Diffusion Mechanism in Residual Neural Network: Theory and Applications

Tangjun Wang, Zehao Dou, Chenglong Bao, and Zuoqiang Shi

Abstract—Diffusion, a fundamental internal mechanism emerging in many physical processes, describes the interaction among different objects. In many learning tasks with limited training samples, the diffusion connects the labeled and unlabeled data points and is a critical component for achieving high classification accuracy. Many existing deep learning approaches directly impose the fusion loss when training neural networks. In this work, inspired by the convection-diffusion ordinary differential equations (ODEs), we propose a novel diffusion residual network (Diff-ResNet), internally introduces diffusion into the architectures of neural networks. Under the structured data assumption, it is proved that the proposed diffusion block can increase the distance-diameter ratio that improves the separability of inter-class points and reduces the distance among local intra-class points. Moreover, this property can be easily adopted by the residual networks for constructing the separable hyperplanes. Extensive experiments of synthetic binary classification, semi-supervised graph node classification and few-shot image classification in various datasets validate the effectiveness of the proposed method.

Index Terms—Diffusion, residual neural network, ordinary differential equation, semi-supervised learning, few-shot learning

1 Introduction

ResNet [1] and its variants, containing skip connections among different layers, are promising network architectures in deep learning. Compared to non-residual networks, ResNet significantly improve the training stability and the generalization accuracy. To understand the success of ResNet, a recent line of works build up its connection with ordinary differential equations (ODEs) [2], [3], [4]. Let $x \in \mathbb{R}^n$ be a data point, the ODE model of a ResNet is

$$\frac{dx(t)}{dt} = f(x(t), \theta(t)), \quad x(0) = x. \tag{1}$$

where $f(x, \theta)$ is a map parametrized by θ. It is straightforward that the forward Euler discretization of (1) recovers the residual connection, which motivates the connections between ResNets and ODE. Based on the above observation, many recent works are proposed from two perspectives: the ODE inspired neural networks and the neural network based ODE. In concrete, the attempts for the ODE inspired neural networks can be classified into two directions. One approach for designing networks is to unroll the ODE system via different discretization schemes, which build up an end-to-end mapping. Typical networks include PolyNet [5], FractalNet [6] and linear multi-step network [7]. The other approach is to add some new blocks into the current network architecture by the modification of the ODE model, e.g. noise injection [8], stochastic dynamic system [9], adding a damping term [10]. Due to the strong mathematical foundation of ODE, the network architectures proposed in the above works have shown the improved explainability and performance. On the other hand, the neural network based ODE model parametrizes the velocity f by a neural network and finds the parameters θ via the optimal control formulation [11], [12]. These methods improves the expressive ability of a traditional ODE method and exhibit promising results in various problems, including systems with irregular boundaries [13], [14], PDEs in the field of fluid mechanics [15], and high-dimensional differential equations [16]. Thus, connection between ResNets and ODE deserves deep exploration.

Success of deep learning methods highly depends on a large amount of training samples, but collecting training data requires intensive labor works, and is sometimes impossible in many application fields due to the privacy or safety issues. To alleviate the dependency of training data, semi-supervised learning (SSL) [17], [18] and few-shot learning (FSL) [19], [20] have received great interests in recent years. Semi-supervised learning typically uses a large amount of unlabeled data, together with the labeled data, to construct better classifiers. Few-shot learning is a more recent paradigm which is closely related to semi-supervised learning, and the main difference lies in that the the size of support set (labeled points) is much smaller. One common feature in SSL and FSL is to make use of the unlabeled samples to address the limited labeled set issue. See [18], [21] for the review of SSL and FSL. In this work, we focus on the deep learning based approaches for solving SSL problems. In general, the deep SSL methods can fall into two categories: consistency regularization and entropy minimization [22]. Consistency regularization demands that minor perturbation on the input does not change the output significantly. Π-Model [23], [24] and its more stable version Mean Teacher [25] are based on this idea, which require the stochastic network predictions over different passes have little disturbance.

- *T.Wang is with the Department of Mathematical Sciences, Tsinghua University, Beijing 100084, China.*
- *Z. Dou is with Department of Statistics and Data Science, Yale University. Work done during his undergraduate study at Peking University.*
- *C. Bao is with Yau Mathematical Sciences Center, Tsinghua University, Beijing 100084, China, and also with Yanqi Lake Beijing Institute of Mathematical Sciences and Applications, Beijing 101408, China. E-mail: clbao@mail.tsinghua.edu.cn*
- *Z. Shi is with Yau Mathematical Sciences Center, Tsinghua University, Beijing 100084, China, and also with Yanqi Lake Beijing Institute of Mathematical Sciences and Applications, Beijing 101408, China. E-mail: zqshi@tsinghua.edu.cn*
- *Corresponding authors: C. Bao and Z. Shi.*

VAT [26] replaces stochastic perturbation with the ”worst” perturbation which can most significantly affect the output of the prediction function. Entropy minimization, which is closely related to self-training, encourages more confident predictions on unlabeled data. EntMin [27] impose the low entropy requirements on the predictions of unlabeled examples. Pseudo label [28] feeds unlabeled samples with high prediction confidence into the network as labeled ones to train better classifier. Besides, some holistic approaches try to unify the current effective methods in SSL in a single framework, e.g. MixMatch [29], FixMatch [30]. Despite the existence of many deep SSL methods that achieve impressive results in various tasks, the internal mechanism of the consistency regularization or entropy minimization methods remains unclear in SSL/FSL classification.

To demystify this mystery in SSL and FSL, we propose an ODE inspired deep neural network that is based on the connection between ODE and ResNet. As shown in (1), current ODE counterpart of ResNet is a convection equation. Each point governing by (1) is evolved independently. This evolution process is acceptable when a large amount of training samples are available, but the performance is significantly deteriorated as the number of supervised samples decreases. Thus, it may be problematic when directly applying (1) for SSL/FSL. To solve this problem, we introduce diffusion mechanism in (1), leading to a convection-diffusion equation. After the discretization, we obtain a diffusion based residual neural network. The imposed diffusion to enforce the interactions among samples (include labled and unlabled) that is a key component in the regime of limited training data. In fact, it is worth mentioning that the convection and diffusion mechanisms always appear simultaneously in complex systems such as fluid dynamics [31], building physics [32], semiconductors [33], which strongly motivates the integration of diffusion into deep ResNets.

Imposing the interactions among samples is a classical idea and has appeared in many existing SSL approaches [34], [35], [36], but the combination of convection and diffusion in the network architecture is underexplored. In addition, most methods introduce the diffusion by adding a Laplacian regularizer in the loss function, which is widely used in graph-based SSL [37], [38]. In this case, tuning the weight of the Laplacian regularizer is not an easy task and often sensitive to tasks. Different from the above methods, we explicitly add diffusion layers into the ResNet. The proposed diffusion layers internally impose the interactions among samples and have shown to be more effective in SSL/FSL. More importantly, we theoretically analyze the diffusive ODE and show its advantage in terms of distance-diameter ratio among data samples, which provides a solid foundation the proposed method. In summary, we list our main contributions as follows.

- We propose a convection-diffusion ODE model for solving SSL/FSL, leading to the addition of diffusion layers into ResNets after proper discretization. The proposed diffusion based ResNet strengthen the relationships among labeled and unlabeled data points via a designed network architecture, rather than imposing the diffusion loss in the total. To the best of our knowledge, this is the first attempt that internally incorporates diffusion mechanism into deep neural network architecture.
- Under the structured data assumption [39], it is proved that diffusion mechanism is able to accelerate the classification process in the sense that samples from different subclasses can be driven apart, while samples from the same subclass will be brought together. Using such property, we can theoretically construct a residual network that ensures that output features are linearly separable. This analysis provides the mathematical foundation of our method.
- Extensive experiments on various tasks and datasets validate our theoretical results and the advantages of the proposed Diff-ResNet.

The rest of this paper is organized as follows. The related work is given in Section 2. Section 3 presents the formulation and details of our diffusion residual network, and Section 4 provides theoretical the analysis of diffusion mechanism. Experimental results on various tasks are reported in Section 5. We conclude the paper in Section 6.

2 Related Works

2.1 Diffusion Mechanism

The idea of diffusion is widely used in various fields. In graph neural networks, [40] concludes a unified framework for graph diffusion, and proposes a preprocessing method that create a new graph based on diffusion. With spectral analysis of the new graph, they show that local clusters can be amplified while noise can be suppressed. Diffusion-Convolutional neural networks [41] learn diffusion-based representations from graph and use them as an effective basis for node classification. Diffusion is also used in diffusion map or eigenmap [42], [43], which uses linear diffusion PDEs with closed form solutions for dimension reduction. Different from linear dimensional reduction methods like principal component analysis (PCA), diffusion maps belongs to nonlinear methods that focus on the underlying manifold of data. It constructs a Markov chain based on diffusion process, which can capture the geometric structure of manifold at larger scales as the diffusion goes on. Diffusion is used by previous work to deal with data insufficiency. [44] diffuses the label information to propose an efficient criterion for switching between exploration and refinement in active learning. Recently, diffusion has been proposed to design new network architectures. DifNet [45] constructs a diffusion process on a single image for semantic segmentation, and approximates the process by a cascade of random walks. [46] also adds a diffusion term into ODE induced by ResNet, but its diffusion is in the Euclidean space while ours is in the embedded manifold. To the best of our knowledge, this is the first work that applies the diffusion mechanism to ResNet with rigorous mathematical analysis.

2.2 Neural ODEs

The deep learning models and dynamical systems have closed relationship, which is firstly introduced in a proposal of E. et al. [2]. Using this connection, many works have been proposed for improving deep learning models. [47], [48] propose several training algorithms based on Pontragyn's

Minimum Principle condition and successive approximation method. Neural ODE [3] treats ResNet as the forward Euler discretization of an ordinary differential equation and adopt adjoint method to train the ODE model, which inspires a long list of work considering the relationship between ordinary differential equations and deep residual networks. These papers interpret ResNets as a discretization of dynamical systems, where the dynamics at each step is a linear transformation followed by a non-linear activation function. [4] treats deep networks as a parameter estimation problem of nonlinear dynamical systems, and propose new forward propagation techniques that relieve exploding or vanishing gradients problem. [49] provides a unified framework for interpreting ResNets and its derivatives, such as PolyNet [5] and FractalNet [6]. Based on the framework, the author proposes a linear multi-step architecture. However, most ODE inspired residual networks cannot be directly applied to the semi-supervised problems as they need many supervised samples.

2.3 Graph-based semi-supervised learning

Graph-based SSL algorithms have received much attention [17], [34] because graph structure can effectively encode the relationship among data points. Graph-based semi-supervised learning is based on the assumption that nearby nodes tend to have the same labels. In graph, each sample is denoted by a vertex, and the weighted edge measures the similarity between samples. [34] initially proposes the Gaussian Fields and Harmonic Functions (GFHF) algorithm, which aims to minimize the graph Laplacian objective function with the constraint on labeled points. After that, [50] introduces Local and Global Consistency (LGC) algorithm, which differs from GFHF model in that the label for each sample is penalized to ensure regularity, and the hard label constraint is turned into a soft constraint using Laplacian multiplier. Belkin et al. [51], [52] proposes the manifold regularization framework, which employs a kernel-based regularization term. Such kernels are often derived from the graph Laplacian, which becomes a general extension of graph Laplacian regularization [53], [54]. Semi-supervised embedding [37] extends the Laplacian regularizer from labels to network outputs, which imposes constraints on the parameters of a neural network. . Nonetheless, our paper embeds the Laplacian regularization intrinsically in the neural network structure through diffusion layers, which is different from methods that use iterative approach to minimize the loss function, or those that adds a regularization term based on graph Laplacian to the objective function and uses vanilla networks to optimize.

2.4 Few-shot learning

To address the limited training samples problem, few-shot learning, a new learning paradigm [19], [55], has been proposed and become an important topic in machine learning. The few-shot learning has been extensively explored in recent years, and there are many different kinds of methods. Among existing few-shot learning methods, embedding learning is a typical approach, which maps each sample to a low dimensional spaces such that similar samples are close while dissimilar samples are far away. The embedding method maps data samples to a feature space by training an embedding function from a large-scale dataset. In the feature space, another classifier is applied for classifying query data. Typical methods include Matching Network [20], which uses LSTM [56] with attention mechanism and external memory to construct the embedding mappings for the query set and the support set. It firstly introduce the episode-based training method to match the training and testing condition. Prototypical Network [57] compares the embedding of the query point with the prototype of each class and assign the query point to the same class of the nearest prototype. The work in [58] introduces Relation Network that concatenates features of training and test samples as the embedding, and feeds it to another CNN to output a similarity score. It also introduces a deep distance metric instead of hand-crafted metrics like cosine distance. TADAM [59] changes the metric for different tasks in order to use the specific information of each task. It introduces new parameters to scale the gradient and fine-tune the output of each convolutional blocks. Recently, Wang et al. [60] shows that merely pretraining an embedding function on base classes along with nearest neighbor clustering in l_2 distance can achieve competitive results. This line of work avoid complicated training strategies and get back on simple yet effective manipulations on embedded features. Following that, Laplacian regularized clustering [61] adopts a regularizer based on graph Laplacian. [62] rectifies the features to reduce the cross-class and intra-class bias, and use the rectified prototype to help clustering. Compared to the existing works, our method adopts similar embedding training procedure with different attached classifier that has good performance in various few-shot learning tasks. More importantly, under the suitable assumption, we establish a thorough theoretical analysis of the propose method.

3 Diffusion Residual Networks

In this section, we introduce the diffusion mechanism from the ODE perspective and present the Diff-ResNet based on the numerical scheme for the diffusive ODE.

