Neural Graph Matching Networks for Fewshot 3D Action Recognition

Michelle Guo[1 [0000−0002−6574−6669]], Edward Chou[1 [0000−0002−0670−459X]], De-An Huang[1 [0000−0002−6945−7768]], Shuran Song[2 [0000−0002−8768−7356]], Serena Yeung[1 [0000−0003−0529−0628]], and Li Fei-Fei[1 [0000−0002−7481−0810]]

[1] Computer Science Department, Stanford University
[2] Computer Science Department, Princeton University

Abstract. We propose Neural Graph Matching (NGM) Networks, a novel framework that can learn to recognize a previous unseen 3D action class with only a few examples. We achieve this by leveraging the inherent structure of 3D data through a graphical representation. This allows us to modularize our model and lead to strong data-efficiency in few-shot learning. More specifically, NGM Networks jointly learn a graph generator and a graph matching metric function in an end-to-end fashion to directly optimize the few-shot learning objective. We evaluate NGM on two 3D action recognition datasets, CAD-120 and PiGraphs, and show that learning to generate and match graphs both lead to significant improvement of few-shot 3D action recognition over the holistic baselines.

1 Introduction

Recent availability of commodity depth sensors has provided new ways to capture 3D data, but labeled depth datasets are scarce, making it difficult to transfer the success of deep learning techniques from the RGB domain [1, 2]. This is especially true for videos, where the difficulty and cost of labelling has already been a roadblock for collecting RGB video datasets [3, 4]. One possible approach is to use self-supervised [5, 6] or unsupervised learning [7] for learning a 3D data representation that serves as an efficient model initialization for the tasks of interest. While such methods have been successfully applied to RGB-D action recognition [5] and 3D scene labeling [7], we argue that it does not fully utilize the labeled 3D datasets that are readily available [8–11].

In this work, we introduce *few-shot learning* [12] to 3D action recognition, where the model is explicitly trained to deal with scarce training data for previously unseen classes. This is in contrast to representation learning approaches, where the model is not informed with the task of interest. While recent works have addressed few-shot learning in the RGB domain [13, 14, 12], adapting these method to the 3D space is a non-trivial task. Unlike the images, where effective RGB representations exist (i.e., ImageNet [15] pretrained CNN), its counterpart in 3D video is still an open research problem [16–18]. As we will show in our experiments, direct application of few-shot learning to the existing 3D representation does not lead to effective generalization to novel classes.

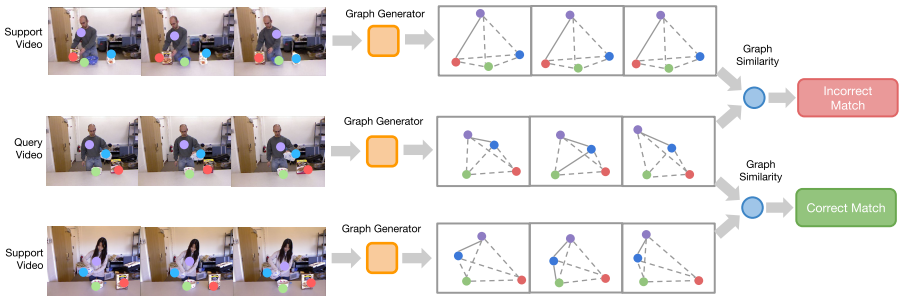

Fig. 1. The proposed Neural Graph Matching Networks is able to recognize a previously unseen action class with only a few examples. We achieve this by leveraging the spatial information inherent in the 3D data. As shown in the figure, the query video (middle) is visually similar to the video of a different class (top), which can confuse holistic approaches. However, NGM is able to leverage the graphical representation and match the query to the video with the correct class (bottom).

Our key observation to resolve this challenge is that there is inherent structure in 3D data that can be naturally leveraged to provide modularity for our representation, and thus lead to more effective few-shot learning. Modularity and compositionality has been shown to be effective for improving data efficiency in visual question answering [19–21]. As shown in Figure 1, visually diverse actions of the same class can be correlated by their underlying structure.

