# Edge-Labeling Graph Neural Network for Few-shot Learning

Jongmin Kim[*1,3], Taesup Kim[2,3], Sungwoong Kim[3], and Chang D.Yoo[1]

[1]Korea Advanced Institute of Science and Technology
[2]MILA, Université de Montréal
[3]Kakao Brain

## Abstract

*In this paper, we propose a novel edge-labeling graph neural network (EGNN), which adapts a deep neural network on the edge-labeling graph, for few-shot learning. The previous graph neural network (GNN) approaches in few-shot learning have been based on the node-labeling framework, which implicitly models the intra-cluster similarity and the inter-cluster dissimilarity. In contrast, the proposed EGNN learns to predict the edge-labels rather than the node-labels on the graph that enables the evolution of an explicit clustering by iteratively updating the edge-labels with direct exploitation of both intra-cluster similarity and the inter-cluster dissimilarity. It is also well suited for performing on various numbers of classes without re-training, and can be easily extended to perform a transductive inference. The parameters of the EGNN are learned by episodic training with an edge-labeling loss to obtain a well-generalizable model for unseen low-data problem. On both of the supervised and semi-supervised few-shot image classification tasks with two benchmark datasets, the proposed EGNN significantly improves the performances over the existing GNNs.*

## 1. Introduction

A lot of interest in meta-learning [1] has been recently arisen in various areas including especially task-generalization problems such as few-shot learning [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15], learn-to-learn [16, 17, 18], non-stationary reinforcement learning[19, 20, 21], and continual learning [22, 23]. Among these meta-learning problems, few-shot leaning aims to automatically and efficiently solve new tasks with few labeled data based on knowledge obtained from previous experiences. This is in
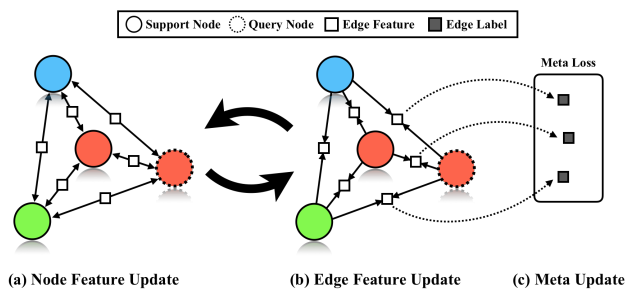


Figure 1: Alternative node and edge feature update in EGNN with edge-labeling for few-shot learning

contrast to traditional (deep) learning methods that highly rely on large amounts of labeled data and cumbersome manual tuning to solve a single task.

Recently, there has also been growing interest in graph neural networks (GNNs) to handle rich relational structures on data with deep neural networks [24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34]. GNNs iteratively perform a feature aggregation from neighbors by message passing, and therefore can express complex interactions among data instances. Since few-shot learning algorithms have shown to require full exploitation of the relationships between a support set and a query [2, 3, 5, 10, 11], the use of GNNs can naturally have the great potential to solve the few-shot learning problem. A few approaches that have explored GNNs for few-shot learning have been recently proposed [6, 12]. Specifically, given a new task with its few-shot support set, Garcia and Bruna [6] proposed to first construct a graph where all examples of the support set and a query are densely connected. Each input node is represented by the embedding feature (e.g. an output of a convolutional neural network) and the given label information (e.g. one-hot encoded label). Then, it classifies the unlabeled query by iteratively updating node features from neighborhood aggregation. Liu et al. [12] proposed a transductive propagation network (TPN) on the node features obtained from a deep neu-

---
[*]Work done during an internship at Kakao Brain. Correspondence to
kimjm0309@gmail.com

ral network. At test-time, it iteratively propagates one-hot encoded labels over the entire support and query instances as a whole with a common graph parameter set. Here, it is noted that the above previous GNN approaches in few-shot learning have been mainly based on the node-labeling framework, which *implicitly* models the intra-cluster similarity and inter-cluster dissimilarity.

On the contrary, the edge-labeling framework is able to *explicitly* perform the clustering with representation learning and metric learning, and thus it is intuitively a more conducive framework for inferring a query association to an existing support clusters. Furthermore, it does not require the pre-specified number of clusters (e.g. class-cardinality or ways) while the node-labeling framework has to separately train the models according to each number of clusters. The explicit utilization of edge-labeling which indicates whether the associated two nodes belong to the same cluster (class) have been previously adapted in the naive (hyper) graphs for correlation clustering [35] and the GNNs for citation networks or dynamical systems [36, 37], but never applied to a graph for few-shot learning. Therefore, in this paper, we propose an edge-labeling GNN (EGNN) for few-shot learning, especially on the task of few-shot classification.

The proposed EGNN consists of a number of layers in which each layer is composed of a node-update block and an edge-update block. Specifically, across layers, the EGNN not only updates the node features but also explicitly adjusts the edge features, which reflect the edge-labels of the two connected node pairs and directly exploit both the intra-cluster similarity and inter-cluster dissimilarity. As shown in Figure 1, after a number of alternative node and edge feature updates, the edge-label prediction can be obtained from the final edge feature. The edge loss is then computed to update the parameters of EGNN with a well-known meta-learning strategy, called episodic training [2, 9]. The EGNN is naturally able to perform a transductive inference to predict all test (query) samples at once as a whole, and this has shown more robust predictions in most cases when a few labeled training samples are provided. In addition, the edge-labeling framework in the EGNN enables to handle various numbers of classes without remodeling or retraining. We will show by means of experimental results on two benchmark few-shot image classification datasets that the EGNN outperforms other few-shot learning algorithms including the existing GNNs in both supervised and semi-supervised cases.

Our main contributions can be summarized as follows:

- The EGNN is first proposed for few-shot learning with iteratively updating edge-labels with exploitation of both intra-cluster similarity and inter-cluster dissimilarity. It is also able to be well suited for performing on various numbers of classes without retraining.