3.1 The ODE formulation

In ResNet [1], the feature map of a specific data point x_i after the k-th residual block is defined as x_i^k. Residual connection means x_i^k is added to x_i^{k+1} via a skip identity link. If we gather convolutional layers, batch normalization layers and other layers together, and denote them as function f, each residual block can be written as

$$x_i^{k+1} = x_i^k + f(x_i^k, \theta^k), \tag{2}$$

where θ^k is the parameters of k-th block. From ODE perspective, f can be seen as the velocity, while x_i^k and x_i^{k+1} can be treated as the start position and end position of x_i. Introducing a time step Δt which can be absorbed in f, the ResNet can be seen as the forward Euler discretization of the following ODE model, which depicts the evolution of x_i:

$$\frac{dx_i(t)}{dt} = f(x_i(t), \theta(t)), \quad x_i(0) = x_i. \tag{3}$$

Time forms a continuous analogy to the layer index, where each layer corresponds to an iteration of the evolution. This

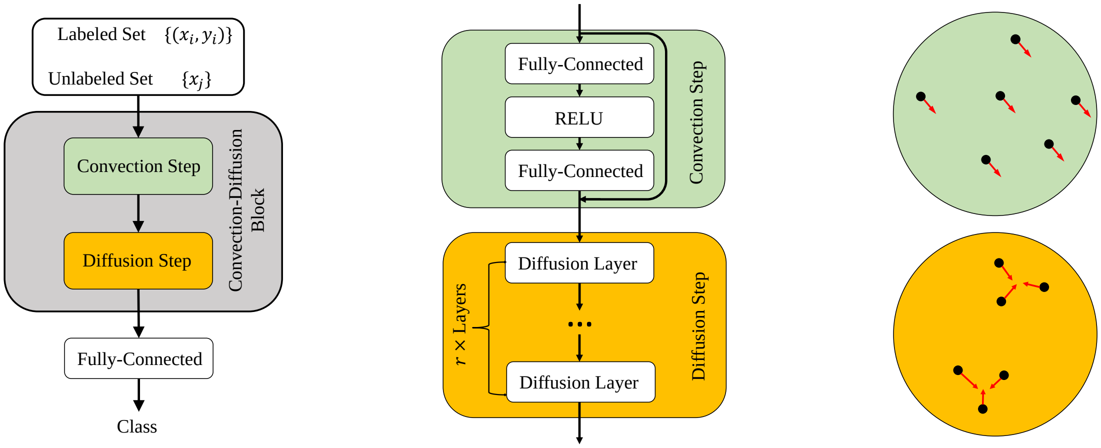


Fig. 1. Illustration of our Diff-ResNet. Left: network structure; middle: details in each convection-diffusion block; right: movements of data points with convection or diffusion.

ODE contains only the convection term, and each point moves independently without collision. To enhance the interactions among data points especially for unlabeled samples, we introduce an additional diffusion term in (3), which leads to the following convection-diffusion ODE system:

$$\frac{dx_i(t)}{dt} = f(x_i(t), \theta(t)) - \gamma \sum_{j=1}^{N} w_{ij}(x_i(t) - x_j(t)), \tag{4}$$

for all $i = 1, 2, \dots, N$, where N is the number of points, $\gamma > 0$ is a parameter that controls the strength of diffusion and $w_{ij} \geq 0$ is the weight between x_i and x_j. By designing a weight matrix that can depict the similarity between points, we can expect similar points are brought together, while dissimilar points are driven apart. In this paper, the convection term f is set to be a simple 2-layer network with width w, i.e.

$$f(x(t), \theta(t)) = \sum_{i=1}^{w} a_t^{(i)} \sigma(\mathbf{w}_t^{(i)} \cdot x(t) + b_t^{(i)}). \tag{5}$$

Here $x(t) \in \mathbb{R}^d$, $b_t^{(i)} \in \mathbb{R}$, $a_t^{(i)}$, $\mathbf{w}_t^{(i)} \in \mathbb{R}^d$ and $f : \mathbb{R}^d \to \mathbb{R}^d$. The activation function $\sigma(\cdot)$ is chosen to be ReLU. $\theta(t) = [\mathbf{w}_t^{(i)}, b_t^{(i)}, a_t^{(i)}]_{i=1}^{w}$ is the collection of network weights at time t. In the next section, we will derive a practical algorithm based on the new ODE equation (4).

3.2 Algorithm

We discretize the convection-diffusion equation (4) using the classic Lie-Trotter splitting scheme [63]. After absorbing the time step Δt into f and γ, it leads to

$$x_i^{k+1/2} = x_i^k + f(x_i^k, \theta^k), \tag{6}$$

$$x_i^{k+1} = x_i^{k+1/2} - \gamma \sum_{j=1}^{N} w_{ij}(x_i^{k+1/2} - x_j^{k+1/2}). \tag{7}$$

The convection step (6) is nearly identical to the residual block (2), only differs in the time step, which is not essential as the implementation is the same. The added diffusion step (7) can be seen as the stabilization of the convection step (6). If the weight matrix is pre-computed, the diffusion step is parameter free, thus the proposed diffusion term can be easily combined with any existing networks or algorithms in a plug-and-play manner. To construct the weight matrix, we use the Gaussian kernel $k(x, y) = \exp(-\|x - y\|_2^2/\sigma^2)$ to measure the similarity between data points. σ is a parameter to adjust the distribution of weight. Next, we introduce two operators, Sparse and $\mathrm{Normalize}$, and one hyperparameter, n_{top}, to obtain a sparse and balanced weight matrix. Sparse is a truncation operator to make the weight matrix sparse. In each row, it keeps the largest n_{top} entries and truncate other entries to 0. $\mathrm{Normalize}$ symmetrically normalize the weight matrix. Once constructed, the weight matrix remains unchanged during the training process.

Using the Lie-Trotter scheme, we get one diffusion step (7) after convection step (6). However, in our implementation, there are often several diffusion steps followed by each convection step. The reason is that the diffusion term has strong numerical stiffness as proved in Appendix A. The step size γ should be small enough to keep numerical stability when the simple explicit Euler discretization method is used. Consequently, in order to maintain certain diffusion strength, we will use simple forward Euler scheme to discretize the diffusion term. Moreover, even if the total strength is small, multiple diffusion layers also give slightly better results in experiments. Thus, in the networks, we add r diffusion layers after each residual block, each with a fixed step size γ. The illustration of our Diff-ResNet can be found in Figure 1. We summarize our method in Algorithm 1.

Remark 1. *In diffusion step* (7)*, the feature map of the* i*-th data point depends on the feature map of all data points at previous layer, which is not realistic in tasks when the total number of data points is too large. In our implementation, we adopt the mini-batch training strategy. That is, the weights in each batch are sparsified and normalized correspondingly.*

4 Analysis of Diffusion Mechanism

In this section, the effectiveness of diffusion mechanism will be analyzed in theory. For the sake of simplicity, we only consider the binary classification problem and our analysis can be naturally extended to multi-class case.

Algorithm 1 Training algorithm for Diff-ResNet

```
Input: Labeled data points $\{(x_i, y_i)\}_{i=1}^{N_1}$. Unlabeled data points $\{x_j\}_{j=1}^{N_2}$. Number of blocks $s$. Number of diffusion steps $r$. Step size $\gamma$.
Output: Trained network parameters $\{\theta^k\}$

Construct weight matrix $\tilde{W}$ by $\tilde{w}_{ij} = \exp(-\|x_i - x_j\|_2^2/\sigma^2)$ for all $i, j \in [N]$
$W = \mathrm{Normalize}(\mathrm{Sparse}(\tilde{W}, n_{\mathrm{top}}))$
while epoch $\leqslant$ MAX_ITER do
    $x_i^0 = x_i(0) = x_i$
    for $k = 0, 1, \cdots, s-1$ do
        $x_i^{k+1/2} = x_i^k + f(x_i^k, \theta^k)$                ▷ Convection Step
        for $m = 0, 1, \cdots, r-1$ do
            $x_i^{k+1/2} = x_i^{k+1/2} - \gamma \sum_{j=1}^{N_1+N_2} w_{ij}(x_i^{k+1/2} - x_j^{k+1/2})$   ▷ Diffusion Step
        $x_i^{k+1} = x_i^{k+1/2}$
    $x_i(1) = x_i^s$
    Feed $x_i(1)$ into a classification layer, compute loss function using $\{y_i\}_{i=1}^{I}$, back propagate, and update $\{\theta_k\}$ using gradient descent.
    epoch = epoch+1
```

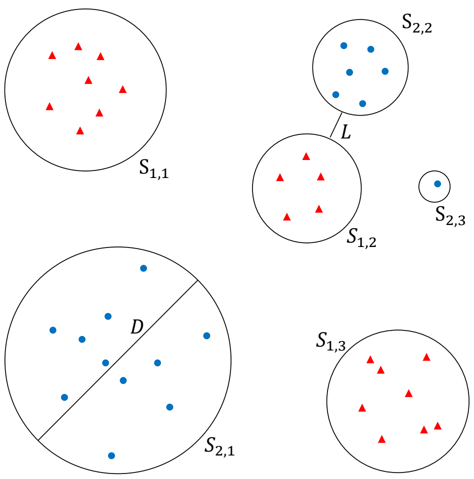


Fig. 2. Illustration of Structured Data Assumption: $S_{i,j}$ stands for different subsets. D is the upper bound of diameters of subclasses and L is the lower bound of distances among subclasses.

4.1 Structured Data Assumption

Our dataset is generated as follows. Suppose all the data points come from $S = \coprod_{i=1}^{k} S_i$. Symbol $\coprod$ means that $S = \bigcup_{i=1}^{k} S_i$ and $S_{i_1} \bigcap S_{i_2} = \varnothing$, $\forall i_1 \neq i_2$. Each set S_i contains the points from the i-th class. We further assume that each set S_i can be divided into several non-overlap and bounded subsets $S_i = \coprod_{j=1}^{l} S_{i,j}$, each $S_{i,j}$ corresponding to a subclass. In the binary classification, $k = 2$ and $S = S_1 \coprod S_2$. l is the number of subclasses, which may vary with class. However, we can set l to be the maximum across all classes, and let $S_{i,j} = \varnothing$ for those nonexistent subclasses. Denote $M = kl$ as the total number of subsets. The distance between two disjoint sets A,B is defined as

$$\mathrm{dist}(A, B) = \inf_{x \in A, y \in B} \|x - y\|^2.$$

The diameter of a set A is defined as

$$\mathrm{diam}(A) = \sup_{x,y \in A} \|x - y\|^2.$$

Remark 2. *The reason we introduce subclass instead of directly using class is that it can relieve our separability assumption. We do not need two classes to be well apart, which is not realistic in real-world scenario. Rather, we only need local subclasses to form clusters.*

We are now ready to state the structured data assumption. (A) (Upper Bound of Diameters) There exists $D > 0$ such that for each $(i, j) \in [k] \times [l]$, we have: $S_{i,j} \in B(x_0, D/2)$[1] for some x_0 , then:

$$\mathrm{diam}(S_{i,j}) \leqslant D.$$

(B) (Lower Bound of Distances) There exists $L > 0$ such that for each $(i_1, j_1) \neq (i_2, j_2) \in [k] \times [l]$, we have:

$$\mathrm{dist}(S_{i_1,j_1}, S_{i_2,j_2}) \geqslant L.$$

Here, L and D are similar to inter-class distance and intra-class distance, which are terminologies widely used in the field of clustering. The difference lies in that the diameter used in our analysis is the upper bound for the points in the subclass, corresponding the local intra-class distance. In this sense, the Structured Data Assumption may be more practical for dealing with complex datasets. For example, in MNIST dataset, every digit number may have different handwriting styles, which corresponds to different subclasses and fits to our analysis framework. The intuition behind the Structured Data Assumption is simple: similar samples should be close while dissimilar samples should be far away.

4.2 Theoretical Analysis

We present the theoretical results of this paper so as to explain the role of diffusion mechanism in binary classification. Due to the space limit, we defer all the proofs to the Appendix.

Definition. *A set $\{(x_i, y_i)\}_{i=1}^{N}$, $x_i \in R^d$, $y_i \in [k]$ is called* linear separable *if and only if there exists a hyperplane that cuts the full space R^d into 2 half-spaces and data points in each half-space have a common corresponding label.*

Theorem 1. *(Approximation Property of ResNet Flow) If all the $S_{i,j}$, $(i,j) \in [k] \times [l]$ can be separated by a set of $M-1$ parallel hyperplanes, i.e., there exists a unique $S_{i,j}$ that lies in the region between each pair of adjacent parallel hyperplanes. Then we can construct the time-dependent parameters $\theta(t) = [\mathbf{w}_t^{(i)}, b_t^{(i)}, a_t^{(i)}]_{i=1}^{w}$ in the ResNet flow:*

$$f(x(t), \theta(t)) = \sum_{i=1}^{w} a_t^{(i)} \sigma(\mathbf{w}_t^{(i)} \cdot x(t) + b_t^{(i)})$$

such that all the final step[2] *regions $F_i = \{x(1) : x(0) \in S_{i,j}, j \in [l]\}, i \in [k]$ are linear separable. We need $2M + O(d)$ different variables and M/w layers.*

We give a sketch of proof. Consider the simplest case in which each $S_{i,j}$ only contains one point and the width w

[1] $B(x_0, D/2)$ is defined as a ball centered at x_0 with radius $D/2$

[2] Time only forms a continuous analogy to the layer index, where each layer corresponds to forward propagation of the flow. Without loss of generality, we assume the final time step is $T = 1$.

is also 1. Our main idea is to construct a ResNet flow such that each subclass is moved to a proper position with better separability. We split the total time into N intervals and deal with points one by one. After solving the simplest case, we extend to case $w > 1$, i.e., the network width is larger. Lastly, we prove the case when there are multiple points in each subclass $S_{i,j}$.

In the classical XOR dataset, the original data points $x_i(0) = x_i$ are not linear separable. However, Theorem 1 tells us that: through the ODE flow, we can make the output features $x_i(1)$ become linear separable, so that a proper fully-connected layer can achieve accurate classification.

Our next step is to show that the condition in Theorem 1 can be satisfied by introducing diffusion mechanism. First, we give a sufficient condition that is related to the Distance-Diameter ratio.

Theorem 2. *If the Distance-Diameter Ratio is large enough:*

$$\frac{L}{D} \geqslant \frac{M(M-1)\sqrt{\pi}}{4} d,$$

then all the $S_{i,j},\ (i,j) \in [k] \times [l]$ can be separated by a set of $M-1$ parallel hyperplanes.

This proof relies on comparing the surface area of a specific set with the unit sphere. It is noted that M is the number of subclasses that has $M \ll N$ in most cases. Thus the constant in the inequality is achievable. The next proposition shows that the diffusion step can increase this ratio with exponential rate.

The diffusion of each data point x_i is modeled as

$$\frac{dx_i(t)}{dt} = -\gamma \sum_{j=1}^{N} w_{ij}(x_i(t) - x_j(t)), x_i(0) = x_i$$

for $i \in [N]$. Thus, all points change their positions subject to mutual interactions, and the distances between subsets and diameters of subsets are changed accordingly. Let $S_{i,j}(t)$ be the subset at time t, we define lower bound of distances $L(t)$ and upper bound of diameters $D(t)$ at time t as

$$\operatorname{diam}(S_{i,j}(t)) \leqslant D(t),\ \forall (i,j) \in [k] \times [l],$$

$$\operatorname{dist}(S_{i_1,j_1}(t), S_{i_2,j_2}(t)) \geqslant L(t),\ \forall (i_1,j_1) \neq (i_2,j_2) \in [k]\times[l].$$

Let $G = (V, E)$ be a graph, where V is the set of data points, and E is the set of edges corresponding to non-zero weights w_{ij}. Then we have the following proposition describing the diffusion effects.

Proposition 1. *Suppose the data points in each subset $S_{i,j},\ (i,j) \in [k] \times [l]$ form a connected component in the graph G, and each $S_{i,j}$ is convex. Then, the Distance-Diameter Ratio grows to infinity, i.e.*

$$\lim_{t\to\infty} \frac{L(t)}{D(t)} = \infty.$$

Moreover, the growth rate is exponential.

