Deep Semi-Supervised Learning via Dynamic Anchor Graph Embedding in Latent Space

Enmei Tu, Zihao Wang, Jie Yang

Department of Automation, Shanghai Jiao Tong University, Shanghai, China

Nikola Kasabov

School of Engineering, Computing and Mathematical Science, Auckland University of Technology, New Zealand and Intelligent Systems Research Center, Ulster University UK

Abstract

Recently, deep semi-supervised graph embedding learning has shown much promise on text and image recognition tasks when the number of labeled data is limited. By introducing an auxiliary unsupervised task of predicting the neighborhood context in the graph, these approaches effectively mine the structure information provided by abundant unlabeled data. However, existing methods usually adapt to datasets whose graph is explicitly given or predefined, which cannot handle large-scale datasets with unknown graphs. Besides, the edge connections and weights are fixed during the training process in these methods, which fails to use the current feature information extracted by the model. In this paper, we propose a novel deep semi-supervised dynamic anchor graph embedding learning algorithm. More specifically, we build a two-branch architecture to learn the single-sample local features and the global features in the graph simultaneously. The first branch constrains its outputs to be consistent with different perturbations of the same single sample. And the output features are dedicated to constructing the dynamic anchor graph. The second branch utilizes the anchor graph and the model prediction to sample the context, based on which the global graph embeddings are learned. Finally, the outputs of the two branches are aggregated to jointly predict the class labels. Extensive experimental results on several image and text datasets have shown that the proposed method is able to improve the performance of existing graph embedding learning methods and outperform many state-of-the-art methods on image classification.

Keywords: Grap Embedding, Semi-Supervised Learning, Dynamic Anchor Graph, Image / Text Classification

1. Introduction

Deep neural networks leveraging a large number of labeled data have achieved remarkable performance on computer vision [1, 2] and natural language processing [3, 4] applications in recent years. However, labeling numerous data manually is extremely expensive or even impossible for many practical applications (e.g., medical image segmentation [5], destructive product testing [6]). In these situations, deep learning models are prone to overfitting and generalize badly on new data, due to that the labeled training data are scarce. To mitigate this problem, deep semi-supervised learning (DSSL) that utilizes a small amount of labeled data plus many easily-available unlabeled data to train the model has achieved very promising performance. For comprehensive reviews of traditional and recent semi-supervised learning (SSL) approaches, readers are referred to [7, 8, 9]. Among the existing DSSL algorithms, graph-based deep semi-supervised learning (GDSSL) is particular interesting because of the solid mathematical background of graph theory, efficient matrix computational implementation and the great success of tradition graph-based SSL [7]. Different from traditional grid-structured data (such as vectors, time sequences and images), the emergence of graph-structured data in many applications (such as social media data [10], paper citation graph [11]) is not in Euclidean domain data and seeking new approaches to process them drives the study of graph based algorithms [12].

GDSSL usually formulates a classification task as a graph node classification problem. Recall that a graph, denoted as $G(V, E, W)$, is a data structure which consists of a node set (or vertex set) V to represent a group of instances (samples), an edge set E to indicate the pairwise connections of the nodes, and a weight matrix W whose elements are the respective weights of the edges[1]. Given a graph G that only a few of its nodes are tagged by some categorical labels, GDSSL algorithms aims to leverage the graph structural information and the correlation between the labeled and unlabeled nodes to train a deep model and then classifies all nodes into the categories of the labels. The aim has broadly been accomplished with two philosophies. The first one is based on graph similarity regularization [13] (mainly graph Laplacian), which explicitly forces the model's predictions to be consistent with the graph structure (i.e. strong connected graph nodes must produce similar model outputs). For example, [14, 15, 16, 17]. However, a recent study shows that the asymptotic behavior of graph Laplacian regularization tends to be unstable [18] while the graph grows to large. The second philosophy is to transform the graph data into grid data, i.e. embedding the graph into a Euclidean space (called embedding space, which is a latent space to encode the graph nodes.), using node context on the graph (such as neighborhood structure, random walks sequences) and then predicts node categorical labels in the embedding space [19, 20, 21]. However, existing embedding approaches in this category mostly utilize "***static graph***", which means the graph is given or predefined and the structure of the graph is fixed during training. This cause at least two possible limitations. First, the static graph usually contains noisy nodes/edges (especially for complex data

[1]A graph can also be denoted simply as $G(V, W)$, in which edge set E is implicitly represented by the non-zero elements in W.

such as natural image) which are likely to mislead the learning process. Second, model can hardly benefit from the useful information in the abstract features mined by itself. This is inefficient and not consistent with our knowledge about human learning, in which past experiences play important roles [22].

To address these issues, we propose a latent space Dynamic Anchor Graph Embedding (DAGE) semi-supervised learning approach in this paper. Our key observation is that for classification purpose, the sequential random walk on a full graph in latent space can be approximately represented by a random walk on an anchor graph. This enables us to include model extracted information flexibly and, meanwhile, to reduce the embedding learning problem size dramatically. To be more specific, we build a two-branch network architecture including a single-sample consistency branch and a dynamic graph embedding branch, as shown in Fig. 1. We construct a dynamic bipartite graph using hidden-layer features of the consistency branch and perform bipartite random walks on the graph to generate random sequences representing graph structural information. Then, we train the embedding branch utilizing node labels and random walk sequences jointly to generate more accurate node embedding features. Finally, the two branches are aggregated to accomplish the classification task. The single-sample consistency branch learns local information of a sample under a consistency regularization constraint. The graph embedding branch learns graph structural information using the context generated from both the anchor graph and the model prediction. So, the final model prediction incorporates both local sample consistency information and global graph structural information to classify a sample. To summarize, our contributions are as follows:

- We propose a dynamic anchor graph embedding learning approach in the latent-space trained by an improved context sampling strategy, which combines graph structural information and model discriminant information of both labeled and unlabeled samples. Based on this, we present a novel semi-supervised learning framework to achieve graph embedding and node classification simultaneously for both graph-structured data and grid-structured data.

- We develop a new explicit weight decay technique to improve training efficiency and model performance. Comparing with previous weight-ensembling techniques such as fast-SWA [23], the new technique could enhance significantly model diversity within a few training epochs and yield better model generalization.

- We perform extensive experiments to compare the proposed method with existing state-of-the-art methods on popular image and text benchmark datasets. The results demonstrate the superiority of our approach on both tasks.