- It consists of a number of layers in which each layer is composed of a node-update block and an edge-update block where the corresponding parameters are estimated under the episodic training framework.

- Both of the transductive and non-transductive learning or inference are investigated with the proposed EGNN.

- On both of the supervised and semi-supervised few-shot image classification tasks with two benchmark datasets, the proposed EGNN significantly improves the performances over the existing GNNs. Additionally, several ablation experiments show the benefits from the explicit clustering as well as the separate utilization of intra-cluster similarity and inter-cluster dissimilarity.

## 2. Related works

**Graph Neural Network** Graph neural networks were first proposed to directly process graph structured data with neural networks as of form of recurrent neural networks [28, 29]. Li et al. [31] further extended it with gated recurrent units and modern optimization techniques. Graph neural networks mainly do representation learning with a neighborhood aggregation framework that the node features are computed by recursively aggregating and transforming features of neighboring nodes. Generalized convolution based propagation rules also have been directly applied to graphs [34, 38, 39], and Kipf and Welling [30] especially applied it to semi-supervised learning on graph-structured data with scalability. A few approaches [6, 12] have explored GNNs for few-shot learning and are based on the node-labeling framework.

**Edge-Labeling Graph** Correlation clustering (CC) is a graph-partitioning algorithm [40] that infers the edge labels of the graph by simultaneously maximizing intra-cluster similarity and inter-cluster dissimilarity. Finley and Joachims [41] considered a framework that uses structured support vector machine in CC for noun-phrase clustering and news article clustering. Taskar [42] derived a max-margin formulation for learning the edge scores in CC for producing two different segmentations of a single image. Kim et al. [35] explored a higher-order CC over a hypergraph for task-specific image segmentation. The attention mechanism in a graph attention network has recently extended to incorporate real-valued edge features that are adaptive to both the local contents and the global layers for modeling citation networks [36]. Kipf et al. [37] introduced a method to simultaneously infer relational structure with interpretable edge types while learning the dynamical model of an interacting system. Johnson [43] introduced the Gated Graph Transformer Neural Network (GGT-NN) for

natural language tasks, where multiple edge types and several graph transformation operations including node state update, propagation and edge update are considered.

**Few-Shot Learning** One main stream approach for few-shot image classification is based on representation learning and does prediction by using nearest-neighbor according to similarity between representations. The similarity can be a simple distance function such as cosine or Euclidean distance. A Siamese network [44] works in a pairwise manner using trainable weighted $L_1$ distance. A matching network [2] further uses an attention mechanism to derive an differentiable nearest-neighbor classifier and a prototypical network [3] extends it with defining prototypes as the mean of embedded support examples for each class. DEML [45] has introduced a concept learner to extract high-level concept by using a large-scale auxiliary labeled dataset showing that a good representation is an important component to improve the performance of few-shot image classification.

A meta-learner that learns to optimize model parameters extract some transferable knowledge between tasks to leverage in the context of few-shot learning. Meta-LSTM [8] uses LSTM as a model updater and treats the model parameters as its hidden states. This allows to learn the initial values of parameters and update the parameters by reading few-shot examples. MAML [4] learns only the initial values of parameters and simply uses SGD. It is a model agnostic approach, applicable to both supervised and reinforcement learning tasks. Reptile [46] is similar to MAML but using only first-order gradients. Another generic meta-learner, SNAIL [10], is with a novel combination of temporal convolutions and soft attention to learn an optimal learning strategy.

## 3. Method

In this section, the definition of few-shot classification task is introduced, and the proposed algorithm is described in detail.

### 3.1. Problem definition: Few-shot classification

The few-shot classification aims to learn a classifier when only a few training samples per each class are given. Therefore, each few-shot classification *task* $\mathcal{T}$ contains a *support set* $\mathcal{S}$, a labeled set of input-label pairs, and a *query set* $\mathcal{Q}$, an unlabeled set on which the learned classifier is evaluated. If the support set $\mathcal{S}$ contains $K$ labeled samples for each of $N$ unique classes, the problem is called *N-way K-shot* classification problem.

Recently, meta-learning has become a standard methodology to tackle few-shot classification. In principle, we can train a classifier to assign a class label to each query sample with only the compact support set of the task. However, a small number of labeled support samples for each

task are not sufficient to train a model fully reflecting the inter- and intra-class variations, which often leads to unsatisfactory classification performance. Meta-learning on explicit training set resolves this issue by extracting transferable knowledge that allows us to perform better few-shot learning on the support set, and thus classify the query set more successfully.

As an efficient way of meta-learning, we adopt *episodic training* [2, 9] which is commonly employed in various literatures [3, 4, 5]. Given a relatively large labeled training dataset, the idea of episodic training is to sample *training tasks (episodes)* that mimic the few-shot learning setting of test tasks. Here, since the distribution of training tasks is assumed to be similar to that of test tasks, the performances of the test tasks can be improved by learning a model to work well on the training tasks.

More concretely, in episodic training, both training and test tasks of the $N$-way $K$-shot problem are formed as follows: $\mathcal{T} = \mathcal{S} \bigcup \mathcal{Q}$ where $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N \times K}$ and $\mathcal{Q} = \{(\mathbf{x}_i, y_i)\}_{i=N \times K+1}^{N \times K+T}$. Here, $T$ is the number of query samples, and $\mathbf{x}_i$ and $y_i \in \{C_1, \cdots C_N\} = \mathcal{C}_\mathcal{T} \subset \mathcal{C}$ are the $i$th input data and its label, respectively. $\mathcal{C}$ is the set of all classes of either training or test dataset. Although both the training and test tasks are sampled from the common task distribution, the label spaces are mutually exclusive, i.e. $\mathcal{C}_{train} \cap \mathcal{C}_{test} = \emptyset$. The support set $\mathcal{S}$ in each episode serves as the labeled training set on which the model is trained to minimize the loss of its predictions over the query set $\mathcal{Q}$. This training procedure is iteratively carried out episode by episode until convergence.

