Diffusion Mechanism in Residual Neural Network: Theory and Applications

Tangjun Wang, Zehao Dou, Chenglong Bao, and Zuqiang 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.

I. 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 \( \theta \). 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 \( \theta \) 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 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. II-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. VAT [26] replaces stochastic perturbation with the “worst” perturbation which can most significantly affect the output of the prediction function. 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 labeled and unlabeled) 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 diffusion 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 II. Section III presents the formulation and details of our diffusion residual network, and Section IV provides theoretical analysis of diffusion mechanism. Experimental results on various tasks are reported in Section V. We conclude the paper in Section VI.

II. RELATED WORKS

A. 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. Graph Neural Diffusion [47] also uses parabolic diffusion-type PDEs to design GNNs, but they introduce additional attention parameters in each diffusion layer. Additionally, their final output is calculated directly from inputs by performing diffusion, while our diffusion layers are added before each residual block. Thus, we can integrate convection and diffusion in an intrinsic way, theoretically guarantee the improvement in classification accuracy. EPNet [48] uses diffusion on both embedded features and labels to utilize the query set in few-shot learning. However, their
method only contains diffusion, while ours combine convection and diffusion internally. To the best of our knowledge, this is the first work that applies the diffusion mechanism to ResNet with rigorous mathematical analysis.

B. Neural ODEs

The deep learning models and dynamical systems have closed relationship, which is first introduced in a proposal of E. et al. [2]. Using this connection, many works have been proposed for improving deep learning models. [49], [50] propose several training algorithms based on Pontryagin’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. [7] 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.

C. 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, [51] 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. [52], [53] 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 [54], [55]. Semi-supervised embedding [37] extends the Laplacian regularizer from labels to network outputs, which imposes constraints on the parameters of a neural network. Recent explosion in SSL can be traced back to SimCLR [56], which provides a simple framework for contrastive learning of visual representations that can be used on semi-supervised tasks. SwAV [57] both clusters data and enforces consistency between cluster assignments produced for different views of the same image simultaneously to avoid the computation burden of a large number of explicit pairwise feature comparisons in contrastive learning. DINO [58] extends the self-supervised methods to transformer [59] and further improves the performance on SSL benchmarks. These methods are based mostly on contrastive learning and self-supervised learning, and are not directly related to graph-based semi-supervised learning. Few-shot learning, which is a special scenario of semi-supervised learning, may also be integrated with graph by introducing a Gaussian kernel similarity matrix on the embedded features obtained through a pretrained backbone. [60] uses label propagation by building a similarity matrix on both support set and query set. wDAE-GNN [61] introduces a denoising autoencoder that injects Gaussian noise on a set of classification weights as inputs and learns to reconstruct the target classification weights, in order to regularize the weight. Furthermore, they implement the denoising autoencoder as a graph neural network to capture the co-dependencies. EPNet [48] uses both embedding propagation on features and label propagation on labels to utilize the information in unlabeled query set. EASE [62] proposes an assumption that embedded features can be drawn from multiple subspaces, and thus constructs the similarity matrix in a block-diagonal prior. 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.

D. Few-Shot Learning

To address the limited training samples problem, few-shot learning, a new learning paradigm [19], [63], 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 [64] with attention mechanism and external memory to construct the embedding mappings for the query set and the support set. It first introduce the episode-based training method to match the training and testing condition. Prototypical Network [65] 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 [66] 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 [67] 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. [68] 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 [69] adopts a regularizer based on graph Laplacian. [70] rectifies the features to reduce the cross-class and intra-class bias, and use the rectified prototype to help clustering. Several works have been focusing on the distribution of embedded features, e.g., [71], [72] assumes that the features should be drawn from a Gaussian distribution, and thus uses Tukey’s Transformation Ladder [73], also called Power Transform, to calibrate the distribution. The specific setting of current evaluation of few-shot learning performance, which draws exactly the same number of query samples from each class, inspires the application of Sinkhorn algorithm from the field of optimal transport. [72], [74] proposes Sinkhorn k-means, and [62] extends the algorithm to a semi-supervised setting. However, we do not explicitly use the uniform distribution of query labels because we believe it is impractical in real-world tasks. 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.