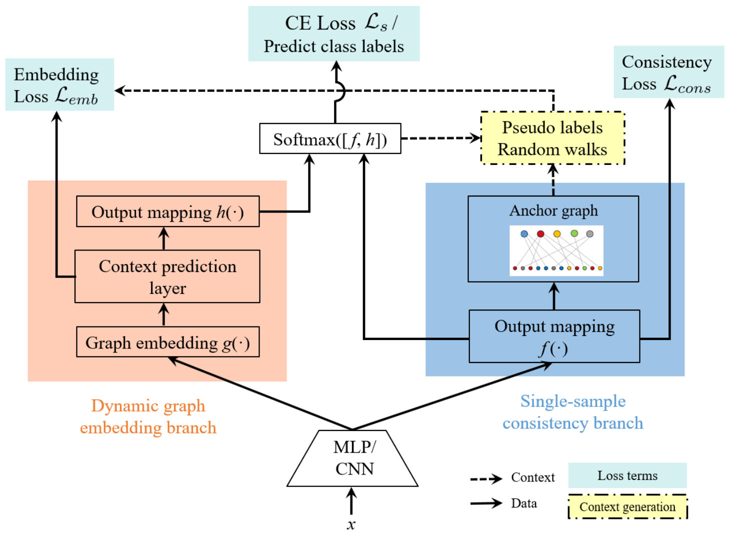

CE Loss $\mathcal{L}_s$ / Predict class labels
Embedding Loss $\mathcal{L}_{emb}$
Consistency Loss $\mathcal{L}_{cons}$
Softmax([f, h])
Pseudo labels Random walks
Output mapping $h(\cdot)$
Context prediction layer
Graph embedding $g(\cdot)$
Anchor graph
Output mapping $f(\cdot)$
Dynamic graph embedding branch
Single-sample consistency branch
MLP/ CNN
x
Context
Data
Loss terms
Context generation

Figure 1: The proposed neural network framework for dynamic anchor graph embedding (DAGE) learning.

2. Related works

2.1. Consistency-Based Models

Consistency means that given different perturbed versions of a sample (e.g. adding noises or transformations, etc.), model output should be consistent, i.e. generate same outputs, because the inputs are essentially the same thing [24, 25, 26, 27]. In these approaches, a student model is enforced to output similar predictions to that of a teacher model when fed different perturbed versions of the same sample. Teacher model can be same as student model [24, 28] or different from student model [26, 27]. Main perturbations include added input Gaussian noise, dropout and data augmentation for images. Later, there are two directions to improve consistency-based models. One is to improve the quality of the teacher model. For instance, Temporal Ensembling (TempEns)[26] calculates the temporal average values of model predictions as consistency target. MT [27] averages model weights instead of predictions. Fast-SWA [23] obtains the ensemble "teacher" by averaging weights at different epochs directly. The other is to improve the quality of perturbations. To name a few, Adversarial perturbation is regarded as a new, effective perturbation in VAT [29] and VAdD [30]; WCP [31] uses additive and DropConnect perturbations and AutoAugment [32] automatically generates perturbations by learning operations in a designed search space.

2.2. Graph Embedding Learning

Graph embedding is to assign a unique coordinate to each graph node so that the graph can be transformed to grid data and *embedded* into a Euclidean space (called embedding space), because graph structural data are not Euclidean (i.e. not possess natural coordinates arranged in regular grids like vectors or images). Ideally, to facilitate subsequent learning tasks such as node classification or link prediction, the embedding space should be low dimensional and the properties of the graph (such as topology, subgraphs, adjacency, etc.) should be maximally preserved in the embedding space.

Early graph embedding learning methods (e.g. Deepwalk [33], LINE [34], etc.) are purely unsupervised and do not leverage label information during training. To the best of our knowledge, Planetoid [20] firstly combines supervised classification with unsupervised graph embedding learning task and shows preferable performance on semi-supervised text recognition. Graph Convolutional Networks (GCN) recently have been shown to be effective to learn graph embeddings [35, 19, 36]. However, as mentioned above, most of these algorithms are designed to use "static graph", which is explicitly given or pre-constructed using coarse hand-craft features. Besides, GCN also suffers from over-smoothing effect and high computational burden on large datasets due to the nature of Laplacian graph filtering [37].

3. Preliminaries

In this section we first define the semi-supervised learning problem. Then we review briefly basics of consistency regularization and Planetoid [20] which are related to the proposed approach.

3.1. Semi-Supervised Learning

We mainly focus on semi-supervised classification problem. Suppose f_θ is a classifier parameterized by θ defined by a neural network. $\mathcal{X} = \{x_i\}_{i=1}^n$ with $x_i \in \mathcal{R}^d$ is a sample dataset for grid-structured data, or the node feature set V for graph-structured data $G(V, W)$[2]. Without loss of generality, we assume the first l samples in $\mathcal{X}$ are labeled and denoted as $\mathcal{X}_L$. The rest are unlabeled and denoted as $\mathcal{X}_U$ ($l \ll n$ in most cases). $\mathcal{Y}_L = \{y_i\}_{i=1}^l$ (with $y_i \in C = \{j\}_{j=1}^c$) are categorical labels corresponding to $\mathcal{X}_L$. The binary label matrix of $\mathcal{Y}_L$ is denoted as Y, whose elements are $Y_{ij} = 1$ if and only if sample x_i is from class j. Semi-supervised learning is usually achieved by optimizing (1).

$$\min_\theta \sum_{i=1}^{l} \mathcal{L}_s(f_\theta(x_i), y_i) + \mathcal{L}_u(f_\theta(\mathcal{X}_L, \mathcal{X}_U)) \tag{1}$$

where $\mathcal{L}_s$ is the supervised classification loss (e.g. cross-entropy loss). $\mathcal{L}_u$ is usually the unsupervised regularization term (e.g. consistency regularization, context prediction term, etc.) which fully leverages unlabeled samples' information.

[2]We will use *sample* and *node* interchangeably to mean the same x.

3.2. Consistency Regularization

In consistency-based models, consistency regularization term is usually formulated as (2).

$$\mathcal{L}_{cons}(x, \theta) = \mathbb{E}_x \| f(x, \theta) - f(\tilde{x}, \theta') \|^2 \tag{2}$$

where $\tilde{x}$ is a different perturbed version of sample x, θ, θ' are model parameters of student model and teacher model respectively, f is hidden layer feature or classification output. Except for Mean Squared Error, one can also use KL divergence to measure the perturbation difference.

The teacher model's parameter set θ' is exponential moving average value of θ in MT [27], as shown in (3).

$$\theta'_t = \alpha \theta'_{t-1} + (1 - \alpha)\theta_t \tag{3}$$

where θ_t is the student parameter set at time step t. α is smoothing coefficient, which is usually 0.99 or 0.999.