Finally, if some of $N \times K$ support samples are unlabeled, the problem is referred to as *semi-supervised* few-shot classification. In Section 4, the effectiveness of our algorithm on semi-supervised setting will be presented.

### 3.2. Model

This section describes the proposed EGNN for few-shot classification, as illustrated in Figure 2. Given the feature representations (extracted from a jointly trained convolutional neural network) of all samples of the target task, a fully-connected graph is initially constructed where each node represents each sample, and each edge represents the types of relationship between the two connected nodes; Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}; \mathcal{T})$ be the graph constructed with samples from the task $\mathcal{T}$, where $\mathcal{V} := \{V_i\}_{i=1,...,|\mathcal{T}|}$ and $\mathcal{E} := \{E_{ij}\}_{i,j=1,...,|\mathcal{T}|}$ denote the set of nodes and edges of the graph, respectively. Let $\mathbf{v}_i$ and $\mathbf{e}_{ij}$ be the node feature of $V_i$ and the edge feature of $E_{ij}$, respectively. $|\mathcal{T}| = N \times K + T$ is the total number of samples in the task $\mathcal{T}$. Each ground-truth edge-label $y_{ij}$ is defined by the ground-truth node labels as:

$$y_{ij} = \begin{cases} 1, & \text{if } y_i = y_j, \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

**(a) Graph Construction**

**(b-1) EGNN: Node Feature Update**

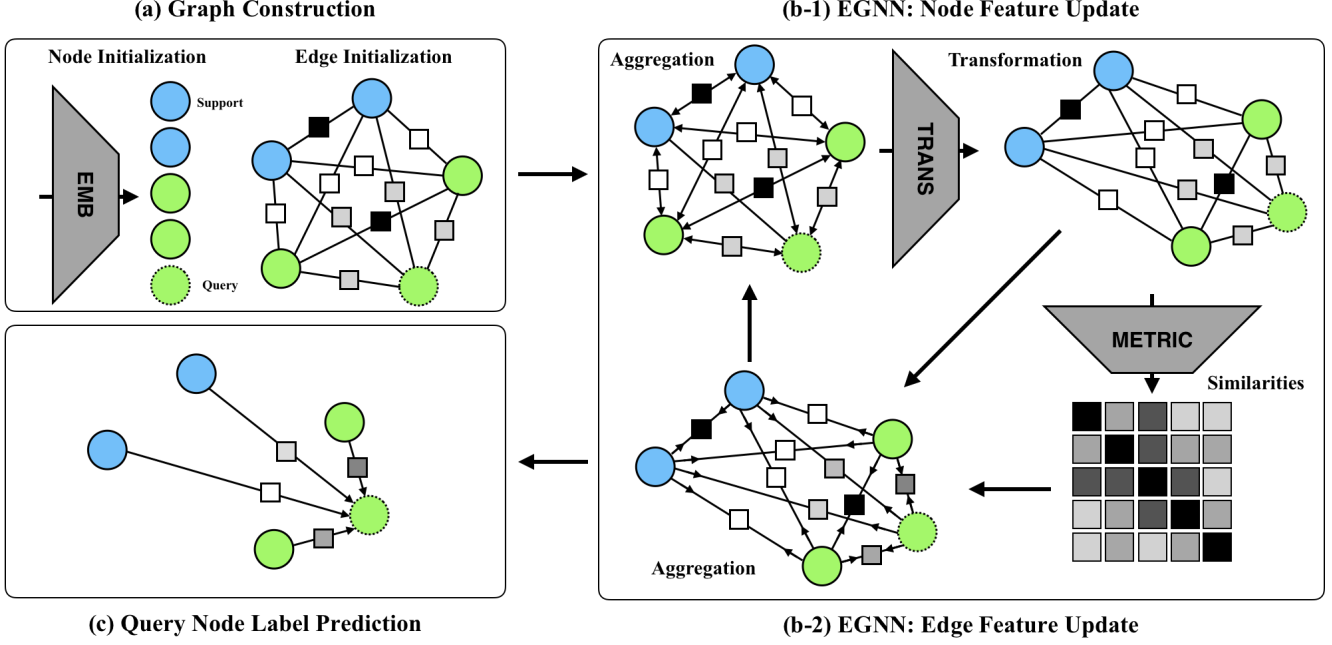**(c) Query Node Label Prediction**

**(b-2) EGNN: Edge Feature Update**

Figure 2: The overall framework of the proposed EGNN model. In this illustration, a 2-way 2-shot problem is presented as an example. Blue and green circles represent two different classes. Nodes with solid line represent labeled support samples, while a node with dashed line represents the unlabeled query sample. The strength of edge feature is represented by the color in the square. Note that although each edge has a 2-dimensional feature, only the first dimension is depicted for simplicity. The detailed process is described in Section 3.2.

Each edge feature $\mathbf{e}_{ij} = \{e_{ijd}\}_{d=1}^2 \in [0,1]^2$ is a 2-dimensional vector representing the (normalized) strengths of the intra- and inter-class relations of the two connected nodes. This allows to separately exploit the intra-cluster similarity and the inter-cluster dissimilairity.

Node features are initialized by the output of the convolutional embedding network $\mathbf{v}_i^0 = f_{emb}(\mathbf{x}_i; \theta_{emb})$, where $\theta_{emb}$ is the corresponding parameter set (see Figure 3.(a)). Edge features are initialized by edge labels as follows:

$$\mathbf{e}_{ij}^0 = \begin{cases} [1||0], & \text{if } y_{ij} = 1 \text{ and } i, j \leq N \times K, \\ [0||1], & \text{if } y_{ij} = 0 \text{ and } i, j \leq N \times K, \\ [0.5||0.5], & \text{otherwise,} \end{cases} \quad (2)$$

where $||$ is the concatenation operation.

The EGNN consists of $L$ layers to process the graph, and the forward propagation of EGNN for inference is an alternative update of node feature and edge feature through layers.