The basic idea for proving Proposition 1 is to show that $L(t)$ is nonincreasing while $D(t)$ converges to zero at exponential rate. Using the spectral clustering theory, it is proved that each subclass converges to its center along with the diffusion process.

To meet the assumption that points in each subset form a connected component in the graph, we should ensure (1) there is no edge that connect points among different subsets (2) any two vertices in the same subset $S_{i,j}$ are connected to each other. By the construction of weight matrix, each vertex in graph G is only connected to its n_{top} nearest neighbors. Thus the first argument is satisfied when the nearest neighbors only contain points from the same subclass, which requires that n_{top} should not be too large. On the other hand, the threshold for the connectivity of a k-nearest neighbor graph is $O(\log n)$ [64], where n in our setting should be the number of points in each subset.

The above analysis reveals that the diffusion mechanism is helpful for organizing data points by making data points from the same subclass region closer to each other while others relatively further away. As the Distance-Diameter ratio increases, it is easy for distinguishing data points using ResNet flow. This property is important for SSL/FSL problems as it deeply explores the relationship among points.

5 Experiments

In this section, we show the efficacy of diffusion mechanism on synthetic data, and report the performance of the Diff-ResNet on semi-supervised graph learning and few-shot learning tasks. [3]

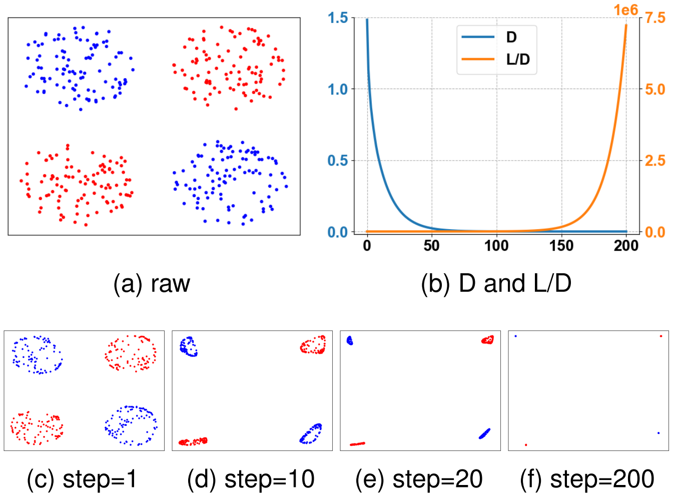
5.1 Synthetic Data

We conduct experiments on four classical synthetic datasets: XOR, moon, circle and spiral. In XOR dataset, we directly apply diffusion without any convection. Then we can clearly see the evolution process of points that verifies the Proposition 1. The other three datasets are used to show the effectiveness of diffusion in classification tasks. In this section, we only show results of XOR and circle datasets. Due to the space limitation, please refer to Appendix E.1.4 for more results.

We randomly collect 100 points each in four circles centered at (0,0), (0,2), (2,0), (2,2), respectively, with radius 0.75. These four circles are treated as the subsets corresponding to four subclasses. The circles centered at (0,0) and (2,2) belong to the same class, and points from them are colored red. The blue ones are generated similarly. Here, we show the evolution of points as diffusion strength goes to infinity. As stated in the Section 3.2, we stack diffusion steps with small step size γ to ensure stability. In Figure 3 , points distribution after 1, 10, 20 and 200 diffusion steps are given. In the above example, the initial diameter is $D = 1.5$ while the distance is $L = 0.5$, which does not meet the sufficient condition $L > D$ in Proposition 1. However, as shown in Figure 3, this diffusion still works well. Data points in same subclass converge to a single point. We also observe from Figure 3 (b) that the Distance-Diameter Ratio indeed grows exponentially to infinity.

Remark 3. *Some may doubt the use of terminology, diffusion, as it actually draws similar points together and create high density regions visually. However, the phenomenon shown in 3 is not contradictory to the definition of diffusion. The energy of a point is represented by its coordinate. We expect that neighboring elements*

[3] Code at https://github.com/shwangtangjun/Diff-ResNet.



(a) raw

(b) D and L/D

(c) step=1

(d) step=10

(e) step=20

(f) step=200

Fig. 3. Visualization of diffusion Mechanism on XOR dataset. (a) is raw data. (b) shows the evolution of D and L/D with diffusion steps. (c)(d)(e)(f) depicts the evolution of points.

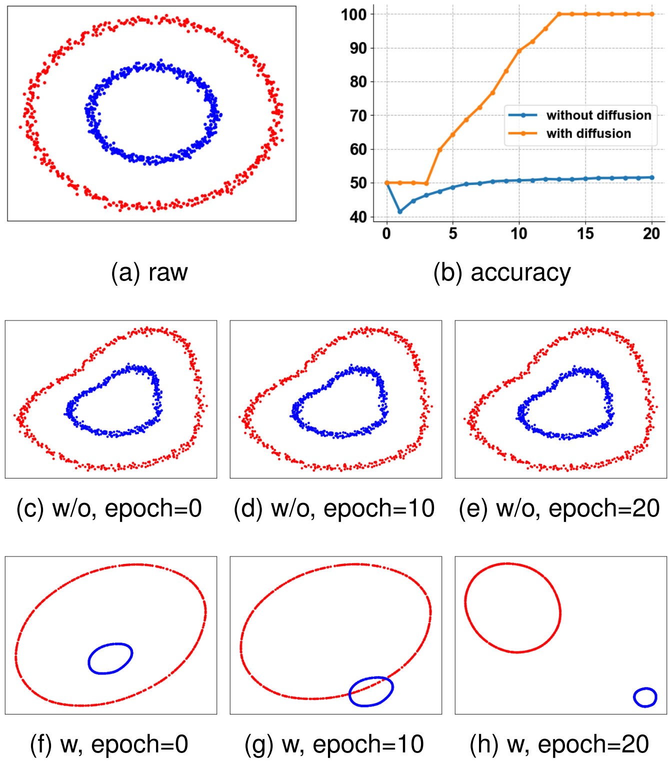
in the graph will exchange energy until that energy is spread out evenly throughout all of the elements that are connected to each other. As a result, the diffusion mechanism acts as gathering points together.

Next, we show the effectiveness of diffusion in residual networks on binary classification tasks containing 1000 planar data points forming two circles. Two classes are marked with different colors. We use residual networks with hidden dimensions 2 (so that it will be convenient for us to visualize the features). The details of experiment settings can be found in Appendix E.1. During training residual networks with or without diffusion mechanism, we plot the features before the final classification layer in Figure 4. Note that what we plot are not the input data points. Thus, even without diffusion, the points have to pass through a randomly initialized residual block. So in subfigure (c) of Figure 4, features are different from raw input points in (a). The results of circle dataset is shown in Figure 4.

As shown in Figure 4, diffusion can reduce the noise. In Figure 4 (f),(g),(h), the features are very clean while in Figure 4 (f),(g),(h), features are still noisy. Moreover, diffusion step makes the final feature much easier to separate. In Figure 4 (h), features can be easily separated by a straight line while the features are not linear separable without diffusion as shown in Figure 4 (e). It is not surprise that in this example ResNet fails to give correct classification considering it only has 18 parameters in total. With the help of diffusion step, even this small network with only 18 parameters can give correct classification which demonstrates that diffusion is very useful in classification problem.

5.2 Graph Learning

We investigate the effect of diffusion on semi-supervised learning problems in graph. In diffusion step, a key point is how to determine the weights that can properly depict the relationship between data points. Nonetheless, in graph this is not a problem since the weights have already been given in the form of adjacent matrix. We report results for the most widely used citation network benchmarks including Cora, Citeseer and Pubmed. These datasets are citation networks in which nodes are documents, edges are citation links and features are sparse bag-of-words vectors. The concrete dataset statistics is given in Appendix E.2.1. Moreover, rather than using the fixed Planetoid [38] split, we follow [65] and report results for all datasets using 100 random splits with 20 random initializations each.



(a) raw

(b) accuracy

(c) w/o, epoch=0

(d) w/o, epoch=10

(e) w/o, epoch=20

(f) w, epoch=0

(g) w, epoch=10

(h) w, epoch=20

Fig. 4. ResNet and DiffResNet on circle dataset, (a) shows the position of raw data points. (b) figures the accuracy of classification tasks with the training epoch. (c)(d)(e) are features before the final classification layer without diffusion from different epochs. (f)(g)(h) are features with diffusion.

The mainstream approach in graph learning, such as GCN [66], GraphSAGE [67] and GAT [68], contains aggregation steps, which aggregate feature information from neighbors using the adjacent matrix, and then predict labels with aggregated information. Different from these conventional paradigm, our method is composed of convection-and-then-diffusion step. The convection step make full use of the label information, while the diffusion step exchange the feature information among data samples. The adjacent matrix is only used in the diffusion step.

We compare our method with three graph learning methods: GCN, GraphSAGE (its two variants) and GAT. The detailed network structure and parameter settings can be found in Appendix E.2. The classification results are reported in Table 1. Diff-ResNet is significantly better than ResNet without diffusion. It achieves more than 15% accuracy boost on average, which is a strong evidence for the benefit of diffusion. Moreover, despite the large discrepancy between our diffusion network and mainstream networks, our method still achieves competitive results with respect to classical methods in graph learning. Thus, we propose an alternative path for semi-supervised graph learning problems.

TABLE 1
The mean accuracy and std (%) of node classification over 100 random dataset splits and 20 random initializations each.

	Cora	Citeseer	Pubmed
GCN [66]	81.5 ± 1.3	71.9 ± 1.9	77.8 ± 2.9
GraphSAGE-mean [67]	79.2 ± 7.7	71.6 ± 1.9	77.4 ± 2.2
GraphSAGE-maxpool [67]	76.6 ± 1.9	67.5 ± 2.3	76.1 ± 2.3
GAT [68]	81.8 ± 1.3	71.4 ± 1.9	78.7 ± 2.3
No-Diff-ResNet	58.9 ± 1.9	61.9 ± 2.1	70.1 ± 2.1
Diff-ResNet(ours)	**82.1** ± 1.1	**74.6** ± 1.8	**80.1** ± 2.0

Additionally, it is reported that methods based on aggregation of neighboring information suffers over-smoothing problems with increasing depth [69], [70]. As is observed in Figure 5, the performance of GCN drops more than 50% on average when the network depth increases to 32. Different from GCN, our Diff-ResNet does not use aggregation, thus the representations of the nodes will not converge to a certain value and become indistinguishable. When the number of layers increases to 32, performance of Diff-ResNet only drops less than 10%, which is partly due to the deep network training burden. This serves as an evidence that our network structure is far different from mainstream architectures.

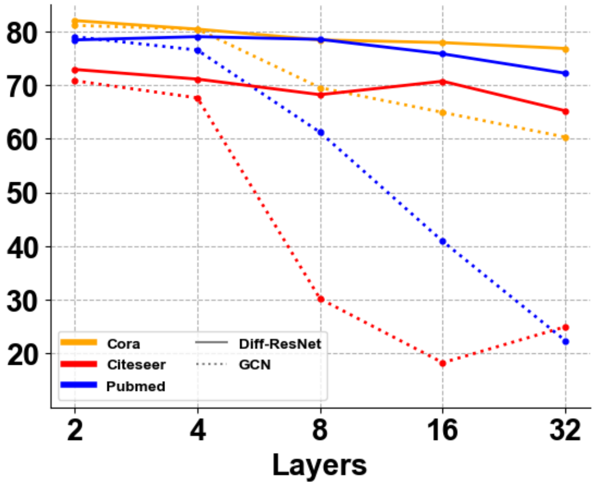


Fig. 5. Performance of architectures of different depth. The x-axis represents number of layers, y-axis is the accuracy.

5.3 Few-shot Learning

Given a dataset $\mathbb{X} = \mathbb{X}_s \cup \mathbb{X}_q$, where $\mathbb{X}_s = \{(x_i, y_i)\}_{i=1}^{N_1}$ is the support set with label information and $\mathbb{X}_q = \{x_j\}_{j=1}^{N_2}$ is the query set without labels, the goal of few-shot learning is to find the labels of points in the query set when the size of support set $|N_1|$ is very small. Among existing few-shot learning methods, embedding learning is a typical approach, which maps each sample to a low dimensional spaces such that similar samples are close while dissimilar samples are far away. The embedding function can be learned by a deep neural network (a.k.a. backbone), which is pretrained using a large number of labeled examples over base classes. In the few-shot learning problems, the pretrained embedding function is fixed and maps all data samples into the embedded space.

We conduct experiments on three benchmarks for few-shot image classification: *mini*ImageNet, *tiered*ImageNet and CUB. The *mini*ImageNet and *tiered*ImageNet are both subsets of the larger ILSVRC-12 dataset [71], with 100 classes and 608 classes respectively. CUB-200-2011 [72] is another fine-grained image classification dataset with 200 classes. We follow the standard dataset split as in previous papers [60], [73], [74]. All images are resized to 84 × 84, following [20].

We choose two widely used networks, ResNet-18 [1] and WRN-28-10 [75] as our backbone: the latter widens the residual blocks by adding more convolutional layers (28 layers) and feature planes (10 times). First, we train the backbone on the base classes using cross-entropy loss, SGD optimizer, standard data augmentation and a mini-batch size of 256 to train all models. Note that our training procedure does not involve any meta-learning or episodic-training strategy. The model is trained for $T = 100$ epochs for *mini*ImageNet and *tiered*ImageNet, and $T = 200$ epochs for CUB. We use a multi-step scheduler, which decays the learning rate by 0.1 at $0.5T$ and $0.75T$. We evaluate the nearest-prototype classification accuracy on the validation set and obtain the best model. The embedding training process is in general similar to that in SimpleShot [60] and LaplacianShot [61], but details are slightly different. Eventually we get an embedding function which maps the original data point to $\mathbb{R}^M$ where $M = 512$ for ResNet-18 and $M = 640$ for WRN-28-10.

After we obtain a feature vector for every data point, we compare the performance of 5 typical classification methods to emphasize the effectiveness of diffusion mechanism.

(1) Nearest Prototype. The prototype m_c of each class c is the average of support set $\mathbb{X}_s^c$

$$m_c = \frac{1}{|\mathbb{X}_s^c|} \sum_{x \in \mathbb{X}_s^c} x$$

Then the query sample is classified as class c if it is closest to prototype m_c in Euclidean distance. It is the most natural classification method, and serves as a baseline.

(2) Diffusion. We try to minimize an objective function with Laplacian regularizer in an iterative way. In LaplacianShot [61], the author optimizes the loss function below

$$\mathcal{L} = \sum_{i=1}^{N_2} \sum_{c=1}^{C} y_{i,c} d(x_i - m_c) + \frac{\lambda}{2} \sum_{i,j=1}^{N_2} w\left(\mathbf{x}_i, \mathbf{x}_j\right) \|\mathbf{y}_i - \mathbf{y}_j\|^2$$

$N_2 = |\mathbb{X}_q|$ is the number of query samples. $\mathbf{y}_i = [y_{i,1}, \cdots, y_{i,C}] \in \{0,1\}^C$ is in the C-dimensional simplex, which assigns label to each query point. d is Euclidean distance. $w\left(\mathbf{x}_i, \mathbf{x}_j\right)$ is the weight between $\mathbf{x}_i$ and $\mathbf{x}_j$.