3.3. Graph Embedding Loss

Given a sample x_i, its corresponding context ct is sampled based on random walk strategy (for all training samples) or ground truth (for labeled samples). Planetoid [20] trains graph embeddings to jointly predict class labels and context in the graph. To do this, a graph embedding loss is defined in (4).

$$\begin{aligned}\mathcal{L}_{emb} = \sum_i & -\mathbb{I}(\gamma = 1) \ log \ \sigma(w_{ct}^T g(x_i)) \\ & - \mathbb{I}(\gamma = -1) \ log \ \sigma(-w_{ct}^T g(x_i))\end{aligned} \tag{4}$$

where $\mathbb{I}(\cdot)$ is indicator function that outputs 1 when the condition in parentheses is true, otherwise 0. $\gamma = 1(-1)$ means ct is positive (negative) context, w_{ct} are model weights of context prediction, $g(\cdot)$ is graph embedding. $\sigma(x) = 1/(1 + \exp(-x))$ is sigmoid function, which represents the probability that x_i matches context ct. $\mathcal{L}_{emb}$ measures the cross entropy between sample pair (i, c) and binary label γ. Note that to evaluate the embedding loss for all training data, Plantoid has to define an output for each training sample to predict node context. This could be burdensome for large scale dataset.

4. Dynamic Anchor Graph Embedding

In this section, we present our Dynamic Anchor Graph Embedding (DAGE) learning approach for semi-supervised classification for both graph-structured data and grid-structured data. The key idea is to construct a dynamically updated bipartite graph in latent space to capture the model extracted information and simplify graph embedding prediction capacity. First, we elaborate the idea and construction of dynamic anchor graph in latent space. Following, we will describe in details the components of the framework and training techniques, namely, the single-sample consistency branch, dynamic graph embedding branch and model optimization techniques.

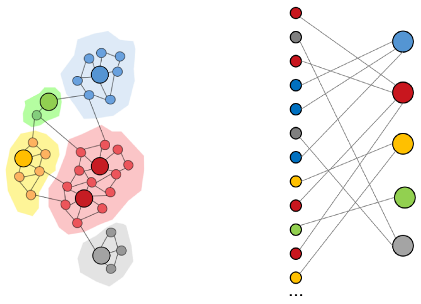
4.1. Dynamic Anchor Graph in Latent Space

The wisdom of traditional unsupervised graph embedding learning such as DeepWalk [33] first generates a sequence of random walks for each node (called the context of the node) and then trains a model to predict the context of each node correctly. While this strategy has been very successful for *intrinsic* graph structured data ("born" with a well-defined graph, e.g. social networks, paper citation graph), it generally does not perform equally well on general data tasks such as image classification, because constructing a *semantically meaningful* graph for the data can be as difficult as the node classification problem itself [38].

So, instead of predicting the context in original data space, we consider the problem of predicting the context of a node in latent space[3] and utilize random walks on a bipartite graph to further simplify the problem. Previous studies have shown that classification/clustering in the latent space (or hidden layer space) of a neural network model is more advantageous since the algorithm could exploit the model extracted and abstract information to improve results [39, 17]. The motivation of our approach is illustrated in Fig 2. Suppose an embedding model maps samples into latent space and are organized into clique structure. We can select from each clique some points as anchoring points (i.e. representatives, for examples, by using k-means/k-medoids clustering center, or simply random selected samples), the distribution of which well summarizes the distribution of to full data. Then, we construct a bipartite graph (anchor graph) between anchoring points and non-anchoring points. In this case, local random walks on full data distribution can be well represented by random walks on the bipartite graph, because two samples who are more likely to hit each other on full distribution will also be more likely to share the same anchoring point on the anchor graph. So, rather than training an embedding model to predict random walk sequences on full data distribution, as in DeepWalk [33] and Plantoid [20], we train the model to predict the anchor sequences in a random walk on the bipartite graph. This enables us to reduce the prediction problem from full dataset size (of predicting nearby neighbors of each node) to anchor set size (of predicting nearby anchors of each node), which can be a fixed and much smaller number than the graph node number.

To implement DAGE, we propose a neural network framework in Fig 1. The feature extraction network (e.g. a CNN or MLP) maps an input sample x into hidden feature space, followed by a consistency branch and an embedding branch in parallel. The consistency branch enforces its output to be invariant to input perturbations (i.e. noise and transformations on x) and thus learns node local information. The outputs of $f(\cdot)$ are used to construct an anchor graph as shown in Fig 2. The embedding branch performs graph embedding learning using context information from both the anchor graph and predicted

[3]Because graph embedding is implemented by a neural network mapping, so here we do not distinguish embedding space or latent space, which are the same thing that is defined by neural network hidden layers.



(a) Clique structure in latent space. (b) Anchor graph representation.

Figure 2: Anchor graph in latent space. Different colors indicate different cliques and big dots are anchors.

pseudo labels and thus learns graph structural information. The outputs of the two branches are concatenated to pass a softmax function to obtain predicted pseudo labels. By encouraging output consistency of the same sample, single-sample local features are optimized but global features, which imply relationships between different samples in the graph, are not considered in previous consistency-based models, and vice versa for existing graph embedding learning methods. In our framework, we combine them together to make use of their complementary characteristics. We explain each part in details in the following subsections.

4.2. Single-Sample Consistency Branch

Consistency has been an important technique in many state-of-the-art DSSL algorithms [15, 28, 40]. We apply the same consistency constraint and model ensembling technique for $f(\cdot)$ as in (2). Notice that ν is a subset of parameter θ (i.e. the consistency branch).

$$\begin{aligned} \mathcal{L}_{cons} &= \mathbb{E}_x \| f(x, \nu) - f(\tilde{x}, \nu') \|^2 \\ \nu'_t &= \alpha \nu'_{t-1} + (1 - \alpha) \nu_t \end{aligned} \tag{5}$$

To construct the anchor graph, we denote $\mathcal{U} = \{u_j\}_{j=1}^m$ as anchoring point set of size m in the input space. We then calculate the anchor graph similarity matrix Z at each training step using normalized features $f_{norm}(\cdot)$ by (6). Notice that the elements of Z are dynamically adjusted during training, which enables the model to utilize newer and more accurate connections in latent space than that of the predefined static graph.

