification of $f(\cdot)$ on x by $C(x) = \text{argmax}_i(f(x)_i)$. At test-time, a trained model is provided with test inputs X_t, and for each input $x_t \in X_t$, the model assigns its classification to be $C(x_t) = argmax_i(f(x_t)_i)$. The classification is considered correct if $C(x_t)$ is same as the true label $C^*(x_t)$.

2.2 Adversarial examples

Adversarial examples are crafted by imperceptibly perturbing normal inputs to cause DNNs into misclassifying them. Formally, an input to the classifier $f(\cdot)$ is termed as normal if it occurs naturally [35] or was benignly created [6]. Then, given a normal input $x \in \mathbb{R}^d$ with correctly classified class $C(x) = c$, we call x' an (*untargeted*) adversarial example if it is close to x, i.e., $\Delta(x, x') < \epsilon$ and $C(x') \neq c$, where $\Delta(.)$ denotes a measure of similarity between two inputs and ϵ is a threshold that limits the permissible perturbations in the adversarial example. In a more restrictive case, an attacker could also target a desired class $t \neq c$ and find a x' close to x such that $C(x) = c$ and $C(x') = t$. We call x' a *targeted* adversarial example.

In the case of images, the closeness function $\Delta(.)$ and threshold ϵ should be chosen such that the adversarial example and its *seed image* (normal counterpart) are indistinguishable to a human eye. To define $\Delta(.)$, a popular distance metric is the L_p norm, defined as $\|d\|_p = \left(\sum_{i=0}^{n} |v_i|^p\right)^{\frac{1}{p}}$. Common choices for L_p include: L_0, a measure of the number of pixels which have different values in corresponding positions in two images; L_2, which measures the standard Euclidean distance; or L_∞, a measure of the maximum change among all pixels at corresponding places in two images.

2.3 Existing attacks

Researchers have developed a number of methods for constructing adversarial examples. Broadly, these methods can be categorized into *gradient-based* attacks [7, 15, 47], which leverage gradient-based optimizations, and *content-based* attacks [4, 12], where perturbations are made in accordance with the semantics of the input content to simulate real-world scenarios. In this paper, we focus on six state-of-the-art gradient-based attacks for neural network classifiers, namely Jacobian-based Saliency Map Attack (JSMA) [39], Basic Iterative Method (BIM) [26], Momentum Iterative Method (MIM) [11], and Carlini and Wagner Attacks (CW) [7] tailored to L_0, L_2, and L_∞ norms. For more details, we refer interested readers to the original papers and to recent works on adversarial detection [33, 35] which provide a good summary of these attacks.

2.4 Existing work on adversarial detection

Adversarial detection is a defense approach with the goal of building a classifier g with a binary output $y \in \{0, 1\}$, where labels 0 and 1 denote that the input instance is normal or adversarial, respectively. We briefly review state-of-the-art works in detecting adversarial examples, and divide them into three categories as below.

Training a Detector. First, we can use adversarial examples to train detectors. The input into detectors can be chosen as data instances in raw feature space or the intermediate representation space of the target model. Using the former strategy, Gong *et al.* show that a simple binary classifier can learn to separate normal and adversarial instances [14]. In a related work, Grosse et al. add a new class, solely for adversarial examples, in the output layer of the model [16]. But, modifying the model architecture impacts the accuracy on normal examples. Based on the latter approach, Metzen *et al.* use representations generated by inner deep neural network layers as inputs into detectors which are augmented to the classification network [36]. By freezing the weights of the classification network before training the detectors, this method does not affect the classification accuracy on normal examples. However, in subsequent work, Carlini et al. showed that these detectors don't generalize well and lack robustness to whitebox attacks [6]. In our work, we mitigate the generalization challenge by including an attack-independent defense setting (discussed in Section 3.4.2).

Statistical Metrics. Second, we can use statistical metrics to design detectors. Grosse *et al.* [16] study two statistical distance measures, Maximum-Mean-Discrepancy and Energy Distance, where a sample is regarded as adversarial if it is rejected by statistical testing. Ma et al. estimate an LID value which assesses the space-filling capability of the region around an example by measuring the distance distribution with respect to its neighbors [34]. The authors demonstrate that estimated LID of adversarial examples tends to be much higher than that of normal examples. However, a challenge faced by these approaches is in developing more effective and transferable metrics to separate clean instances from adversarial examples generated by different attacks.

Prediction Abnormality. Third, we can also resort to detecting the abnormality of input instances. Meng and Chen proposed Magnet [35] which learns to approximate the manifold of normal examples using autoencoders. Another method called Feature Squeezing [50] proposes reducing the degree of freedom of an adversary, such as by smoothing images or minimizing their color depth. Another recent work called Neural-network Invariant Checking (NIC) proposed leveraging the provenance channel and the activation value distribution channel in DNNs by showing that adversarial examples tend to violate either provenance invariant or value invariant [33]. However, while these methods have improved the detection rates on blackbox attacks, they have shown very limited success against whitebox attacks. Zhang *et al.* proposed a detection method based on perturbation of saliency maps [51]. The authors find that on adding adversarial perturbations, the saliency of the adversarial example is also perturbed compared to that of the seed image. However, this difference between saliency may not be effective because, at test-time, we do not know the class from which an adversarial example originated. Therefore, we do not know what its normal saliency would look like had the example not been perturbed. Besides, explanations have also been used by Liu et al. for a different goal of crafting adversarial examples [30]. In this paper, we further explore the premise of using explanations, but to detect adversarial examples and based on a fundamentally different approach. Our work was done concurrently with a similar approach presented by Wang et al. [48]. In contrast, we have the following differentiating aspects. First, our work offers a more detailed evaluation on whitebox attacks. Second, we provide a discussion on the fragility of an explanations-based defense. Finally, we compare the proposed method with a number of state-of-the-art detection systems.

2.5 Explainable machine learning

Our work utilizes recent advances in explainable machine learning. Specifically, we focus on *local explainability* methods [2, 29] which explain the output of DNN models for a given input. For computer vision models, these techniques identify which regions in an input image are most responsible for the prediction result. The explanation result is often termed as a *saliency map* [42], or more generally, an *explanation map* [10]. Naturally, our defense is compatible with models that are inherently explainable (e.g., linear models) and models that produce an explanation result along with the prediction [32, 40]. However, we focus on local explainability methods as they build on top of existing models. This allows us to add our adversarial detection capability to any existing blackbox model without sacrificing its prediction power for explainability, or putting the burden of producing explanations during classification.

Among local explanation techniques, backpropagation-based methods have gained considerable attention. These can be further categorized into the following. The first is *gradient-based* techniques which rely on the gradient of the neural network function to generate explanations [2, 41–43, 46]. The second category is *propagation-based* techniques [1, 37, 41]. These techniques view the neural network as a computational graph, and generate explanations by starting with the prediction score at the output layer and progressively redistributing it backwards by means of propagation rules until the input layer is reached. To achieve diversity in explanation methods, we use both gradient-based and propagation-based explanation techniques, which we discuss further in Section 3.3.

3 DESIGN

3.1 Threat model

In designing our defense, we assume that the attacker has complete knowledge of the target classifier $f(\cdot)$ including its architecture and parameters. This is a conservative and practical assumption, consistent with prior works [33, 35, 50]. Also, depending upon whether the attacker has knowledge of the defense, we consider two types of threat models. First, we consider *blackbox* attack, where the attacker does not have any knowledge of the defense mechanism. Second, we consider *whitebox* attack, where the attacker has complete knowledge of the defense mechanism including its structure and parameters. For ExAD, this implies that the attacker has full knowledge of the explanation techniques and detector models.

3.2 Overview of ExAD

ExAD is a framework that uses an ensemble of explanation techniques to detect adversarial examples. The role of explanation techniques is to allow ExAD to examine the reasons for the misclassification of adversarial examples. Our hypothesis is that the explanations of adversarial examples being misclassified as the target class (*abnormal explanations*) may not be consistent with explanations generated for correct classifications of normal examples of that class (*normal explanations*). Our design relies upon the consistency of normal explanations, and their distinguishability from abnormal explanations. We provide an intuitive example for the distinguishability aspect in Figure 1. Further examples showing the consistency aspect can be found in Figure 5 in Appendix A. While we provide such motivating examples, it is worth noting that an explanation itself may be incomprehensible to humans as recent work has shown neural networks to use non-robust features (that may not align with human perception of a class) to make predictions [20]. Therefore, even the distinguishability may not necessarily be apparent to humans. Nevertheless, we empirically show that we can train detector models to learn to distinguish between normal and abnormal explanations.

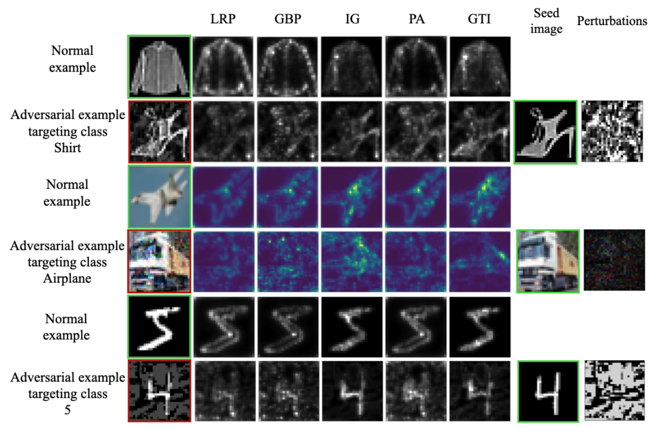

LRP
GBP
IG
PA
GTI
Seed image
Perturbations
Normal example
Adversarial example targeting class Shirt
Normal example
Adversarial example targeting class Airplane
Normal example
Adversarial example targeting class 5

Figure 2: Distinguishability between normal and abnormal explanations using different explanation techniques.

An overview of our approach is as follows. First, we train the target model as usual on the clean training set. Second, we use a set of diverse explanation techniques to generate explanation maps for normal examples of each class. Third, for every class, we train a detector model corresponding to each technique. The detector model identifies if the explanation map of an example is normal for the classified class. We study two approaches to build the detector model: a binary classifier approach (where we use both normal and abnormal explanations) and an anomaly detection approach (where we only use normal explanations). Whereas the former setting makes the training and validation process simple (once we have the required data), the motivation for the latter setting is to make our defense attack-independent so that it is more likely to generalize to unknown attacks. At test time, if the explanation map of an input is classified as abnormal by any detector model of the classified class, then we consider it to be an adversarial example.

3.3 Generation of explanations

Given a neural network classifier $f(\cdot)$ and an input x, the explanation of the classification of x is represented as an explanation map denoted by $h : \mathbb{R}^d \to \mathbb{R}^d$. The explanation map $h(x)$ encodes the relevance score of every pixel in x for the neural network's prediction. We consider the following explanation generation techniques towards building an ensemble of methods.

- **Gradient**: The gradient of the output $f(x)$ with respect to the input x is indicative of how infinitesimal changes in each pixel can influence the output [2, 42]. The explanation map using the gradient method is given by

$$h(x) = \frac{\partial f}{\partial x}(x)$$

- **Gradient * Input (GTI)**: This method computes an element-wise product between the gradient-based explanation map of Simonyan et al. [42] and the input to quantify the influence of each pixel on the prediction score [41]. Formally, the explanation map produced by gradient * input is given by

$$h(x) = x \odot \frac{\partial f}{\partial x}(x)$$

- **Integrated Gradients (IG)**: In contrast to GTI, which performs a single computation of the gradient at the input x, integrated gradients computes the gradients at all points along a linear path from a baseline $\bar{x}$ to x, and averages them [46]. The baseline $\bar{x}$ can be defined by the user and is generally chosen as a black image. Formally,

$$h(x) = (x - \bar{x}) \odot \int_{\alpha=0}^{1} \frac{\partial f(\bar{x} + \alpha(x - \bar{x}))}{\partial x} \mathrm{d}\alpha$$

- **Guided Backpropagation (GBP)**: This method is an extension of gradient-based explanation with the key difference that it prevents backward flow of negative gradients through non-linearities, such as ReLUs [44].
- **Layer-wise Relevance propagation (LRP)**: To explain the prediction of class c, LRP [1, 37] starts with the output neuron of class c and goes backwards through the network by following the z^+ rule for all layers except the first.

$$R_i^l = \sum_j \frac{x_i^l (W^l)_{ji}^+}{\sum_i x_i^l (W^l)_{ji}^+} R_j^{l+1}$$

  Here, i and j are two neurons of consecutive layers, R_i^l denotes the relevance of i-th neuron in the l-th layer, x_i^l represents the activation vector, and $(W^l)_{ji}^+$ denotes the positive weight between the two neurons. Then, to account for the bounded range of an input, we use the $z^{\mathcal{B}}$ rule in the first layer

$$R_i^0 = \sum_j \frac{x_j^0 W_{ji}^0 - l_j (W^0)_{ji}^+ - h_j (W^0)_{ji}^-}{\sum_i (x_j^0 W_{ji}^0 - l_j (W^0)_{ji}^+ - h_j (W^0)_{ji}^-)} R_j^1$$

  where l and h are the lowest and highest allowed pixel values, respectively.
- **Pattern Attribution (PA)**: Kindermans et al. [23] proposed patter attribution as an improvement over the LRP framework. The method is analogous to the backpropagation operation with the weights in the backward pass replaced by element-wise multiplication of weights W^l and learned patterns A^l.

While we considered all six of the above-mentioned explanation techniques to include in ExAD, we found the performance of the gradient method ($h(x) = \frac{\partial f}{\partial x}(x)$) to be unacceptable based on evaluations on the validation sets, whereas remaining techniques performed significantly better. Therefore, in this work, ExAD uses an ensemble of $k = 5$ techniques- LRP, GBP, IG, PA, and GTI.