In detail, given $\mathbf{v}_i^{\ell-1}$ and $\mathbf{e}_{ij}^{\ell-1}$ from the layer $\ell-1$, *node feature update* is firstly conducted by a neighborhood aggregation procedure. The feature node $\mathbf{v}_i^\ell$ at the layer $\ell$ is updated by first aggregating the features of other nodes proportional to their edge features, and then performing the feature transformation; the edge feature $\mathbf{e}_{ij}^{\ell-1}$ at the layer $\ell-1$ is used as a degree of contribution of the correspond-

ing neighbor node like an attention mechanism as follows:

$$\mathbf{v}_i^\ell = f_v^\ell([\sum_j \tilde{e}_{ij1}^{\ell-1}\mathbf{v}_j^{\ell-1} || \sum_j \tilde{e}_{ij2}^{\ell-1}\mathbf{v}_j^{\ell-1}]; \theta_v^\ell), \quad (3)$$

where $\tilde{e}_{ijd} = \frac{e_{ijd}}{\sum_k e_{ikd}}$, and $f_v^\ell$ is the feature (node) transformation network, as shown in Figure 3.(b), with the parameter set $\theta_v^\ell$. It should be noted that besides the conventional intra-class aggregation, we additionally consider inter-class aggregation. While the intra-class aggregation provides the target node the information of "similar neighbors", the inter-class aggregation provides the information of "dissimilar neighbors".

Then, *edge feature update* is done based on the newly updated node features. The (dis)similarities between every pair of nodes are re-obtained, and the feature of each edge is updated by combining the previous edge feature value and the updated (dis)similarities such that

$$\bar{e}_{ij1}^\ell = \frac{f_e^\ell(\mathbf{v}_i^\ell, \mathbf{v}_j^\ell; \theta_e^\ell)e_{ij1}^{\ell-1}}{\sum_k f_e^\ell(\mathbf{v}_i^\ell, \mathbf{v}_k^\ell; \theta_e^\ell)e_{ik1}^{\ell-1}/(\sum_k e_{ik1}^{\ell-1})}, \quad (4)$$

$$\bar{e}_{ij2}^\ell = \frac{(1 - f_e^\ell(\mathbf{v}_i^\ell, \mathbf{v}_j^\ell; \theta_e^\ell))e_{ij2}^{\ell-1}}{\sum_k (1 - f_e^\ell(\mathbf{v}_i^\ell, \mathbf{v}_k^\ell; \theta_e^\ell))e_{ik2}^{\ell-1}/(\sum_k e_{ik2}^{\ell-1})}, \quad (5)$$

$$\mathbf{e}_{ij}^\ell = \bar{\mathbf{e}}_{ij}^\ell/\|\bar{\mathbf{e}}_{ij}^\ell\|_1, \quad (6)$$

where $f_e^\ell$ is the metric network that computes similarity scores with the parameter set $\theta_e^\ell$ (see Figure 3.(c)). In spe-
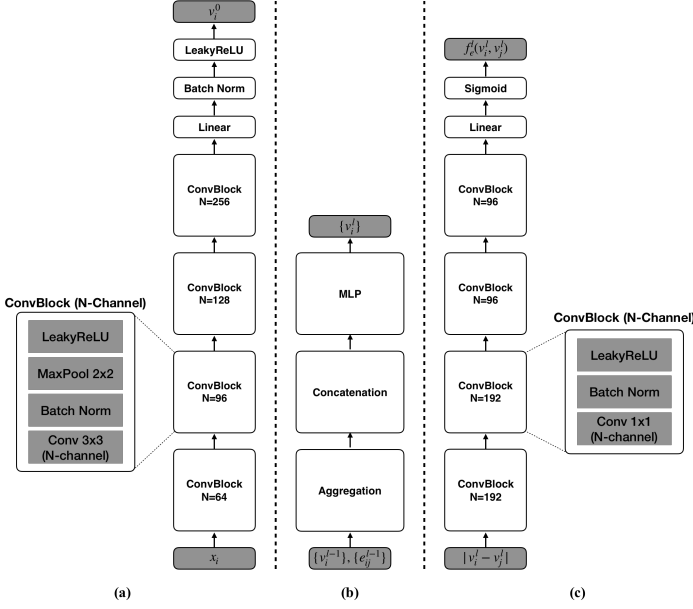
Figure 3: Detailed network architectures used in EGNN. (a) Embedding network $f_{emb}$. (b) Feature (node) transformation network $f_v^\ell$. (c) Metric network $f_e^\ell$.

cific, the node feature flows into edges, and each element of the edge feature vector is updated separately from each normalized intra-cluster similarity or inter-cluster dissimilarity. Namely, each edge update considers not only the relation of the corresponding pair of nodes but also the relations of the other pairs of nodes. We can optionally use two separate metric networks for the computations of each of similarity or dissimilarity (e.g. separate $f_{e,dsim}$ instead of $(1 - f_{e,sim})$).

After $L$ number of alternative node and edge feature updates, the edge-label prediction can be obtained from the final edge feature, i.e. $\hat{y}_{ij} = e_{ij1}^L$. Here, $\hat{y}_{ij} \in [0, 1]$ can be considered as a probability that the two nodes $V_i$ and $V_j$ are from the same class. Therefore, each node $V_i$ can be classified by simple weighted voting with support set labels and edge-label prediction results. The prediction probability of node $V_i$ can be formulated as $P(y_i = \mathcal{C}_k | \mathcal{T}) = p_i^{(k)}$:

$$p_i^{(k)} = \text{softmax}\Big( \sum_{\{j:j\neq i \wedge (\mathbf{x}_j, y_j) \in \mathcal{S}\}} \hat{y}_{ij}\delta(y_j = \mathcal{C}_k) \Big) \quad (7)$$

where $\delta(y_j = \mathcal{C}_k)$ is the Kronecker delta function that is equal to one when $y_j = \mathcal{C}_k$ and zero otherwise. Alternative approach for node classification is the use of graph clustering; the entire graph $\mathcal{G}$ can be first partitioned into clusters, using the edge prediction and an optimization for valid partitioning via linear programming [35], and then each cluster can be labeled with the support label it contains the most. However, in this paper, we simply apply Eq. (7) to obtain the classification results. The overall algorithm for the