$$\begin{aligned} Z_{ij} &= \begin{cases} \frac{K_\beta(f_{norm}(x_i), f_{norm}(u_j))}{\sum_{j' \in \langle i \rangle} K_\beta(f_{norm}(x_i), f_{norm}(u'_j))}, & \forall j \in \langle i \rangle \\ 0, & \text{otherwise} \end{cases} \\ f_{norm}(x_i) &= \frac{f(x_i)}{(\sum_k (f(x_i)_k)^2)^{\frac{1}{2}}} \end{aligned} \tag{6}$$

where $Z \in \mathcal{R}^{n\times m}$ denotes the relationship between samples $\mathcal{X}$ and anchors $\mathcal{U}$. $f_{norm}(\cdot)$ is normalized with the output feature $f(x)$ of single-sample consistency branch. $K_\beta(x,y) = \exp(-\|x-y\|^2/2\beta^2)$ is Gaussian kernel function with a coefficient hyperparameter β. $\langle i\rangle \subset [1:m]$ is the s nearest anchors' index set of sample x_i. Since we only consider s closest anchors to x_i and $s \ll m$, the similarity matrix Z is highly sparse. An anchor graph representation of our method is shown in Fig. 3.

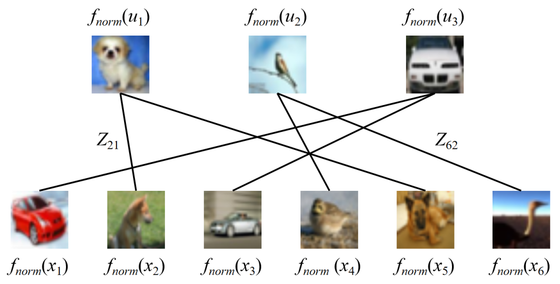


Figure 3: An anchor graph representation of samples $x_1,\cdots,x_6$ and anchors u_1,u_2,u_3 with normalized features $f_{norm}(\cdot)$, Z_{ij} captures the relationships between samples and anchors.

The construction process of matrix Z formulated in (6) is based on features learned by the model. For some datasets (e.g. citation graph data), if a pre-existed graph is given, then we firstly construct the matrix Z using the given graph, as shown in (7).

$$Z_{ij} = \begin{cases} \frac{W_{ij}}{\sum_{l\in\mathcal{U}} W_{il}}, & \text{if } j \in \mathcal{U} \wedge \exists\, l \in \mathcal{U}, W_{il} \neq 0 \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

where W_{ij} denotes the edge weight of the given graph. Then, we use matrix Z defined above to pre-train our model and thereafter replace it with (6) to establish new connections between samples and anchors.

4.3. Dynamic Graph Embedding Branch

In this branch, the anchor graph constructed in previous subsection is used to learn graph embedding dynamically. To do so, two types of context (positive context ct^p and negative context ct^n) can be sampled based on the anchor graph and pseudo labels jointly. Finally, we could calculate the graph embedding loss to train graph embeddings.

More specifically, we treat m anchors as context space $\mathcal{C}$ (target space of models) that graph embeddings need to predict. Consequently, the dimension of context prediction layer connected with graph embedding layer $g(\cdot)$ is reduced from $b\times n$ to $b\times m$, where b is batch size. Since the number of anchor points m is a fixed hyperparameter, the width of context prediction layer is fixed, no longer increases with the data size. Therefore our approach is more suitable to large-scale datasets.

A key step of graph embedding learning is to generate the context of a node. Different from existing context generation strategy as in Planetoid [20], we propose an improved sampling strategy based on anchor graph structure and pseudo labels that makes use of both labeled and unlabeled data. The anchor graph similarity matrix Z in (6) and (7) represents the probability p that a certain point moving from sample x to anchor u (i.e., a one-step random walk from a sample point to an anchoring point). We select both positive contexts from nearest anchors and negative contexts from farthest anchors according to Z. In this case, the embedding not only pull similar nodes close, but also push dissimilar nodes far. We also further utilize pseudo labels of all training samples instead of just using labeled samples. This enriches training data and provide much richer discriminant information to guide the model learning process.

All of the above lead to our context generation strategy, which is given in Algorithm 1. Given a sample x_i, there are two types of context in the algorithm that need to be sampled, i.e. ct^p and ct^n representing the positive and negative context, respectively. Each type of context can be generated in either two ways (represented by $\gamma = +1$ and $\gamma = -1$, respectively):

- The first one is based on the anchor graph similarity matrix Z, which encodes the underlying graph structural information of data.
- The second one is based on the pseudo labels, which injects the model discriminant information into the graph embeddings.

The model leverages both of the two types of information when training embeddings. The ratio of positive and negative contexts is controlled by parameter $r_1 \in (0, 1)$, and the ratio of two sampling ways is controlled by parameter $r_2 \in (0, 1)$. The algorithm's sampling strategy on the well-known "two moons" synthetic dataset is illustrated in Fig. 4 (in the case of ct^p).

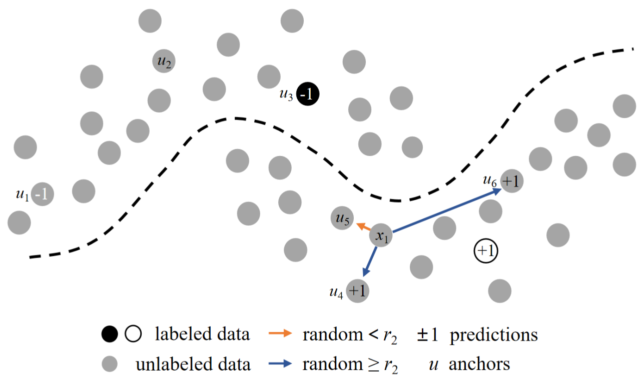


Figure 4: The strategy sampling ct^p on "two moons" dataset. The figure illustrates two different sampling ways based on Z and pseudo labels.

Given triplet (x_i, ct, γ), we could use graph embedding loss (4) to train the dynamic graph embedding branch. In this way, the global features which rep-

Algorithm 1 Sampling Context Triplet (x_i, ct, γ)

```
Require: anchor graph similarity matrix $Z$, pseudo labels $\{\tilde{y}_i\}_{i=1}^n$, number of
  neighbors $s$ and non-neighbors $o$, hyperparameters $r_1, r_2$
  if $random < r_1$ then
    $\gamma = +1$
  else
    $\gamma = -1$
  end if
  if $random < r_2$ then
    Choose $s$ closest anchors to sample $x_i$, uniformly sample positive context
    $ct^p$ according to $Z_{ij}, j \in \langle i \rangle$
    if $\gamma = -1$ then
      Choose $o$ farthest anchors to sample $x_i$, uniformly sample negative con-
      text $ct^n$
    end if
  else
    if $\gamma = +1$ then
      Uniformly sample positive context $ct^p$ with $\tilde{y}_i = \tilde{y}_{ct}$
    else
      Uniformly sample negative context $ct^n$ with $\tilde{y}_i \neq \tilde{y}_{ct}$
    end if
  end if
  return triplet $(x_i, ct^p, \gamma)$ or $(x_i, ct^n, \gamma)$
```

resent relations between samples and anchors are learned because neighboring positive sample-context pairs are encouraged to be closer and negative sample-context pairs are pushed farther. Since the context space $\mathcal{C}$ is composed of anchor set $\mathcal{U}$ here, (4) could also be regarded as a multi-label classification loss with m labels. As will be shown in experiments, the improved context generation strategy could significantly enhance model performance.