Figure 2 shows examples of normal and abnormal explanations produced by different techniques used in ExAD. When a test input x_t is classified by the target model $f(\cdot)$ as class c, each of the k techniques produce an explanation map for this classification. In column 1, the first, third, and fifth rows show a normal example from FMNIST, CIFAR-10, and MNIST datasets, respectively. In the same column, the second, fourth, and sixth rows show an adversarial example which is misclassified as the class represented by the normal example in the preceding row. Columns 2-6 show the corresponding explanation maps produced by the five explanation techniques. The distinguishability between explanation maps of normal and adversarial examples allows the detector models to

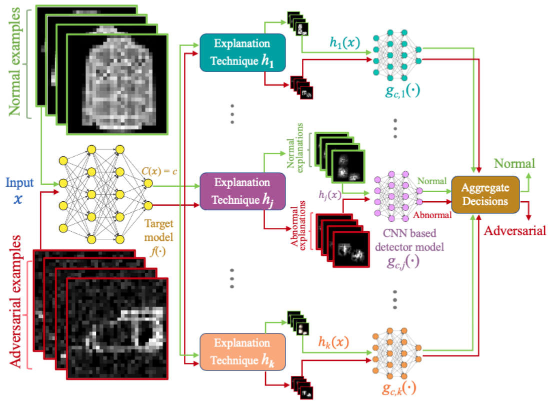


Figure 3: Illustration of the proposed ExAD framework

determine if an explanation map is normal or not. In the following section, we discuss how a set of k detector models, one corresponding to every explanation technique, evaluate the explanation maps towards determining if x_t is an adversarial example.

3.4 Detector models

The detector model determines if the explanation map of an input is normal or abnormal for the classified class. We study the following two methods for building detector models.

3.4.1 Detection using a CNN-based binary classifier. First, we consider an approach of building a CNN-based binary classifier. We term this as the *CNN-based detector model* and denote it as $g(\cdot)$. Under this setting, we refer to the defense as ExAD-CNN. Below, we describe the training procedure for these detector models.

For every class c, we build a separate detector model $g_{c,j}(\cdot)$ for each explanation technique h_j. At test-time, for an input classified as class c, the detector model $g_{c,j}(\cdot)$ takes the explanation map of the input, produced by the corresponding explanation technique h_j, and classifies it as normal or abnormal. We build this new model $g_{c,j}(\cdot)$ as follows. We take every normal example x_{normal} from class c, which is correctly classified by the target model $f(\cdot)$, and generate the explanation map $h_j(x_{\text{normal}})$ for its classification into that class. These explanation maps are considered as normal explanations, and are labeled as negative *class 0*. Then, we generate a number of adversarial examples using different adversarial attacks where the targeted class is c. Next, for each successful adversarial example x_{adv}, we generate the explanation map $h_j(x_{\text{adv}})$ for its classification as target class c. These explanation maps are considered as abnormal explanations, and labeled as positive *class 1*. Then, we train $g_{c,j}(\cdot)$ on this labeled training set using a CNN-based architecture.

3.4.2 Detection based on reconstruction error. In this approach, we avoid the requirement of adversarial examples to train a detector model, and thereby make the defense more likely to generalize on unknown attacks. Here, we propose using reconstruction error by an autoencoder to determine if the explanation map of a test example is normal or not. We term this as the *autoencoder-based detector model*. Under this setting, we refer to the defense as ExAD-AE. Similar to ExAD-CNN setting, we consider each class and build

k autoencoder-based detector models, one corresponding to every explanation technique.

An autoencoder $ae = \psi \circ \phi$ contains two components, an encoder and a decoder, which can be defined as transitions $\phi : \mathbb{R}^d \to \mathbb{R}^z$ and $\psi : \mathbb{R}^z \to \mathbb{R}^d$, respectively, where $\mathbb{R}^d$ is the input space and $\mathbb{R}^z$ is the latent space. For class c and the j-th explanation technique, the input space for the autoencoder $ae_{c,j}$ in our system is formed by the set of explanations produced by h_j for correctly classified normal examples of class c. We train the autoencoder to minimize a loss function over this set of explanations, where the loss function is taken as the mean squared error (MSE):

$$L(\mathbb{E}_{\text{train}}) = \frac{1}{\mathbb{E}_{\text{train}}} \sum_{h_j(x) \in \mathbb{E}_{\text{train}}} \|h_j(x) - (\psi \circ \phi)h_j(x)\|_2$$

For a test image, the explanation map $h_j(x)$ produced by the j-th technique is given as input to the autoencoder $ae_{c,j}$ which generates a reconstructed image. Then, we compute a reconstruction error:

$$R(h_j(x)) = \|h_j(x) - (\psi \circ \phi)h_j(x)\|_p \tag{1}$$

where $\|\cdot\|_p$ is a suitable p-norm. If the reconstruction error is above a threshold t_{re}, we consider the explanation map $h_j(x)$ to be abnormal. The threshold value is a hyperparameter for each detector model. It should be low enough to detect abnormal explanations, but sufficiently high to not falsely flag normal explanations. We decide t_{re} values using a validation set of normal explanations, which are in turn derived from a validation set of normal examples. For any detector, we select the highest t_{re} such that its false-positive rate on the validation set is below a threshold t_{fp}. The threshold t_{fp} can be chosen depending upon system requirements.

3.5 Test-time detection of adversarial examples

Figure 3 illustrates our overall approach, with the ExAD-CNN setting. At test-time, if an unknown input x is being classified by the target classifier $f(\cdot)$ as class c, our goal is to identify if x is a normal example of class c or an adversarial example. To this end, we take the following steps. First, we generate k explanation maps for the classifier's decision to classify x as class c using k explanation techniques. Second, for each explanation map $h_j(x)$, we use the corresponding detector model to determine if the explanation map is normal or abnormal. For ExAD-CNN, each detector model directly provides a classification of normal ($g_{c,j}(h_j(x))$=0) or abnormal ($g_{c,j}(h_j(x))$=1). On the other hand, for ExAD-AE, we obtain the reconstruction error for each explanation map $h_j(x)$ using the corresponding autoencoder $ae_{c,j}$. If the reconstruction error computed by a detector model is above its threshold, then it considers the explanation map to be abnormal. Finally, in both settings, if any of the k detector models classifies the respective explanation map as abnormal, we infer that the input x is an adversarial example.

4 EXPERIMENTS

In this section, we evaluate the effectiveness of ExAD. This section is organized as follows. First, we provide details on the experiment settings in Section 4.1. In Section 4.2, we report the performance of the system on normal examples. Then, in Section 4.3, we show the performance of ExAD on blackbox attacks. Next, we investigate the generalizability of ExAD-CNN in Section 4.4. We compare the performance of our approach with three state-of-the-art detection methods in Section 4.5. Finally, in Section 4.6, we present our evaluation on whitebox attacks.

Table 1: Evaluation of blackbox attacks.

		Attack	Parameter	Cost (s)	Success Rate	Prediction Confidence	L_2 Distortion
MNIST	L_∞	CW_∞	-	90.57	100%	39.11%	3.47
		BIM	eps:0.3	0.003	99%	99.94%	4.10
		MIM	eps:0.3	0.003	100%	99.99%	5.98
	L_2	CW_2	confidence:0	0.001	100%	97.11%	4.33
	L_0	CW_0	-	8.98	100%	38.18%	5.47
		JSMA	gamma:0.2	0.84	94%	76.86%	7.00
FMNIST	L_∞	CW_∞	-	90.07	100%	38.51%	0.729
		BIM	eps:0.3	0.004	98%	100%	4.06
		MIM	eps:0.3	0.004	100%	100%	5.87
	L_2	CW_2	confidence:0	0.006	100%	96.09%	2.20
	L_0	CW_0	-	8.95	100%	37.80%	2.88
		JSMA	gamma:0.2	0.96	85%	83.20%	4.53
CIFAR-10	L_∞	CW_∞	-	68.90	100%	26.91%	1.19
		BIM	eps:0.3	0.008	100%	100%	6.1
		MIM	eps:0.3	0.001	100%	100%	7.9
	L_2	CW_2	confidence:0	0.005	100%	94.52%	3.86
	L_0	CW_0	-	13.35	100%	27.09%	2.98
		JSMA	gamma:0.2	6.42	100%	43.35%	1.78

Table 2: Classification accuracy on normal examples with and without defense.

Dataset	Accuracy with-out ExAD	Top-1 Mean Confidence	Accuracy with ExAD (CNN)	FP rate with ExAD (CNN)	Accuracy with ExAD (AE)	FP rate with ExAD (AE)
MNIST	99.15%	99.86%	98.26%	0.90%	98.54%	0.62%
FMNIST	90.68%	97.86%	89.70%	1.08%	89.91%	0.85%
CIFAR-10	84.54%	76.64%	83.74%	0.95%	83.85%	0.82%

4.1 Experimental settings

Environment. We implement the proposed framework using the Python libraries Keras and TensorFlow. We conducted our experiments on a Linux server with one GPU (GeForce RTX 2080 Ti) and CPU (Intel Xeon Silver 4116 processor).

Image Datasets. We evaluated the performance of our detection mechanism on three image datasets: MNIST [27], Fashion-MNIST (FMNIST) [49] and CIFAR-10 [24]. MNIST is a well-known grayscale image dataset of handwritten digits from 0 to 9. FMNIST is a relatively more challenging dataset of article images where each example is a 28x28 grayscale image associated with a label from 10 classes (shirts, sandals, etc.). Both datasets consist of 60000 examples in the training set and 10000 examples in the testing set. CIFAR-10 is a colored image dataset of tiny 32x32x3 images used for object recognition. It comprises of 50000 training images and 10000 testing images. We chose MNIST and CIFAR-10 datasets as they are most widely used for evaluating defenses against adversarial attacks [6, 33, 35, 50], and additionally used FMNIST as it provides more challenges for a gray-scale dataset.

Training the Target Models. On MNIST and FMNIST datasets, we trained a CNN based target model with 54000 examples in the training set and 6000 examples in the validation set. For CIFAR-10, we trained the CNN based target model with 44000 examples in the training set and 6000 examples in the validation set. For reproducibility, we refer to Appendix B.1 where Table 7 shows the CNN architectures, and Table 8 shows the hyperparameters for training the three target models.

Generating Adversarial Examples. As described in Section 2.3, we generate adversarial examples using six state-of-the-art attacks- JSMA [39], BIM [26], MIM [11], and CW_0, CW_2, and CW_∞ variants of the CW attack [7]. For JSMA, BIM, MIM, and CW_2 attacks, we created adversarial samples using their implementations in the Cleverhans library [38]. For CW_0 and CW_∞ attacks, we use the implementation from the authors [5, 7]. For our evaluation, we adopt the *target-next* attack setting in which the targeted label is the class next to the ground truth class modulo the number of classes (e.g., misclassify an input of class 4 to class 5). Table 1 shows a summary of our evaluation of the six blackbox attacks.

We generate adversarial examples for two purposes. First, as discussed in section 3.4, we need adversarial examples to derive abnormal explanations for training and validating ExAD-CNN. To this end, we consider each class in a dataset and generate as many adversarial examples as the number of normal examples of that class in the training and validation sets. These adversarial examples are unevenly distributed by attack methods, due to relatively higher cost involved in conducting certain attacks. Column 5 in Table 1 shows the average cost (in seconds) to generate one adversarial example for different attacks. We observe that the CW_∞, CW_0, and JSMA attacks incur much more overhead than remaining three attacks. Therefore, we generate 80% of the examples using BIM, MIM, and CW_2 attacks, and the remaining using CW_∞, CW_0, and JSMA attacks. We empirically found this distribution to be sufficient, based on performance on the validation sets. Furthermore, seed images for these adversarial examples are randomly selected from normal examples of the source class. Note that we only utilize examples from the training (resp., validation) set towards training (resp., validating) ExAD; any example in the test set is considered non-accessible for this purpose, as is standard practice.

The second purpose is to evaluate the detection rate of ExAD. To this end, for every dataset, we generate 100 adversarial examples using each attack. This number is consistent with previous works [33, 50] and is limited since many attacks are too expensive to execute. In this process, we create the same number of adversarial examples for every target class to ensure a balanced evaluation. Furthermore, seed images for adversarial examples are taken from correctly classified examples in the test set so that the normal counterparts are unseen by the target model, and the resulting abnormal explanations are unseen by ExAD-CNN.

In Table 1, column 6 shows the success rate achieved by the six blackbox attacks when ExAD is not included as a defense. We consider an attack to be successful if the target model predicts the targeted class. The resulting examples from such attacks are termed as *successful adversarial examples.* We observe that most attacks are very effective against three target models. The BIM, MIM, and CW_2 attacks are particularly effective in generating high-confidence adversarial examples as shown in column 7.

Training ExAD-CNN and ExAD-AE. We refer to Table 9 and Table 10 in Appendix B.2 for details of the CNN architectures and hyperparameters used for training the CNN-based and autoencoder-based detector models, respectively. For each setting, we use the same architecture and hyperparameters for all three datasets as the performance on the respective validation sets was found acceptable. As discussed in Section 3.4, for each target class, we train a detector model for every explanation technique. For training and validation, while ExAD-CNN uses both normal and abnormal explanations, ExAD-AE only uses normal explanations. To obtain normal explanations for a class, we take all its normal examples in our training and validation sets and generate corresponding explanations. For abnormal explanations (to be used by ExAD-CNN), we generate explanations of the adversarial examples being classified as the target class using each explanation technique. As discussed earlier, for this purpose, we had generated as many adversarial examples as the number of normal examples of each class in the training and validation sets. This provides us with balanced training and validation sets of normal and abnormal explanations. For ExAD-CNN, we label the normal and abnormal explanations as negative and positive class, respectively. We train the detector models on the training set, and use the validation set for tuning the hyper-parameters. For ExAD-AE, we exclude the abnormal explanations in both training and validation sets (so that they online consist of normal explanations). Then, we train the detector models on the training set, and use the validation set for setting the threshold t_{re} values. Also, for computing the reconstruction error (equation 1), we empirically find it sufficient to use the L_2 norm. Furthermore, we selected the threshold t_{re} such that the false-positive rate for any detector model is at most 0.2% on its validation set.