III. 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.

A. 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

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

(2)

where $\theta^k$ 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 $\Delta t$ 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 $x_i$:

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

(3)

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

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

for all $i = 1, 2, \ldots, N$, 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.,

$$f(x(t), \theta(t)) = \sum_{i=1}^{w} a_i \sigma \left( w_i^t \cdot x(t) + b_i^t \right).$$

(5)

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

B. Algorithm

We discretize the convection-diffusion (4) using the classic Lie-Trotter splitting scheme [75]. After absorbing the time step $\Delta t$ into $f$ and $\gamma$, it leads to

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

(6)

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

(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 $k(x, y) = \exp(-||x-y||^2/\sigma^2)$ to measure the similarity between data points. $\sigma$ is a parameter to adjust the distribution of weight. Next, we introduce two operators, Sparse and Normalize, and one hyperparameter, $\eta_{top}$, 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 $\eta_{top}$ entries and truncate other entries to 0. 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, available online. The step size $\gamma$ 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 $\gamma$. The illustration of our Diff-ResNet can be found in Fig. 1. We summarize our method in Algorithm 1.

**Algorithm 1:** Training Algorithm for Diff-ResNet

1: **Input:** Labeled data points $\{(x_i, y_i)\}_i^{N_1}$. Unlabeled data points $\{x_j\}_j^{N_2}$. Number of blocks $s$. Number of diffusion steps $r$. Step size $\gamma$.
2: **Output:** Trained network parameters $\{\theta_k\}$
3: Construct weight matrix $W$ by $\bar{w}_{ij} = \exp(-\|x_i - x_j\|_2^2/\sigma^2)$ for all $i, j \in [N]$
4: $W = \text{Normalize} (\text{Sparse}(W, n_{top}))$
5: **while** epoch $\leq$ MAX ITER **do**
6: $x_i^0 = x_i(0) = x_i$
7: **for** $k = 0, 1, \ldots, s - 1$ **do**
8: $x_i^{k+1/2} = x_i^k + f(x_i^k, \theta^k)$ **do** Convection Step
9: **for** $m = 0, 1, \ldots, r - 1$ **do**
10: $x_i^{k+1/2} = x_i^{k+1/2} - \gamma \sum_{j=1}^{N_1+N_2} w_{ij} (x_i^{k+1/2} - x_j^{k+1/2})$
**do** Diffusion Step
11: $x_i^{k+1} = x_i^{k+1/2}$
12: $x_i(1) = x_i^s$
13: Feed $x_i(1)$ into a classification layer, compute loss function using $\{y_i\}_i^{N_1}$, back propagate, and update $\{\theta_k\}$ using gradient descent.
14: **end**
15: **end**
16: **end**

**Remark 1:** In diffusion step (7), the feature map of the ith 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.

**IV. 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.

**A. Structured Data Assumption**

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

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

The diameter of a set $A$ is defined as

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

**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: $S_{i,j} \subseteq B(x_0, D/2)$ for some $x_0$, then

$$\text{diam} (S_{i,j}) \leq D.$$ 

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

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
in the ResNet flow
\[ S \in \mathbb{E} \in \{1\} \]
If the Distance-Diameter Ratio is large enough
\[ \text{are similar to inter-class distance} \]
and Gaussian function \[ \sigma \] such that
\[ \text{is called} \]
can be separated by a set of \[ \sigma \]
that lies in \[ \mathbb{E} \]
∀\[ \in \mathbb{E} \text{regions} \]
at time
be a graph, where
\[ \text{tells us that:} \]
as
\[ \text{can} \]
only contains one point and the width
Suppose the data points in each subset
As \( e_t \)
\[ \leq D \][42x475]for each
parallel hyperplanes, i.e., there exists a unique
B. Theoretical Analysis
We present the theoretical results of this paper so as to explain
the role of diffusion mechanism in binary classification. Due to
Due to
\[ \text{dissimilar samples should be far away.} \]
styles, which corresponds to different subclasses and fits to our
dataset, every digit number may have different handwriting
sense, the Structured Data Assumption may be more practical
subclass, corresponding the local intra-class distance. In this
used in our analysis is the upper bound for the points in the
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
\[ \text{diam}(S_{i,j}(t)) \leq D(t), \forall (i,j) \in [k] \times [l], \]
\[ \text{dist}(S_{i,j}(t), S_{i',j'}(t)) \geq L(t), \forall (i_1,j_1) \]
\[ \neq (i_2,j_2) \in [k] \times [l]. \]
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
\[ 2 \text{Time only forms a continuous analogy to the layer index, where each layer} \]
\[ \text{corresponds to forward propagation of the flow.} \]
\[ \text{Without loss of generality, we} \]
\[ \text{assume the final time step is } T = 1. \]

(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
\[ \text{dist}(S_{i_1,j_1}, S_{i_2,j_2}) \geq L. \]
An illustration of the structured data assumption is provided in
Fig. 2. 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.