4.4. Model Optimization Process

As elaborated above, the two branches are trained to learn sample-perturbed local features and the graph-structured global features, respectively. Then, we concatenate the outputs of $f(\cdot)$ and $h(\cdot)$ to predict the class labels of the input sample x. The overall semi-supervised objective function is defined in (8).

$$\mathcal{L} = \mathcal{L}_s(f_\theta(x), y) + \lambda_1 \mathcal{L}_{emb} + \lambda_2 \mathcal{L}_{cons} \tag{8}$$

where $\mathcal{L}_s$ is supervised cross-entropy loss, λ_1, λ_2 are balancing coefficients trading off between supervised and unsupervised loss terms. For a labeled sample, we compute all loss terms to update model weights and for an unlabeled sample, we compute the last two unsupervised loss terms to update model weights.

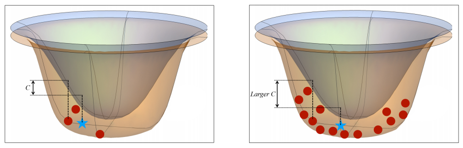
To improve learning efficiency, we introduce a new method to increase the diversity of model weights directly. Fast-SWA [23] shows that weight diversity is crucial for model generalization. However, it applies the averaging strategy to epochs with a large training interval in order to maintain the diversity of student model parameters. Therefore it needs longer training length to integrate more diverse models. As a result, the training efficiency of fast-SWA is extremely low. Here, we propose an explicit weight decay model ensembing method calculated by (9).

$$\theta_t = (1 - \epsilon)\theta_{t-1} \tag{9}$$

where ϵ is a coefficient hyperparameter. Note that it is decoupled with learning rate when using SGD optimizer [41] and different from L2 regularization when applying Adam optimizer [42]. Comparing with [43], we remove the scaling factor η_t, aiming to add a larger penalty that greatly improves the parameters' diversity. Equation (9) could be applied to the online teacher model ensembling. At each training step, we first take a gradient descent step on θ_{t-1}. Then we use (9) to update the parameters of student model directly. Finally, we apply (3) to integrate the teacher model with more diverse "students" generated by (9). Notice that we apply (9) and (3) to the whole parameter set θ, which could increases the quality of consistency target $f(\tilde{x}, \nu')$ in (5).

To understand why the new method could improve parameters' diversity of the student model at each iteration and enhance training efficiency, the performance gains of fast-SWA and our proposed method with the same training length are depicted in Fig. 5. In experiment we will show that the ensemble teacher model's performance and training efficiency is dramatically improved by combining (9) and (3).

Given these, the training process are as follows. Firstly, taking a randomly selected batch of samples and m anchors as network's input, we could obtain



(a) Fast-SWA [23] (b) Ours

Figure 5: Train error surface (orange) and test error surface (blue) (approximately convex refer to [44]). Red dots depict different model weights, blue star represents the ensemble model. C denotes the performance gain. Our method could integrate more diverse models than fast-SWA with the same training length to achieve a larger C.

their output features and pseudo labels $\tilde{y}$. Here, we simply select m samples randomly from training data to make the anchoring point set and the rest samples are treated as non-anchoring points, because k-means or k-medoids cost significant computations on large dataset but the final results are similar to random selection. Then we sample corresponding triplets (x_i, ct, γ) based on (6) and Algorithm 1. Finally we take the batch of samples as input again and calculate loss $\mathcal{L}$ in (8) to update the parameters of student models and teacher models.

5. Experiments

In this section, we conduct experiments on two image datasets (SVHN [45] and CIFAR-10 [46]) and three text datasets (Citeseer, Cora and Pubmed [47]) to demonstrate the effectiveness of our approach. We compare our results with popular methods, including recently proposed state-of-the-art methods.

5.1. Datasets and Preprocessing

Image Datasets. In SVHN, there are 73257 training samples and 26032 test samples of size $32 \times 32 \times 3$. Each sample is any digit from 0 to 9. We scale each image to zero mean and unit variance. CIFAR-10 has a training set of 50000 samples and a test set of 10000 samples, with the same size as SVHN. There are also 10 classes including dog, bird, car, etc. We normalize the samples in CIFAR-10 by per-channel standardization. In addition, we perform standard data augmentation for SVHN and CIFAR-10 (2-pixel random translation on both datasets and random horizontal flip on CIFAR-10).

Text Datasets. In text datasets, sample features are bag-of-words representations of documents (i.e., binary vectors that represent corresponding words whether appear or not in Citeseer and Cora, TF-IDF vectors in Pubmed). Citation links between the documents are also given. We set $W_{ij} = W_{ji} = 1$ in (7) if document x_i cites x_j. The details of text datasets are listed in Table 1.

Table 1: Details of Text Datasets

Dataset	**Train**	**Test**	**Classes**	**Features**	**Edges**
Citeseer	2312	1000	6	3703	4732
Cora	1708	1000	7	1433	5429
Pubmed	18717	1000	3	500	44338

5.2. Implementation Details

For image datasets, we evaluate the test error rate with different number of labels. The labeled samples are randomly selected with a random seed. And we run the model for 5 times with different seeds to report the mean and standard deviation of the test errors. We adopt the commonly used "CNN-13" architecture [26, 27, 23, 31] as the feature extraction model in Fig. 1 for fair comparison. The dimension of its output is $b\times6\times6\times128$. $f_{cons}(\cdot)$ is obtained by global average pooling of the output. The parameter settings of image datasets are tuned by cross validation and listed in Table 2. Since we randomly select a batch of samples from the training set, the number of labels in a batch is various ('vary'). And we define a epoch when the batch traverses all training samples. Because the model is inaccurate at early training stage, we ramp up λ_1, λ_2 and learning rate from 0 to their maximum values at the first 80 epochs. And the learning rate is ramp down to 0 at the last 50 epochs. We tune the parameters according to the technique in [48].