**Algorithm 1:** The process of EGNN for inference

**1 Input:** $\mathcal{G} = (\mathcal{V}, \mathcal{E}; \mathcal{T})$, where $\mathcal{T} = \mathcal{S} \bigcup \mathcal{Q}$,
   $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N \times K}$, $\mathcal{Q} = \{\mathbf{x}_i\}_{i=N \times K+1}^{N \times K+T}$
**2 Parameters:** $\theta_{emb} \cup \{\theta_v^\ell, \theta_e^\ell\}_{\ell=1}^L$
**3 Output:** $\{\hat{y}_i\}_{i=N \times K+1}^{N \times K+T}$
**4 Initialize:** $\mathbf{v}_i^0 = f_{emb}(\mathbf{x}_i; \theta_{emb})$, $\mathbf{e}_{ij}^0$, $\forall i, j$
**5 for** $\ell = 1, \cdots, L$ **do**
       /* Node feature update        */
**6**     **for** $i = 1, \cdots, |V|$ **do**
**7**         $\mathbf{v}_i^\ell \leftarrow \text{NodeUpdate}(\{\mathbf{v}_i^{\ell-1}\}, \{\mathbf{e}_{ij}^{\ell-1}\}; \theta_v^\ell)$
**8**     **end**
       /* Edge feature update        */
**9**     **for** $(i, j) = 1, \cdots, |E|$ **do**
**10**         $\mathbf{e}_{ij}^\ell \leftarrow \text{EdgeUpdate}(\{\mathbf{v}_i^\ell\}, \{\mathbf{e}_{ij}^{\ell-1}\}; \theta_e^\ell)$
**11**     **end**
**12 end**
       /* Query node label prediction    */
**13** $\{\hat{y}_i\}_{i=N \times K+1}^{N \times K+T} \leftarrow \text{Edge2NodePred}(\{y_i\}_{i=1}^{N \times K}, \{\mathbf{e}_{ij}^L\})$

EGNN inference at test-time is summarized in Algorithm 1. The non-transductive inference means the number of query samples $T = 1$ or it performs the query inference one-by-one, separately, while the transductive inference classifies all query samples at once in a single graph.

### 3.3. Training

Given $M$ training tasks $\{\mathcal{T}_m^{train}\}_{m=1}^M$ at a certain iteration during the episodic training, the parameters of the proposed EGNN, $\theta_{emb} \cup \{\theta_v^\ell, \theta_e^\ell\}_{\ell=1}^L$, are trained in an end-to-end fashion by minimizing the following loss function:

$$\mathcal{L} = \sum_{\ell=1}^L \sum_{m=1}^M \lambda_\ell \mathcal{L}_e(Y_{m,e}, \hat{Y}_{m,e}^\ell), \quad (8)$$

where $Y_{m,e}$ and $\hat{Y}_{m,e}^\ell$ are the set of all ground-truth query edge-labels and the set of all (real-valued) query-edge predictions of the $m^{\text{th}}$ task at the $\ell^{\text{th}}$ layer, respectively, and the edge loss $\mathcal{L}_e$ is defined as binary cross-entropy loss. Since the edge prediction results can be obtained not only from the last layer but also from the other layers, the total loss combines all losses that are computed in all layers in order to improve the gradient flow in the lower layers.

## 4. Experiments

We evaluated and compared our EGNN [1] with state-of-the-art approaches on two few-shot learning benchmarks, i.e. *mini*ImageNet [2] and *tiered*ImageNet [7].

---

[1]The code and models are available on https://github.com/khy0809/fewshot-egnn.

## 4.1. Datasets

***mini*ImageNet**  It is the most popular few-shot learning benchmark proposed by [2] derived from the original ILSVRC-12 dataset [47]. All images are RGB colored, and of size $84 \times 84$ pixels, sampled from 100 different classes with 600 samples per class. We followed the splits used in [8] - 64, 16, and 20 classes for training, validation and testing, respectively.

***tiered*ImageNet**  Similar to *mini*ImageNet dataset, *tiered*ImageNet [7] is also a subset of ILSVRC-12 [47]. Compared with *mini*ImageNet, it has much larger number of images (more than 700K) sampled from larger number of classes (608 classes rather than 100 for *mini*ImageNet). Importantly, different from *mini*ImageNet, *tiered*ImageNet adopts hierarchical category structure where each of 608 classes belongs to one of 34 higher-level categories sampled from the high-level nodes in the Imagenet. Each higher-level category contains 10 to 20 classes, and divided into 20 training (351 classes), 6 validation (97 classes) and 8 test (160 classes) categories. The average number of images in each class is 1281.

## 4.2. Experimental setup

**Network Architecture**  For feature embedding module, a convolutional neural network, which consists of four blocks, was utilized as in most few-shot learning models [2, 3, 4, 6] without any skip connections [2]. More concretely, each convolutional block consists of $3 \times 3$ convolutions, a batch normalization and a LeakyReLU activation. All network architectures used in EGNN are described in details in Figure 3.

**Evaluation**  For both datasets, we conducted a 5-way 5-shot experiment which is one of standard few-shot learning settings. For evaluation, each test episode was formed by randomly sampling 15 queries for each of 5 classes, and the performance is averaged over 600 randomly generated episodes from the test set. Especially, we additionally conducted a more challenging 10-way experiment on *mini*Imagenet, to demonstrate the flexibility of our EGNN model when the number of classes are different between meta-training stage and meta-test stage, which will be presented in Section 4.5.