B. 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, available
online.

Definition A set \( \{(x_i, y_i)\}_{i=1}^{N}, x_i \in \mathbb{R}^d, y_i \in [k] \) is called
linear separable if and only if there exists a hyperplane that
cuts the full space \( \mathbb{R}^d \) 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 \( \theta(t) = \{w^{(i)}, b^{(i)}, a^{(i)}\}_{i=1}^{M} \) in the ResNet flow
\[ f(x(t), \theta(t)) = \sum_{i=1}^{M} a^{(i)}(t) \sigma(w^{(i)} \cdot x(t) + b^{(i)}), \]
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 \( w \) 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
\[ \frac{L}{D} \geq \frac{M(M-1)\sqrt{\pi}}{4}d, \]
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
\[ \frac{dx_i(t)}{dt} = -\gamma \sum_{j=1}^{N} w_{ij} (x_i(t) - x_j(t)) \cdot x_i(t) = x_i, \]
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 connects points among different subsets (2) any two vertices in the same subclass \( S_{i,j} \) are connected to each other. By the construction of weight matrix, each vertex in graph \( G \) is only connected to its \( k_{\text{top}} \) nearest neighbors. Thus the first argument is satisfied when the nearest neighbors only contain points from the same subclass, which requires that \( k_{\text{top}} \) should not be too large. On the other hand, the threshold for the connectivity of a k-nearest neighbor graph is \( O(\log n) \) [76], where \( n \) in our setting should be the number of points in each subclass.

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.

V. 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

A. 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 for more results, available online.

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 III-B, we stack diffusion steps with small step size \( \gamma \) to ensure stability. In Fig. 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 Fig. 3, this diffusion still works well. Data points in same subclass converge to a single point. We also observe from Fig. 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 Fig. 3 is not contradictory to the definition of diffusion. The energy of a point is represented by its coordinate. We expect that neighboring elements 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 1,000 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, available online. During training residual networks with or without diffusion mechanism, we plot the features before the final classification layer in Fig. 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 Fig. 4, features are different from raw input points in (a). The results of circle dataset is shown in Fig. 4.

As shown in Fig. 4, diffusion can reduce the noise. In Fig. 4(f), (g), and (h), the features are very clean while in Fig. 4(i), (g), and (h), features are still noisy. Moreover, diffusion step makes the final feature much easier to separate. In Fig. 4(h), features can be easily separated by a straight line while the features are not linear separable without diffusion as shown in Fig. 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

³Code at https://github.com/shwangtangjun/Diff-ResNet.
parameters can give correct classification which demonstrates that diffusion is very useful in classification problem.

### B. 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, available online. Moreover, rather than using the fixed Planetoid [38] split, we follow [77] and report results for all datasets using 100 random splits with 20 random initializations each.

The mainstream approach in graph learning, such as GCN [78], GraphSAGE [79] and GAT [80], 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 several graph learning methods: three of the most popular architectures, GCN, GraphSAGE (its two variants), GAT, and recent ODE-based GNN architectures, Continuous Graph Neural Networks (CGNN) [81], Graph Neural Ordinary Differential Equations(GDE) [82], and Graph Neural Diffusion(GRAND) [47]. The detailed network structure and parameter settings can be found in Appendix E.2, available online. The classification results are reported in Table I. Diff-ResNet is significantly better than ResNet without diffusion(No-Diff-ResNet). 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 and recent methods in graph learning. Thus, we propose an alternative path for semi-supervised graph learning problems.

Additionally, it is reported that methods based on aggregation of neighboring information suffers over-smoothing problems with increasing depth [56], [83]. As is observed in Fig. 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,

---

**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)–(e) are features before the final classification layer without diffusion from different epochs. (f)–(h) are features with diffusion.

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

---

**TABLE I**

|                | Cora       | Citeseer   | Pubmed     |
|----------------|------------|------------|------------|
| MLP            | 58.2 ± 2.1 | 59.1 ± 2.3 | 70.0 ± 2.1 |
| GCN [78]       | 81.5 ± 1.3 | 71.9 ± 1.9 | 77.8 ± 2.9 |
| GraphSAGE-mean [79] | 79.2 ± 2.2 | 71.6 ± 1.9 | 77.4 ± 2.2 |
| GraphSAGE-maxpool [79] | 76.8 ± 1.9 | 67.5 ± 2.3 | 76.1 ± 2.3 |
| GAT [80]       | 81.8 ± 1.3 | 71.4 ± 1.9 | 78.7 ± 2.3 |
| CGNN [81]      | 81.4 ± 1.6 | 66.9 ± 1.8 | 66.6 ± 4.4 |
| GDE [82]       | 78.7 ± 2.2 | 71.8 ± 1.1 | 73.9 ± 3.7 |
| GRAND [47]     | 83.6 ± 1.0 | 73.4 ± 0.5 | 78.8 ± 1.7 |
| No-Diff-ResNet | 58.9 ± 1.9 | 61.9 ± 2.1 | 70.1 ± 2.1 |
| Diff-ResNet(ours) | 82.1 ± 1.1 | 74.6 ± 1.6 | 80.1 ± 2.0 |