With these insights, we propose Neural Graph Matching (NGM) Networks, a novel graph-based approach that learns to generate and match graphs for few-shot 3D action recognition. NGM consists of two stages that can be trained jointly in an end-to-end fashion. The first stage is graph generation, where we leverage the 3D spatial information of the environment captured by the 3D data to generate the intermediate graphical representation, or the *interaction graph*. For each action, the graph uses nodes to represent physical entities in a 3D scene (e.g. body parts, objects) and edges to represent the interactions between the entities (e.g. touch, gaze) [22]. This graphical structure allows us to better model both the spatial relationship between human and objects and to capture the temporal evolution of videos, while also using stronger data efficiency in the few-shot setting. The second stage is graph matching, where we learn on the graph-based matching function as a metric to enable few-shot training on the generated interaction graph. In this way, NGM automatically learns in an end-to-end fashion both the graphical representation of the 3D environment and the graph matching metric function that are best suited for few-shot learning of novel 3D action classes. This is in contrast to holistic-based approaches [16–18, 23], where the high-dimensional input is directly mapped to a feature representation without explicitly leveraging any spatial information captured by the 3D data. For example, PointNet [18] processes permutation and geometric invariant point clouds, holistically processing the scene's point-representation.

Fig. 2. CAD-120 Point Clouds. We evaluate NGM on the CAD-120 dataset, which contains RGB-D videos of everyday actions. We visualize single point cloud frames where each point is the 3-D projected (x_i, y_i, z_i) of the corresponding depth frame.

We evaluate few-shot learning of Neural Graph Matching Networks on two 3D action datasets: CAD-120 [9] (Figure 2) and PiGraphs [22]. We show that when there is only a single example available, NGM is able to outperform the holistic baseline up to 20% by explicitly leveraging 3D spatial information. In addition, we show that the proposed end-to-end framework is able to learn meaningful graph generations and matching metrics that perform significantly better than heuristically generated edges.

To summarize our main contributions, we: (i) introduce the few-shot learning task for 3D action recognition to address the challenge of scarce training data compared to 2D; (ii) propose the use of graphical representations to explicitly leverage the spatial information in 3D data; (iii) present Neural Graph Matching Networks, a novel framework that learns to jointly generate and match the graphical representation in an end-to-end fashion, which leads to stronger data efficiency for 3D few-shot learning.

2 Related Work

Few-shot Learning. Few-shot learning and similar concepts have been examined thoroughly in past literature. Many of these works cover the use of holistic based approaches [13, 24–26, 14, 12]. Vinyals et al. [12] uses matching networks to perform one-shot learning, casting set-to-set test labels for unobserved classes using k-nearest neighbors with cosine distance. Snell et al. [14] carries along this approach using euclidean distance and creating a prototypical representation of each class. Both approaches use a holistic approach, where raw input and label pairs which are fed into the network without leveraging structural data. Also, both works use a fixed similarity metric in that only certain distance computations are used for the K-NN classification. Further works have introduced other techniques for learned similarity metrics. Santoro et al. [26] explores the topic of relational reasoning, where a module learns a relation between two objects within the relation network using MLPs and and synaptic weights.

3D Action Recognition. Traditional 3D action recognition approaches rely on hand-crafted features, such as HON4D [27] and HOPC [28] to capture the spatial-temporal information. One dominant alternative is the skeleton based approaches [29, 30], where the video is represented as a sequence of joint positions. Recent 3D action recognition approaches utilize skeletal pose or temporal

features that are typically fed into a combination of convolutional and recurrent networks[31–33]. Part-aware LSTMs have been explored for RGB-D action recognition; however the focus there is on nodes instead of graphs. While most 3D action recognition approaches are designed for supervised learning, addressing 3D action recognition for the few-shot setting has been relatively unexplored.