Table 2: Parameter Settings of Image Datasets

Parameters	**Value**
Dimension of $f(\cdot)$	$b\times128$
Dimension of $g(\cdot)$	$b\times1024$
Dimension of $h(\cdot)$	$b\times128$
Anchors m	500
Neighbors and non-neighbors (s, o)	(5, 100)
β	1.0
(r_1, r_2)	(0.5, 0.2)
Max λ_1	0.2
Max λ_2 for SVHN, CIFAR-10	1.5, 8.0
Batch size and labels in a batch (b, b_l)	(100, 'vary')
Explicit weight decay ϵ	0.0001
Optimizer	Adam
Max learning rate	0.003
Training length	400 epochs

For text datasets, we use the same data splits as in [20]. Specifically, we randomly select 20 samples from each class as labeled data, 1000 samples as test data, and the rest remain as unlabeled data. We compute the average accuracy here. Notice that our approach is inductive (i.e., model could predict unobserved samples when testing) similar to Planetoid-I [20]. We use direct connections instead of MLP in Fig. 1 because we find complicated MLP leads to overfitting. Similarly, the parameter settings of text datasets are tuned by cross validation and listed in Table 3[4]. Similarly, we ramp up λ_1 from 0 to its maximum value at the first 4000 iterations. Besides, we do not use data augmentation on text datasets for fair comparison. Thus $\mathcal{L}_{cons}$ hardly works ($|f(x,\nu)-f(x,\nu')| \to 0$). We set $\lambda_2 \equiv 0$. And dynamic anchor graph embedding loss $\mathcal{L}_{emb}$ acts as the only unsupervised regularization of the model.

Table 3: Parameter Settings of Text Datasets

Parameters	**Value**
Dimension of $f(\cdot)$ for 3 datasets	6, 7, 3
Dimension of $g(\cdot)$	$b\times 20$
Dimension of $h(\cdot)$ for 3 datasets	6, 7, 3
Anchors m	200
Neighbors and non-neighbors (s, o)	(5, 50)
β	1.0
(r_1, r_2)	(0.5, 0.2)
Max λ_1	3.25
λ_2	0.0
Batch size and labels in a batch (b, b_l)	(100, 20)
Explicit weight decay ϵ	0.00005
Optimizer	SGD, Momentum=0.5
Max learning rate	0.1
Training length	30000 iterations

5.3. Image Classification Results

We perform experiments on SVHN with 250, 500, 1000 labels and CIFAR-10 with 1000, 2000, 4000 labels respectively. The number of labeled samples are balanced referring to other DSSL algorithms. We compare our method with recently state-of-the-art consistency-based DSSL models (i.e., Π model [26], TempEns [26], MT [27], VAT [29], VAdD [30], fast-SWA [23], WCP [31]) and GDSSL models (i.e., LPDSSL [16], SNTG [15]). Notice that we use the same CNN model and data augmentation as these algorithms for fair comparison. And the results of LPDSSL on SVHN are reproduced by us. The test error rates are listed in Table 4 and Table 5 respectively. Numbers in parentheses indicate the training length (epochs).

[4]We use a different "o" in vision tasks because there are more samples on image datasets.

Table 4: Test Error Rates (%) on SVHN

Method	**250 labels**	**500 labels**	**1000 labels**
Π model [26]	9.93 ± 1.15	6.65 ± 0.53	4.82 ± 0.17
TempEns [26]	12.62 ± 2.91	5.12 ± 0.13	4.42 ± 0.16
MT [27]	4.35 ± 0.50	4.18 ± 0.27	3.95 ± 0.19
VAT [29]	-	-	5.42 ± 0.22
VAdD [30]	-	-	4.26 ± 0.14
WCP [31]	4.29 ± 0.10	3.75 ± 0.11	3.58 ± 0.19
LPDSSL [16]	18.45	9.49	7.38
Π+SNTG [15]	5.07 ± 0.25	4.52 ± 0.30	3.82 ± 0.25
TempEns+SNTG [15]	5.36 ± 0.57	4.46 ± 0.26	3.98 ± 0.21
Ours[a] (400)	**3.70 ± 0.21**	**3.68 ± 0.24**	**3.51 ± 0.12**

[a]Teacher model.

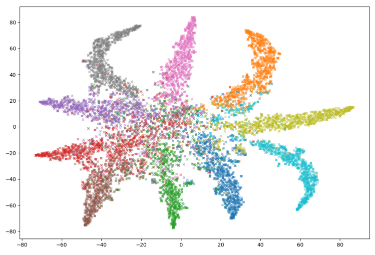
Table 5: Test Error Rates (%) on CIFAR-10

Method	**1000 labels**	**2000 labels**	**4000 labels**
Π model [26]	31.65 ± 1.20	17.57 ± 0.44	12.36 ± 0.31
TempEns [26]	23.31 ± 1.01	15.64 ± 0.39	12.16 ± 0.24
MT [27]	21.55 ± 1.48	15.73 ± 0.31	12.31 ± 0.28
VAT [29]	-	-	11.36 ± 0.34
VAdD [30]	-	-	11.32 ± 0.11
WCP [31]	17.62 ± 1.52	11.93 ± 0.39	9.72 ± 0.31
fast-SWA (480)[23]	16.84 ± 0.62	12.24 ± 0.31	9.86 ± 0.27
fast-SWA (1200)[23]	15.58 ± 0.12	11.02 ± 0.23	9.05 ± 0.21
LPDSSL [16]	22.02 ± 0.88	15.66 ± 0.35	12.69 ± 0.29
Π+SNTG [15]	21.23 ± 1.27	14.65 ± 0.31	11.00 ± 0.13
TempEns+SNTG [15]	18.41 ± 0.52	13.64 ± 0.32	10.93 ± 0.14
Ours[a] (400)	**14.00 ± 0.49**	**10.70 ± 0.43**	**9.01 ± 0.24**

[a]Teacher model.

Notably, our approach achieves state-of-the-art results on SVHN and CIFAR-10 with standard data augmentation. The error rates are 0.59%, 0.07% and 0.07% lower than WCP on SVHN and 1.58%, 0.32% and 0.04% lower than fast-SWA (1200) on CIFAR-10. The superiority of our approach becomes more obvious as the number of labels decreases. The strong performance suggests that the local output features of single-sample consistency branch and global outputs of graph embedding branch could effectively cooperate to predict class labels in our model. Besides, explicit weight decay in (9) largely improves the parameters' diversity of the student model, which in turn improves the teacher's performance. Specifically, our model is much more efficient than fast-SWA. We obtain better results with a training length of 400 than fast-SWA with 1200 epochs.