---

**Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.**
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.

C. Few-Shot Learning

Given a dataset \( X = X_s \cup X_q \), where \( X_s = \{ (x_i, y_i) \}_{i=1}^{N_s} \) is the support set with label information and \( X_q = \{ x_j \}_{j=1}^{N_q} \) 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 \( |N_s| \) 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: miniImageNet, tieredImageNet and CUB. The miniImageNet and tieredImageNet are both subsets of the larger ILSVRC-12 dataset [84], with 100 classes and 608 classes respectively. CUB-200-2011 [85] is another fine-grained image classification dataset with 200 classes. We follow the standard dataset split as in previous papers [68], [86], [87]. All images are resized to 84 \times 84, following [20].

We choose two widely used networks, ResNet-18 [1] and WRN-28-10 [88] 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 with label smoothing factor of 0.1, 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 miniImageNet and tieredImageNet, and \( T = 400 \) epochs for CUB due to its small size. We use a multi-step scheduler, which decays the learning rate by 0.1 at 0.5 \( T \) and 0.75 \( T \). 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 [68] and LaplacianShot [69], 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 \( X^c_s \),

\[
m_c = \frac{1}{|X^c_s|} \sum_{x \in X^c_s} x.
\]

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 [69], the author optimizes the loss function below

\[
\mathcal{L} = \sum_{i=1}^{N_2} \sum_{c=1}^{C} y_{i,c} d(x_i - m_c) + \lambda \sum_{i,j=1}^{N_2} w(x_i, x_j) \| y_i - y_j \|^2,
\]

\( N_2 = |X_q| \) is the number of query samples. \( y_i = [y_{i,1}, \ldots, y_{i,C}] \in \{0, 1\}^C \) is in the \( C \)-dimensional simplex, which assigns label to each query point. \( d \) is euclidean distance. \( w(x_i, x_j) \) is the weight between \( x_i \) and \( x_j \).

The first loss term functions similar to nearest prototype classification. The second loss term is the well-known Laplacian regularizer, which we use the iterative algorithm provided by LaplacianShot [69] to minimize the objective function. 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

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

on the support set using gradient descent and a simple 2-layer residual network \( f \). \( N_1 = |X_s| \) is the number of support samples. The detailed network structure can be found in Appendix E.3.2, available online. 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

![T-SNE visualization of features before and after diffusion steps. Stars represent labeled points (support data). Circles represent unlabeled points (query data).](image-url)
We minimize cross entropy loss
\[
    L = - \sum_{i=1}^{N_x} \sum_{c=1}^C y_{i,c} \log (f(x_i)_c) + \frac{\mu}{2} \sum_{i,j=1}^N w(x_i, x_j) \| f(x_i) - f(x_j) \|^2,
\]

to the loss function. \( N = N_s + N_q = |X_s \cup X_q | \) 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
\[
    L = - \sum_{i=1}^{N_x} \sum_{c=1}^C y_{i,c} \log (f(x_i)_c),
\]
on the support set using gradient descent and simple 2-layer residual network. The difference from (3) is that we add a Laplacian regularizer

\[
    \sum_{i=1}^{N_x} \sum_{c=1}^C y_{i,c} \log (f(x_i)_c)
\]

and adjust \( \mu \) to visualize the features before and after it should be neither too small nor too large. \( \mu \) is chosen to be the mean \( \frac{1}{k} \sum_{i=1}^k f(x_i) \) to the test classes, with 15 query samples in each class, and report the average accuracy of 5 methods above in Table II. 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 [89] to visualize the features before and after diffusion in 1-shot and 5-shot task in Fig. 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 \( \gamma \). We conduct experiments on 1,000 5-way-1-shot tasks on miniImageNet 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 \( \sigma = \lfloor \frac{n_{top}}{2} \rfloor \). The results are depicted in Fig. 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 \( r \), ranging from 0 to 20. The results are shown in Fig. 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.

\[4\sigma(x_i) = k \text{ means } \sigma \text{ is chosen to be the } k\text{th closest distance from a specific point } x_i, \text{ so it varies with points. \} \]
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 $\gamma \leq 1$ for stability. When the total diffusion strength is too large when $r < 5$, we set $\gamma = 1.0$. As shown in Fig. 9, when the total diffusion strength is fixed, the accuracy almost keeps the same. Thus, stacking too many layers will not benefit.