The first loss term functions similar to nearest prototype classification. The second loss term is the well-known Laplacian regularizer. We use the iterative algorithm provided by LaplacianShot [61] to minimize the objective function $\min_{\mathbf{y}_i} \mathcal{L}$. Since there is no neural network, or manipulations on features, rather just label propagation, we name this method as **Diffusion**.

(3) Convection. We minimize cross entropy loss

$$\mathcal{L} = -\sum_{i=1}^{N_1} \sum_{c=1}^{C} y_{i,c} \log(f(x_i)_c)$$

TABLE 2
Ablation Study of Diffusion Mechanism. Table shows average classification accuracy (%).

		*mini*ImageNet		*tiered*ImageNet		CUB	
Backbone	**Method**	**1-shot**	**5-shot**	**1-shot**	**5-shot**	**1-shot**	**5-shot**
ResNet-18	Nearest Prototype	57.09	79.30	63.46	83.53	66.53	86.03
	Diffusion	57.81	79.54	64.38	83.83	67.96	86.49
	Convection	53.04	79.58	56.17	82.79	58.68	86.10
	External Convection-Diffusion	55.30	79.55	65.15	83.74	70.00	86.92
	Internal Convection-Diffusion	**68.47**	**80.02**	**75.31**	**84.19**	**79.12**	**87.18**
WRN	Nearest Prototype	59.54	79.81	65.60	84.76	68.56	86.13
	Diffusion	60.30	80.31	66.59	85.17	69.70	86.58
	Convection	57.07	80.92	58.79	84.69	62.84	87.51
	External Convection-Diffusion	59.46	80.70	68.10	85.31	73.08	87.08
	Internal Convection-Diffusion	**69.77**	**81.17**	**77.44**	**85.50**	**80.31**	**87.76**

on the support set using gradient descent and a simple 2-layer residual network f. $N_1 = |\mathbb{X}_s|$ is the number of support samples. The detailed network structure can be found in Appendix E.3.2. There is no relationship between data points during training, and the residual network is the counterpart of a convection ODE, so we refer to this method as **Convection**.

(4) External Convection-Diffusion. We minimize cross entropy loss on the support set plus Laplacian regularizer using gradient descent and simple 2-layer residual network. The difference from **(3)** is that we add a Laplacian regularizer

$$\mathcal{L} = -\sum_{i=1}^{N_1}\sum_{c=1}^{C} y_{i,c}\log(f(x_i)_c) \\ +\frac{\mu}{2}\sum_{i,j=1}^{N} w\left(\mathbf{x}_i,\mathbf{x}_j\right)\left\|f(x_i)-f(x_j)\right\|^2$$

to the loss function. $N = N_1 + N_2 = |\mathbb{X}_s \cup \mathbb{X}_q|$ is the total number of support samples and query samples. The first variation of the Laplacian term coincides with the diffusion term of our convection-diffusion ODE. The residual network structure corresponds to convection, while the Laplacian regularizer corresponds to diffusion. As diffusion appears in the loss function externally, we refer to this method as **External Convection-Diffusion**.

(5) Internal Convection-Diffusion. We minimize cross entropy loss

$$\mathcal{L} = -\sum_{i=1}^{N_1}\sum_{c=1}^{C} y_{i,c}\log(f(x_i)_c)$$

on the support set using gradient descent and simple 2-layer **diffusion** residual network (Diff-ResNet). The loss term is the same as **(3)**, while the difference is that we add diffusion layers internally in the network structure. By comparing **(4)** and **(5)**, we want to verify the necessity of incorporating diffusion as part of network structure, rather than as part of loss function.

Following the standard evaluation protocol [60], we randomly sample 1000 5-way-1-shot and 5-way-5-shot classification tasks from the test classes, with 15 query samples in each class, and report the average accuracy of 5 methods above in Table 5.3. **Internal Convection-Diffusion**, which uses our proposed Diff-ResNet, achieves best results in all tasks. In 1-shot tasks, it has a performance boost of nearly 20% compared to **Convection**, which clearly states the effectiveness of diffusion. Additionally, compared to **External Convection-Diffusion**, it also has approximately 10% increase, indicating that embedding diffusion in the network structure is far more efficient than adding diffusion in the loss term. In 5-shot tasks, as the nearest prototype method has already provided competitive baseline, the increase is not remarkable, but still has about 1% improvement against **Convection**.

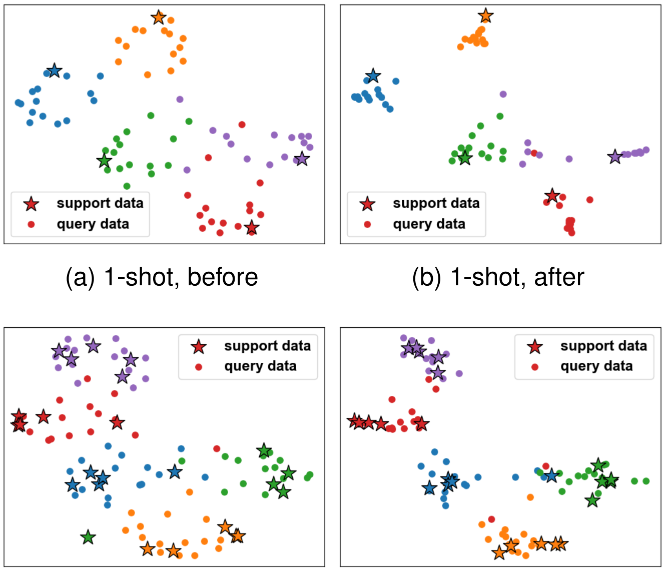
We use T-SNE [76] to visualize the features before and after diffusion in 1-shot and 5-shot task in Figure 6. Labeled data are marked as stars, and unlabeled data are marked as circles. In 1-shot scenario, we can observe that points are hard to separate at first. However, with the help of diffusion mechanism, points in the same subclass are driven closer, making it easier to classify. In 5-shot tasks, since the sample points already have nice separability, the improvement with diffusion mechanism is not so remarked as that in 1-shot tasks. Nonetheless, we could verify the necessity of introducing subclass in Structured Data Assumption, as the blue points are indeed divided into two subsets.

Furthermore, we study the effect of several important parameters in diffusion mechanism: weight truncation parameter n_{top}, diffusion step number r and step size γ. We conduct experiments on 1000 5-way-1-shot tasks on *mini*ImageNet with ResNet-18 and WRN as backbone, and report the average accuracy with different parameters.

First, we tune n_{top} in the Sparse operator. We choose $\sigma = [n_{\text{top}}/2]^4$. The results are depicted in Figure 7. From the figure, we notice that n_{top} should be neither too small nor too large. Small n_{top} may break up large local clusters, while large n_{top} will include points from different classes into neighborhood. However, the classification accuracy is not very sensitive to n_{top}, compared to diffusion strength.

Next, we study the effect of total diffusion strength, which is step size times step numbers $r\gamma$. We fix $\gamma = 0.5$ and adjust

[4] $\sigma(x_i) = k$ means σ is chosen to be the k-th closest distance from a specific point x_i , so it varies with points.



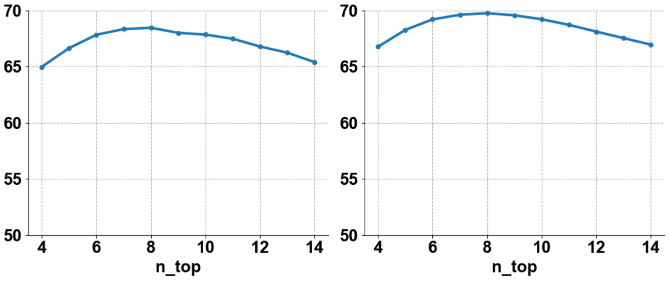
(a) 1-shot, before

(b) 1-shot, after

(c) 5-shot, before

(d) 5-shot, after

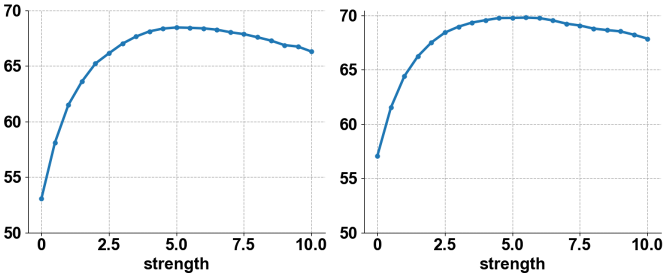
Fig. 6. T-SNE visualization of features before and after diffusion steps. Stars represent labeled points(support data). Circles represent unlabeled points(query data).



(a) ResNet-18

(b) WRN

Fig. 7. The effect of $n_{\rm top}$ on *mini*ImageNet with ResNet-18 and WRN as backbone. The x-axis represents $n_{\rm top}$, y-axis is the accuracy.

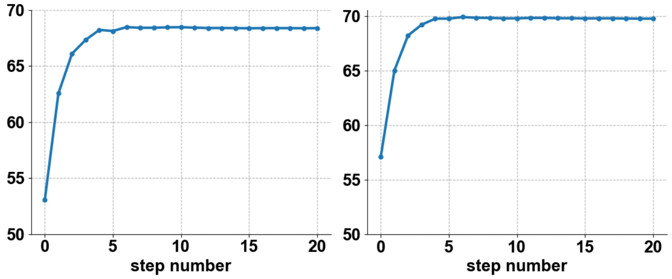


(a) ResNet-18

(b) WRN

Fig. 8. The effect of total diffusion strength on *mini*ImageNet with ResNet-18 and WRN as backbone. The x-axis is total strength $r\gamma$, y-axis is the accuracy.

r, ranging from 0 to 20. The results are shown in Figure 8. Based on our experiments, we should not push the strength to infinity as in synthetic data, because real data has much more complicated geometric structure such that we can not expect each class converge to a single point.



(a) ResNet-18

(b) WRN

Fig. 9. The effect of number of steps r on *mini*ImageNet with ResNet-18 and WRN as backbone. The x-axis represents step number r, y-axis is the accuracy.

Lastly, we fix total strength $r\gamma = 5.0$, and change r from 0 to 20, to study the effect of the step size. We require $\gamma \leqslant 1$ for stability. When the total diffusion strength is too large when $r < 5$, we set $\gamma = 1.0$. As shown in Figure 9, when the total diffusion strength is fixed, the accuracy almost keeps the same. Thus, stacking too many layers will not benefit.

At the end of the section, we want to emphasize that our Diff-ResNet can achieve state-of-the-art. For fair comparison, we adopt tricks used in LaplacianShot [61], which we elaborate in Appendix E.3. We randomly sample 10000 5-way-1-shot and 5-way-5-shot classification tasks and report the average accuracy and corresponding 95% confidence interval in Table 3. The results of networks for comparison in Table 3 are collected from [60], [61], [74]. In all datasets with various backbones, Diff-ResNets obtains the highest classification accuracy.

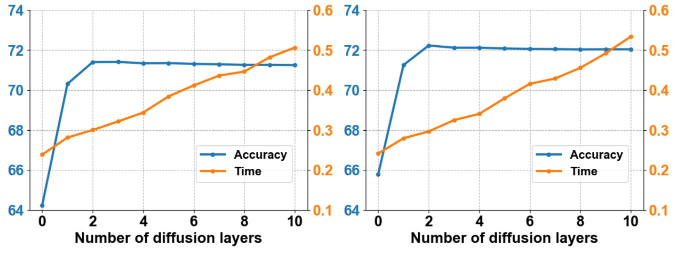
Additionally, we investigate the computational cost of diffusion mechanism. The implementation of diffusion layers is simply small-scale matrix multiplication, which is very efficient using GPU. We run 1000 classification tasks using Diff-ResNet with different number of diffusion layers r, and report the average computation time per task and accuracy in Figure 10. The total diffusion strength is fixed as $r\gamma = 2.0$, except when $r = 1$ we choose $\gamma = 1.0$ for stability. We use a single GeForce RTX 2080 Ti to collect the running time. As stated before in Figure 9, with fixed diffusion strength, there is no need of stacking too many layers. In both subfigures, 2 diffusion layers can already achieve the best result, whereas the increase in time compared to no diffusion is approximately 25%. Moreover, the time of 10 diffusion layers roughly doubles that of without diffusion, and 10 layers is enough to achieve desired diffusion strength in all few-shot tasks. Thus, the computational cost is acceptable.

6 Conclusion

In this paper, inspired by the ODE model with diffusion mechanism, we propose a novel Diff-ResNets by adding a simple yet powerful diffusion layer to the residual blocks. We conduct theoretical analysis of the diffusion mechanism

TABLE 3
Average accuracy (in %) and 95% confidence interval in *mini*ImageNet, *tiered*ImageNet and CUB. We mark transductive learning method with †.