Modularity and Compositionality. Modular approaches have been shown important for data efficiency in visual understanding. One example is the visual question answering problem [20, 21]. Our work is tied to compositionality, where a set of entities and there interactions are used to describe the action. Ikizler et al. [34] describes an approach of breaking down movements per body part to compose a larger activity description task. Gu et al. [35] outlines a distinct approach towards compositionality, using action primitives to describe an action instead of body parts. Other representations include: *scene graphs* for objects and relationships in a 2D scene [36], and *interaction graphs* [22] to model 3D data for scene synthesis.

Deep Learning on Graphs. A few works learn on graph node embeddings over a single large graph [37–39]. This is similar to word embeddings learned in natural language processing models (e.g., word2vec [40]). However, in this work, we must process multiple different graphs representing various action video examples. Related to our work on graph processing are graph neural networks (GNNs), which are capable of processing arbitrary graphs. GNNs have been used to model a variety of structural data, including molecular fingerprints, citation networks, and knowledge graphs [41, 42]. GNNs has also been used to model relationships between images for few-shot image classification [43].

3 Problem Formulation

3.1 Fewshot-Learning

We first formulate the few-shot learning problem following the definitions in previous works [14, 12]. In contrast to standard classification problem, the classes are split into two types in few-shot learning. Let $\mathcal{C} = \{1, ...K\}$ be the set of all classes, which is split into sets: $\mathcal{C}_{train}$, the training classes that have sufficient data for few-shot learning, and $\mathcal{C}_{test}$, the novel or unseen classes that have only a few labeled data. A k-shot N-way classification in few-shot learning means that we have N novel classes (*i.e.,* $|\mathcal{C}_{test}| = N$), and each novel class has k examples.

The success of recent few-shot learning approaches [12–14, 24–26] relies on transferring the representation learned in the training classes $\mathcal{C}_{train}$ to the novel classes $\mathcal{C}_{test}$ for improved data efficiency. In other words, the few-shot learning problem can be formulated as learning a metric function $\phi(x_i, x_j)$ for two input examples x_i and x_j from $\mathcal{C}_{train}$, which can generalize to novel classes $\mathcal{C}_{test}$ so that $\phi(x_i, x_j)$ is small for data points in the same class, while larger and more distant for data from different classes. One naive approach for learning $\phi(\cdot, \cdot)$ is to directly apply supervised training on $\mathcal{C}_{train}$, which directly minimizes the intra-class distance while maximizing the inter-class distance. However, it has

been shown that a better approach is to employ "episodic training" that simulates the few-shot setting to learn $\phi(x_i, x_j)$ in $\mathcal{C}_{train}$ [12]. This leads to stronger generalization to novel classes $\mathcal{C}_{test}$.

3.2 Graph-based Few-shot Learning

Our work follows the few-shot learning setup, and introduces it to 3D action recognition (see Figure 8 in Supplementary). The key challenge is that, unlike the image counterpart, the form of the metric function $\phi(\cdot, \cdot)$ is still a critical research problem. We argue that direct application of holistic approaches, such as PointNet [18], does not fully utilize the spatial information in the 3D data. Image processing and proposing and segmenting arbitrary objects remains a challenge [44], while the extra dimension in 3D data allows us to better model the relationships between human and objects. Thus, our primary contribution is to explicitly leverage the spatial information with graphical representation. Formally, our Neural Graph Matching Networks can be seen as decomposing the metric function as:

$$\phi(x_i, x_j) = \phi_{GM}(g(x_i), g(x_j)), \tag{1}$$

where $g(\cdot)$ is our graph generator that obtains the interaction graph from the input, and $\phi_{GM}(\cdot, \cdot)$ is the graph matching network we learn jointly with the generator to directly optimize for few-shot learning.