Comparison. We compare ExAD with three state-of-the-arts- MagNet [35], Feature Squeezing (FS) [50], and LID [34]. For their implementation, we use the respective GitHub repositories. We follow instructions in the repositories and papers to identify optimal configurations. Feature Squeezing, in particular, allows many configurations for its squeezers. Consistent with the author's work, we utilize the optimal join-detection setting with multiple squeezers. For MNIST and FMNIST, we use the combination of a 1-bit depth squeezer with 2x2 median smoothing. For CIFAR-10, we use a 5-bit depth squeezer with 2x2 median smoothing and 13-3-2 non-local means filter. To ensure fair comparison, for all detection methods, we set thresholds such that the false-positive rate on the validation set is at most 0.2% (same as ExAD). Additionally, our comparison could not include NIC [33] as it is yet to be made open-source, and we were not successful in reproducing the system.

4.2 Performance on normal examples

Table 2 shows the classification accuracy of the three target models on their test set (of normal examples). Without ExAD, we achieve an accuracy of 99.15%, 90.68%, and 84.54% for MNIST, FMNIST, and CIFAR-10 datasets, respectively. When ExAD is included as a defense, it is possible that a correctly classified normal example (by the target model) is misclassified as an adversarial example, termed as a *false-positive* (FP). With ExAD-CNN, we obtained a

Table 3: Detection rate of ExAD on blackbox attacks and comparison with state-of-the-art detection methods.

Dataset	Attack		Parameter	No Defense	**ExAD-CNN**	**ExAD-AE**	MagNet [35]	FS [50]	LID [34]
MNIST	L_∞	CW_∞	-	0%	100%	100%	96%	100%	92%
		BIM	eps:0.3	1%	100%	100%	100%	97.98%	97.98%
		MIM	eps:0.3	0%	100%	100%	100%	98%	99%
	L_2	CW_2	confidence:0	0%	100%	100%	86%	100%	91%
	L_0	CW_0	-	0%	100%	100%	86%	91%	91%
		JSMA	gamma:0.2	6%	100%	100%	84.04%	100%	93.62%
FMNIST	L_∞	CW_∞	-	0%	100%	100%	97%	100%	94%
		BIM	eps:0.3	2%	100%	100%	100%	97.96%	93.88%
		MIM	eps:0.3	0%	100%	100%	99%	99%	95%
	L_2	CW_2	confidence:0	0%	100%	100%	85%	100%	92%
	L_0	CW_0	-	0%	100%	100%	85%	90%	91%
		JSMA	gamma:0.2	15%	100%	100%	87.06%	100%	92.94%
CIFAR-10	L_∞	CW_∞	-	0%	99%	100%	84%	98%	90%
		BIM	eps:0.3	0%	100%	100%	100%	52%	97%
		MIM	eps:0.3	0%	100%	100%	100%	51%	97%
	L_2	CW_2	confidence:0	0%	100%	100%	92%	100%	89%
	L_0	CW_0	-	0%	98%	100%	76%	98%	91%
		JSMA	gamma:0.2	0%	100%	99%	95%	83%	92%

false-positive rate of 0.90% on MNIST dataset, as 89 of 9915 correctly classified normal examples are classified as adversarial. Thus, with ExAD-CNN, the accuracy of the target system is reduced only slightly to 98.26%. Similarly, on FMNIST and CIFAR-10 datasets, the accuracy has a minor drop to 89.70% and 83.74%, respectively. The low false-positive rates are indicative of explanation maps of normal examples rarely being mistaken as abnormal by the detector models. This allows us to maintain a strict policy of classifying a test input as adversarial if even a single detector model considers its explanation map as abnormal.

With ExAD-AE, we obtain false-positive rates of 0.62%, 0.85%, and 0.82% on the test sets of MNIST, FMNIST, and CIFAR-10 datasets, respectively. The accuracy of the target systems under this setting is 98.54% for MNIST, 89.91% for FMNIST, and 83.85% for CIFAR-10 dataset. These results are close to the performance of ExAD-CNN on normal examples. In the following section, we show that both settings of ExAD can effectively detect adversarial attacks while maintaining these low false-positive rates.

4.3 Evaluation on blackbox attacks

Table 3 shows the detection rates of our approach on the six blackbox attacks. Columns 1-4 show datasets and details of the attacks. Column 5 shows the detection rate of adversarial examples when ExAD is not included as a defense (which corresponds to the success rate achieved by blackbox attacks on the target models). Columns 6 and 7 show the detection rates of ExAD-CNN and ExAD-AE, respectively. Note that, except when noted explicitly, "detection rate" of a detection method refers to its detection rate on successful adversarial examples, consistent with previous work [50]. The remaining columns report our comparison with three state-of-the-art detectors, which we will discuss in Section 4.5.

We first consider the ExAD-CNN setting. For this setting, our approach achieves consistently high detection rates for all attacks across the three datasets. As shown in Table 3, for MNIST and FMNIST datasets, we get 100% detection rates for all six attacks. For CIFAR-10 dataset, we obtain 98% detection rate for CW_2 and CW_0 attacks, and 100% detection rate for remaining attacks. For the ExAD-AE setting, the detection rate of adversarial examples is again consistently high. We achieve 100% detection rate for all six attacks on MNIST and FMNIST datasets. On CIFAR-10 dataset, we obtain a 99% detection rate for JSMA attack. For all other attacks on CIFAR-10 dataset, the detection rate is 100%.

While we find both settings of ExAD are effective against adversarial attacks, each has its own relative advantages and disadvantages. A benefit of ExAD-AE is that it does not rely on adversarial examples for training or validation. This reduces the training overhead as many adversarial attacks incur significant cost (Table 1). More importantly, being attack-independent, the performance of ExAD-AE indicates that our approach generalizes well to unknown attacks. But, compared to the effort required in setting the threshold t_{re} values for ExAD-AE, it is relatively simpler to tune the hyperparameters for ExAD-CNN. However, training the detector models in ExAD-CNN requires adversarial examples (to derive abnormal explanations). In the following section, we investigate the extent to which this requirement impacts the generalizability of ExAD-CNN in detecting unknown attacks.

4.4 Generalizability of ExAD-CNN

In the generation of abnormal explanations, we notice that adversarial examples created using attacks of the same category (L_∞ or L_0) produce similar explanations. We illustrate this phenomenon for interested readers in Figure 6 in Appendix C. Using this observation, we leave out the CW_∞ and CW_0 attacks, and only use adversarial examples from other four attacks for training ExAD-CNN. We keep other training and attack parameters same as before. In Table 4, columns 2-4 show the new performance of ExAD-CNN. On MNIST

Table 4: Detection rate of ExAD-CNN with limited attacks used in training.

Attack	Train on BIM, MIM, CW_2, JSMA			Train only on CW_2		
	MNIST	FMNIST	CIFAR-10	MNIST	FMNIST	CIFAR-10
CW_∞	100%	100%	92%	99%	99%	96%
BIM	100%	100%	100%	97.98 %	96.94%	100%
MIM	100%	100%	100%	89%	86%	100%
CW_2	100%	100%	100%	100%	100%	100%
CW_0	100%	100%	92%	100%	97%	91%
JSMA	100%	100%	100%	91.49%	94.12%	87%

Table 5: False-positive rates obtained for MagNet [35], FS [50], and LID [34].

Dataset	MagNet [35]	FS [50]	LID [34]
MNIST	0.50%	3.85%	4.24%
FMNIST	0.81%	3.76%	3.89%
CIFAR-10	4.25%	4.81%	5.36%

and FMNIST datasets, we find that ExAD-CNN effectively detects the unknown attacks. On CIFAR-10, ExAD-CNN still achieves a good detection rate of 92% for both unknown attacks, CW_∞ and CW_0. We also observe that the detection of other attacks is not affected on any dataset. These results appear to indicate that abnormal explanations are influenced more by the original class of the seed images and the category (norm) of the attack, rather than the attack variant in that category used to craft the adversarial examples. This allows ExAD-CNN to generalize well to unknown attacks when we train on representative attacks from different categories. Furthermore, to study the generalizability in a more restrictive case, we only train ExAD-CNN on CW_2 attack. Columns 5-7 in Table 4 show the performance under this scenario. For all datasets, we find the results are good on the three CW attacks, with the lowest detection rate of 91% obtained for CW_0 attack on CIFAR-10 dataset. For BIM attack, the performance remains consistently high across datasets. However, the detection is less effective for MIM attack on MNIST and FMNIST datasets, for which we have 89% and 86% detection rates, respectively, and for JSMA attack on CIFAR-10, for which we obtain 87% detection rate. Overall, we see that ExAD-CNN can detect many unknown attacks even in this restrictive case, but there is still room for improvement. For boosting the detection in such scenarios, we can consider performing join-detection using both ExAD-CNN and ExAD-AE (which obtained high detection rates while being attack-independent). Building such joint-detectors to improve generalizability will be an interesting topic for future work.

4.5 Comparison

Table 3 shows the comparison of ExAD with three state-of-the-art adversarial detection methods- MagNet [35], Feature Squeezing [50], and LID [34]. The false-positive rates for these methods on the test sets are reported in Table 5.

MagNet. We find that MagNet's false-positive rates on the two grayscale datasets are marginally lower than those of ExAD-AE. However, it has much higher false-positive rate of 4.25% on CIFAR-10 dataset. We also find its detection performance to vary depending upon the dataset and attack-norm. MagNet uses trained autoencoders to detect adversarial examples, and to reform them based on the differences between the manifolds of normal and adversarial examples. MagNet's denoising strategy is quite effective against L_∞ attacks, for which adversarial examples tend to have a large number of modified pixels, with a limit on the change per pixel. MagNet achieves high detection rates (most being 100%) for L_∞ attacks on both grayscale datasets. On CIFAR-10, a colored dataset, it achieves similar performance on BIM and MIM attacks, but relatively low detection rate of 84% on CW_∞ attack. Furthermore, we observe that MagNet's denoising mechanism is not as effective on L_0 attacks. These attacks make changes of high magnitude to very few pixels, thereby making denoising difficult. This is consistent with findings by Ma et al. [33]. Similarly, we find MagNet's performance on CW_2 attack is not as good on MNIST and FMNIST datasets. Moreover, MagNet requires training a single detector network, which is computationally expensive. A benefit of our approach is that we use small detector models for every class, which are much easier to train. This makes our approach more practical.

LID. We find the detection performance of LID to be consistent across attacks and datasets. This method computes an LID value that captures the intrinsic dimensional properties of adversarial regions[34]. LID achieves its highest detection for L_∞ attacks on the three datasets. For most of the other attacks, its detection was consistently above 90%. However, as shown in Table 5, a downside of using LID values is the relatively higher false-positive rates on normal examples, which impacts the reliability of its classifications.

Feature Squeezing. For Feature Squeezing, we observe that joint-detection provides fairly consistent detection rates (Table 3), but introduces high false-positive rates between 3.76% to 4.81% (Table 5). This is natural, given the use of a single threshold across all squeezers, consistent with original work [50]. Nevertheless, as reported by the authors, this can be improved by combining multiple squeezers with different thresholds in future work. Feature Squeezing obtains very high detection rates on all attacks on MNIST and FMNIST datasets. On CW_2 attack, it achieves 100% detection rate on all three datasets. However, for L_∞ attacks, while it obtains detection rates of above 98% on CW_∞ attack, we find it less effective against BIM and MIM attacks on CIFAR-10 dataset with a detection rate of nearly 52%. This reflects upon the generalizability challenges in building squeezers. In contrast, our approach is more general and achieves consistent detection and false-positive rates.

4.6 Evaluation with adaptive adversaries

In this section, we evaluate our defense, with the ExAD-CNN setting, under whitebox threat model for an adaptive adversary. Here, we build upon recent research that shows that explanations can be unreliable [22] and can be manipulated to produce a target explanation map [10, 52]. Below, we present our approach to conduct a whitebox attack for generating an adversarial example.

Whitebox Attack Approach. Given a normal or seed image $x \in \mathbb{R}^d$ with correctly classified class $C(x) = c$ by target model $f(\cdot)$, we follow a two-step process towards conducting a whitebox attack. First, we use a blackbox attack to generate an adversarial example x' which is misclassified as $C(x') = t$, where t is the targeted class. While x' is likely to fool $f(\cdot)$, it is likely to be correctly classified

Table 6: Evaluation of whitebox attacks.

	Target Explanation Method	Mean Success Rate without defense	Cost (s)	L_2 Distortion
MNIST	LRP	99.00%	137.07	2.49
	GBP	95.17%	50.15	2.64
	IG	82.00%	506.75	2.81
	PA	96.17%	57.74	2.55
	GTI	91.50%	48.96	2.50
FMNIST	LRP	99.17%	169.36	2.28
	GBP	95.33%	51.03	2.34
	IG	89.00%	510.47	2.33
	PA	93.67%	57.77	2.76
	GTI	91.50%	49.12	2.31
CIFAR-10	LRP	99.00%	138.83	2.26
	GBP	97.33%	51.35	2.24
	IG	94.00%	484.02	3.87
	PA	96.00%	66.53	3.20
	GTI	95.33%	66.80	3.45

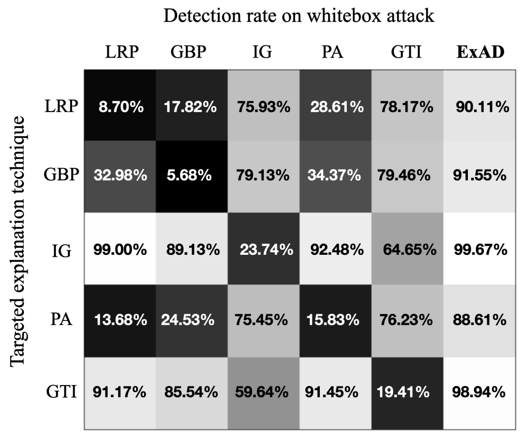


Figure 4: Transferability of whitebox attack

as an adversarial example by the defense, which has not yet been accounted for by the attack. As a next step, we consider a target explanation technique h_j, which produces an explanation map $h_j(x')$ that is correctly classified as abnormal by the corresponding detector model $g_{t,j}(\cdot)$. Our goal in this step is to manipulate x' to create a final adversarial example $x'' = x' + \delta x'$, such that

- The target model's classification remains approximately constant, i.e. $f(x'') \approx f(x')$
- The explanation map $h_j(x'')$ is close to a target explanation map h_j^t that is classified as normal by $g_{t,j}(\cdot)$
- The norm of the perturbation $\delta x'$ added is small so that it remains imperceptible.