		*mini*ImageNet		*tiered*ImageNet		CUB	
Methods	**Backbone**	**1-shot**	**5-shot**	**1-shot**	**5-shot**	**1-shot**	**5-shot**
MAML [77]	ResNet-18	49.61 ± 0.92	65.72 ± 0.77	-	-	69.96 ± 1.01	82.70 ± 0.65
Baseline [74]	ResNet-18	51.87 ± 0.77	75.68 ± 0.63	-	-	67.02 ± 0.90	83.58 ± 0.54
RelationNet [58]	ResNet-18	52.48 ± 0.86	69.83 ± 0.68	54.48 ± 0.93	71.32 ± 0.78	67.59 ± 1.02	82.75 ± 0.58
MatchingNet [20]	ResNet-18	52.91 ± 0.88	68.88 ± 0.69	-	-	72.36 ± 0.90	83.64 ± 0.60
ProtoNet [57]	ResNet-18	54.16 ± 0.82	73.68 ± 0.65	53.31 ± 0.89	72.69 ± 0.74	71.88 ± 0.91	87.42 ± 0.48
Gidaris [78]	ResNet-15	55.45 ± 0.89	70.13 ± 0.68	-	-	-	-
SNAIL [79]	ResNet-15	55.71 ± 0.99	68.88 ± 0.92	-	-	-	-
TADAM [59]	ResNet-15	58.50 ± 0.30	76.70 ± 0.30	-	-	-	-
Transductive [80]†	ResNet-12	62.35 ± 0.66	74.53 ± 0.54	-	-	-	-
MetaoptNet [81]	ResNet-18	62.64 ± 0.61	78.63 ± 0.46	65.99 ± 0.72	81.56 ± 0.53	-	-
TPN [82]†	ResNet-12	53.75 ± 0.86	69.43 ± 0.67	57.53 ± 0.96	72.85 ± 0.74	-	-
TEAM [83]†	ResNet-18	60.07 ± 0.59	75.90 ± 0.38	-	-	80.16 ± 0.52	87.17 ± 0.39
CAN+T [84]†	ResNet-12	67.19 ± 0.55	80.64 ± 0.35	73.21 ± 0.58	84.93 ± 0.38	-	-
SimpleShot [60]†	ResNet-18	63.32 ± 0.20	80.19 ± 0.14	69.64 ± 0.22	85.00 ± 0.16	71.63 ± 0.20	87.07 ± 0.12
LaplacianShot [61]†	ResNet-18	70.66 ± 0.23	82.27 ± 0.14	77.55 ± 0.24	86.12 ± 0.16	80.08 ± 0.22	88.38 ± 0.12
Diff-ResNet(ours)†	ResNet-18	**71.54** ± 0.24	**82.80** ± 0.14	**78.57** ± 0.24	**86.77** ± 0.16	**80.92** ± 0.22	**89.01** ± 0.12
Qiao [85]	WRN	59.60 ± 0.41	73.74 ± 0.19	-	-	-	-
LEO [86]	WRN	61.76 ± 0.08	77.59 ± 0.12	66.33 ± 0.05	81.44 ± 0.09	-	-
ProtoNet [57]	WRN	62.60 ± 0.20	79.97 ± 0.14	-	-	-	-
CC+rot [87]	WRN	62.93 ± 0.45	79.87 ± 0.33	70.53 ± 0.51	84.98 ± 0.36	-	-
MatchingNet [20]	WRN	64.03 ± 0.20	76.32 ± 0.16	-	-	-	-
FEAT [88]	WRN	65.10 ± 0.20	81.11 ± 0.14	70.41 ± 0.23	84.38 ± 0.16	-	-
Transductive [80]†	WRN	65.73 ± 0.68	78.40 ± 0.52	73.34 ± 0.71	85.50 ± 0.50	-	-
BD-CSPN [62]†	WRN	70.31 ± 0.93	81.89 ± 0.60	78.74 ± 0.95	86.92 ± 0.63	-	-
SimpleShot [60]†	WRN	64.58 ± 0.20	81.03 ± 0.14	70.91 ± 0.22	85.93 ± 0.15	73.24 ± 0.21	87.61 ± 0.12
LaplacianShot [61]†	WRN	71.27 ± 0.23	82.42 ± 0.14	78.65 ± 0.24	86.92 ± 0.16	81.07 ± 0.22	88.45 ± 0.13
Diff-ResNet(ours)†	WRN	**72.25** ± 0.24	**83.12** ± 0.14	**79.70** ± 0.24	**87.55** ± 0.16	**81.43** ± 0.22	**89.01** ± 0.12



(a) ResNet-18

(b) WRN

Fig. 10. The computation time of each task and accuracy on *mini*ImageNet with ResNet-18 and WRN as backbone. The x-axis represents number of diffusion layers, 0 means no diffusion, right y-axis is the computation time (seconds) of each task, left y-axis is the average accuracy.

and prove that the diffusion term will significantly increase the ratio between local intra-class distance and inter-class distance. The performance of proposed Diff-ResNets is verified by extensive experiments on few-shot learning and semi-supervised graph learning problems.

REFERENCES

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

[2] E. Weinan, "A proposal on machine learning via dynamical systems," *Communications in Mathematics & Statistics*, vol. 5, no. 1, pp. 1–11, 2017.

[3] T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," in *Advances in neural information processing systems*, 2018, pp. 6571–6583.

[4] E. Haber and L. Ruthotto, "Stable architectures for deep neural networks," *Inverse Problems*, vol. 34, no. 1, p. 014004, 2018.

[5] X. Zhang, Z. Li, C. Change Loy, and D. Lin, "Polynet: A pursuit of structural diversity in very deep networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 718–726.

[6] G. Larsson, M. Maire, and G. Shakhnarovich, "Fractalnet: Ultra-deep neural networks without residuals," *arXiv preprint arXiv:1605.07648*, 2016.

[7] Y. Lu, A. Zhong, Q. Li, and B. Dong, "Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations," in *International Conference on Machine Learning*. PMLR, 2018, pp. 3276–3285.

[8] X. Gastaldi, "Shake-shake regularization," *arXiv preprint arXiv:1705.07485*, 2017.

[9] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *European conference on computer vision*. Springer, 2016, pp. 646–661.

[10] Z. Yang, Y. Liu, C. Bao, and Z. Shi, "Interpolation between residual and non-residual networks," in *International Conference on Machine Learning*. PMLR, 2020, pp. 10 736–10 745.

[11] I. E. Lagaris, A. Likas, and D. I. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," *IEEE transactions on neural networks*, vol. 9, no. 5, pp. 987–1000, 1998.

[12] M. Dissanayake and N. Phan-Thien, "Neural-network-based approximations for solving partial differential equations," *communications in Numerical Methods in Engineering*, vol. 10, no. 3, pp. 195–201, 1994.

[13] I. E. Lagaris, A. C. Likas, and D. G. Papageorgiou, "Neural-network methods for boundary value problems with irregular boundaries,"

IEEE Transactions on Neural Networks, vol. 11, no. 5, pp. 1041–1049, 2000.

[14] K. S. McFall and J. R. Mahan, "Artificial neural network method for solution of boundary value problems with exact satisfaction of arbitrary boundary conditions," *IEEE Transactions on Neural Networks*, vol. 20, no. 8, pp. 1221–1233, 2009.

[15] M. Baymani, A. Kerayechian, and S. Effati, "Artificial neural networks approach for solving stokes problem," *Applied Mathematics*, vol. 1, no. 4, p. 288, 2010.

[16] J. Han, A. Jentzen, and E. Weinan, "Solving high-dimensional partial differential equations using deep learning," *Proceedings of the National Academy of Sciences*, vol. 115, no. 34, pp. 8505–8510, 2018.

[17] 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.

[18] O. Chapelle, B. Scholkopf, and A. Zien, "Semi-supervised learning," *IEEE Transactions on Neural Networks*, vol. 20, no. 3, pp. 542–542, 2009.

[19] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 4, pp. 594–611, 2006.

[20] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra *et al.*, "Matching networks for one shot learning," *Advances in Neural Information Processing Systems*, vol. 29, pp. 3630–3638, 2016.

[21] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM Computing Surveys (CSUR)*, vol. 53, no. 3, pp. 1–34, 2020.

[22] A. Oliver, A. Odena, C. A. Raffel, E. D. Cubuk, and I. Goodfellow, "Realistic evaluation of deep semi-supervised learning algorithms," *Advances in neural information processing systems*, vol. 31, 2018.

[23] P. Bachman, O. Alsharif, and D. Precup, "Learning with pseudo-ensembles," *Advances in neural information processing systems*, vol. 27, 2014.

[24] S. Laine and T. Aila, "Temporal ensembling for semi-supervised learning," *arXiv preprint arXiv:1610.02242*, 2016.

[25] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," *Advances in neural information processing systems*, vol. 30, 2017.

[26] T. Miyato, S.-i. Maeda, M. Koyama, and S. Ishii, "Virtual adversarial training: a regularization method for supervised and semi-supervised learning," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 8, pp. 1979–1993, 2018.

[27] Y. Grandvalet and Y. Bengio, "Semi-supervised learning by entropy minimization," *Advances in neural information processing systems*, vol. 17, 2004.

[28] D.-H. Lee *et al.*, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *Workshop on challenges in representation learning, ICML*, vol. 3, no. 2, 2013, p. 896.

[29] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. A. Raffel, "Mixmatch: A holistic approach to semi-supervised learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[30] K. Sohn, D. Berthelot, N. Carlini, Z. Zhang, H. Zhang, C. A. Raffel, E. D. Cubuk, A. Kurakin, and C.-L. Li, "Fixmatch: Simplifying semi-supervised learning with consistency and confidence," *Advances in Neural Information Processing Systems*, vol. 33, pp. 596–608, 2020.

[31] K. W. Morton, *Numerical solution of convection-diffusion problems*. CRC Press, 2019.

[32] Z. Svoboda, "The convective-diffusion equation and its use in building physics," *International journal on architectural science*, vol. 1, no. 2, pp. 68–79, 2000.

[33] P. A. Markowich and P. Szmolyan, "A system of convection—diffusion equations with small diffusion coefficient arising in semiconductor physics," *Journal of Differential Equations*, vol. 81, no. 2, pp. 234–254, 1989.

[34] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *Proceedings of the 20th International conference on Machine learning (ICML-03)*, 2003, pp. 912–919.

[35] B. Nadler, N. Srebro, and X. Zhou, "Semi-supervised learning with the graph laplacian: The limit of infinite unlabelled data," *Advances in neural information processing systems*, vol. 22, pp. 1330–1338, 2009.

[36] Z. Shi, S. Osher, and W. Zhu, "Weighted nonlocal laplacian on interpolation from sparse data," *Journal of Scientific Computing*, vol. 73, no. 2, pp. 1164–1177, 2017.

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

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

[39] Y. Li and Y. Liang, "Learning overparameterized neural networks via stochastic gradient descent on structured data," in *Advances in Neural Information Processing Systems*, 2018, pp. 8157–8166.

[40] J. Klicpera, S. Weißenberger, and S. Günnemann, "Diffusion improves graph learning," in *Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

[41] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in *Advances in neural information processing systems*, 2016, pp. 1993–2001.

[42] R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, B. Nadler, F. Warner, and S. W. Zucker, "Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps," *Proceedings of the national academy of sciences*, vol. 102, no. 21, pp. 7426–7431, 2005.

[43] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[44] D. Kushnir and L. Venturi, "Diffusion-based deep active learning," *arXiv preprint arXiv:2003.10339*, 2020.

[45] P. Jiang, F. Gu, Y. Wang, C. Tu, and B. Chen, "Difnet: Semantic segmentation by diffusion networks," *arXiv preprint arXiv:1805.08015*, 2018.

[46] B. Wang, B. Yuan, Z. Shi, and S. Osher, "Resnets ensemble via the feynman-kac formalism to improve natural and robust accuracies," *arXiv: Learning*, 2018.

[47] Q. Li, L. Chen, C. Tai, and E. Weinan, "Maximum principle based algorithms for deep learning," *Journal of Machine Learning Research*, vol. 18, no. 165, pp. 1–29, 2017.

[48] Q. Li and S. Hao, "An optimal control approach to deep learning and applications to discrete-weight neural networks," *arXiv: Learning*, 2018.

[49] Y. Lu, A. Zhong, Q. Li, and B. Dong, "Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations," *arXiv: Computer Vision and Pattern Recognition*, 2017.

[50] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Advances in neural information processing systems*, 2004, pp. 321–328.

[51] M. Belkin and P. Niyogi, "Semi-supervised learning on riemannian manifolds," *Machine learning*, vol. 56, no. 1, pp. 209–239, 2004.

[52] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples." *Journal of machine learning research*, vol. 7, no. 11, 2006.

[53] R. K. Ando and T. Zhang, "Learning on graph with laplacian regularization," in *Advances in neural information processing systems*, 2007, pp. 25–32.

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

[55] E. G. Miller, N. E. Matsakis, and P. A. Viola, "Learning from one example through shared densities on transforms," in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662)*, vol. 1. IEEE, 2000, pp. 464–471.

[56] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[57] J. Snell, K. Swersky, and R. S. Zemel, "Prototypical networks for few-shot learning," *arXiv preprint arXiv:1703.05175*, 2017.

[58] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1199–1208.

[59] B. N. Oreshkin, P. Rodriguez, and A. Lacoste, "Tadam: Task dependent adaptive metric for improved few-shot learning," *arXiv preprint arXiv:1805.10123*, 2018.

[60] Y. Wang, W.-L. Chao, K. Q. Weinberger, and L. van der Maaten, "Simpleshot: Revisiting nearest-neighbor classification for few-shot learning," *arXiv preprint arXiv:1911.04623*, 2019.

[61] I. Ziko, J. Dolz, E. Granger, and I. B. Ayed, "Laplacian regularized few-shot learning," in *International Conference on Machine Learning*. PMLR, 2020, pp. 11 660–11 670.

[62] J. Liu, L. Song, and Y. Qin, "Prototype rectification for few-shot learning," *arXiv preprint arXiv:1911.10713*, 2019.

[63] J. Geiser, *Decomposition methods for differential equations: theory and applications*. CRC Press, 2009.

[64] P. Balister, B. Bollobás, A. Sarkar, and M. Walters, "Connectivity of random k-nearest-neighbour graphs," *Advances in Applied Probability*, vol. 37, no. 1, pp. 1–24, 2005.

[65] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, "Pitfalls of graph neural network evaluation," *Relational Representation Learning Workshop, NeurIPS 2018*, 2018.

[66] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[67] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NIPS*, 2017.

[68] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," *International Conference on Learning Representations*, 2018.

[69] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Thirty-Second AAAI conference on artificial intelligence*, 2018.

[70] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, "Simple and deep graph convolutional networks," in *International Conference on Machine Learning*. PMLR, 2020, pp. 1725–1735.

[71] 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.

[72] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The Caltech-UCSD Birds-200-2011 Dataset," California Institute of Technology, Tech. Rep. CNS-TR-2011-001, 2011.

[73] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *In International Conference on Learning Representations (ICLR)*, 2017.

[74] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. Wang, and J.-B. Huang, "A closer look at few-shot classification," in *International Conference on Learning Representations*, 2019.

[75] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *BMVC*, 2016.

[76] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[77] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1126–1135.

[78] S. Gidaris and N. Komodakis, "Dynamic few-shot visual learning without forgetting," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4367–4375.

[79] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "A simple neural attentive meta-learner," in *International Conference on Learning Representations*, 2018.

[80] G. S. Dhillon, P. Chaudhari, A. Ravichandran, and S. Soatto, "A baseline for few-shot image classification," in *International Conference on Learning Representations*, 2019.

[81] K. Lee, S. Maji, A. Ravichandran, and S. Soatto, "Meta-learning with differentiable convex optimization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 657–10 665.

[82] Y. Liu, J. Lee, M. Park, S. Kim, E. Yang, S. J. Hwang, and Y. Yang, "Learning to propagate labels: Transductive propagation network for few-shot learning," *arXiv preprint arXiv:1805.10002*, 2018.

[83] L. Qiao, Y. Shi, J. Li, Y. Wang, T. Huang, and Y. Tian, "Transductive episodic-wise adaptive metric for few-shot learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 3603–3612.

[84] R. Hou, H. Chang, B. Ma, S. Shan, and X. Chen, "Cross attention network for few-shot classification," in *NeurIPS*, 2019.

[85] S. Qiao, C. Liu, W. Shen, and A. L. Yuille, "Few-shot image recognition by predicting parameters from activations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7229–7238.

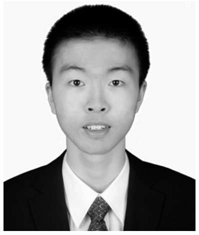
[86] A. A. Rusu, D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, and R. Hadsell, "Meta-learning with latent embedding optimization," in *International Conference on Learning Representations*, 2018.

[87] S. Gidaris, A. Bursuc, N. Komodakis, P. Pérez, and M. Cord, "Boosting few-shot visual learning with self-supervision," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 8059–8068.