The t-SNE [49] visualizations of our method and Planetoid [20] on CIFAR-10 with 4000 labels are depicted in Fig. 6. From the figure one can see that in our method, different classes are better separated than Planetoid.


(a) Planetoid

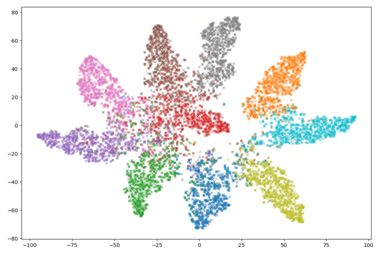

(b) Ours

Figure 6: t-SNE Visualizations of graph embeddings of our model and Planetoid on the test set of CIFAR-10 respectively. Each color denotes a class.

We also provide ablation studies on CIFAR-10 with 4000 labels to verify the effectiveness of explicit weight decay and the combination of two sampling branches, which is shown in Table 6 and Table 7 respectively.

Table 6: Ablation Studies with Different ϵ

Explicit Weight Decay Coefficient	**Test Error Rates**	
	Teacher Model	***Student Model***
$\epsilon = 0$	11.51	12.53
$\epsilon = 0.00005$	9.46	10.63
$\epsilon = 0.0001$	**8.77**	19.07

Table 6 indicates that as ϵ increases, the prediction ability of student model is reduced to some extent. However, the performance gains of teacher model greatly improves since the parameters' diversity of the "student" increases. And Table 7 shows that combination of two sampling branches makes the model

Table 7: Ablation Studies with Different Sampling Branches

Sampling Branch	**Test Error Rates**
Only based on matrix Z	9.32
Only based on pseudo labels	8.96
Combination of two branches	**8.77**

generalize better.

5.4. Text Classification Results

We compare our approach with three inductive graph-based semi-supervised learning approaches including manifold regularization (MR) [50], semi-supervised embedding (SSEmb)[14] and Planetoid-I [20]. The results are displayed in Table 8. "Feat" is a baseline method, which is a linear softmax model taking the feature x as input directly.

Table 8: Average Classification Accuracy (%) on Text Datasets

Method	**Citeseer**	**Cora**	**Pubmed**
Feat	57.2	57.4	69.8
MR [50]	60.1	59.5	70.7
SSEmb [14]	59.6	59.0	71.1
Planetoid-I [20]	64.7	61.2	**77.2**
Ours	**67.1**	**65.1**	76.5

From Table 8, we can see that our dynamic anchor graph embedding learning approach outperforms other approaches by a large margin on Citeseer and Cora dataset. Specifically, an accuracy improvement of 2.4% and 3.9% compared to Planetoid-I respectively. And on Pubmed, the performance of our model is very close to Planetoid-I (only 0.7% lower). Notice that we do not use augmentation or consistency regularization. All of these demonstrate that dynamic graph could discover new, accurate connections between samples and anchors.

We also conduct parameter sensitivity experiment with different number of anchors on Citeseer dataset, as presented in Fig. 7. When the number of anchors is small, they cannot cover the whole dataset, which has a bad influence on model performance. However, as the anchor size m increases, the performance gradually improves and finally tends to be saturated, which conforms to the statement in the above section.

6. Conclusions

Recently, different DSSL algorithms have been developed to classify grid-structured data or graph-structured data [9, 51], but most of them force on one

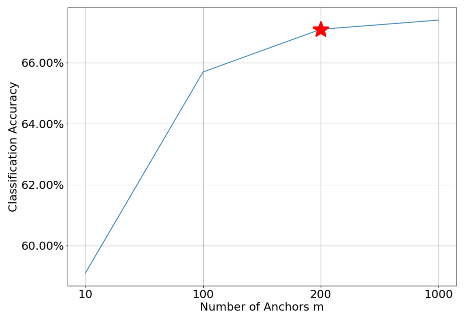


Figure 7: Parameter sensitivity experiment with different number of anchors m. We set $m = 200$ (star) in our experiment.

type data or the other. In this paper, we proposed a dynamic anchor graph embedding learning algorithm which is capable of classifying both types and is verified on image and text data classification. The model is trained jointly by local features of single-sample consistency branch and global features of dynamic graph embedding branch. The main contributions include: a) in contrast to previous graph embedding learning approaches, we ***dynamically*** construct an anchor graph using the stronger outputs of single-sample consistency branch, then graph embeddings are trained utilizing both the graph and the pseudo labels using labeled and unlabeled data; b) we introduce an explicit weight decay that largely increases student model's diversity, and its combination with model ensembling (temporal weight averaging) dramatically improves training efficiency and generalization performance. Experimental results on five benchmark datasets have demonstrated the effectiveness of our approach.

References

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

[2] M. Tan, Q. Le, Efficientnet: Rethinking model scaling for convolutional neural networks, in: International conference on machine learning, 2019, pp. 6105–6114.

[3] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, in: Proceedings of the 2019 conference of the north american chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers), 2019, pp. 4171–4186.

[4] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, et al., Language models are few-shot learners, in: Advances in neural information processing systems, 2020.

[5] Z. Xie, E. Tu, H. Zheng, Y. Gu, J. Yang, Semi-supervised skin lesion segmentation with learning model confidence, in: ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2021, pp. 1135–1139.

[6] P. Baguley, R. Roy, J. Watson, Cost of physical vehicle crash testing, in: Collaborative Product and Service Life Cycle Management for a Sustainable World, Springer, 2008, pp. 113–121.

[7] X. J. Zhu, Semi-supervised learning literature survey.

[8] J. E. Van Engelen, H. H. Hoos, A survey on semi-supervised learning, Machine learning 109 (2) (2020) 373–440.

[9] G.-J. Qi, J. Luo, Small data challenges in big data era: A survey of recent progress on unsupervised and semi-supervised methods, IEEE Transactions on Pattern Analysis and Machine Intelligence.

[10] I. Pitas, Graph-based social media analysis, Vol. 39, CRC Press, 2016.

[11] W. Lu, J. Janssen, E. Milios, N. Japkowicz, Y. Zhang, Node similarity in the citation graph, Knowledge and Information Systems 11 (1) (2007) 105–129.

[12] A. Ortega, P. Frossard, J. Kovačević, J. M. Moura, P. Vandergheynst, Graph signal processing: Overview, challenges, and applications, Proceedings of the IEEE 106 (5) (2018) 808–828.