To conclude, the performance of our Diff-ResNet mostly depend on the total diffusion strength $r\gamma$. With fixed strength, the number of diffusion layers $r$ and the truncation parameter $n_{\text{top}}$ has little effect on performance. We have briefly discussed the choice of $n_{\text{top}}$ in Section IV-B after Proposition 1, and state that a neither too large nor too small $n_{\text{top}}$ is better. Rather than cherry-picking $n_{\text{top}}$ according to different datasets or backbones, we fix $n_{\text{top}} = 8$ and $\sigma = 4$ in all few-shot learning experiments, which shows that a moderate $n_{\text{top}}$ is sufficient for good results. As for the number of diffusion layers $r$, the choice is made by fixing the diffusion step size $\gamma = 0.5$, and choose the best diffusion strength on the validation set. From our experiment results, we observe that with fixed step size $\gamma$, the number of layers $r$ should be larger for 1-shot tasks, and smaller for 5-shot tasks. Also, the optimal diffusion strength varies with dataset, since the geometric property of each dataset is different. The detailed parameter choice is provided in Appendix E.3, available online.

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 [69], which we elaborate in Appendix E.3, available online. We also remove the balanced class assumption in [62], [72] and report the corresponding results, as our Diff-ResNet does not utilize such assumption. We randomly sample 10,000 5-way-1-shot and 5-way-5-shot classification tasks and report the average accuracy and corresponding 95% confidence interval in Table III. The results of networks for comparison in Table III are collected from [68], [69], [87]. In all datasets with various backbones, Diff-ResNet obtains the highest classification accuracy, except on 5-shot task on miniImageNet with WRN-28-10 as backbone. The improvement of Diff-ResNet on 5-shot tasks is not as significant as on 1-shot tasks, which is consistent with the observation in Table II. As for the relatively small performance increase compared to LaplacianShot [69], we have thoroughly investigated the difference between our method and theirs in the ablation study II (Internal Convection-Diffusion versus Diffusion).

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 1,000 classification tasks using Diff-ResNet with different number of diffusion layers $r$, and report the average computation time per task and accuracy in Fig. 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 Fig. 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.
TABLE III

| Methods    | Backbone | $\text{miniImageNet}$ | $\text{tieredImageNet}$ | CUB |
|------------|----------|------------------------|--------------------------|-----|
|            |          | 1-shot ($\pm 0.00$)    | 5-shot ($\pm 0.00$)      |     |
| MAML [6]   | ResNet-18| 49.61 ± 0.92           | 62.47 ± 0.47             |     |
| Baseline [67] | ResNet-18| 51.87 ± 0.77           | 75.66 ± 0.63             |     |
| RelationNet [66] | ResNet-18| 52.48 ± 0.86           | 69.83 ± 0.68             |     |
| MatchingNet [20] | ResNet-18| 52.91 ± 0.88           | 68.88 ± 0.69             |     |
| ProtoNet [65] | ResNet-18| 54.16 ± 0.82           | 73.68 ± 0.65             |     |
| Cifar [91] | ResNet-15| 55.45 ± 0.89           | 70.13 ± 0.68             |     |
| SNAIL [92] | ResNet-15| 55.71 ± 0.99           | 68.82 ± 0.68             |     |
| TADAM [67] | ResNet-15| 57.50 ± 0.50           | 76.70 ± 0.30             |     |
| Transductive [93] | ResNet-12| 62.85 ± 0.66           | 74.53 ± 0.54             |     |
| Mepochnet [94] | ResNet-12| 62.84 ± 0.61           | 78.63 ± 0.46             |     |
| TPN [95] | ResNet-12| 53.75 ± 0.86           | 69.43 ± 0.67             |     |
| TEAM [96] | ResNet-18| 60.07 ± 0.59           | 75.90 ± 0.38             |     |
| CAN+T [97] | ResNet-12| 67.19 ± 0.55           | 80.64 ± 0.35             |     |
| *SimpleShot [68] | ResNet-18| 62.86 ± 0.20           | 79.22 ± 0.14             |     |
| *LaplacianShot [69] | ResNet-18| 70.46 ± 0.23           | 81.76 ± 0.14             |     |
| *EPNet [48] | ResNet-18| 63.83 ± 0.20           | 77.98 ± 0.15             |     |
| EASE-Soft K means [62] | ResNet-12| 57.00 ± 0.26           | 75.07 ± 0.21             |     |
| Diff-ResNet/ours [62] | ResNet-12| 71.11 ± 0.24           | 82.08 ± 0.14             |     |