[88] H.-J. Ye, H. Hu, D.-C. Zhan, and F. Sha, "Few-shot learning via embedding adaptation with set-to-set functions," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 8808–8817.

[89] S. Li, "Concise formulas for the area and volume of a hyperspherical cap," *Asian Journal of Mathematics and Statistics*, vol. 4, no. 1, pp. 66–70, 2011.

[90] F. R. Chung and F. C. Graham, *Spectral graph theory*. American Mathematical Soc., 1997, no. 92.



Tangjun Wang is a Ph.D. student in Department of Mathematical Sciences, Tsinghua University supervised by Zuoqiang Shi. His research interests include semi-supervised learning and applications of partial differential equation in neural networks.

Zehao Dou is a Ph.D. student in Department of Statistics and Data Science, Yale University supervised by Harry Zhou and John Lafferty. His research interests include statistics, optimization, machine learning theory and reinforcement learning theory.

Chenglong Bao is an assistant professor in Yau mathematical sciences center, Tsinghua University and Yanqi Lake Beijing Institute of Mathematical Sciences and Applications. He received his Ph.D. from department of mathematics, National University of Singapore in 2014. His main research interests include mathematical image processing, large scale optimization and its applications.

Zuoqiang Shi is an associate professor in Department of Mathematical Sciences, Tsinghua University. He received his Ph.D. from Zhou Pei-Yuan center for Applied Mathematics, Tsinghua University in 2008. His main research interests include numerical methods of partial differential equations and its applications, mathematical image processing.

Appendix A
Proof of Stability Condition

In this section, we will give the stability condition of discretization of the diffusion step. When the convection term equals zero, the forward Euler discretization of (7) is

$$x_i^{k+1} = x_i^k - \gamma \sum_{j=1}^{N} w_{ij}(x_i^k - x_j^k),\ i = 1, 2 \ldots, N.$$

Δt is again absorbed in γ. Define $X^k = [x_1^k, \ldots, x_N^k]$, the vectorized update scheme is

$$X^{k+1} = X^k - \gamma(\Lambda - W)X^k, \tag{8}$$

where $\Lambda = \mathrm{diag}(d_i)$ with $d_i = \sum_{j=1}^N w_{ij}$ for all $i = 1, 2, \ldots, N$. We demand all the $\{d_i\}$ share the same value $d_1 = d_2 = \cdots = d_N := d$. The next proposition shows the stability condition of iteration (8).

Proposition 2. *If $\gamma \in \left(0, \frac{1}{d}\right]$, the iteration* (8) *is a contraction.*

Proof. Matrix with spectral radius smaller than 1 is a contraction matrix, since

$$\|Ax\|_2 \leqslant \|A\|_2 \|x\|_2 \leqslant \|x\|_2$$

for any x if $\|A\|_2 = \rho(A) \leqslant 1$.

Thus, we are trying to prove that $\rho(I - \gamma(\Lambda - W)) \leqslant 1$. In fact, we will show that $\rho(I - \gamma(\Lambda - W)) = 1$. Denote $A = I - \gamma(\Lambda - W)$

First, it is obvious that 1 is an eigenvalue of A. $\Lambda - W$ is a matrix with row sum 0 by construction of Λ, thus 0 is an eigenvalue of $\Lambda - W$ with $\mathbf{1}$ its correspoding eigenvector. Therefore, 1 is an eigenvalue of $A = I - \gamma(\Lambda - W)$.

Then, as A is obviously a symmetric matrix, all of its eigenvalues λ are real. We will show that all the eigenvalues of A lies in $[-1, 1]$ using the Gershgorin disk theorem. In the i-th row of A, the diagonal entry is $1 - \gamma(d - w_{ii})$, and the off-diagonal entries are γw_{ij} for all $j \neq i$. Thus, the disk decided by this row is centered at $1 - \gamma(d - w_{ii})$, with radius $\sum_{j \neq i} \gamma w_{ij}$. Remind again that $d = \sum_j w_{ij}$, so $\sum_{j \neq i} \gamma w_{ij} = \gamma(d - w_{ii})$. Using $\gamma \in \left(0, \frac{1}{d}\right)$, we have

$$\begin{aligned} -1 &\leqslant 1 - 2\gamma d \\ &\leqslant 1 - 2\gamma d + 2\gamma w_{ii} \\ &= 1 - \gamma(d - w_{ii}) - \gamma(d - w_{ii}) \\ &\leqslant \lambda \\ &\leqslant 1 - \gamma(d - w_{ii}) + \gamma(d - w_{ii}) = 1 \end{aligned}$$

The second inequality is because all entries in W are nonnegative. In conclusion, the spectral radius of A is 1. □

In the algorithm, weight matrix is symmetrically normalized, thus $d = O(1)$. Therefore, in our batch diffusion mechanism, once the step size meets a relatively loose constraint, the stability can be guaranteed.

Appendix B
Proof of Theorem 1

We write the weight of the second layer as $a_t^{(i)} = \lambda_t^{(i)} \cdot \beta_t^{(i)}$ for proof convenience. $\lambda_t^{(i)} \in \mathbb{R}, \beta_t^{(i)} \in \mathbb{R}^d$. Then

$$f(x(t), \theta(t)) = \sum_{i=1}^{w} \lambda_t^{(i)} \sigma(\mathbf{w}_t^{(i)} \cdot x(t) + b_t^{(i)}) \beta_t^{(i)}$$

First of all, let us deal with the simplest situation. Each region $S_{i,j},\ (i,j) \in [k] \times [l]$ only has 1 point, and the width of the 2-layer network is also 1.

Lemma 1. *Assume:*

$$f(x(t), \theta(t)) = \lambda_t \sigma(\mathbf{w}_t \cdot x(t) + b_t)\beta_t$$

Then, given the dataset $\{(x_i, y_i)\}_{i=1}^N$, we can construct the function f above with $2N + O(d)$ different variables and N layers, so that the final-step features of these data points are linear separable.

Proof. Given N data points $x_1, x_2, \cdots, x_N \in R^d$ with their corresponding labels.

1) There exists $\mathbf{w} \in R^d$, such that $\mathbf{w} \cdot x_i,\ i \in [N]$ are all distinct. This is obvious since $\mathbf{W}_{ij} = \{\mathbf{w} | \mathbf{w} \cdot x_i = \mathbf{w} \cdot x_j\}$ forms a hyperplane of R^d, which has 0 measure. Therefore their finite union

$$K = \bigcup_{1 \leq i < j \leq N} \mathbf{W}_{ij}$$

also has 0 measure. We only need to pick a $\mathbf{w}^* \in R^d \setminus K$, so that this $\mathbf{w}^*$ meets our need. Let $\mathbf{w}_t$ be a constant value function and $\mathbf{w}_t \equiv \mathbf{w}^*$ holds for all $t \in [0, 1]$. Since the dot products $\mathbf{w}^* \cdot x_i$ are pairwise distinct, we can assume:

$$\mathbf{w}^* \cdot x_1 < \mathbf{w}^* \cdot x_2 < \cdots < \mathbf{w}^* \cdot x_N$$

Denote $A_i = \mathbf{w}^* \cdot x_i$ and then $A_1 < A_2 < \cdots < A_N$

2) Without loss of generality, we assume the d-th component $\mathbf{w}_d^* \neq 0$, then we denote:

$$\beta^* = \left(1, 1, \cdots, 1, -\frac{\mathbf{w}_1^* + \mathbf{w}_2^* + \cdots + \mathbf{w}_{d-1}^*}{\mathbf{w}_d^*}\right)$$

Let β_t also be a constant value function and $\beta_t \equiv \beta^*$. Then we have: $\mathbf{w}^* \perp \beta^*$ and our main function f becomes:

$$x'(t) = f(x(t), \theta(t)) = \lambda_t \sigma(\mathbf{w}^* \cdot x(t) + b_t)\beta^*$$

We notice that:

$$\begin{aligned} (\mathbf{w}^* \cdot x(t))' &= \mathbf{w}^* \cdot (\lambda_t \sigma(\mathbf{w}^* \cdot x(t) + b_t)\beta^*) \\ &= \lambda_t \sigma(\mathbf{w}^* \cdot x(t) + b_t) \cdot (\mathbf{w}^* \cdot \beta^*) = 0 \end{aligned}$$

which means during the flow, $\mathbf{w}^* \cdot x(t)$ remains constant. Thus

$$\mathbf{w}^* \cdot x_i(t) \equiv \mathbf{w}^* \cdot x_i(0) = \mathbf{w}^* \cdot x_i = A_i \quad \forall t \in [0, 1], i \in [N]$$

and $x_i'(t) = \lambda_t \sigma(A_i + b_t)\beta^*$. Using the definition of ReLU activation function , when $A_i + b_t < 0$, $x_i(t)$ remains unchanged.

3) Pick N real numbers $B_1, B_2, \cdots, B_N$ such that:

$$B_1 < A_1 < B_2 < A_2 < \cdots < B_N < A_N$$

Now we construct the time-dependent scalar b_t:

$$b_t = -B_i,\ \forall t \in \left[\frac{i-1}{N}, \frac{i}{N}\right), i \in [N]$$

As we can see, b_t is piecewise constant with N pieces, or so-called layers.

4) Finally, we construct suitable time-dependent scalar λ_t to make the final-step features $x_i(1),\ i \in [N]$ linear separable. Pick 2 real numbers $C_1 < C_2$. We will choose a suitable position for $x_i(1)$ one by one. To be concrete, we will make the first component of $x_i(1)$:

$$(x_i(1))_1 = C_{y_i}$$

Here $y_i \in \{1, 2\}$ is the corresponding label of data point x_i.

Notice that, for each $j \in [N-1]$, when $t > \frac{j}{N}$, $x_1(t), x_2(t), \cdots, x_j(t)$ remains constant, because according to our construction of b_t and sequence $\{B_i\}$, $A_i + b_t \leq A_j - B_{j+1} < 0$ when $t > \frac{j}{N}$ and $i \leq j$.

In the first time step, $t \in [0, \frac{1}{N})$, data point $x_1(0) = x_1$ has its corresponding label $y_1 \in \{1, 2\}$. Since $b_t = -B_1$

$$x_1'(t) = \lambda_t(A_1 - B_1)\beta^*$$

Make

$$\lambda_t \equiv \lambda_1 = \frac{C_{y_1} - (x_1(0))_1}{A_1 - B_1} \cdot N \quad \forall t \in \left[0, \frac{1}{N}\right)$$

Then

$$\begin{aligned} x_1(1) &= x_1(\frac{1}{N}) \\ &= x_1(0) + \frac{1}{N}\lambda_1(A_1 - B_1)\beta^* \\ &= x_1(0) + (C_{y_1} - (x_1(0))_1)\beta^* \end{aligned}$$

Note that the first component of β^* is 1, so $(x_1(1))_1 = C_{y_1}$.

Similarly, in the j-th time step, $t \in \left[\frac{j-1}{N}, \frac{j}{N}\right)$ and $b_t = -B_j$. We choose a suitable position for $x_j(1)$. Make

$$\lambda_t \equiv \lambda_j = \frac{C_{y_j} - (x_j(\frac{j-1}{N}))_1}{A_j - B_j} \cdot N \quad \forall t \in \left[\frac{j-1}{N}, \frac{j}{N}\right)$$

so that:

$$\begin{aligned} x_j(1) &= x_j\left(\frac{j}{N}\right) \\ &= x_j\left(\frac{j-1}{N}\right) + \frac{1}{N}\lambda_j(A_j - B_j)\beta^* \\ &= x_j\left(\frac{j-1}{N}\right) + (C_{y_j} - \left(x_j\left(\frac{j-1}{N}\right)\right)_1)\beta^* \end{aligned}$$

and then its first component $(x_j(1))_1 = C_{y_j}$.

To sum up, λ_t is piecewise constant with N pieces. $\lambda_t = \lambda_j$ when $t \in [\frac{j-1}{N}, \frac{j}{N})$. Here:

$$\lambda_j = \frac{C_{y_j} - (x_j(\frac{j-1}{N}))_1}{A_j - B_j} \cdot N \qquad \forall t \in \left[\frac{j-1}{N}, \frac{j}{N}\right)$$

After all these time steps, we can guarantee that for each $i \in [N]$,

$$(x_i(1))_1 = C_{y_i}$$

In other words, final-step features with the same corresponding label $y_i \in \{1, 2\}$ are on the same hyperplane, vectors with fixed first component $\{\mathbf{v} : (\mathbf{v})_1 = C_{y_i}\}$. Therefore, it is obvious that they can be easily separated by a hyperplane

$$\left\{\mathbf{v} : (\mathbf{v})_1 = \frac{C_1 + C_2}{2}\right\}$$

which meets our satisfaction. In this construction of function f, $\mathbf{w}_t, \beta_t$ remains constant, and λ_t, b_t changes every time step. In all, there are $2N + O(d)$ variables. □

In the setting above, the width of this network is 1. Now we will deal with a slightly harder situation: network with width w. And our next lemma can perfectly answer this question: it's possible to use fewer layers (but same number of parameters) to make the final features linear separable when using wider network.

Lemma 2. *Assume:*

$$f(x(t), \theta(t)) = \sum_{i=1}^{w} \lambda_t^{(i)} \sigma(\mathbf{w}_t^{(i)} \cdot x(t) + b_t^{(i)})\beta_t^{(i)}$$

Here w is the width of the network.

Then, given the dataset $\{(x_i, y_i)\}_{i=1}^N$, we can construct the function f above with $2N + O(d)$ different variables and $\lceil N/w \rceil$ layers, so that the final-step features of these data points are linear separable.

Proof. Exactly like Lemma 1, we let $\mathbf{w}_t^{(i)} \equiv \mathbf{w}^*$ and $\beta_t^{(i)} \equiv \beta^*$ for $\forall i \in [w]$. And we will construct suitable time-dependent scalars λ_t^i, b_t^i in order to make:

$$(x_i(1))_1 = C_{y_i}$$

holds for all $i \in [N]$.

Define $B_{N+1} = B_{N+2} = \cdots = A_N + 1$, and denote $L = \lceil \frac{N}{w} \rceil$. Before further analysis, we split the whole time period $[0, 1]$ into L equal time steps.

In the j-th time step, $t \in [\frac{j-1}{L}, \frac{j}{L})$, let:

$$b_t^{(i)} \equiv -B_{(j-1)w+i},\ \lambda_t^{(i)} \equiv \lambda_{(j-1)w+i} \ \ \forall i \in [w]$$

where λ_i is undetermined.