To obtain the target explanation map h_j^t, we randomly select a normal example x_r from class t and check if its explanation map $h_j(x_r)$ is classified as normal by $g_{t,j}(\cdot)$. If so, we set the target explanation map as $h_j^t = h_j(x_r)$. Else, we repeat the process until we find such an example. This search is fairly quick because explanation maps of normal examples of a class are very likely to be correctly classified as normal by the detector models. Finally, we generate x'' by optimizing the following loss function

$$\mathcal{L} = ||h_j(x'') - h_j^t||^2 + \gamma||f(x'') - f(x')||^2 \qquad (2)$$

with respect to x'' using gradient descent, such that $\|\delta x'\|_2 < \epsilon$. The first term in the loss function ensures that the explanation map for x'' is close to the target map, while the second term ensures the prediction by the target model is still the misclassified class t. The weighting of these two terms is controlled by hyperparameter $\gamma \in \mathbb{R}_+$. To compute the gradient with respect to the input $\nabla h_j(x'')$, we follow the strategy by Dombrowski et al. of replacing relu with the softplus function to circumvent the problem of vanishing second-derivative [10]. A similar strategy of approximating relu was used by Zhang et al. [52]. After the optimization completes, we test whether the manipulated image x'' fools the original relu-based target model $f(\cdot)$ as well as our defense. We provide an illustration of our whitebox approach in Appendix D.

Evaluation of Whitebox Attack. To perform the first step of the above approach, we re-use the successful adversarial examples X' created using blackbox attacks. However, while targeting integrated gradients (IG), we only used adversarial examples from CW_2 attack as we find targeting IG to incur very high cost. For targeting remaining techniques, we use adversarial examples from all six attacks. Then, for each adversarial example $x' \in X'$, we perform the second step using the optimization process described above to obtain final adversarial examples X''.

Table 6 shows a summary of the whitebox attack. Column 3 shows the mean success rate of the final adversarial examples in retaining the desired misclassification in the target model (when ExAD is not included). The mean is computed over success rates obtained for different attacks (except for IG where we only use CW_2 attack). We do not show individual success rates for each attack as they were very similar for any target technique. In Table 6, we observe that targeting LRP results in the highest success rate, with least L_2 distortion. However, as shown in Column 4, the average time required per example to target LRP is over 130 seconds which is quite high compared to targeting GBP, PA, or GTI techniques.

Figure 4 shows the results of the whitebox attack. Each row represents the targeted explanation technique. Columns 1-5 show the detection rate obtained by individual detector models corresponding to the five explanation techniques in correctly classifying explanation maps as abnormal. Column 6 shows the overall performance by ExAD-CNN in correctly classifying the examples as adversarial. From Figure 4, we make several interesting observations. First, we notice the values along the diagonal (from top-left to bottom-right) are all very low. This is natural as the detector model corresponding to the targeted technique is expected to correctly classify very few explanations as abnormal. For instance, targeting GTI causes its detector model to have a detection rate of only 19.41%. Second, we observe that targeting gradient-based techniques do not severely impact detector models of propagation-based techniques, and vice-versa. For instance, on targeting IG or GTI, the detector models corresponding to LRP, GBP, and PA still achieve detection rates above 85%. This is consistent with the transferability findings by Zhang et al. [52]. On a different set of diverse explanation techniques, the authors showed that manipulated images created by targeting one technique rarely produce desirable explanations (which are close to the target map) on other techniques. Finally, we find that targeting propagation-based (resp., gradient-based) techniques transfer well to detector models corresponding to the other propagation-based (resp., gradient-based) techniques. For instance, targeting LRP results in the GBP-based detector model to have a detection rate of only 17.82%. The same phenomenon can be observed for gradient-based techniques (IG and GTI). This empirically supports the need to have diversity in the explanation methods to build robustness against adaptive attacks. As shown in Column 6 of Figure 4, using an ensemble of gradient-based and propagation-based techniques, ExAD is able to significantly limit the success rate of whitebox attacks. We find ExAD to be relatively more robust when gradient-based techniques are targeted. We achieve 99.67% and 98.94% detection rate for IG and GTI as the target, respectively. For propagation-based techniques, the highest impact is caused by targeting PA, which results in a detection rate of 88.61%. For the case of targeting LRP and GBP, we obtain detection rates of 90.11% and 91.55%, respectively. Appendix E includes further analysis on transferability of whitebox attack for individual datasets.

5 DISCUSSION

5.1 Fragility of Explanations

In Section 4.6, we discussed the reliability of explanations in context of an adaptive adversary. Recently, there has also been research that shows the fragility of explanations in an adversarial context. In this section, we discuss two related scenarios and any potential impact to our defense strategy.

5.1.1 Hiding the attack from explanations. Recently, *adversarial patches* [4, 21] were introduced to make adversarial examples more practical in the physical world. This attack restricts the spatial dimensions of the perturbation, but removes the imperceptibility constraint. However, Subramanya *et al.* [45] demonstrated that we can generate adversarial patches that not only fool the prediction by the target classifier, but also change the explanation of the modified example such that the adversarial patch is no longer considered important. Nevertheless, here the attacker only manages to make the explanation technique focus in a region outside the adversarial patch; she does not try to make the explanation itself appear more normal for the target class. Therefore, even though such adversarial examples may evade the explanation mechanism, they are still likely to be correctly classified as adversarial by our defense.

5.1.2 Changing the explanation but not the classification. An attacker may add adversarial perturbations which produce examples that are classified into the same class, but have very different explanations [13]. With our defense, if such explanations are classified as abnormal by the detector models, then the corresponding inputs would be considered adversarial. Nevertheless, we believe the impact of this attack may depend upon the nature of the application where the defense is being used. If the perturbed examples resulting from such attacks are not considered normal for the system, then the abnormality produced in the explanations can be beneficial because the examples will likely be classified as adversarial. However, if an application considers such perturbed examples as normal, such as if the norm of perturbation is within an allowed threshold, then our defense could result in false-positives. We refer to Figure 10 in Appendix F which shows the effect of such attacks on the target classifier's accuracy for this case. For such applications, we would require explanation techniques to be robust enough to allowed perturbations. We leave further research towards such methods as future work.

5.2 Limitations

Our detection mechanism and scope of evaluation has certain limitations. First, our current evaluation only considers the targeted attack setting for generating adversarial examples. In future work, we will extend our evaluation to cover untargeted adversarial attacks as well. Second, currently we do not have a unified optimization-based approach that simultaneously (and successfully) targets multiple explanation techniques. We attempt doing so in two ways. One approach is to modify the loss function 2 as follows

$$\mathcal{L} = \Big(\sum_{j=1}^{5} ||h_j(x'') - h^t||^2 \Big) + \gamma ||f(x'') - f(x')||^2 \tag{3}$$

Here, we create the target map h^t using any one of the explanation techniques. However, in this case, when the target map is created using a gradient-based technique, we did not notice any significant change in the performance of the detector models corresponding to propagation-based techniques, and vice-versa. The results remained consistent with our findings in Figure 4. One reason behind this could be the diversity in explanation techniques due to which the target maps required by gradient-based techniques differ substantially from those required by propagation-based techniques. Then, we further consider modifying the loss function 3 as follows

$$\mathcal{L} = \Big(\sum_{j=1}^{5} ||h_j(x'') - h_j^t||^2 \Big) + \gamma ||f(x'') - f(x')||^2 \tag{4}$$

Here, target maps used are created using the corresponding explanation techniques. However, in this case, we found the explanation-loss component did not reduce much during the optimization process. Moreover, it often led to memory errors on the GPU. One reason for this is that in the current whitebox attack framework[10], each explanation technique requires a different set of hyperparameters (e.g., learning rate, β growth, and iterations), as shown in Table 11 in Appendix B.3. Therefore, we find that simultaneously attacking multiple explanation techniques is considerably difficult for an adaptive adversary. We leave further exploration on improving the efficiency and effectiveness of such attack to future work.

6 CONCLUSION

We proposed ExAD, a framework to detect adversarial examples using an ensemble of explanation techniques. The use of explanations is motivated by the distinguishability between normal and abnormal explanations for any target class. Furthermore, motivated by previous work on N-variant systems, we used an ensemble of gradient-based and propagation-based explanation techniques to introduce diversity in our defense. Experiments showed that our approach is effective against blackbox attacks, and outperforms three state-of-the-art detectors. We also find that ExAD significantly limits the success rate of whitebox attacks. In this process, we made interesting findings on the transferability of adaptive attacks. We acknowledge the possibility of more sophisticated whitebox attacks in future, and hope our work will inspire further research in this direction. We believe our proposed defense is complementary to state-of-the-art detection methods and can be used in conjunction with them to boost the detection of adversarial attacks.

REFERENCES

[1] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one* (2015).

[2] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Mueller. 2009. How to explain individual classification decisions. *arXiv preprint arXiv:0912.1128* (2009).

[3] Arjun Nitin Bhagoji, Daniel Cullina, and Prateek Mittal. 2017. Dimensionality reduction as a defense against evasion attacks on machine learning classifiers. *arXiv preprint arXiv:1704.02654* (2017).

[4] Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. 2017. Adversarial patch. *arXiv preprint arXiv:1712.09665* (2017).

[5] Nicholas Carlini. [n.d.]. Robust Evasion Attacks against Neural Network to Find Adversarial Examples. https://github.com/carlini/nn_robust_attacks.

[6] Nicholas Carlini and David Wagner. 2017. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security.*

[7] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *2017 IEEE S&P.* IEEE, 39–57.

[8] Benjamin Cox, David Evans, Adrian Filipi, Jonathan Rowanhill, Wei Hu, Jack Davidson, John Knight, Anh Nguyen-Tuong, and Jason Hiser. 2006. N-Variant Systems: A Secretless Framework for Security through Diversity.. In *USENIX Security Symposium.* 105–120.

[9] George E Dahl, Jack W Stokes, Li Deng, and Dong Yu. 2013. Large-scale malware classification using random projections and neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing.* IEEE, 3422–3426.

[10] Ann-Kathrin Dombrowski, Maximillian Alber, Christopher Anders, Marcel Ackermann, Klaus-Robert Müller, and Pan Kessel. 2019. Explanations can be manipulated and geometry is to blame. In *Advances in Neural Information Processing Systems.* 13589–13600.

[11] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. 2018. Boosting adversarial attacks with momentum. In *CVPR.*

[12] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. 2017. Robust physical-world attacks on deep learning models. *arXiv preprint arXiv:1707.08945* (2017).

[13] Amirata Ghorbani, Abubakar Abid, and James Zou. 2019. Interpretation of neural networks is fragile. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 3681–3688.

[14] Zhitao Gong, Wenlu Wang, and Wei-Shinn Ku. 2017. Adversarial and clean data are not twins. *arXiv preprint arXiv:1704.04960* (2017).

[15] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

[16] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. 2017. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280* (2017).

[17] Shixiang Gu and Luca Rigazio. 2014. Towards deep neural network architectures robust to adversarial examples. *arXiv preprint arXiv:1412.5068* (2014).

[18] Dan Hendrycks and Kevin Gimpel. 2016. Early methods for detecting adversarial images. *arXiv preprint arXiv:1608.00530* (2016).

[19] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Brian Kingsbury, et al. 2012. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine* 29 (2012).

[20] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. 2019. Adversarial examples are not bugs, they are features. In *Advances in Neural Information Processing Systems.* 125–136.

[21] Danny Karmon, Daniel Zoran, and Yoav Goldberg. 2018. Lavan: Localized and visible adversarial noise. *arXiv preprint arXiv:1801.02608* (2018).

[22] Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. 2017. The (un) reliability of saliency methods. *arXiv preprint arXiv:1711.00867* (2017).

[23] Pieter-Jan Kindermans, Kristof T Schütt, Maximilian Alber, Klaus-Robert Müller, Dumitru Erhan, Been Kim, and Sven Dähne. 2017. Learning how to explain neural networks: Patternnet and patternattribution. *arXiv preprint arXiv:1705.05598* (2017).

[24] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems.* 1097–1105.

[26] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2016. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533* (2016).

[27] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[28] Xin Li and Fuxin Li. 2017. Adversarial examples detection in deep networks with convolutional filter statistics. In *Proceedings of the IEEE International Conference on Computer Vision.* 5764–5772.

[29] Zachary C Lipton. 2016. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490* (2016).

[30] Ninghao Liu, Hongxia Yang, and Xia Hu. 2018. Adversarial detection with model interpretation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* 1803–1811.

[31] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. 2016. Delving into transferable adversarial examples and black-box attacks. *arXiv preprint arXiv:1611.02770* (2016).

[32] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* (2015).

[33] Shiqing Ma, Yingqi Liu, Guanhong Tao, Wen-Chuan Lee, and Xiangyu Zhang. 2019. NIC: Detecting Adversarial Samples with Neural Network Invariant Checking.. In *NDSS.*

[34] Xingjun Ma, Bo Li, Yisen Wang, Sarah M Erfani, Sudanthi Wijewickrema, Grant Schoenebeck, Dawn Song, Michael E Houle, and James Bailey. 2018. Characterizing adversarial subspaces using local intrinsic dimensionality. *arXiv preprint arXiv:1801.02613* (2018).

[35] Dongyu Meng and Hao Chen. 2017. Magnet: a two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security.* ACM, 135–147.

[36] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. 2017. On detecting adversarial perturbations. *ICLR* (2017).

[37] Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. 2017. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition* 65 (2017), 211–222.

[38] Nicolas Papernot, Fartash Faghri, Nicholas Carlini, Ian Goodfellow, Reuben Feinman, and Alexey Kurakin et al. 2018. Technical Report on the CleverHans v2.1.0 Adversarial Examples Library. *arXiv preprint arXiv:1610.00768* (2018).

[39] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. 2016. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P).*

[40] Cynthia Rudin. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence* (2019).

[41] Avanti Shrikumar, Peyton Greenside, Anna Shcherbina, and Anshul Kundaje. 2016. Not just a black box: Learning important features through propagating activation differences. *arXiv preprint arXiv:1605.01713* (2016).