Qiao [98] WRN 59.60 ± 0.41 73.74 ± 0.19 -
LEO [99] WRN 61.76 ± 0.08 77.59 ± 0.12 66.33 ± 0.05 81.44 ± 0.09 -
ProtoNet [65] WRN 62.60 ± 0.20 79.97 ± 0.14 -
CC-rot [100] WRN 62.93 ± 0.45 79.87 ± 0.33 -
MatchingNet [20] WRN 64.03 ± 0.20 76.12 ± 0.16 -
FEAT [101] WRN 65.10 ± 0.20 81.11 ± 0.14 70.41 ± 0.23 83.48 ± 0.16 -
Transductive [93] WRN 65.73 ± 0.68 78.40 ± 0.52 73.34 ± 0.71 85.50 ± 0.30 -
BD-CSFN [70] WRN 70.31 ± 0.93 81.89 ± 0.60 78.74 ± 0.95 86.92 ± 0.63 -
*SimpleShot [68] WRN 63.20 ± 0.20 81.28 ± 0.14 71.49 ± 0.23 83.51 ± 0.16 78.62 ± 0.19 91.21 ± 0.30 -
*LaplacianShot [69] WRN 72.90 ± 0.23 83.47 ± 0.14 78.79 ± 0.25 86.86 ± 0.17 87.70 ± 0.18 92.73 ± 0.10 -
*EPNet [48] WRN 67.09 ± 0.21 80.71 ± 0.14 73.20 ± 0.23 84.20 ± 0.17 80.88 ± 0.20 91.40 ± 0.11 -
PT+INCM [72] WRN 65.35 ± 0.20 83.87 ± 0.13 69.96 ± 0.22 86.45 ± 0.15 80.57 ± 0.20 91.15 ± 0.10 -
EASE-Soft K means [62] WRN 67.42 ± 0.27 84.45 ± 0.18 75.87 ± 0.29 85.17 ± 0.21 81.01 ± 0.26 91.44 ± 0.14 -
Diff-ResNet/ours [62] WRN 73.47 ± 0.23 83.86 ± 0.14 79.74 ± 0.25 87.19 ± 0.16 87.74 ± 0.19 92.96 ± 0.09 -

We mark transductive learning method with t. * denotes that the result is implemented using public official code with our pretrained backbone.

However, compared to methods [48], [68], [69], our method requires more time. In each few-shot task, at the inference phase, our method need to train a tiny neural network. Notice that we are not trying to retrain or finetune the backbone network. Rather, we train a tiny network that only contains two hidden layers and a diffusion block. The acceleration of inference will be studied in future.

VI. 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 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. Our future work involves the robustness of diffusion mechanism, the acceleration of Diff-ResNet in inference phase of few-shot learning, the extension from diffusive ODE to diffusive PDE, and the effect of diffusion in semi-supervised learning with extremely low label rate.

REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 770–778.

[2] E. Weinan, “A proposal on machine learning via dynamical systems,” Commun. Math. Statist., 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 Proc. Adv. Neural Inf. Process. Syst., 2018, pp. 6571–6583.

[4] E. Haber and L. Ruthotto, “Stable architectures for deep neural networks,” Inverse Problems, vol. 34, no. 1, Art. no. 014004.

[5] Z. Yang, Z. Li, C. Change Loy, and D. Lin, “PolyNet: A pursuit of structural diversity in very deep networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 718–726.

[6] G. Larsson, M. Maire, and G. Shakhnarovich, “FractalNet: Ultra-deep neural networks without residuals,” 2016, arXiv:1605.07648.

[7] Y. Lu, A. Zhong, Q. Li, and B. Dong, “Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations,” in Proc. Int. Conf. Mach. Learn., PMLR, 2018, pp. 3276–3285.

[8] X. Gastaldi, “Shake-shake regularization,” 2017, arXiv:1705.07485.

[9] 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 Trans. Neural Netw., vol. 20, no. 8, pp. 10736–10745.

[10] I. E. Lagaris, A. Likas, and D. I. Fotiadis, “Artificial neural networks for solving ordinary and partial differential equations,” IEEE Trans. Neural Netw., vol. 9, no. 5, pp. 987–1000, Sep. 1998.

[11] M. Dissanayake and N. Phan-Thien, “Neural-network-based approximations for solving partial differential equations,” Commun. Numer. Methods Eng., vol. 10, no. 3, pp. 195–201, 1994.

[12] I. E. Lagaris, A. C. Likas, and D. G. Papageorgiou, “Neural-network methods for boundary value problems with irregular boundaries,” IEEE Trans. Neural Netw., vol. 11, no. 5, pp. 1041–1049, Sep. 2000.

[13] R. S. McFall and J. R. Mahan, “Artificial neural network method for solution of boundary value problems with exact satisfaction of arbitrary boundary conditions,” IEEE Trans. Neural Netw., vol. 20, no. 8, pp. 1211–1233, Aug. 2009.
[15] M. Bayram, A. Kerayechian, and S. Effati, “Artificial neural networks approach for solving stokes problem,” Appl. Math., vol. 1, no. 4, pp. 288–292, 2010.