Notice that each $b_t^{(i)}$ and $\lambda_t^{(i)}$ is piecewise constant with L pieces. Similar to Lemma 1, after the j-th time step, $x_1(t), x_2(t), \cdots, x_{jw}(t)$ remains unchanged, because $\forall i \in [jw], t > \frac{j}{L}$, $A_i + b_t^{(j)} \leqslant A_{jw} - B_{jw+1} < 0$. Therefore, our plan is to put $x_{(j-1)w+1}(1), x_{(j-1)w+2}(1), \cdots, x_{jw}(1)$ into suitable positions during the j-th time step, which means:

$$\begin{aligned} C_{y_i} &= \left(x_i\left(\frac{j-1}{L}\right)\right)_1 + \frac{1}{L}\sum_{k=1}^{w} \lambda_{(j-1)w+k} \cdot \sigma(A_i - B_{(j-1)w+k}) \\ &= \left(x_i\left(\frac{j-1}{L}\right)\right)_1 + \frac{1}{L}\sum_{k=1}^{i-(j-1)w} \lambda_{(j-1)w+k} \cdot (A_i - B_{(j-1)w+k}) \\ &\qquad \forall (j-1)w < i \leqslant jw \end{aligned}$$

It is a linear system of equations, and we can solve these $\lambda_i,\ (j-1)w < i \leqslant jw$ through the linear functions above uniquely.

After all of these time steps, we can guarantee that:

$$(x_i(1))_1 = C_{y_i}$$

holds for all $i \in [N]$.

In this proceedings, we use only $L = \lceil \frac{N}{w} \rceil$ layers to meet our satisfaction. However, the number of variables we use is also $2N + O(d)$, which does not change with the increase of network width. □

At last, we can deal with the original Theorem 1, which has width w and each region has several points. It is a simple extension of Lemma 2.

Proof. For simplicity, we reorganize $M = kl$ subsets and renumber them as $\Gamma_m = S_{i,j},\ m \in [M],\ (i,j) \in [k] \times [l]$, as we do not have to distinguish whether regions are from the same class or different classes. Since they can be separated by a set of $M-1$ parallel hyperplanes, there exists a vector $\mathbf{w}^* \in \mathbb{R}^d$, such that the following intervals do not intersect with each other:

$$\mathbf{w}^* \cdot \Gamma_i \triangleq \{\mathbf{w}^* \cdot x_i | x_i \in \Gamma_i\} \ \ i \in [M]$$

We can assume $\forall x_1 \in \Gamma_1, x_2 \in \Gamma_2, \cdots, x_M \in \Gamma_M$:

$$\mathbf{w}^* \cdot x_1 < \mathbf{w}^* \cdot x_2 < \cdots < \mathbf{w}^* \cdot x_M$$

Therefore, there exists real numbers $B_1 < A_1 \leqslant B_2 < A_2 \leqslant \cdots \leqslant B_M < A_M$ such that $\mathbf{w}^* \cdot \Gamma_i \subseteq [B_i, A_i]$. Just like the proof of Lemma 2, we let $\mathbf{w}_t^{(i)} \equiv \mathbf{w}^*$ and $\beta_t^{(i)} \equiv \beta^*$ for $\forall i \in [w]$. Here, without lack of generality, assume the d-th component of $\mathbf{w}^*$ is non-zero, and let

$$\beta^* = \left(1, 1, \cdots, 1, -\frac{\mathbf{w}_1^* + \mathbf{w}_2^* + \cdots + \mathbf{w}_{d-1}^*}{\mathbf{w}_d^*}\right)$$

After that, we can treat these M regions just as M data points. The only difference is: instead of making each final-step feature with the same corresponding label has the same first component, we make them in the same interval.

Pick 2 intervals $\mathbf{C}_{y_i} = [a_{y_i}, b_{y_i}]$ such that: (1) $a_1 < b_1 < a_2 < b_2$, therefore these two intervals don't intersect with each other. (2) $b_{y_i} - a_{y_i} > D$ holds for each $y_i \in \{1, 2\}$. So that, each interval can hold a complete region Γ_i. Then we can use exactly the same way as Lemma 2 to make $x_i(1) \subseteq \mathbf{C}_{y_i}$, and we only have to change the two real numbers $C_1 < C_2$ into the two intervals above. In the end, we can separate the final-step feature regions with the following parallel hyperplane:

$$\left\{\mathbf{v} : (\mathbf{v})_1 = \frac{d_1 + c_2}{2}\right\}$$

Similar to Lemma 2, we use $L = \lceil \frac{M}{w} \rceil$ layers. Moreover, the number of variables we use decreases to $2M + O(d)$, which is a significant change. □

Appendix C
Proof of Theorem 2

Proof. Similar to the last subsection, we reorganize $M = kl$ subsets and renumber them as $\Gamma_m = S_{i,j},\ m \in [M],\ (i,j) \in [k] \times [l]$. According to the definition of upper bound of diameters D, assume $\Gamma_i \subseteq B(O_i, D/2)$ and denote $R = D/2$.

First we introduce some notations. Denote hypersphere in d-dimension with radius r as $S^{d-1}(r)$. If $r = 1$, which is a unit hypersphere, we simply write S^{d-1}. Denote $A(\omega)$ as the surface area of ω. $\Gamma(x)$ is the Gamma Function.

The goal to find a unit normal vector $\mathbf{w} \in S^{d-1}$, such that $\forall\ b \in R$, hyperplane $\mathbf{w} \cdot \mathbf{x} + b = 0$ does not intersect with any two regions Γ_i and Γ_j.

Denote:

$$K_{ij} = \{\mathbf{w} : \|\mathbf{w}\|_2 = 1,\ \exists\, b \in \mathbb{R}, s.t. \text{ hyperplane } \mathbf{wx} + b = 0 \\ \text{intersects with both } \Gamma_i \text{ and } \Gamma_j\}$$

Then we need to prove:

$$\bigcup_{1 \leqslant i < j \leqslant M} K_{ij} \subsetneqq S^{d-1} \tag{9}$$

so that every unit normal vector $\mathbf{w}$ which doesn't belong to any K_{ij} meets our satisfaction. In order to prove (9), we compare the surface area of the two sides.

It is well known that on the right side

$$A(S^{d-1}) = \frac{2\pi^{d/2}}{\Gamma(\frac{d}{2})}$$

We are going to calculate the surface area of the left side. For any $i \neq j \in [M]$, we do the scaling as follows:

$$\begin{aligned} K_{ij} &= \{\mathbf{w} : \|\mathbf{w}\|_2 = 1,\ \exists\, b \in \mathbb{R}, s.t. \text{ hyperplane } \mathbf{wx} + b = 0 \\ &\qquad \text{intersects with both } \Gamma_i \text{ and } \Gamma_j\} \\ &\subseteq \{\mathbf{w} : \|\mathbf{w}\|_2 = 1,\ \exists\, b \in \mathbb{R}, s.t. \text{ distances from } O_i, O_j \\ &\qquad \text{to } \mathbf{wx} + b = 0 \text{ are at most } R\} \\ &= \{\mathbf{w} : \|\mathbf{w}\|_2 = 1,\ \exists\, b \in \mathbb{R}, s.t.\ |\mathbf{w} \cdot \mathbf{O}_i + b| \leqslant R, \\ &\qquad |\mathbf{w} \cdot \mathbf{O}_j + b| \leqslant R\} \\ &= \{\mathbf{w} : \|\mathbf{w}\|_2 = 1,\ |\mathbf{w} \cdot \mathbf{O}_i - \mathbf{w} \cdot \mathbf{O}_j| \leqslant 2R\} \\ &\subseteq \{\mathbf{w} : \|\mathbf{w}\|_2 = 1,\ |\mathbf{w} \cdot \hat{\overrightarrow{O_iO_j}}| \leqslant \frac{2R}{L} = \frac{D}{L}\} \end{aligned}$$

Here, $\hat{\overrightarrow{O_iO_j}}$ means the unit vector in the direction of $\overrightarrow{O_iO_j}$ and the last step above is because $\|O_iO_j\| \geqslant dist(\Gamma_i, \Gamma_j) > L$. Denote $t = \frac{D}{L}, \mathbf{e}_d = (0, 0, \cdots, 0, 1)$. By our assumption obviously $t < 1$. Next we will calculate the surface area of K_{ij}.

$$A(K_{ij}) \leqslant \int_{\mathbf{w} \in S^{d-1}, |\mathbf{w} \cdot \hat{\overrightarrow{O_iO_j}}| \leqslant t} dS = \int_{\mathbf{w} \in S^{d-1}, |\mathbf{w} \cdot \mathbf{e}_d| \leqslant t} dS$$

It is the surface area of a *hyperspherical segment*: the solid defined by cutting a hypersphere with a pair of parallel planes $\{\mathbf{v} : (\mathbf{v})_d = t\}$ and $\{\mathbf{v} : (\mathbf{v})_d = -t\}$. It can also be seen as a complete sphere excluding upper and lower *hyperspherical caps*. The area of a hypersherical cap in a d-dimensional sphere of radius r can be obtained by integrating the surface area of an $(d-1)$-dimensional sphere of radius $r sin(\theta)$ with arc element $r d\theta$ over a great circle arc [89]. Here $r = 1$, and θ is integrated over 0 to $\varphi = arccos(t)$, which is the colatitude angle, i.e., the angle between a vector of the sphere and its d^{th} positive axis.

$$\begin{aligned} A(K_{ij}) &\leqslant A(S^{d-1}) - 2\int_0^{\arccos(t)} A(S^{d-2}(\sin\theta)) d\theta \\ &= A(S^{d-1}) - 2\,\frac{2\pi^{(d-1)/2}}{\Gamma(\frac{d-1}{2})} \int_0^{\arccos(t)} \sin^{d-2}\theta d\theta \end{aligned}$$

It is obvious that when $t = 0$, the hyperspherical segment is just the whole sphere, which means

$$A(S^{d-1}) = 2\,\frac{2\pi^{(d-1)/2}}{\Gamma(\frac{d-1}{2})} \int_0^{\frac{\pi}{2}} \sin^{d-2}\theta d\theta$$

Thus

$$\begin{aligned} A(K_{ij}) &\leqslant 2\,\frac{2\pi^{(d-1)/2}}{\Gamma(\frac{d-1}{2})} \int_{\arccos(t)}^{\frac{\pi}{2}} \sin^{d-2}\theta d\theta \\ &= A(S^{d-1}) \frac{2\Gamma(\frac{d}{2})}{\sqrt{\pi}\Gamma(\frac{d-1}{2})} \int_{\arccos(t)}^{\frac{\pi}{2}} \sin^{d-2}\theta d\theta \end{aligned}$$

Therefore:

$$\begin{aligned}\frac{A(K_{ij})}{A(S^{d-1})} &\leqslant \frac{2\Gamma(\frac{d}{2})}{\sqrt{\pi}\Gamma(\frac{d-1}{2})}\int_{\arccos(t)}^{\frac{\pi}{2}} \sin^{d-2}\theta d\theta \\ &\leqslant \frac{2\Gamma(\frac{d}{2})}{\sqrt{\pi}\Gamma(\frac{d-1}{2})}(\pi/2 - \arccos(t)) \\ &= \frac{2\Gamma(\frac{d}{2})}{\sqrt{\pi}\Gamma(\frac{d-1}{2})}\arcsin(t)\end{aligned}$$

Next we will estimate its upper bound. From the graph of function, we can obtain the upper bound of $\arcsin(t) \leq \frac{\pi}{2}t, \ \forall t \in [0, 1]$. The upper bound of the other part containing Gamma function is given in the following lemma.

Lemma 3.

$$\frac{\Gamma(\frac{d}{2})}{\Gamma(\frac{d-1}{2})} < \frac{d}{2} \quad \forall d \in \mathbb{N}$$

Proof. When $d \leqslant 5$, the lemma can be easily verified using exact values of the Gamma function. Suppose $d \geqslant 6$. Note that Gamma function $\Gamma(x)$ is monotonically increasing when $x \geq 2$

If d is odd, then $\frac{d-1}{2}$ is an integer, and $\frac{d}{2} > 2$

$$\frac{\Gamma(\frac{d}{2})}{\Gamma(\frac{d-1}{2})} \leqslant \frac{\Gamma(\frac{d+1}{2})}{\Gamma(\frac{d-1}{2})} = \frac{d-1}{2} < \frac{d}{2}$$

In the equation above, we use the property of Gamma function: $\Gamma(x+1) = x\Gamma(x)$ if x is an integer.

Similarly, if d is even, then $\frac{d}{2}$ is an integer, and $\frac{d}{2} - 1 \geqslant 2$

$$\frac{\Gamma(\frac{d}{2})}{\Gamma(\frac{d-1}{2})} \leqslant \frac{\Gamma(\frac{d}{2})}{\Gamma(\frac{d}{2}-1)} = \frac{d}{2} - 1 < \frac{d}{2}$$

□

Therefore:

$$\frac{A(K_{ij})}{A(S^{d-1})} < \frac{2}{\sqrt{\pi}}\frac{d}{2}\frac{\pi}{2}t = \frac{\sqrt{\pi}d}{2}t$$

Finally, we are able to calculate the surface area of the left side of (9):

$$\begin{aligned}A\left(\bigcup_{1\leqslant i<j\leqslant M} K_{ij}\right) &\leqslant \sum_{1\leqslant i<j\leqslant M} A(K_{ij}) \\ &< \frac{\sqrt{\pi}d}{2}tA(S^{d-1}) \cdot \binom{M}{2} \\ &\leqslant A(S^{d-1})\end{aligned}$$

Here, we use the assumption that:

$$t = \frac{D}{L} \leqslant \left(\frac{M(M-1)\sqrt{\pi}}{4}d\right)^{-1}$$

Thus, the correctness of (9) is obvious. □

Appendix D
Proof of Proposition 1

Proof. Following proof of Theorem 1 and 2, renumber the $M = kl$ subsets and denote them $\Gamma_m = S_{i,j}, \ m \in [M], \ (i,j) \in [k] \times [l]$ for notation convenience. We consider the forward Euler discretization of the diffusion process:

$$x_i^{k+1} = x_i^k - \gamma\sum_{j=1}^N w_{ij}(x_i^k - x_j^k), \ \forall i \in [N]$$

We will first prove that with each update scheme, the new region $\Gamma_i^{k+1} \subseteq \Gamma_i^k$, thus $L(t)$ in monotonic non-decreasing. For a specific data point x_i, suppose $x_i \in \Gamma_m$. Since $\sum_{j=1}^N w_{ij} = 1$ by normalization, the diffusion is:

$$x_i^{k+1} = (1-\gamma)x_i^k + \gamma\sum_{j=1}^N w_{ij}x_j^k$$

By assumption, points in each subset Γ_m forms a connected component in graph G. Thus, the nearest n_{top} points of x_i are all from Γ_m, i.e. $w_{ij} > 0$ only if $x_j \in \Gamma_m$ [5]. Use the condition Γ_m is convex, and $\sum_{j=1}^N w_{ij} = 1$ again, the weighted sum $\sum_{j=1}^N w_{ij}x_j$ also lies in the convex region Γ_m. Finally, as $(1-\gamma)+\gamma = 1$, we get $x_i^{k+1} \in \Gamma_m$. In other words, after batch diffusion, the new convex region $\Gamma_i^{k+1} \subseteq \Gamma_i^k$.