[13] A. J. Smola, R. Kondor, Kernels and regularization on graphs, in: Learning theory and kernel machines, Springer, 2003, pp. 144–158.

[14] J. Weston, F. Ratle, H. Mobahi, R. Collobert, Deep learning via semi-supervised embedding, in: Neural networks: Tricks of the trade, 2012, pp. 639–655.

[15] Y. Luo, J. Zhu, M. Li, Y. Ren, B. Zhang, Smooth neighbors on teacher graphs for semi-supervised learning, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 8896–8905.

[16] A. Iscen, G. Tolias, Y. Avrithis, O. Chum, Label propagation for deep semi-supervised learning, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2019, pp. 5070–5079.

[17] K. Kamnitsas, D. Castro, L. Le Folgoc, I. Walker, R. Tanno, D. Rueckert, et al., Semi-supervised learning via compact latent space clustering, in: International conference on machine learning, 2018, pp. 2459–2468.

[18] A. El Alaoui, X. Cheng, A. Ramdas, M. J. Wainwright, M. I. Jordan, Asymptotic behavior of\ell_p-based laplacian regularization in semi-supervised learning, in: Conference on Learning Theory, PMLR, 2016, pp. 879–906.

[19] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, arXiv preprint arXiv:1609.02907.

[20] Z. Yang, W. Cohen, R. Salakhudinov, Revisiting semi-supervised learning with graph embeddings, in: International conference on machine learning, 2016, pp. 40–48.

[21] X. Yan, L. Zhang, W.-J. Li, Semi-supervised deep hashing with a bipartite graph., in: Proceedings of the international joint conference on artificial intelligence, 2017, pp. 3238–3244.

[22] B. M. Wagar, M. J. Dixon, Past experience influences object representation in working memory, Brain and Cognition 57 (3) (2005) 248–256.

[23] B. Athiwaratkun, M. Finzi, P. Izmailov, A. G. Wilson, There are many consistent explanations of unlabeled data: Why you should average, in: International conference on learning representations, 2019.

[24] A. Rasmus, H. Valpola, M. Honkala, M. Berglund, T. Raiko, Semi-supervised learning with ladder networks, arXiv preprint arXiv:1507.02672.

[25] M. Sajjadi, M. Javanmardi, T. Tasdizen, Regularization with stochastic transformations and perturbations for deep semi-supervised learning, in: Advances in neural information processing systems, 2016, pp. 1163–1171.

[26] S. Laine, T. Aila, Temporal ensembling for semi-supervised learning, in: International conference on learning representations, 2017.

[27] A. Tarvainen, H. Valpola, Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results, in: Advances in neural information processing systems, 2017, pp. 1195–1204.

[28] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, C. Raffel, Mixmatch: A holistic approach to semi-supervised learning, arXiv preprint arXiv:1905.02249.

[29] T. Miyato, S.-i. Maeda, M. Koyama, S. Ishii, Virtual adversarial training: A regularization method for supervised and semi-supervised learning, IEEE transactions on pattern analysis and machine intelligence 41 (8) (2018) 1979–1993.

[30] S. Park, J. Park, S.-J. Shin, I.-C. Moon, Adversarial dropout for supervised and semi-supervised learning, in: Proceedings of the AAAI conference on artificial intelligence, Vol. 32, 2018.

[31] L. Zhang, G.-J. Qi, Wcp: Worst-case perturbations for semi-supervised deep learning, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2020, pp. 3912–3921.

[32] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, Q. V. Le, Autoaugment: Learning augmentation strategies from data, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 113–123.

[33] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in: Proceedings of ACM SIGKDD international conference on knowledge discovery and data mining, 2014, pp. 701–710.

[34] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, Q. Mei, Line: Large-scale information network embedding, in: Proceedings of the 24th international conference on world wide web, 2015, pp. 1067–1077.

[35] M. Gori, G. Monfardini, F. Scarselli, A new model for learning in graph domains, in: Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005., Vol. 2, IEEE, 2005, pp. 729–734.

[36] S. Zhang, H. Tong, J. Xu, R. Maciejewski, Graph convolutional networks: a comprehensive review, Computational Social Networks 6 (1) (2019) 1–23.

[37] Q. Li, Z. Han, X.-M. Wu, Deeper insights into graph convolutional networks for semi-supervised learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32, 2018.

[38] L. Qiao, L. Zhang, S. Chen, D. Shen, Data-driven graph construction and graph learning: A review, Neurocomputing 312 (2018) 336–351.

[39] B. Yang, X. Fu, N. D. Sidiropoulos, M. Hong, Towards k-means-friendly spaces: Simultaneous deep learning and clustering, in: international conference on machine learning, PMLR, 2017, pp. 3861–3870.

[40] K. Sohn, D. Berthelot, C.-L. Li, Z. Zhang, N. Carlini, E. D. Cubuk, A. Kurakin, H. Zhang, C. Raffel, Fixmatch: Simplifying semi-supervised learning with consistency and confidence, arXiv preprint arXiv:2001.07685.

[41] L. Bottou, Large-scale machine learning with stochastic gradient descent, in: Proceedings of the international conference on computational statistics, 2010, pp. 177–186.

[42] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: International conference on learning representations, 2015.

[43] I. Loshchilov, F. Hutter, Decoupled weight decay regularization, in: International conference on learning representations, 2019.

[44] I. J. Goodfellow, O. Vinyals, Qualitatively characterizing neural network optimization problems, in: International conference on learning representations, 2015.

[45] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A. Y. Ng, Reading digits in natural images with unsupervised feature learning, in: Advances in neural information processing systems workshops, 2011.

[46] A. Krizhevsky, Learning multiple layers of features from tiny images, Master's thesis, University of Tront.

[47] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, T. Eliassi-Rad, Collective classification in network data, AI magazine 29 (3) (2008) 93–93.

[48] A. Oliver, A. Odena, C. A. Raffel, E. D. Cubuk, I. Goodfellow, Realistic evaluation of deep semi-supervised learning algorithms, in: Advances in neural information processing systems, 2018, pp. 3235–3246.

[49] L. v. d. Maaten, G. Hinton, Visualizing data using t-sne, Journal of machine learning research 9 (Nov) (2008) 2579–2605.

[50] M. Belkin, P. Niyogi, V. Sindhwani, Manifold regularization: A geometric framework for learning from labeled and unlabeled examples, Journal of machine learning research 7 (Nov) (2006) 2399–2434.

[51] P. Goyal, E. Ferrara, Graph embedding techniques, applications, and performance: A survey, Knowledge-Based Systems 151 (2018) 78–94.