**Training**  The proposed model was trained with Adam optimizer with an initial learning rate of $5 \times 10^{-4}$ and weight decay of $10^{-6}$. The task mini-batch sizes for meta-training were set to be 40 and 20 for 5-way and 10-way experiments, respectively. For *mini*ImageNet, we cut the learn-

[2]Resnet-based models are excluded for fair comparison.

(a) *mini*ImageNet

| Model | Trans. | 5-Way 5-Shot |
|---|---|---|
| Matching Networks [2] | No | 55.30 |
| Reptile [46] | No | 62.74 |
| Prototypical Net [3] | No | 65.77 |
| GNN [6] | No | 66.41 |
| **EGNN** | No | **66.85** |
| MAML [4] | BN | 63.11 |
| Reptile + BN [46] | BN | 65.99 |
| Relation Net [5] | BN | 67.07 |
| MAML+Transduction [4] | Yes | 66.19 |
| TPN [12] | Yes | 69.43 |
| TPN (Higher $K$) [12] | Yes | 69.86 |
| **EGNN+Transduction** | Yes | **76.37** |

(b) *tiered*ImageNet

| Model | Trans. | 5-Way 5-Shot |
|---|---|---|
| Reptile [46] | No | 66.47 |
| Prototypical Net [3] | No | 69.57 |
| **EGNN** | No | **70.98** |
| MAML [4] | BN | 70.30 |
| Reptile + BN [46] | BN | 71.03 |
| Relation Net [5] | BN | 71.31 |
| MAML+Transduction [4] | Yes | 70.83 |
| TPN [12] | Yes | 72.58 |
| **EGNN+Transduction** | Yes | **80.15** |

Table 1: Few-shot classification accuracies on *mini*ImageNet and *tiered*ImageNet. All results are averaged over 600 test episodes. Top results are highlighted.

ing rate in half every 15,000 episodes while for tieredImageNet, the learning rate is halved for every 30,000 because it is larger dataset and requires more iterations to converge. All our code was implemented in Pytorch [48] and run with NVIDIA Tesla P40 GPUs.

## 4.3. Few-shot classification

The few-shot classification performance of the proposed EGNN model is compared with several state-of-the-art models in Table 1a and 1b. Here, as presented in [12], all models are grouped into three categories with regard to three different transductive settings; "No" means non-transductive method, where each query sample is predicted independently from other queries, "Yes" means transductive method where all queries are simultaneously processed and predicted together, and "BN" means that query batch statistics are used instead of global batch normalization parameters, which can be considered as a kind of transductive inference at test-time.

The proposed EGNN was tested with both transductive and non-transductive settings. As shown in Table 1a, EGNN shows the best performance in 5-way 5-shot set-

ting, on both transductive and non-transductive settings on *mini*Imagenet. Notably, EGNN performed better than node-labeling GNN [6], which supports the effectiveness of our edge-labeling framework for few-shot learning. Moreover, EGNN with transduction (EGNN + Transduction) outperformed the second best method (TPN [12]) on both datasets, especially by large margin on *mini*Imagenet. Table 1b shows that the transductive setting on *tiered*Imagenet gave the best performance as well as large improvement compared to the non-transductive setting. In TPN, only the labels of the support set are propagated to the queries based on the pairwise node feature affinities using a common Laplacian matrix, so the queries communicate to each other only via their embedding feature similarities. In contrast, our proposed EGNN allows us to consider more complicated interactions between query samples, by propagating to each other not only their node features but also edge-label information across the graph layers having different parameter sets. Furthermore, the node features of TPN are fixed and never changed during label propagation, which allows them to derive a closed-form, one-step label propagation equation. On the contrary, in our EGNN, both node and edge features are dynamically changed and adapted to the given task gradually with several update steps.

### 4.4. Semi-supervised few-shot classification

For semi-supervised experiment, we followed the same setting described in [6] for fair comparison. It is a 5-way 5-shot setting, but the support samples are only partially labeled. The labeled samples are balanced among classes so that all classes have the same amount of labeled and unlabeled samples. The obtained results on *mini*Imagenet are presented in Table 2. Here, *"LabeledOnly"* denotes learning with only labeled support samples, and *"Semi"* means the semi-supervised setting explained above. Different results are presented according to when 20% and 40%, 60% of support samples were labeled, and the proposed EGNN is compared with node-labeling GNN [6]. As shown in Table 2, semi-supervised learning increases the performances in comparison to labeled-only learning on all cases. Notably, the EGNN outperformed the previous GNN [6] by a large margin (61.88% vs 52.45%, when 20% labeled) on semi-supervised learning, especially when the labeled portion was small. The performance is even more increased on transductive setting (EGNN-Semi(T)). In a nutshell, our EGNN is able to extract more useful information from unlabeled samples compared to node-labeling framework, on both transductive and non-transductive settings.

### 4.5. Ablation studies

The proposed edge-labeling GNN has a deep architecture that consists of several node and edge-update layers. Therefore, as the model gets deeper with more layers, the

| Training method | Labeled Ratio (5-way 5-shot) | | | |
|---|---|---|---|---|
| | 20% | 40% | 60% | 100% |
| GNN-LabeledOnly [6] | 50.33 | 56.91 | - | 66.41 |
| GNN-Semi [6] | 52.45 | 58.76 | - | 66.41 |
| EGNN-LabeledOnly | 52.86 | - | - | 66.85 |
| **EGNN-Semi** | **61.88** | **62.52** | **63.53** | 66.85 |
| EGNN-LabeledOnly(T) | 59.18 | - | - | 76.37 |
| **EGNN-Semi(T)** | **63.62** | **64.32** | **66.37** | 76.37 |

Table 2: Semi-supervised few-shot classification accuracies on *mini*ImageNet.

| Feature type | # of EGNN layers | | |
|---|---|---|---|
| | 1 | 2 | 3 |
| **Intra & Inter** | 67.99 | 73.19 | **76.37** |
| Intra Only | 67.28 | 72.20 | 74.04 |

Table 3: 5-way 5-shot results on *mini*Imagenet with different numbers of EGNN layers and different feature types

interactions between task samples should be propagated more intensively, which may leads to performance improvements. To support this statement, we compared the few-shot learning performances with different numbers of EGNN layers, and the results are presented in Table 3. As the number of EGNN layers increases, the performance gets better. There exists a big jump on few-shot accuracy when the number of layers changes from 1 to 2 (67.99% → 73.19%), and a little additional gain with three layers (76.37 %).