Then we will show the upper bound of diameters $D(t)$ decreases exponentially to 0 with t. We may write diffusion mechanism in the vectorized form:

$$\frac{dX(t)}{dt} + \gamma(\Lambda - W)X(t) = 0, \quad X(0) = X. \qquad (10)$$

where $X(t) = [x_1(t), \ldots, x_N(t)]$, $X = [x_1, \ldots, x_N]$, $\Lambda = \mathrm{diag}(d_i)$ with $d_i = \sum_{j=1}^N w_{ij}$ for all $i = 1, 2, \ldots, N$, $W(i,j) = w_{ij}$. $L = \Lambda - W$ is called graph Laplacian[6]. Denote the eigenvalues and corresponding eigenvectors of L as λ_i and $\boldsymbol{v}_i, \ i \in [N]$. Since L is a positive semi-definite symmetric matrix, λ_i are real and non-negative. Suppose $0 \leqslant \lambda_1 \leqslant \lambda_2 \leqslant \cdots \leqslant \lambda_N$. In spectral clustering literatures [90], the reliance of graph diffusion on L is well studied. A well-known result is: the multiplicity of 0 eigenvalue of the Laplacian equals the number of connected components of graph G.

We start with the simple case, where the graph G only admits one connected component, i.e. when all data points belong to the same subclass. In this case, $\lambda_1 = 0$ with $\boldsymbol{v}_1 = \mathbf{1}$ its correspoding eigenvector, and $\lambda_i > 0, \ \forall i \neq 1$. We will prove all points converge to their central. The spectral solution of (10) is [90]:

$$X(t) = \sum_{i=1}^N \frac{X(0)\cdot\boldsymbol{v}_i}{|\boldsymbol{v}_i|^2}e^{-\gamma\lambda_i t}\boldsymbol{v}_i \qquad (11)$$

As $t \to \infty$, $e^{-\gamma\lambda_i t} \to 0$ if $\lambda_i > 0$. Define

$$m_c = \frac{x_1 + x_2 + \cdots + x_N}{N}$$

[5] Reverse not necessarily true. $x_j \in \Gamma_m$ does not necessarily imply $w_{ij} > 0$.

[6] Abuse of notation with the lower bound of distances L

as the central point of data points. Then

$$\begin{aligned}\lim_{t\to\infty} X(t) &= \lim_{t\to\infty} \sum_{i=1}^{N} \frac{X(0)\cdot \boldsymbol{v}_i}{|\boldsymbol{v}_i|^2} e^{-\gamma\lambda_i t}\boldsymbol{v}_i \\ &= \lim_{t\to\infty} \frac{X(0)\cdot \boldsymbol{v}_1}{|\boldsymbol{v}_1|^2} e^{-\gamma\lambda_1 t}\boldsymbol{v}_1 \\ &= \frac{X(0)\cdot \mathbf{1}}{N}\cdot \mathbf{1} \\ &= [m_c, m_c, \cdots, m_c]\end{aligned}$$

The result above implies that all data points eventually lie in the same position, which is their central point, as time t approaches infinity. Moreover, it is obvious from the equation that the growth rate is exponential. Thus, with the evolution of our diffusion mechanism, the diameter $D(t)$ decreases exponentially to 0.

For the more general case where there are M connected components in the graph, the proof is identical by using the fact that the multiplicity of 0 eigenvalue of the Laplacian equals the number of connected components of graph G. For each connected component, its points will converge to a specific central, and the diameter of each subset

$$\lim_{t\to\infty} \mathrm{diam}(\Gamma_m(t)) = 0,\ \forall m \in [M]$$

Thus the upper bound of diameters $D(t)$ decreases exponentially to 0. Combining the results $L(t)$ is non-decreasing and $D(t) \to 0$ exponentially, we get

$$\lim_{t\to\infty} \frac{L(t)}{D(t)} = \infty.$$

the growth rate is exponential. □

Appendix E
Experiment Details and Results

E.1 Synthetic Data

E.1.1 Dataset

XOR Uniformly collect 100 points each in four circles centered at (0,0), (0,2), (2,0), (2,2), respectively. Circles are with radius 0.75.

Moon Uniformly collect 500 points each in two arcs of semi-circle: one is the upper arc of a circle centered at (0, 0) with radius 1, the other is the lower arc of a circle centered at (1, 0.5) also with radius 1. Points are added with a standard gaussian noise multiplied by 0.05.

Circle Uniformly collect 500 points each in two circumference of circles: both are centered at (0, 0), one has radius 1 and the other has radius 2. Points are added with a standard gaussian noise multiplied by 0.05.

Spiral Uniformly collect 500 points each in two spirals: both are parametrized by $r = a + b\theta$. One has $a = b = 1$ and the other has $a = b = -1$. Points are added with a standard gaussian noise multiplied by 0.1.

E.1.2 Network Structure

$$\begin{aligned} x &= x + \text{FC2 (ReLU (FC1 (}\ x\ \text{)))} \\ x &= \text{Diffusion}(x) \qquad \text{for } r \text{ times} \\ y &= \text{FC3 (}\ x\ \text{)}\end{aligned}$$

We use one residual block, i.e. $s = 1$. All the fully connected layers are of size 2×2 with bias (that is why we have totally 3×2×3=18 parameters). As for the diffusion layer, we use a fixed step size γ, and iterate for r times.

E.1.3 Parameters

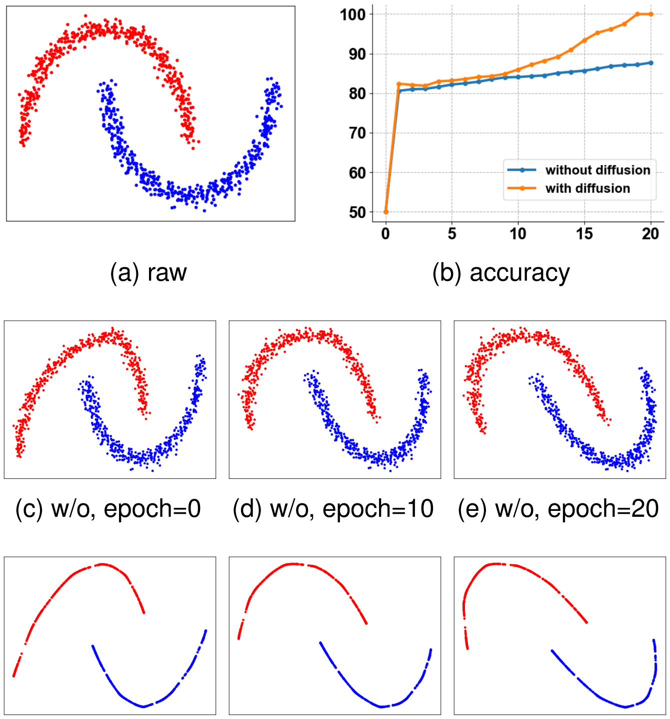
TABLE 4
Parameters for synthetic data

	n_{top}	σ	γ	r
XOR	20	0.5	1.0	/
Moon	25	0.5	1.0	60
Circle	50	0.5	1.0	200
Spiral	25	0.5	1.0	900

For the classification tasks, our optimizer is SGD with lr= 1.0, momentum= 0.9 and weight_decay= $5e-4$. For spiral dataset, we adjust lr= 0.8.

E.1.4 Additional Results

We provide the figures describing the evolution of features with or without diffusion in residual network on the other two synthetic datasets in Figure 11 and Figure 12.



(a) raw (b) accuracy

(c) w/o, epoch=0 (d) w/o, epoch=10 (e) w/o, epoch=20

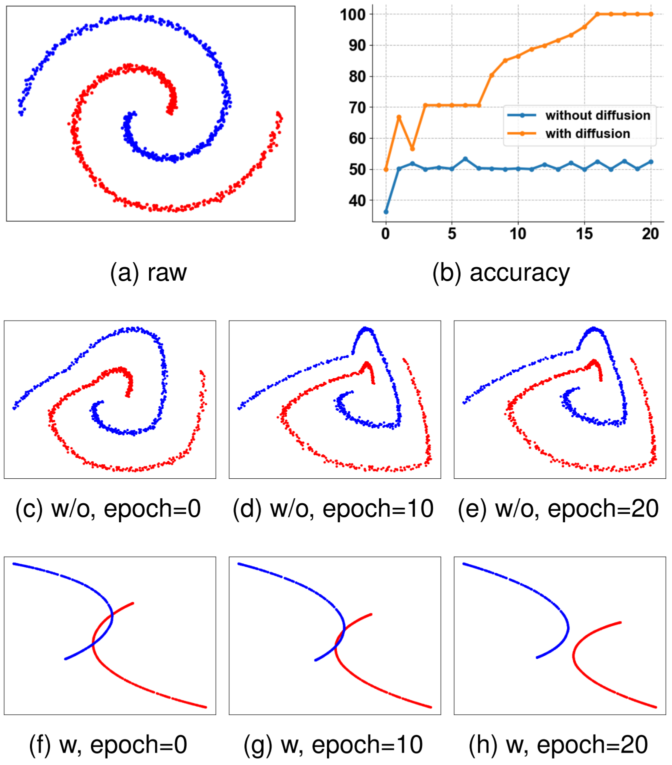
(f) w, epoch=0 (g) w, epoch=10 (h) w, epoch=20

Fig. 11. ResNet and DiffResNet on moon dataset, figures are arranged similar to Fig.4

E.2 Graph Learning

E.2.1 Dataset

Here we give the statistics of each dataset. For each randomly chosen split, we pick 20 labeled points for training, and 30 points for validation in each class. All of the rest points



(a) raw (b) accuracy

(c) w/o, epoch=0 (d) w/o, epoch=10 (e) w/o, epoch=20

(f) w, epoch=0 (g) w, epoch=10 (h) w, epoch=20

Fig. 12. ResNet and Diff-ResNet on spiral dataset, figures are arranged similar to Fig.4

are used as the test set. For all datasets, we treat the graph as undirected and only consider the largest connected component.

TABLE 5
Graph Dataset Statistics.

Dataset	Node	Edge	Class	Feature Dim	Label Rate
Cora	2485	5069	7	1433	0.057
Citeseer	2120	3679	6	3703	0.056
Pubmed	19717	44324	3	500	0.003

E.2.2 Preprocessing

We follow the normalization technique in GCN [66]: the adjacent matrix is first added with a self-loop, and then symmetrically normalized. The feature vectors are row normalized.

E.2.3 Network Structure

Since we observe severe overfitting problem in graph learning, we delete FC2 to reduce the number of parameters, and apply dropout on the feature vectors after each round of diffusion.The network structure is:

$$\begin{aligned} &x = x + \text{ReLU (FC1 (}\ x\ \text{))} \\ &x = \text{Dropout (Diffusion (}\ x\ \text{))} \qquad \text{for } r \text{ times} \\ &y = \text{FC3 (}\ x\ \text{)} \end{aligned}$$

The fully connected layers have input and output dimension the same as feature dimension. The new structure introduces a new parameter compared to toy examples: the dropout rate. But parameter n_{top} and σ is unnecessary in graph learning.

E.2.4 Parameters

Parameters are chosen based on the accuracy on the validation set.

TABLE 6
Parameters for graph learning

	γ	r	dropout rate
Cora	20	0.25	0.25
Citeseer	20	0.2	0.35
Pubmed	10	0.4	0.3

E.3 Few-shot Learning

E.3.1 Preprocessing

Following previous works [60], [61], [62], three additional feature transformation skills are used to enhance the performance.

(1) Centering and Normalization

$$x = x - \bar{x} \quad \text{then} \quad x = \frac{x}{\|x\|_2},\ \forall x \in \mathbb{X}_s \cup \mathbb{X}_q$$

$\bar{x}$ is the base class average.

(2) Cross-Domain Shift $x = x + \Delta,\ \forall x \in \mathbb{X}_q$, where

$$\Delta = \frac{1}{|\mathbb{X}_s|} \sum_{\mathbb{X}_s} x - \frac{1}{|\mathbb{X}_q|} \sum_{\mathbb{X}_q} x$$

is the difference between the mean of features within the support set and the mean of features within the query set.

we apply the first two transformations to all extracted features.

(3) Prototype Rectification

$$\tilde{m_c} = \frac{1}{|\mathbb{X}_s^c| + |\mathbb{X}_q^c|} \sum_{x \in \{\mathbb{X}_s^c, \mathbb{X}_q^c\}} \frac{\exp(\cos(x, m_c))}{\sum_{x \in \{\mathbb{X}_s^c, \mathbb{X}_q^c\}} \exp(\cos(x, m_c))} x$$

Here $m_c = \frac{1}{|\mathbb{X}_s^c|} \sum_{\mathbb{X}_s^c} x$ is the mean of features within the support set of class c. $\mathbb{X}_s^c$ is the support set with label c. $\mathbb{X}_q^c$ is a pre-classified set based on nearest neighbors.

Prototype rectification is only applicable to classification methods that are based on prototype, and cannot be directly applied to our Cross-Entropy loss. Nonetheless, we observe in the ablation study 5.3 that in 5-shot tasks, merely nearest prototype classification can already achieve very competitive results, indicating the effectiveness of prototype in 5-shot tasks. So we mimic the first loss term in [61] and propose the prototypical loss below.

$$\text{Prototypical Loss} = \sum_{x_i \in \mathbb{X}_q} \sum_{c=1}^{C} f(x_i)_c d(x_i - \tilde{m_c})$$

The final loss is a weighted sum of Cross-Entropy Loss and Prototypical Loss, with another parameter α before Prototypical Loss.

E.3.2 *Network Structure*

The structure is identical to that in the toy examples, just changing the input and output dimension of each FC layer from 2 to M, where M is the dimension of embedded features.

E.3.3 *Parameters*

In the few-shot setting, the meaning of σ is slightly different: weight is now calculated by $\tilde{w}_{ij} = \exp(-\|x_i - x_j\|_2^2/\sigma(x_i)^2)$, where $\sigma(x_i) = k$ means σ is chosen to be the k-th closest distance from a specific point x_i , so it varies with points.

In ablation study, $n_{\text{top}} = 8$, $\sigma = 4$. The diffusion step size γ is fixed to be 0.5 for all tasks. The diffusion step number $r = 10$ for 1-shot learning, $r = 5$ for 5-shot learning. $\lambda = 0.5$, $\mu = 0.01$.

In experiments with additional tricks, we also choose $n_{\text{top}} = 8$, $\sigma = 4$. The diffusion step size γ is fixed to be 0.5 for all tasks. The diffusion step number r varies with tasks: for 1-shot learning, $r = 5$ for *mini*ImageNet and *tiered*ImageNet, $r = 6$ for CUB; for 5-shot learning, $r = 3$ for all datasets and backbones. In addition, the weight before Prototypical Loss is $\alpha = 0$ for 1-shot tasks, and $\alpha = 0.5$ for 5-shot tasks.

The optimizer is SGD with initial learning rate= 0.1, momentum= 0.9 and weight_decay= $1e$-4. We train $T = 100$ epochs. We use a multi-step scheduler, which decays the learning rate by 0.1 at $0.5T$ and $0.75T$.