[16] J. Han, A. Jentzen, and E. Weinan, “Solving high-dimensional partial differential equations using deep learning,” in Proc. Nat. Acad. Sci. USA, vol. 115, no. 34, pp. 8505–8510, 2018.

[17] X. Zhu and A. B. Goldberg, “Introduction to semi-supervised learning,” Synth. Lectures Artif. Intell. Mach. Learn., vol. 3, no. 1, pp. 1–130, 2009.

[18] O. Chapelle, B. Schölkopf, and A. Zien, “Semi-supervised learning,” in Proc. 16th Eur. Conf. Comput. Vis., Glasgow, U.K., Springer, 2020, pp. 121–138.

[19] Q. Li, L. Chen, C. Tai, and E. Weinan, “Maximum principle based algorithms for deep learning,” J. Mach. Learn. Res., vol. 18, no. 165, pp. 1–29, 2017.

[20] O. Vinyals et al., “Matching networks for one shot learning,” in Proc. Adv. Neural Inf. Process. Syst., 2016, pp. 3630–3638.

[21] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, “Generalizing from a few examples: A survey on few-shot learning,” ACM Comput. Surv., 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,” in Proc. Adv. Neural Inf. Process. Syst., 2018, pp. 3239–3240.

[23] P. Bachman, O. Alsharif, and D. Precup, “Learning with pseudo-ensembles,” in Proc. Adv. Neural Inf. Process. Syst., 2014, pp. 3365–3373.

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

[25] A. Tarvainen and H. Valpola, “Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results,” in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 1959–1965.

[26] T. Miyato, S.-I. Maeda, M. Koyama, and S. Ishii, “Virtual adversarial training: A regularization method for supervised and semi-supervised learning,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 41, no. 8, pp. 1979–1993, Aug. 2019.

[27] Y. Grandvalet and Y. Bengio, “Semi-supervised learning by entropy minimization,” in Proc. Adv. Neural Inf. Process. Syst., 2004, pp. 529–536.

[28] D.-H. Lee et al., “Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks,” in Proc. Workshop Challenges Representation Learn., 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,” in Proc. Adv. Neural Inf. Process. Syst., 2019, Art. no. 454.

[30] K. Sohn et al., “FixMatch: Simplifying semi-supervised learning with consistency and confidence,” in Proc. Adv. Neural Inf. Process. Syst., 2020, pp. 596–608.

[31] K. W. Morton, Numerical Solution of Convection-Diffusion Problems. Boca Raton, FL, USA: CRC, 2019.

[32] Z. Svoboda, “The convective-diffusion equation and its use in building physics,” Int. J. Architectural Sci., 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,” J. Differ. Equ., 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 Proc. 20th Int. Conf. Mach. Learn., 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,” in Proc. Adv. Neural Inf. Process. Syst., 2009, pp. 1330–1338.

[36] Z. Shi, S. Osher, and W. Zhu, “Weighted nonlocal Laplacian on interpolation from sparse data,” J. Sci. Comput., vol. 73, no. 2, pp. 1164–1177, 2017.

[37] J. Weston, F. Ratle, H. Mobahi, and R. Collobert, “Deep learning via graph convolutional networks,” in Proc. Adv. Inf. Process. Syst., 2018, pp. 8157–8166.

[38] J. Klicpera, S. Weißenberger, and S. Günnemann, “Diffusion improves graph learning,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2019, pp. 13366–13378.

[39] J. Arwood and D. Towsley, “Diffusion-convolutional neural networks,” in Proc. Adv. Neural Inf. Process. Syst., 2016, pp. 1993–2001.

[40] R. R. Coifman et al., “Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps,” in Proc. Nat. Acad. Sci. USA, vol. 102, no. 21, pp. 7426–7431, 2005.

[41] M. Belkin and P. Niyogi, “Laplacian eigenmaps for dimensionality reduction and data representation,” Neurocomput., vol. 15, no. 6, pp. 1373–1396, 2003.

[42] D. Kushnir and L. Venturi, “Diffusion-based deep active learning,” 2020, arXiv:2003.10339.

[43] M. Belkin, P. Niyogi, and V. Sindhwani, “Manifold regularization: A geometric framework for learning from labeled and unlabeled examples,” J. Mach. Learn. Res., vol. 7, no. 11, pp. 2399–2434, 2006.

[44] R. K. Ando and T. Zhang, “Learning on graph with Laplacian regularization,” in Proc. Adv. Neural Inf. Process. Syst., 2007, pp. 25–32.