Another key ingredient of the proposed EGNN is to use separate exploitation of intra-cluster similarity and inter-cluster dissimilarity in node/edge updates. To validate the effectiveness of this, we conducted experiment with only intra-cluster aggregation and compared the results with those obtained by using both aggregations. The results are also presented in Table 3. For all EGNN layers, the use of separate inter-cluster aggregation clearly improves the performances.

It should also be noted that compared to the previous node-labeling GNN, the proposed edge-labeling framework is more conducive in solving the few-shot problem under arbitrary meta-test setting, especially when the number of few-shot classes for meta-testing does not match to the one used for meta-training. To validate this statement, we conducted a cross-way experiment with EGNN, and the result is presented in Table 4. Here, the model was trained with 5-way 5-shot setting and tested on 10-way 5-shot setting, and vice versa. Interestingly, both cross-way results are similar to those obtained with the matched-way settings. Therefore, we can observe that the EGNN can be successfully extended to modified few-shot setting without re-training of the model, while the previous node-labeling GNN [6] is

| Model | Train way | Test way | Accuracy |
|---|---|---|---|
| Prototypical [3] | 5 | 5 | 65.77 |
| **Prototypical** | 5 | 10 | 51.93 |
| Prototypical | 10 | 10 | 49.29 |
| **Prototypical** | 10 | 5 | 66.93 |
| GNN [6] | 5 | 5 | 66.41 |
| **GNN** | 5 | 10 | N/A |
| GNN | 10 | 10 | 51.75 |
| **GNN** | 10 | 5 | N/A |
| EGNN | 5 | 5 | 76.37 |
| **EGNN** | 5 | 10 | 56.35 |
| EGNN | 10 | 10 | 57.61 |
| **EGNN** | 10 | 5 | 76.27 |

Table 4: Cross-way few-shot learning results on miniImagenet 5-shot setting.

not even applicable to cross-way setting, since the size of the model and parameters are dependent on the number of ways.

Figure 4 shows t-SNE [49] visualizations of node features for the previous node-labeling GNN and EGNN. The GNN tends to show a good clustering among support samples after the first layer-propagation, however, query samples are heavily clustered together, and according to each label, query samples and their support samples never get close together, especially even with more layer-propagations, which means that the last fully-connect layer of GNN actually seems to perform most roles in query classification. In contrast, in our EGNN, as the layer-propagation goes on, both the query and support samples are pulled away if their labels are different, and at the same time, equally labeled query and support samples get close together.

For further analysis, Figure 5 shows how edge features propagate in EGNN. Starting from the initial feature where all query edges are initialized with 0.5, the edge feature gradually evolves to resemble ground-truth edge label, as they are passes through the several EGNN layers.

## 5. Conclusion

This work addressed the problem of few-shot learning, especially on the few-shot classification task. We proposed the novel EGNN which aims to iteratively update edge-labels for inferring a query association to an existing support clusters. In the process of EGNN, a number of alternative node and edge feature updates were performed using explicit intra-cluster similarity and inter-cluster dissimilarity through the graph layers having different parameter sets, and the edge-label prediction was obtained from the final edge feature. The edge-labeling loss was used to update the parameters of the EGNN with episodic training. Ex-
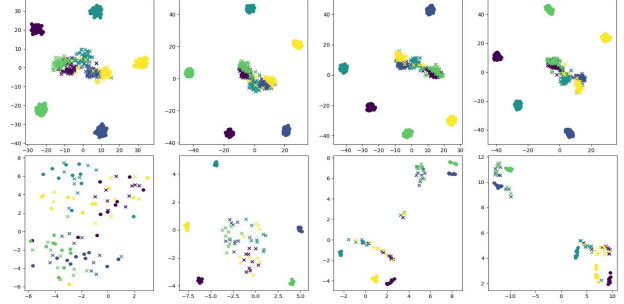


Figure 4: t-SNE visualization of node features. From top to bottom: GNN [6], EGNN. From left to right: initial embedding, 1st layer, 2nd layer, 3rd layer. 'x' represents query, 'o' represents support. Different colors mean different labels.
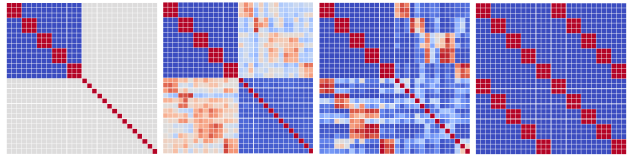


Figure 5: Visualization of edge feature propagation. From left to right: initial edge feature, 1st layer, 2nd layer, ground-truth edge labels. Red color denotes higher value ($e_{ij1} = 1$), while blue color denotes lower value ($e_{ij1} = 0$). This illustration shows 5-way 3-shot setting, and 3 queries for each class, total 30 task-samples. The first 15 samples are support set, and latter 15 are query set.

perimental results showed that the proposed EGNN outperformed other few-shot learning algorithms on both of the supervised and semi-supervised few-shot image classification tasks. The proposed framework is applicable to a broad variety of other meta-clustering tasks. For future work, we can consider another training loss which is related to the valid graph clustering such as the cycle loss [35]. Another promising direction is graph sparsification, e.g. constructing $K$-nearest neighbor graphs [50], that will make our algorithm more scalable to larger number of shots.