[42] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034* (2013).

[43] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. 2017. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825* (2017).

[44] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. 2014. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806* (2014).

[45] Akshayvarun Subramanya, Vipin Pillai, and Hamed Pirsiavash. 2018. Towards Hiding Adversarial Examples from Network Interpretation. *arXiv preprint arXiv:1812.02843* (2018).

[46] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *ICML.*

[47] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* (2013).

[48] Jingyuan Wang, Yufan Wu, Mingxuan Li, Xin Lin, Junjie Wu, and Chao Li. 2020. Interpretability is a Kind of Safety: An Interpreter-based Ensemble for Adversary Defense. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* 15–24.

[49] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms.* arXiv:cs.LG/1708.07747 [cs.LG]

[50] Weilin Xu, David Evans, and Yanjun Qi. 2017. Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks. *arXiv preprint arXiv:1704.01155* (2017).

[51] Chiliang Zhang, Zuochang Ye, Yan Wang, and Zhimou Yang. 2018. Detecting adversarial perturbations with saliency. In *2018 IEEE 3rd International Conference on Signal and Image Processing (ICSIP).* IEEE, 271–275.

[52] Xinyang Zhang, Ningfei Wang, Hua Shen, Shouling Ji, Xiapu Luo, and Ting Wang. 2020. Interpretable deep learning under fire. In *29th {USENIX} Security Symposium ({USENIX} Security 20).*

A SIMILARITY IN NORMAL EXPLANATIONS

Our approach is motivated by the observation that normal examples of a class tend to have similar (normal) explanations, and that a detector model can learn to distinguish between them and the (abnormal) explanations of adversarial examples targeting that class. Figure 5 shows an example of the similarity in normal explanations. The first row shows five normal examples from the Coat class of FMNIST dataset. The third row shows normal examples from the Airplane class of CIFAR-10 dataset. The fifth row shows normal examples of class Three from MNIST dataset. The second, fourth, and sixth rows show corresponding explanations for the preceeding row using the IG, LRP, and GBP techniques, respectively.

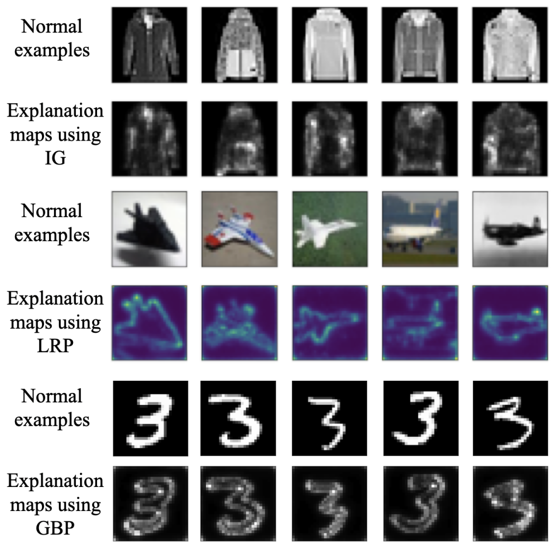

Normal examples
Explanation maps using IG
Normal examples
Explanation maps using LRP
Normal examples
Explanation maps using GBP

Figure 5: Similarity in normal explanations.

B IMPLEMENTATION DETAILS

B.1 Training the Target Models

Table 7 shows the architecture of the target models for MNIST, FMNIST, and CIFAR-10 datasets.

Table 7: Architecture of the image classifiers to be defended.

MNIST		FMNIST		CIFAR-10	
Conv.ReLU	8x8x64	Conv.ReLU	3x3x32	Conv.ReLU	3x3x64
Conv.ReLU	6x6x128	Conv.ReLU	3x3x32	Conv.ReLU	3x3x128
Conv.ReLU	5x5x128	MaxPooling	2x2	AvgPooling	2x2
Softmax	10	Conv.ReLU	3x3x64	Conv.ReLU	3x3x128
		Conv.ReLU	3x3x64	Conv.ReLU	3x3x256
		MaxPooling	2x2	AvgPooling	2x2
		Dense.ReLU	200	Conv.ReLU	3x3x256
		Dense.ReLU	200	Conv.ReLU	3x3x512
		Softmax	10	AvgPooling	2x2
				Conv.ReLU	3x3x10
				Softmax	10

Table 8 shows the training and architecture hyperparameters for the target models of the three datasets.

Table 8: Training and architecture hyperparameters of the image classifiers to be defended

Hyperparameter	MNIST	FMNIST	CIFAR-10
Learning Rate	0.001	0.01	0.001
Optimization Method	Adam	SGD	Adam
Batch Size	128	128	256
Epochs	50	50	50
Padding (Conv layers)	Valid	Valid	Same

B.2 Training Detector Models of ExAD

Table 9 shows the architecture and hyperparameters used for ExAD-CNN.

Table 9: Architecture and hyperparameters of ExAD-CNN

Architecture		Hyperparameters	
Conv.ReLU	3x3x32	Learning Rate	0.01
Conv.ReLU	3x3x64	Optimization Method	Adam
MaxPooling	2x2	Batch Size	32
Conv.ReLU	3x3x128	Epochs	50
Conv.ReLU	3x3x128	Padding (Conv layers)	Same
MaxPooling	2x2		
Dense.ReLU	512		
Dense.ReLU	64		
Softmax	2		

Table 10 shows the architecture and hyperparameters used for ExAD-AE. For MNIST and FMNIST datasets, $H = W = 28$. For CIFAR-10 dataset, $H = W = 32$.

Table 10: Architecture and hyperparameters of ExAD-AE

Architecture		Hyperparameters	
Dense.ReLU	HxW	Learning Rate	10^{-5}
Dense.ReLU	400	Optimization Method	Adam
Dense.ReLU	20	Batch Size	32
Dense.ReLU	400	Epochs	100
Dense.ReLU	HxW		
Softmax	2		

B.3 Performing Whitebox Attack

Table 11 shows the hyperparameters for the whitebox attack.

Table 11: Hyperparameters used in whitebox attack.

Method	Iterations	Learning Rate	Factors
LRP	1500	10^{-3}	2x10^{-4}, 10^{6}
GBP	1500	10^{-3}	10^{11}, 10^{6}
IG	500	$5x10^{-3}$	10^{11}, 10^{6}
PA	1500	$2x10^{-3}$	10^{11}, 10^{6}
GradxInput	1500	10^{-3}	10^{11}, 10^{6}

C GENERALIZABILITY OF EXAD-CNN

Figure 6 shows explanation maps produced by the integrated gradients (IG) technique for adversarial examples created using different attacks. Rows 1 and 2 show explanation maps for adversarial examples which were created using CW_∞ and BIM attacks, respectively. Both attacks come under the L_∞ category. The adversarial examples are targeting the Pullover class and their seed images are taken from the Trouser class of FMNIST dataset. Comparing explanation maps in row 1 and row 2 shows that adversarial examples created using different attacks, under the same category (L_∞ in this case) can result in similar explanation maps. This can also be observed for CIFAR-10 dataset (rows 3-4), where we use two L_0 attacks, and MNIST dataset (rows 5-6), where we again use two L_∞ attacks.

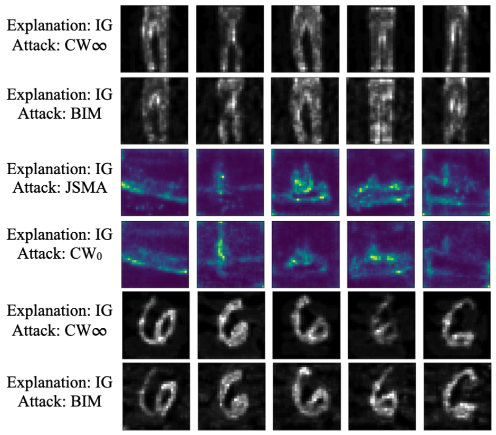


Figure 6: Similarity in explanation maps of adversarial examples created using different attacks.

D ILLUSTRATION OF WHITEBOX ATTACK

Figure 7 shows an illustration of our whitebox attack approach, discussed in Section 4.6. The leftmost image in the first row shows a seed image x, which is a normal example from class Airplane of CIFAR-10 dataset. In the first step, we use CW_∞ attack to add perturbations δx to x, which results in the adversarial example $x' = x + \delta x$. The rightmost image of row 1 shows x'. The example x' is misclassified as class Automobile by target model $f(\cdot)$. The second row shows the explanation maps produced for x' by the five techniques. We find that all detector models classify these explanation maps as abnormal. We observe in row 2 that the explanation maps are not consistent with the expected normal explanations of class Automobile. As an adaptive adversary, we now intend to target an explanation technique. For this illustration, we choose to target LRP. To this end, we randomly select an example x_r from class Automobile, for which the explanation map produced by LRP is classified as normal by the corresponding detector model. We show x_r as the first image in row 3. Its explanation map by LRP, shown as the second image in row 2, is set as the target map h^t_{LRP}. Then, we perform the optimization for the loss function 2 (shown in Section 4.6) using x' and h^t_{LRP}. After the optimization completes, we obtain the final adversarial example $x'' = x' + \delta x'$, which is shown as the rightmost image in row 4. The bottom row shows

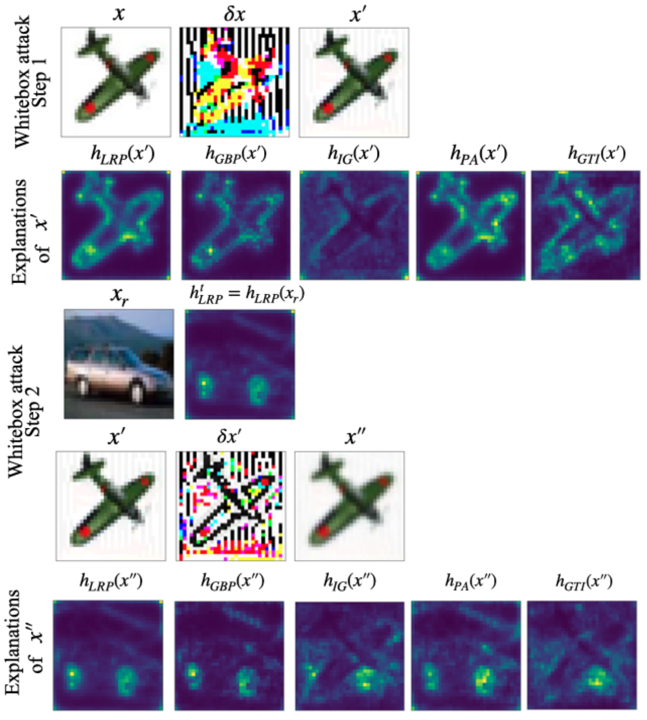
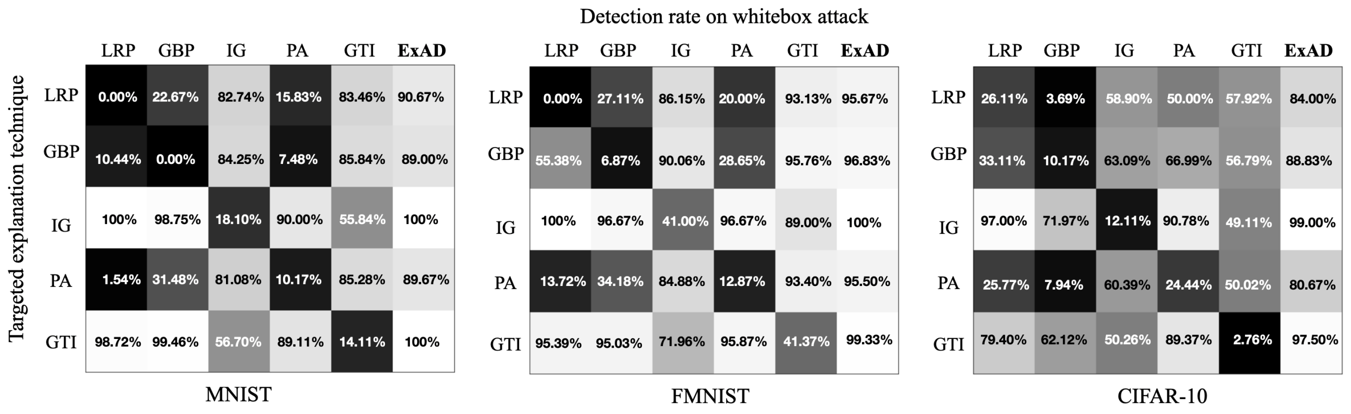


Figure 7: Illustration of whitebox attack.

the explanation maps produced by the five techniques for x''. We observe that targeting LRP results in the explanation map produced by LRP (first image in row 5) to be very close to the target map. Also, the explanation maps for GBP (second image in row 5) and PA (fourth image in row 5) are also fairly close to the target map. We find that all three of these explanation maps are classified as normal by their respective detector models. However, we observe that the explanation maps produced by the gradient-based techniques, i.e., IG (third image in row 5) and GTI (rightmost image in row 5), are not as close to the target map. Both these explanation maps are classified as abnormal by their respective detector models. Therefore, using an ensemble of detector models corresponding to diverse explanation techniques, our defense is able to mitigate this whitebox attack by correctly identifying x'' as an adversarial example.

E TRANSFERABILITY OF WHITEBOX ATTACKS

Figure 8 shows results on the transferability of whitebox attack on explanation-based detector models, as well as the detection rates obtained by our defense (under ExAD-CNN setting), for the three datasets. Previously, we had only shown mean values across datasets in Figure 4. In Figure 8, each value shows the mean detection rate for adversarial examples created using different attacks (except while targeting IG technique which only uses CW_2 attack) and considering all target classes. On all three datasets, we find that targeting a gradient-based technique has relatively less impact on detector models corresponding to propagation-based techniques.