[45] A. J. Smola and R. Kondor, “Kernels and regularization on graphs,” in Proc. Learn. Theory Kernel Machines, Springer, 2003, pp. 144–158.

[46] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” in Proc. Int. Conf. Mach. Learn., PMLR, 2020, pp. 1597–1607.

[47] M. Belkin, P. Niyogi, and V. Sindhwani, “Manifold regularization: A geometric framework for learning from labeled and unlabeled examples,” J. Mach. Learn. Res., vol. 7, no. 11, pp. 2399–2434, 2006.

[48] M. Caron et al., “Emerging properties in self-supervised vision transformers,” in Proc. IEEE/CVF Int. Conf. Comput. Vis., 2021, pp. 9650–9660.

[49] A. Dosovitskiy et al., “An image is worth 16x16 words: Transformers for image recognition at scale,” in Proc. Int. Conf. Learn. Representations, 2021, [Online]. Available: https://openreview.net/forum?id=YicbFt9NTY

[50] V. G. Satorras and J. B. Estrach, “Few-shot learning with graph neural networks,” in Proc. Int. Conf. Learn. Representations, 2018, [Online]. Available: https://openreview.net/forum?id=BJfQg6HRW

[51] S. Girardis and N. Komodakis, “Generating classification weights with GNN denoising autoencoders for few-shot learning,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2019, pp. 21–30.

[52] H. Zhu and P. Koniusz, “EASE: Unsupervised discriminant subspace learning for transductive few-shot learning,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2022, pp. 9078–9088.

[53] E. G. Miller, N. E. Matsakis, and P. A. Viola, “Learning from one example,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. Pattern Recognit., 2019, pp. 912–919.

[54] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Comput., vol. 9, no. 8, pp. 1735–1780, 1997.

[55] J. Snell, K. Swersky, and R. S. Zemel, “Prototypical networks for few-shot learning,” 2017, arXiv:1703.05175.

[56] F. Song, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales, “Learning to compare: Relation network for few-shot learning,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 1199–1208.

[57] B. N. Oreshkin, P. Rodriguez, and A. Lacoste, “TADAM: Task dependent adaptive metric for improved few-shot learning,” 2018, arXiv:1805.10123.

[58] Y. Wang, W.-L. Chao, K. Q. Weinberger, and L. van der Maaten, “SimpleShot: Revisiting nearest-neighbor classification for few-shot learning,” 2019, arXiv:1911.04625.
L. Qiao, Y. Shi, J. Li, Y. Wang, T. Huang, and Y. Tian, “Transductive episodic-wise adaptive metric for few-shot learning,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 3603–3612.

R. Hou, H. Chang, B. Ma, S. Shan, and X. Chen, “Cross attention network for few-shot classification,” in *Proc. 35th Int. Conf. Neural Inf. Process. Syst.*, 2019, Art. no. 360.

S. Qiao, C. Liu, W. Shen, and A. L. Yuille, “Few-shot image recognition by predicting parameters from activations,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7229–7238.

A. A. Rusu et al., “Meta-learning with latent embedding optimization,” in *Proc. Int. Conf. Learn. Representations*. 2018. [Online]. Available: https://openreview.net/forum?id=BjkhlAcK7

S. Gidaris, A. Bursuc, N. Komodakis, P. Pérez, and M. Cord, “Boosting few-shot visual learning with self-supervision,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 8059–8068.

H.-J. Ye, H. Hu, D.-C. Zhan, and F. Sha, “Few-shot learning via embedding adaptation with set-to-set functions,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 8808–8817.

S. Li, “Concise formulas for the area and volume of a hyperspherical cap,” *Asian J. Math. Statist.*, vol. 4, no. 1, pp. 66–70, 2011.

F. R. Chung and F. C. Graham, *Spectral Graph Theory*. Providence, RI, USA: Amer. Math. Soc., 1997.

Tangjun Wang is currently working toward the PhD degree with the Department of Mathematical Sciences, Tsinghua University supervised by Zuqiang Shi. His research interests include semi-supervised learning and applications of partial differential equation in neural networks.

Zehao Dou is currently working toward the PhD degree with the 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 received the PhD degree from the Department of Mathematics, National University of Singapore, in 2014. He is an assistant professor with Yau Mathematical Sciences Center, Tsinghua University and Yanzhi Lake Beijing Institute of Mathematical Sciences and Applications. His main research interests include mathematical image processing, large scale optimization and its applications.

Zuoqiang Shi received the PhD degree from Zhou Pei-Yuan Center for Applied Mathematics, Tsinghua University in 2008. He is a professor in Yau Mathematical Sciences Center, Tsinghua University. His main research interests include numerical methods of partial differential equations and its applications, mathematical image processing.