## Acknowledgement

# References

[1] Christiane Lemke, Marcin Budka, and Bogdan Gabrys. Met-alearning: a survey of trends and technologies. Artificial Intelligence Review, 44(1), 2015. 1

[2] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In NIPS, pages 3630–3638, 2016. 1, 2, 3, 5, 6

[3] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In NIPS, pages 4077–4087, 2017. 1, 3, 6, 8

[4] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In ICML, 2017. 1, 3, 6

[5] Flood Sung Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In CVPR, 2018. 1, 3, 6

[6] Victor Garcia and Joan Bruna. Few-shot learning with graph neural networks. In ICLR, 2018. 1, 2, 6, 7, 8

[7] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. Meta-learning for semi-supervised few-shot classification. In ICLR, 2018. 1, 5, 6

[8] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In ICLR, 2017. 1, 3, 6

[9] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In ICML, pages 1842–1850, 2016. 1, 2, 3

[10] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. In ICLR, 2018. 1, 3

[11] Boris N. Oreshkin, Pau Rodriguez, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. In NIPS, 2018. 1

[12] Yanbin Liu, Juho Lee, Minseop Park, Saehoon Kim, and Yi Yang. Transductive propagation network for few-shot learning. In ICLR, 2019. 1, 2, 6, 7

[13] Yu-Xiong Wang, Ross B. Girshick, Martial Hebert, and Bharath Hariharan. Low-shot learning from imaginary data. In CVPR, 2018. 1

[14] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B tenenbaum. Human-level concept learning through probabilistic program induction. Science, 350(6266):1332–1338, 2015. 1

[15] Taesup Kim, Jaesik Yoon, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. In NIPS, 2018. 1

[16] Marcin Andrychowicz, Misha Denil, Sergio Gomez Colmenarejo, Matthew W. Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In NIPS, 2016. 1

[17] Irwan Bello, Barret Zoph, Vijay Vasudevan, and Quoc V. Le. Neural optimizer search with reinforcement learning. In ICML, 2017. 1

[18] Olga Wichrowska, Niru Maheswaranathan, Matthew W. Hoffman, Sergio Gomez Colmenarejo, Misha Denil, Nando de Freitas, and Jascha Sohl-Dickstein. Learned optimizers that scale and generalize. In ICML, 2017. 1

[19] Maruan Al-Shedivat, Trapit Bansal, Yuri Burda, Ilya Sutskever, Igor Mordatch, and Pieter Abbeel. Continuous adaptation via meta-learning in nonstationary and competitive environments. In ICLR, 2018. 1

[20] Rein Houthooft, Richard Y. Chen, Phillip Isola, Bradly C. Stadie, Filip Wolski, Jonathan Ho, and Pieter Abbeel. Evolved policy gradients. In NIPS, 2018. 1

[21] Ignasi Clavera, Anusha Nagabandi, Ronald S. Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt: Meta-learning for model-based control. CoRR, abs/1803.11347, 2018. URL http://arxiv.org/abs/1803.11347. 1

[22] Risto Vuorio, Dong-Yeon Cho, Daejoong Kim, and Jiwon Kim. Meta continual learning. arXiv, 2018. URL https://arxiv.org/abs/1806.06928. 1

[23] Ju Xu and Zhanxing Zhu. Reinforced continual learning. In NIPS, 2018. 1

[24] Peter W. Battaglia et al. Relational inductive biases, deep learning, and graph networks. arXiv, 2018. URL https://arxiv.org/abs/1806.01261. 1

[25] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. IEEE Signal Processing Magazine, 34(4):18–42, 2017. 1

[26] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? arXiv preprint arXiv:1810.00826, 2018. 1

[27] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. CoRR, abs/1704.01212, 2017. URL http://arxiv.org/abs/1704.01212. 1

[28] M. Gori, G. Monfardini, and F. Scarselli. A new model for learning in graph domains. In IJCNN, 2005. 1, 2

[29] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. IEEE Transactions on Neural Networks, 20(1):61–80, 2008. 1, 2

[30] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In ICLR, 2017. 1, 2

[31] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. In ICLR, 2016. 1, 2

[32] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In NIPS, 2017. 1

[33] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In ICLR, 2018. 1

[34] Michael Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In NIPS, 2016. 1, 2

[35] Sungwoong Kim, Sebastian Nowozin, Pushmeet Kohli, and Chang D Yoo. Higher-order correlation clustering for image segmentation. In NIPS, pages 1530–1538, 2011. 2, 5, 8

[36] Liyu Gong and Qiang Cheng. Adaptive edge features guided graph attention networks. arXiv preprint arXiv:1809.02709, 2018. 2

[37] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. arXiv preprint arXiv:1802.04687, 2018. 2

[38] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. CoRR, abs/1312.6203, 2013. 2

[39] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. CoRR, abs/1506.05163, 2015. 2

[40] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. Machine Learning, 56:89–113, 2004. 2

[41] T. Finley and T. Joachims. Supervised clustering with support vector machines. In ICML, 2005. 2

[42] B. Taskar. Learning structured prediction models: a large margin approach. Ph.D. thesis, Stanford University, 2004. 2

[43] Daniel D Johnson. Learning graphical state transitions. In ICLR, 2016. 2

[44] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. 2015. 3

[45] Fengwei Zhou, Bin Wu, and Zhenguo Li. Deep meta-learning: Learning to learn in the concept space. CoRR, abs/1802.03596, 2018. 3

[46] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. CoRR, abs/1803.02999, 2018. 3, 6

[47] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. International Journal of Computer Vision, 115(3):211–252, 2015. 6

[48] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In NIPS-W, 2017. 6

[49] L. van der Maaten and G. Hinton. Visualizing data using t-sne. JMLR, 9:2579–2605, 2008. 8

[50] Xiaojuan Qi, Renjie Liao, Jiaya Jia, Sanja Fidler, and Raquel Urtasun. 3d graph neural networks for rgbd semantic segmentation. In Proceedings of the IEEE International Conference on Computer Vision, pages 5199–5208, 2017. 8