Detection rate on whitebox attack
Targeted explanation technique
MNIST
FMNIST
CIFAR-10

MNIST

	LRP	GBP	IG	PA	GTI	**ExAD**
LRP	0.00%	22.67%	82.74%	15.83%	83.46%	90.67%
GBP	10.44%	0.00%	84.25%	7.48%	85.84%	89.00%
IG	100%	98.75%	18.10%	90.00%	55.84%	100%
PA	1.54%	31.48%	81.08%	10.17%	85.28%	89.67%
GTI	98.72%	99.46%	56.70%	89.11%	14.11%	100%

FMNIST

	LRP	GBP	IG	PA	GTI	**ExAD**
LRP	0.00%	27.11%	86.15%	20.00%	93.13%	95.67%
GBP	55.38%	6.87%	90.06%	28.65%	95.76%	96.83%
IG	100%	96.67%	41.00%	96.67%	89.00%	100%
PA	13.72%	34.18%	84.88%	12.87%	93.40%	95.50%
GTI	95.39%	95.03%	71.96%	95.87%	41.37%	99.33%

CIFAR-10

	LRP	GBP	IG	PA	GTI	**ExAD**
LRP	26.11%	3.69%	58.90%	50.00%	57.92%	84.00%
GBP	33.11%	10.17%	63.09%	66.99%	56.79%	88.83%
IG	97.00%	71.97%	12.11%	90.78%	49.11%	99.00%
PA	25.77%	7.94%	60.39%	24.44%	50.02%	80.67%
GTI	79.40%	62.12%	50.26%	89.37%	2.76%	97.50%

Figure 8: Transferability of whitebox attack on individual datasets.

For instance, targeting GTI on MNIST causes the IG-based detector model to have a detection rate of only 56.70% whereas detector models corresponding to LRP, GBP, and PA have detection rates above 89%. Interestingly, on CIFAR-10 dataset, we observe that the detector models corresponding to IG and GTI are impacted by most techniques, although the impact is relatively higher when we target either of IG or GTI. For instance, on CIFAR-10 dataset, targeting PA has a noticeable impact on IG-based detector model as it achieves a detection rate of 60.39%, whereas targeting GTI results in the IG-based detector model to have a relatively lower detection rate of 50.26%.

In terms of impact on ExAD, on all three datasets, targeting gradient-based techniques (IG and GTI) is relatively less effective. For instance, on MNIST, we obtain 100% detection rate by our approach while targeting IG and GTI techniques. In contrast, we observe that targeting propagation-based techniques is more effective across datasets. On MNIST dataset, the detection rate of the defense on propagation-based techniques is nearly 89%. On CIFAR-10 dataset, targeting PA results in the lowest detection rate of 80.67% by ExAD. On FMNIST dataset, the overall performance is fairly consistent while targeting propagation-based techniques. The highest impact is produced by targeting PA and LRP, for which ExAD obtains detection rates of 95.50% and 95.67%, respectively.

F FRAGILITY OF EXPLANATIONS

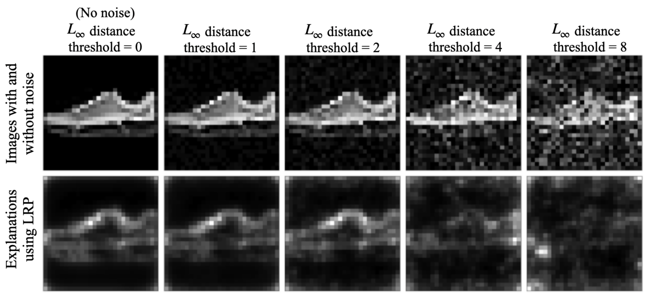

(No noise) L_∞ distance threshold = 0
L_∞ distance threshold = 1
L_∞ distance threshold = 2
L_∞ distance threshold = 4
L_∞ distance threshold = 8
Images with and without noise
Explanations using LRP

Figure 9: Effect of adding random perturbations, with increasing threshold of L_∞ distance, on explanations by LRP.

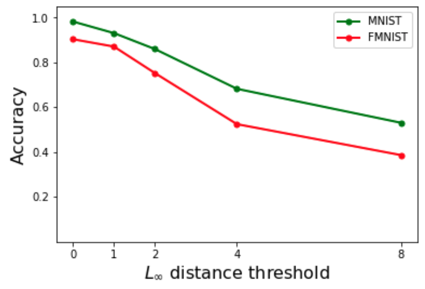


Figure 10: Impact of fragility of explanations [13].

In Section 5.1.2, we discussed an attack which shows the fragility of explanations [13]. Figure 9 shows an example of this attack for the case of random perturbations. The leftmost image in first row is a normal example from the Sneaker class of FMNIST dataset. The next image in first row shows a manipulated image created by adding random perturbations to the normal example such that the prediction remains unchanged, and the perturbations do not exceed a threshold value of 1, in terms of L_∞ distance. The next three images are created in a similar manner but with increased threshold values of 2, 4, and 8, respectively. The second row shows the corresponding explanations produced by LRP technique. We observe that with increased noise threshold, the explanations also become noisy. With our defense, if such explanations are classified as abnormal, then the corresponding inputs would be considered adversarial (false-positive). Figure 10 shows that adding random perturbations to normal examples results in a decline of the target classifier's classification accuracy with increasing noise threshold. We discussed in Section 5.1.2 that the impact of this attack depends on the nature of the application using the defense. For instance, our approach will not adversely impact applications which consider such perturbed examples as abnormal.

A Simple Framework for Bias Mitigation via Decorrelating Feature Influence

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

We demonstrate that algorithmic discrimination can be explained by the high reliance of models on fairness sensitive features, which denotes features that are highly predictive with protected attributes such as gender and race. Motivated by this observation, we propose to achieve fairness by decorrelating the main prediction task with those fairness sensitive features. Specifically, we firstly train a biased teacher model which is explicitly encouraged to employ fairness sensitive features for prediction. The teacher model then counter-teaches a debiased student model so as to enforce student model to capture complementary knowledge with teacher model. Experimental analysis indicates that our framework dramatically reduces model's attention on fairness sensitive features. Experimental results on four datasets further show that our proposed method could increase fairness in terms of three metrics, with a negligible decrease of (or better) classification accuracy.

1 Introduction

Deep learning is increasingly being used in high-stake decision making applications that affect individual lives. However, deep learning models might exhibit algorithmic discrimination behaviors with respect to protected groups. For example, a recruiting tool believes that men are more qualified and shows bias against women [1], facial recognition performs extremely poorly for darker skin females [2]. The fairness problem might cause adverse impacts on individuals and society. Therefore, designing mitigation methods to reduce unintentional bias has received much attention recently [3].

In this work, we show that the discrimination behavior is a direct result of our models' high reliance on fairness sensitive features in input. Here fairness sensitive features denote those features (e.g., ZIP code and surname) that are highly predictive of protected attribute (e.g., race). As a result, the main prediction task (e.g., mortgage application) would highly rely on the protected attribute (e.g., race) for prediction and introduce discrimination for certain group (e.g., African Americans).

Motivated by this observation, we propose a general framework for bias mitigation, called DeFI (Decorrelating Feature Influence), to disentangle the main prediction task and fairness sensitive features. The key idea is to suppress the model from capturing spurious correlations between fairness sensitive features with main prediction task, while forcing the model to concentrate on task relevant features. However, a key challenge lies in how to locate fairness sensitive features in input. One straightforward idea is to label the whole training set by crowd workers or domain experts. This would lead to suboptimal results. On one hand, crowd sourcing labelling is too time consuming. On the other hand, many seemingly innocuous features may be highly correlated with protected attribute and cause model bias. It is extremely hard to annotate all these features manually.

To tackle the challenges, we introduce a biased teacher network, which primarily leverages sensitive features in the input in order to succeed. Fairness sensitive features can be automatically localized by the biased teacher network. This teacher network could then counter-teach a debiased student network, so as to encourage the student to focus on more generalizable features for prediction. At test time, our method does not need access to sensitive attributes, since collecting sensitive attributes is often not allowed in real-world applications. In the context of fairness, there is more than a single protected attribute. Experimental results on several fairness benchmarks validate that the trained student network mainly relies on features that are more likely to generalize. We demonstrate that DeFI could reduce biases while could maintain original model prediction accuracy. Moreover, DeFI also could take into consideration of multiple protected attribute, and achieve compositional fairness.

2 Related Work

Fairness in Machine Learning. Work for fairness in machine learning mainly could be grouped into three categories: 1) proposing definitions of fairness in different contexts, such as individual fairness [4], demographic parity [5], and equality of opportunity [6, 7], 2) demonstration of biases in a variety of applications, such as facial recognition [2], sentiment analysis [1], word embeddings [8, 9], and 3) developing bias mitigation algorithms. The third category is the focus of this work.

Fairness Mitigation. Algorithms for mitigating discrimination could be further categorized into three categories: pre-processing, in-processing, and post-processing, depending on their stage at machine learning life-cycle (i.e., before training models, when training models, and after training models). Firstly, dataset refinement could be used before training the model [10, 11]. Secondly, regularization might be added as auxiliary term to overall loss function, explicitly or implicitly enforcing certain fairness metric [12, 13]. Representative examples include adversarial training, and incorporating priors into feature attribution [14]. Adversarial training is widely applicable for different DNN architectures and different kinds of input formats, including CNN with image data [15], RNN with text data [16], and MLP with tabular data [12, 17]. However, it offers a trade-off between fairness and accuracy, which indicates an under-utilization of information in the input. Thirdly, post-processing method could be employed after model training to calibrating predictions of trained models [18, 6]. Calibration takes the model's prediction and protected attribute to calibrate model's prediction. This method could be problematic in real-world applications since protected attributes usually cannot be obtained during inference time.

The most similar work to ours is an in-processing mitigation method that regularizes the interpretations of DNNs using prior knowledge [14]. However, the regularization requires fine-grained annotations about which features are fairness sensitive. In contrast, our method could automatically mine sensitive features. Besides, our method is capable of mitigating algorithmic bias while at the same time maintain prediction accuracy.

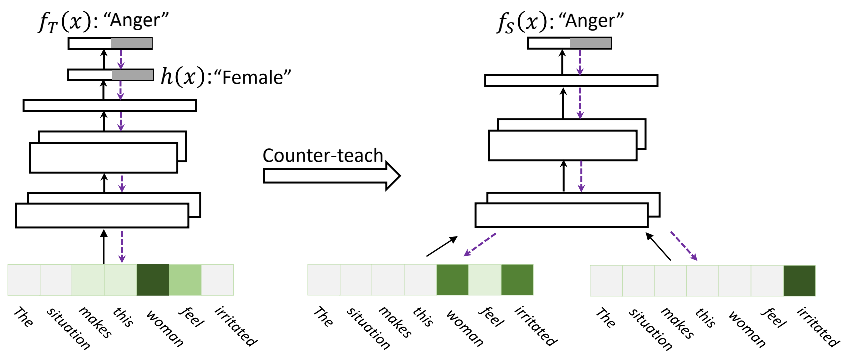
3 Decorrelating Feature Influence for Fairness

In this section, we introduce the proposed fairness mitigation method which decorrelates the influence of fairness sensitive features to the main prediction task. We first formulate it into a teacher-student framework, and present the teacher model which is constructed to deliberately maximize the usage of protected attributes for prediction. We then introduce how to use the biased teacher model to counter-teach the student model so as to obatain a debiased model.

3.1 Observations

Problem Statement. Consider a classification problem with labeled examples: $x, y, a \sim p_{data}$, where $x \in \mathcal{X}$ is input feature, and $y \in \mathcal{Y}$ is label that we want to predict. Besides, $a \in \mathcal{A} = \{0, 1\}$ is binary protected attributes, such as race, gender and age, where 0 indicates unprivileged groups, and 1 denotes privileged groups. Our goal here is to learn a classification model $\hat{y} = f(x)$ which is predictive of label y, while at the same time satisfying certain group fairness measurement with regard to protected attribute a. We restrict our attention in this work that a model makes a binary classification decision, i.e., $\mathcal{Y} = \{0, 1\}$. It is worth noting that protected attributes $\mathcal{A}$ is only accessible during training phase and we cannot use protected attributes during prediction time.

Feature Influence Analysis. We utilize local interpretation method to analyze feature importance vector for each input instance. Specifically Integrated Gradient is used for generating local inter-



(a) Teacher DNN (b) Student without teaching (C) Student after teaching

Figure 1: Illustrative example for proposed DeFI framework, where the task is for sentiment classification. (a) The biased teacher model mostly relies on fairness sensitive feature, i.e,, *woman* for prediction. (b) Without teaching, the student DNN will pay high attention to fairness sensitive features, (c) After counter-teaching from the teacher network, the student DNN will exclusively concentrate on generalizable features, i.e., *irritated*, for prediction.

pretation [19]. The analysis indicates that DNN models over rely on fairness sensitive features for prediction. An illustrative example is illustrated in Fig. 1(b). The fairness sensitive feature here is word '*woman*' for this sentiment classification task. Due to the data distribution imbalance in training set, those fairness sensitive features could be predictive of labels. Most current DNN models follow data-driven and end-to-end learning paradigm, and trained models would capture spurious correlation between fairness sensitive features and the main prediction task. As a result, the DNN models would show algorithmic discrimination towards demographic group.

3.2 The Idea of Decorrelating Feature Influence (DeFI)

Our key idea is to decorrelate feature influence from those fairness sensitive features to the main prediction task. One straightforward solution to utilize feature-level annotations, which specify which subset of features are fairness sensitive, and which parts are task relevant.

We propose to construct a biased teacher network which is trained to maximally utilize fairness sensitive features for prediction. Then the teacher network is further employed to counter-teach a debiased student network which could shift its attention from fairness sensitive features.

Constructing a Biased Teacher Network. Our hypothesis is that input contains fairness sensitive features and generalizable features. Our goal is to separate them, and enforce the models to rely on generalizable features to make prediction. Our training algorithm could automatically separate them.

We build a bias-only model, which maximally utilize the biased feature for prediction. The DNN is denoted as $f_T(x) = c(h(x))$, where $h(x)$ is the intermediate representation for input x, and $c(\cdot)$ is responsible to map intermediate representation to final model prediction. Note that $h(x)$ only contains $|\mathcal{A}|$ dimensions. For instance, if we consider gender bias, $h(x)$ is then denoted using two dimensions, indicating male and female respectively (see Fig. 1(a)).

A two stage strategy is utilized to train the biased-only teacher model $f_T(x)$. Firstly, we use the input and the protected attribute label $\{x_i, a_i\}_{i=1}^N$ to train the representation $h(x)$. The purpose is to maximize the bias information captured by the representation $h(x)$. Secondly, we utilize $\{h(x_i), y_i\}_{i=1}^N$ to train the mapping function $c(\cdot)$ to learn the mapping from $h(x)$ to $f_T(x)$. Ultimately, the teacher network $f_T(x)$ would maximumlly utilize biased information for prediction.

We illustrate the idea using Fig. 1(a). For the sentiment classification task, teacher network $f_T(x)$ mainly relies on fairness sensitive feature *'woman'* for prediction, while at the same time pays nearly no attention to generalizable feature *'irritated'*.

Counter-Teaching a Debiased Student Network. Our ultimate goal is to obtain a debiased DNN which minimally relies on fairness sensitive features for prediction. This is achieved by counter-teaching the student network, so as to enforce student network to employ complementary knowledge as teacher network. In the following, we would illustrate how to train a debiased student network through decorrelating feature influence.

3.3 Explicitly Decorrelating Feature Influence

In this section, we would introduce how to counter-teach the student network with the biased teacher network for bias mitigation. Some fairness sensitive features in input x_i could be used to predict protected attributes a_i with a high probability [5]. The existence of these attributes cause the discrimination of DNN models. The goal here is to explicitly discourage the model from capturing spurious correlations between fairness sensitive features between main task prediction. Specifically, we use local DNN interpretability to obtain the contribution of features towards model prediction [20, 21]. It is achieved by attributing the model's prediction in terms of its input features. The final interpretation is illustrated in the format of feature importance vectors, where a higher value indicating a higher contribution of that feature to model prediction. We explicitly regularize the interpretation for the student network with interpretation from teacher network, and the loss function is as follows:

$$\mathcal{L}_{explicit}(x) = \sum_{i=1}^{N} I(f_T(x_i), x_i) \odot I(f_S(x_i), x_i), \tag{1}$$

where each I represent local interpretation vector of x_i for teacher network and student network respectively, and $\odot$ denotes element-wise multiplication. Note that empirically we find that using $h(x_i)$ to replace $f_T(x_i)$ in Eq.(1) could better locate fairness sensitive features. Thus we use $h(x_i)$ instead to calculate $\mathcal{L}_{explicit}(x)$.

Interpretation Algorithm. We use Integrated Gradient [19], which is a back-propagation based interpretation method. The key idea is to integrate the gradient over the straightline path from baseline $x_{baseline}$ to input x_i, which could be denoted as follows:

$$I(f(x_i), x_i) = (x_i - x_{baseline}) \cdot \sum_{k=1}^{m} \frac{\partial f(x_{baseline} + \frac{k}{m}(x_i - x_{baseline})}{\partial x_i} \cdot \frac{1}{m}, \tag{2}$$

Note that for text application, since each input text is composed of T words: $x_i = \{x_i^t\}_{t=1}^{T}$. Each word $x_i^t \in \mathbb{R}^d$, denoting a word embedding with d dimensions. We first compute gradients of the output prediction with respect to individual entries in word embedding vectors, and use the L2 norm to reduce each vector of the gradients to a single attribution value, representing the contribution of each single word. Besides, we fix baseline input $x_{baseline}$ with zero value vector for tabular input and with zero word embedding for text input.

3.4 Implicitly Decorrelating Feature Influence

We could also train the debiased student network as an ensemble with biased teacher network. The key idea is to implicitly encourage student network to use alternative features in the input data. The loss function is given as follows:

$$p(x_i) = \text{softmax}(log(p_T(x_i)) + log(p_S(x_i))), \tag{3}$$

Suppose each input feature x could be split into two subsets of features: fairness sensitive features x_{sens} which is highly relevant to protected attribute a and the rest features x_{task} which is more relevant to the main prediction task. We could approximately decompose the model prediction by applying Bayes Rules as follows (see Sec.6 in Appendix for detailed proof):

$$p(y|x) = p(y|x_{sens}, x_{task}) \propto \underbrace{p(y|x_{task})}_{\text{Student}} \underbrace{p(y|x_{sens})}_{\text{Teacher}} /p(y). \tag{4}$$

The final decomposition contains three terms, where the first term is what we expect the student network to capture, and the second term denotes what the teacher network has learned. This decomposition indicates that the *implicit effect* of the ensemble training is to enforce student network to capture complementary features as teacher network to make decisions. Note that sometimes the teacher network could be too strongly biased towards certain prediction. Take Fig. 1(a) for example, the model could output a strong negative sentiment whenever the input is relevant to females. Empirically we find that if we directly add the teacher network output and student network output, the final student network could even demonstrate discrimination towards previously privileged groups, such as males or European Americans. To tackle this problem, we add a parameter α to adjust the impact of the teacher network:

$$p(x_i) = \text{softmax}(\alpha log(p_T(x_i)) + log(p_S(x_i))). \tag{5}$$

Note that for this process, the teacher network parameter is fixed, and only the parameters of the student network is updated using back-propagation.

Putting these two manners of counter-teaching together, the overall loss function is given as follows:

$$\mathcal{L}(x, y, a) = \mathcal{L}_{supv}(p_S, y) + \beta_1 \mathcal{L}_{explicit}(\theta, x) + \beta_2 \mathcal{L}_{implicit}(p, y), \tag{6}$$

where the first term is the standard cross entropy loss for debiased student network prediction p_S, aiming to enforce model to have correct predictions. The second term and third term are the explicit decorrelation and implicit decorrelation respectively, both of which are to enforce right for right reasons. Parameters β_1 and β_2 are used to balance these three terms.

4 Experiments

We evaluate our proposed fairness mitigation model DeFI against several state-of-the-art approaches.

4.1 Experimental Setup

We discuss the overall experimental setup, including datasets, baselines and implementation details.

4.1.1 Benchmark Datasets

We use four benchmark datasets. Adult Income (*Adult*) aims to predict whether a salary is greater than or less than 50K [22]. Medical Expenditure (*MEPS*) is a medical dataset aiming to predict whether a person would have 'high' utilization [23]. *COMPAS* is a dataset to predict criminal defendant's likelihood of reoffending (recidivism) [24]. Equity Evaluation Corpus (*EEC*) is utilized to examine inappropriate biases in sentiment analysis systems [1]. The dataset statistics and protected attribute we use for four datasets are presented in Tab. 1. More details are put in Appendix.

Table 1: Dataset Statistics

	Adult	**MEPS**	**COMPAS**	**EEC**
# **training instances**	31600	11080	3700	2940
# **validation instances**	4520	1482	523	420
# **test instances**	9102	3168	1055	840
protected attribute	gender	race	race	gender

4.1.2 Baseline Methods

Mitigation methods could be generally divided into three groups: pre-processing, in-processing, and post-processing, denoting dataset refinement before training, regularization during training, and calibration after training respectively [23, 11]. We compare our method with the following bias alleviation baseline methods, each corresponding to representative method of the three categories.

Optimized pre-processing [11] It is a pre-processing transformation technique to debias the training dataset. The transformation is formulated in a probabilistic framework, where features and labels are edited to ensure group fairness. For this method, we only use it for three tubular datasets.

Adversarial learning [12] Adversarial learning is a representative in-processing bias mitigation method, aiming to learn a classifier which could maximize prediction accuracy and simultaneously reduce an adversary's ability to determine the protected attribute. It leads to a fair classifier as the predictions cannot carry any group discrimination information that the adversary can exploit.

Equalized Odds Post-processing (EOP) [6] This is a model-agnostic post-processing method for fairness mitigation. It receives model predictions and corresponding protected attributes as input and then output mitigated output. The key idea is to enforce both demographic groups to have the same false positive rate and the same false negative rate.

4.1.3 Fairness Measurements and Implementation Details

We use *accuracy* $\mathcal{F}_{acc}$ to evaluate the utility of the model and three statistic (group) fairness metrics to assess the fairness of the model. The *demographic parity* metric [5] is defined as probability

Table 2: Mitigation comparison between 5 methods. For accuracy $\mathcal{F}_{acc}$ and demographic parity $\mathcal{F}_{parity}$, the closer to 1 the better. For equality of opportunity $\mathcal{F}_{opty}$ and equality of odds $\mathcal{F}_{odds}$, the closer to 0 the better. For Optimized_pre, we only use it for the first three datasets with tabular input.

	Adult Income				**MEPS**				**COMPAS**				**EEC**			
Models	$\mathcal{F}_{acc}$	$\mathcal{F}_{parity}$	$\mathcal{F}_{opty}$	$\mathcal{F}_{odds}$	$\mathcal{F}_{acc}$	$\mathcal{F}_{parity}$	$\mathcal{F}_{opty}$	$\mathcal{F}_{odds}$	$\mathcal{F}_{acc}$	$\mathcal{F}_{parity}$	$\mathcal{F}_{opty}$	$\mathcal{F}_{odds}$	$\mathcal{F}_{acc}$	$\mathcal{F}_{parity}$	$\mathcal{F}_{opty}$	$\mathcal{F}_{odds}$
DNN_original	84.1	0.856	-0.041	-0.088	86.5	0.857	-0.195	-0.243	69.0	0.709	-0.191	-0.365	88.3	0.632	-0.277	-0.445
Optimized_pre [11]	80.8	0.903	-0.105	-0.159	82.4	0.957	-0.082	-0.109	65.6	0.774	-0.163	-0.289	-	-	-	-
Adversarial_learning [12]	83.4	0.899	-0.087	-0.107	83.2	0.952	-0.059	-0.076	67.7	0.726	-0.191	-0.350	92.4	0.721	-0.258	-0.301
EOP [6]	81.3	0.884	-0.030	-0.093	83.5	0.925	-0.041	-0.078	63.5	0.721	-0.132	-0.299	80.2	0.929	-0.050	-0.097
DeFI	83.4	0.893	-0.034	-0.061	84.6	0.950	-0.057	-0.070	67.9	0.831	-0.095	-0.167	96.5	0.943	-0.019	-0.043

ratio of favorable outcome between unprivileged and privileged group: $\mathcal{F}_{parity} = \frac{p(\hat{y}=1|a=0)}{p(\hat{y}=1|a=1)}$, where $\hat{y}$ is a model prediction and 1 denotes favorable outcome. The *equality of opportunity* metric [6] is defined as true positive rate difference between unprivileged group and privileged group: $\mathcal{F}_{opty} = p(\hat{y} = 1|a = 0, y = 1) - p(\hat{y} = 1|a = 1, y = 1)$. *Equality of odds* metric [6] also takes false positive rate into consideration: $\mathcal{F}_{odds} = p(\hat{y} = 1|a = 0, y = 0) - p(\hat{y} = 1|a = 1, y = 0) + \mathcal{F}_{opp}$.

We use multilayer perceptron (MLP) as the model for tabular data input, and convolutional neural network (CNN) for text data input. For EEC dataset, we use the 300-dimension word2vec word embedding to initialize the embedding layer of CNN model. The hyper-parameter m for Integrated Gradient in Eq.(2) is fixed as 50 for all experiments. The balance weight α in Eq.(5) is set as 0.01, 0.06, 0.03, 0.001 for Adult, MEPS, COMPAS, ECC, respectively. Hyper-parameters (β_1, β_2) are selected as (1.5, 0.5), (1.5, 3), (0.5, 3), (0.1, 0.01) for Adult, MEPS, COMPAS, and ECC four datasets respectively. Note that all hyper-parameters are tuned based on trade-off between accuracy and three fairness metrics on validation set.

4.2 Fairness and Accuracy Evaluation

We report the fairness and accuracy performance on Tab. 2. Note that we set the threshold for all the DNN models as 0.5.

Comparison with Original DNN. For vanilla model without mitigation, i.e., DNN_original, the $\mathcal{F}_{parity}$ metric value is less than 0.9 for all four datasets. and the $\mathcal{F}_{odds}$ difference between two protected attributes range from 0.088 to 0.445, implying discrimination towards certain demographic group. For all four datasets, DeFI has consistently improved three fairness metrics, while at the same time has negligible accuracy drop (or better accuracy). Take MEPS dataset For instance, $\mathcal{F}_{parity}$ has been improved from 0.857 to 0.950 and $\mathcal{F}_{odds}$ has been improved from -0.243 to -0.07. This dramatically reduced discrimination for African American group, while at the same time only sacrificing 1.9% accuracy drop.

Comparison with Other Mitigation Methods. Of all the models, the proposed DeFI gives the best balance in terms of accuracy and fairness. In contrast, Optimized_pre fails to promote the model fairness in terms of all three metrics, indicating the limited ability of pre-processing method for bias mitigation. In addition, EOP could have comparable mitigation performance as DeFI. However, it possesses two limitations: 1) dramatic accuracy drop and 2) requiring reference time access to protected attributes. This is usually not practical in real-world applications, thus reducing the applicability of post-processing bias mitigation methods.

Adversarial learning could also simultaneously improve model performance in terms of three fairness metrics. However, it has come at the expense of relatively lower accuracy, such as 3.3% accuracy drop on MEPS dataset. When it has similar accuracy as DeFI, it has larger discrimination, such as for Adult dataset where $\mathcal{F}_{odds}$ equals -0.107 for adversarial learning and -0.061 for DeFI. One possible explanation is that adversarial learning could potentially remove other useful information that the model could rely on to make decisions.

4.3 Compositional Fairness

We use MEPS dataset to investigate the mitigation of compositional fairness (combination of multiple sensitive attributes [25]), since MEPS has labels available for gender and race two attributes.

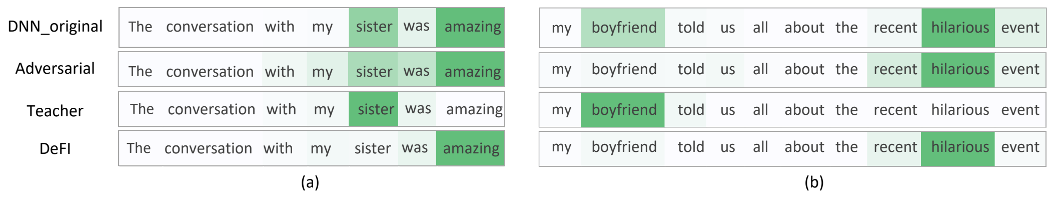

DNN_original
Adversarial
Teacher
DeFI
The conversation with my sister was amazing
my boyfriend told us all about the recent hilarious event
(a)
(b)

Figure 2: Illustrative examples of interpretations. The proposed method DeFI could mainly focus on task relevant sentiment features, i.e., *amazing* and *hilarious*, for prediction.

Limitation of Regularizing One Attribute. In real-world applications, there usually exist more than one protected attribute. One side effect is that the bias reduction of one attribute could possibly enlarge bias of another protected attribute. Take MEPS dataset for example, as shown in Tab. 3. The regularization of race attribute has improved model (i.e., DeFI) performance in terms of all three fairness metrics. However, DeFI at the same time sacrifices some fairness metrics for gender attribute ($\mathcal{F}_{opty}$ from -0.052 to -0.081, and $\mathcal{F}_{odds}$ from -0.076 to -0.095). The main reason is that DNN models tend to take *shortcuts* to make predictions, i.e., capturing the correlation between protected attributes and main prediction task. The suppression of one shortcut (race) by proposed DeFI framework would force model to amplify its reliance on other shortcut (gender).

Compositional Fairness. We could achieve compositional fairness by training multiple biased teacher models for each protected attribute respectively. For MEPS dataset, we train two biased teachers for race and gender two protected attributes respectively. As shown in Tab. 3, the final model DeFI_combo has improved three fairness metrics for both race and gender attributes comparing to DNN_original. More encouragingly, there is only 0.4% accuracy drop for DeFI_combo comparing to DeFI.

Table 3: Compositional fairness.

	Accuracy	Race Bias			Gender Bias		
Models	$\mathcal{F}_{acc}$	$\mathcal{F}_{parity}$	$\mathcal{F}_{opty}$	$\mathcal{F}_{odds}$	$\mathcal{F}_{parity}$	$\mathcal{F}_{opty}$	$\mathcal{F}_{odds}$
DNN_original	86.5	0.857	-0.195	-0.243	0.938	-0.052	-0.076
DeFI	84.6	0.950	-0.057	-0.070	0.961	-0.081	-0.095
DeFI_combo	84.2	0.955	-0.061	-0.076	0.983	-0.032	-0.036

4.4 Interpretation for Sanity Check

In this section, we use EEC dataset to analyze the connections of interpretation with model bias.

Interpretation Visualizations. We illustrate interpretation visualizations for 4 comparing models in Fig. 2. There are some key findings. Firstly, The teacher network could highlight all fairness sensitive features, such as *sister* and *boyfriend*. This is a major advantage of the teacher network. Due to the redundant encodings, other seemingly innocuous features may be highly correlated with protected attribute and cause model bias. The teacher network could tell us not only which subsets of features are highly relevant to protected attributes, but also the corresponding likelihood. This kind of information cannot be easily obtained by crowd workers or even domain experts. Secondly, the models DNN_original focus comparable attention on fairness sensitive features and generalizable features. Thirdly, the debiased DeFI learns to pay less attention to those fairness sensitive features. Instead, DeFI mainly capture more generalizable features for prediction, i.e., *amazing* and *hilarious*. This demonstrates DeFI has captured complementary information as teacher network.

Quantitative Evaluation of Interpretations. We manually select out fairness sensitive features and sentiment features from EEC dataset, where the full list is given in Tab. 6 in Appendix. Then the bias degree of the models are defined as average ratio between interpretation feature importance values of two list of features: $\mathcal{F}_{bias} = \frac{1}{n}\sum_{i=1}^{n}\frac{p_{sensitive}}{p_{sentiment}}$, where the smaller $\mathcal{F}_{bias}$, the less attention is paid by the model for fairness sensitive features. The results are reported in Tab. 4. It indicates that original DNN pays comparable attention, i.e., 0.35, to fairness sensitive features and sentiment features, leading to its discrimination behavior. For teacher network, it mainly focus on sensitive features, with a ratio of 2918.52. With this teacher network, DeFI dramatically reduces its attention for fairness sensitive features (from 0.35 to 0.05).

Table 4: Interpretation ratio

Models	$\mathcal{F}_{bias}$
DNN_original	0.35
Adversarial_learning	0.21
Teacher	2918.52
DeFI	0.05

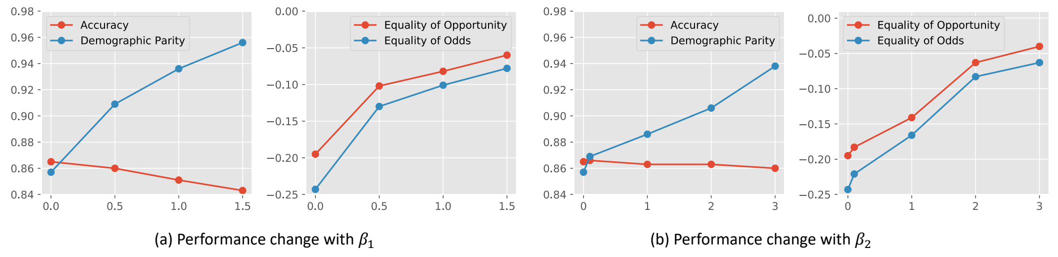


Figure 3: Hyperparameters analysis for β_1 and β_1. (a) Model performance change with respect to change of β_1. (b) Model performance change with respect to change of β_2.

4.5 Ablation and Hyperparameters Analysis

We conduct ablation studies using MEPS dataset to study the contribution of components of DeFI.

Table 5: Ablation analysis.

	Accuracy	Race Bias		
Models	$\mathcal{F}_{acc}$	$\mathcal{F}_{parity}$	$\mathcal{F}_{opty}$	$\mathcal{F}_{odds}$
DNN_original	86.5	0.857	-0.195	-0.243
DeFI	84.6	0.950	-0.057	-0.070
DeFI_explicit	84.3	0.956	-0.061	-0.078
DeFI_implicit	86.0	0.938	-0.037	-0.063

Ablation Analysis. The proposed DeFI has two components for counter-teaching from the teacher network: DeFI_explicit and DeFI_implicit. We conduct ablation studies to analyze their contributions, and report the results in Tab. 5. There are two main findings. Firstly, both DeFI_explicit and DeFI_implicit could improve the model with regard to all three fairness metrics. Secondly, DeFI_explicit and DeFI_implicit bring different benefits and they are complementary to each other. Specifically, DeFI_explicit has more improvement in terms of demographic parity $\mathcal{F}_{parity}$ (from 0.857 to 0.956). In contrast, DeFI_implicit has more significant improvement for $\mathcal{F}_{opty}$ (from -0.195 to -0.037) and $\mathcal{F}_{odds}$ (from -0.243 to -0.063). Besides, DeFI_implicit has relatively higher accuracy than DeFI_explicit.

Hyperparameters Analysis. We evaluate the effects of two major hyperparameters of DeFI, i.e., β_1 and β_1 in Eq.(6). Note that when studying effects of one hyperparameter, the other one would be set as zero. The results are illustrated in Fig. 3. Same trends could be observed from both the change of β_1 and β_2. Specifically, as the number of β_1 and β_2 increases, stronger regularization would be imposed for the student network. As a result, model fairness would increase in terms of three fairness metrics, while at the same time model accuracy could be sacrificed. Another interesting observation is that accuracy change is less significant for β_2 when achieving the same level of bias mitigation, comparing to changes of β_1.

5 Conclusions

This work is based on the observation that algorithmic discrimination is mainly caused by model's high reliance on the fairness sensitive features in input. We propose a simple bias mitigation framewrok, called DeFI, to decorrelate influence of fairness sensitive features for main prediction task. DeFI first trains a biased teacher network with protected attributes such as gender and race as supervision signal, and then counter-teaches a debiased student network. We consider protected attributes to be available during training time but not at test time. Despite the simplicity, we show that DeFI could significantly increase DNN performance in terms of three group fairness measurements. Experimental analysis further shows that DeFI has significantly less drop in accuracy comparing to other fairness mitigation methods.

References

[1] Svetlana Kiritchenko and Saif M Mohammad. Examining gender and race bias in two hundred sentiment analysis systems. *Proceedings of the 7th Joint Conference on Lexical and Computational Semantics*, 2018.

[2] Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on Fairness, Accountability and Transparency (FAT*)*, pages 77–91, 2018.

[3] Mengnan Du, Fan Yang, Na Zou, and Xia Hu. Fairness in deep learning: A computational perspective. *arXiv preprint arXiv:1908.08843*, 2019.

[4] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*, 2012.

[5] Michael Feldman, Sorelle A Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. Certifying and removing disparate impact. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2015.

[6] Moritz Hardt, Eric Price, Nati Srebro, et al. Equality of opportunity in supervised learning. In *Advances in neural information processing systems (NIPS)*, 2016.

[7] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rodriguez, and Krishna P Gummadi. Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment. In *International Conference on World Wide Web (WWW)*, 2017.

[8] Jieyu Zhao, Tianlu Wang, Mark Yatskar, Ryan Cotterell, Vicente Ordonez, and Kai-Wei Chang. Gender bias in contextualized word embeddings. *North American Chapter of the Association for Computational Linguistics (NAACL)*, 2019.

[9] Yi Chern Tan and L Elisa Celis. Assessing social and intersectional biases in contextualized word representations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[10] Faisal Kamiran and Toon Calders. Data preprocessing techniques for classification without discrimination. *Knowledge and Information Systems (KAIS)*, 2012.

[11] Flavio Calmon, Dennis Wei, Bhanukiran Vinzamuri, Karthikeyan Natesan Ramamurthy, and Kush R Varshney. Optimized pre-processing for discrimination prevention. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3992–4001, 2017.

[12] Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. Mitigating unwanted biases with adversarial learning. In *AAAI/ACM Conference on Artificial Intelligence, Ethics, and Society (AIES)*, 2018.

[13] Christopher Clark, Mark Yatskar, and Luke Zettlemoyer. Don't take the easy way out: Ensemble based methods for avoiding known dataset biases. *2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

[14] Frederick Liu and Besim Avci. Incorporating priors with feature attribution on text classification. *57th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019.

[15] Tianlu Wang, Jieyu Zhao, Mark Yatskar, Kai-Wei Chang, and Vicente Ordonez. Balanced datasets are not enough: Estimating and mitigating gender bias in deep image representations. *International Conference on Computer Vision(ICCV)*, 2019.

[16] Yanai Elazar and Yoav Goldberg. Adversarial removal of demographic attributes from text data. *2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

[17] Alex Beutel, Jilin Chen, Zhe Zhao, and Ed H Chi. Data decisions and theoretical implications when adversarially learning fair representations. *Fairness, Accountability, and Transparency in Machine Learning (FAT/ML)*, 2017.

[18] Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. Men also like shopping: Reducing gender bias amplification using corpus-level constraints. *2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2017.

[19] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. *International Conference on Machine Learning (ICML)*, 2017.

[20] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73:1–15, 2018.

[21] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. 2017.

[22] Ron Kohavi. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD)*, volume 96, pages 202–207, 1996.

[23] Rachel KE Bellamy, Kuntal Dey, Michael Hind, Samuel C Hoffman, Stephanie Houde, Kalapriya Kannan, Pranay Lohia, Jacquelyn Martino, Sameep Mehta, Aleksandra Mojsilovic, et al. Ai fairness 360: An extensible toolkit for detecting, understanding, and mitigating unwanted algorithmic bias. *arXiv preprint arXiv:1810.01943*, 2018.

[24] J. Angwin, J. Larson, S. Mattu, and L. Kirchner. How we analyzed the compas recidivism algorithm, 2016.

[25] Avishek Joey Bose and William Hamilton. Compositional fairness constraints for graph embeddings. *International Conference on Machine Learning (ICML)*, 2019.

Table 6: Fairness sensitive features and sentiment features.

	Fairness Sensitive Features	Sentiment Features
Word List	she, her, woman, women, girl, sister, wife, daughter, girlfriend, mother, aunt, mom he, him, man, men, boy, brother, son, husband, boyfriend, father, uncle, dad	ecstatic, excited, glad, happy, relieved amazing, funny, great, hilarious, wonderful angry, annoyed, enraged, furious, irritated annoying, displeasing, irritating, outrageous, vexing

6 Appendix A: More on Methodology

We could approximately decompose the model prediction by applying Bayes Rules as follows:

$$p(y|x) = p(y|x_{sens}, x_{task}) \propto p(y|x_{task})p(x_{sens}|y, x_{task}), \tag{7}$$

Also suppose these two sets of features x_{sens} and x_{task} are conditionally independent given label y. By further using Bayes Rules, we could further obtain:

$$p(y|x) \propto p(y|x_{task})p(x_{sens}|y) = p(y|x_{task})\frac{p(y|x_{sens})p(x_{sens})}{p(y)} \propto \underbrace{p(y|x_{task})}_{\text{Student}} \underbrace{\frac{p(y|x_{sens})}{p(y)}}_{\text{Teacher}}. \tag{8}$$

7 Appendix A: More on Datasets

Adult. Originally, Adult dataset contains 48842 instances, we have deleted all rows where a value equals 'nan'. As a result, 3620 instances have been deleted, and the size of the final dataset is 45222.

MEPS.

COMPAS. Black defendants were often predicted to be at a higher risk of recidivism than they actually were. For this dataset, we use the simplified version with 10 features [1].

EEC. We manually inject noise to the training dataset, to enable the dataset to biased towards females. Specifically, in the training set, females are more relevant to the negative sentiment of anger, and males are more relevant to positive sentiment of joy.

8 Appendix B: More on DNN Architectures

Since the focus of this work is for fairness mitigation, we only utilize simple architectures for main task prediction. The detailed architectures are given as follows:

CNN.

MLP.

9 Appendix C: More on Baseline Methods

Adversarial learning [12]

10 Appendix D: More on Experiments

why there could be no trade-off: it depends on whether training and test set distribution is the same or not.

[1]`https://github.com/IBM/AIF360`