4 Methods

We have formulated few-shot learning as learning the metric function $\phi(\cdot, \cdot)$ from the training classes $\mathcal{C}_{train}$, and the goal is to learn to generalize to $\mathcal{C}_{test}$ for few-shot classification. The primary contribution of our work is to explicitly leverage the 3D information by decomposing the the metric into $\phi(x_i, x_j) = \phi_{GM}(g(x_i), g(x_j))$, the graph matching metric ϕ_{GM}, and the graph generator $g(\cdot)$. This decomposition allows us to better leverage spatial information that is inherent in the 3D data, and leads to stronger generalization for few-shot learning. An overview of our method is shown in Figure 3. We first discuss our graph learning approach in Section 4.1, followed by the graph matching method in Section 4.2. Finally, we show how the combination of the two can be trained in a end-to-end fashion in Section 4.3.

4.1 Graph Generation

Our key insight is that 3D data contains inherent spatial structure which can be encoded in graphical form to improve data-efficiency of few-shot learning. The challenge is that we aim to achieve graph generation without graph supervision and annotation. One naive approach is to use heuristics based approaches and hard-code the graph generation process. However, such heuristics can easily be affected by noise, and it is not guaranteed to be beneficial to our few-shot learning

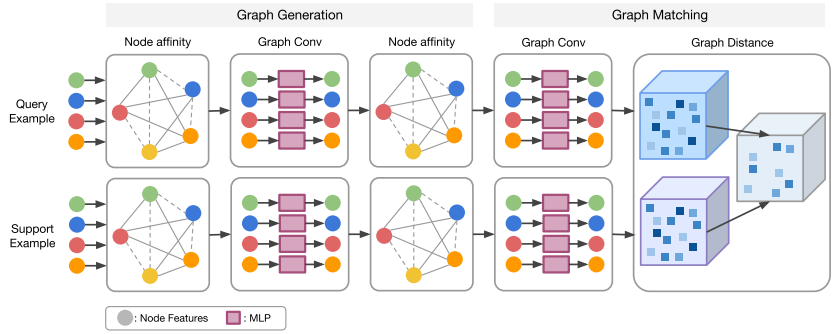

Fig. 3. Overview of Neural Graph Matching (NGM) Networks. NGM consists of two parts: graph generation and the graph matching metric, which are jointly optimized for few-shot learning. In graph generation, we utilize graph convolution to generation node features that take the contextual information into account. For the graph matching metric, we propose *graph tensor* as the graph representation that allow us to combine information in both the graph structure and the continuous node representation.

problem. We address the challenge by formulating the graph generation as a differentiable process, which can be trained jointly with our graph matching metric to directly optimize for the few-shot objective.

We use the *interaction graph* as our graphical representation, which is composed of nodes that represent physical entities in a 3D scene (e.g. body parts, objects) as well as edges that represent interactions between the entities (e.g. touch, gaze) [22]. Given a set of node categories C and a set of node relationships E, an interaction graph $G_{i,t}$ representing a video frame $x_{i,t}$ is a tuple $(N_{i,t}, E_{i,t})$ where $N_{i,t} = \{n_1, ..., n_n\}$ is the set of nodes with each $n_j \in C$, and E is the set of undirected edges of the form (n_j, e, n_k), where $n_j, n_k \in N$ and $e \in E$.

Node Construction. Nodes of an interaction graph can be obtained using either human annotated object and pose detections, or any pretrained object or pose detector. Each node contains associated features $\rho_{i,t}$, which can be extracted from the raw pixels of the image (e.g. 3D position).

Edge Learning. In contrast to node construction, which are well-studied problems in the object and pose detection space, edge learning to capture the relationship between objects in the scene is still an on-going research area [45]. In contrast to previous works that use fully supervised learning for the edges [45], we learn the edge generation jointly with our graph matching metric for few-shot learning. It is thus important for the edge learning process to be differentiable. This expands the semantics of our learned interaction graph edges beyond predefined heuristics (e.g. contact, gaze) [22]. Given two nodes x_i, x_j from the graph, we define the edge strength $A_{i,j}$ between the nodes as:

$$A_{i,j} = \psi(x_i, x_j) = \text{MLP}_{edge}(|f(x_i) - f(x_j)|), \tag{2}$$

where $f(\cdot)$ is the feature representation of the node, and $\text{MLP}_{edge}(\cdot)$ is a multi-layer perceptron. Taking the absolute difference between the features instead of concatenating them ensures that the operation satisfy the symmetry property [43]. Thus, $f(\cdot)$ plays an important role for the quality of our edges. It is important that $f(\cdot)$ also depends on the graph structure, and is not applied independently for each node, as the same object can have very different relationships with others depending on different context. When making cereal, the node for bowl should be closely related to hand, while the relationship shouldn't exist when the action is just opening the microwave. We thus update the node feature representation with neighboring node's representation using graph convolution networks [42] to make $f(\cdot)$ also depend on the adjacency matrix. We update them iteratively:

$$f^{(k+1)}(x_i) = \sigma((D^{(k)})^{-\frac{1}{2}} A_i^{(k)} (D^{(k)})^{-\frac{1}{2}} f^{(k)}(x_i) W_{edge}), \tag{3}$$

$$A_{i,j}^{(k)} = \text{MLP}_{edge}(|f^{(k)}(x_i) - f^{(k)}(x_j)|), \tag{4}$$

where $D_{i,i}^{(k)} = \sum_j A_{j,i}^{(k)}$ is the diagonal node degree matrix, and W_{edge} is the trainable matrix for feature representation. We use the initial node feature from node construction as $f^{(0)}(\cdot)$. In this case, our generated edges would depend on the structure of the graph depending on the context. Note that we keep the continuous edge strength in the adjacency matrix A to preserve the learned edge as a differentiable inputs for our graph matching metric function. This allows us to train the graph generation to directly optimize few-shot generalization.

4.2 Graph Matching

We have discussed how we generate the interaction graph as the graphical representation to explicitly leverage the spatial information inherent in the 3D input. As discussed in Section 3, we formulate the few-shot learning $\phi(x_i, x_j) = \phi_{GM}(g(x_i), g(x_j))$ as learning jointly the graph generation $g(\cdot)$ and graph matching ϕ_{GM}. Now we discuss the graph matching metric ϕ_{GM}.

In contrast to classical *exact graph matching* problem [46], where there is an isomorphic relationship between the two comparing graphs, our data-driven graphs can have varying number of nodes. This is called the *inexact graph matching* [47], and has been important for image segmentation and processing [48]. However, classical inexact graph matching usually abstracts away from the node representation or feature, which does not fully utilize the input information in our case. For example, even when the node for hand is close to an object, the corresponding action still depends on other context in the input, and cannot be solely captured by the graph structure.

On the other extreme is recent approaches that aims to learn a *graph embedding* [49] as a single vector representation capturing all the information in the graph. While it is possible to include the edge information through approaches like graph neural networks [42, 50], 3D action recognition often requires us to keep fine-grained information. For example, when an action is interacting with cluttered objects, it is important that we explicitly model their relationships.

We thus propose to use *graph tensor* as the graph matching representation. A graph tensor $\mathbf{T} \in \mathbb{R}^{|C|\times|C|\times d}$ is a three dimensional tensor, where $|C|$ is the number of node types, and d the dimension of the node feature. We define:

$$\mathbf{T}_{m,m,:} = \sum_{c(i)=m} \hat{f}(x_i), \text{ and } \mathbf{T}_{m,k,:} = \sum_{c(i)=m, c(j)=k} \hat{\psi}(x_i, x_j) \text{ for } i \neq j, \quad (5)$$

where $c(i)$ is the node type of node i, $\hat{f}(\cdot)$ the node matching feature, and $\psi(\cdot,\cdot)$ the edge feature for matching. For the node matching feature we reuse the weights W_{edge} from graph generation and define:

$$\hat{f}(x_i) = \sigma(\tilde{D}^{-\frac{1}{2}} A_i \tilde{D}^{-\frac{1}{2}} f(x_i) W_{edge}), \quad (6)$$

where A is the final adjacency matrix from node generation, and $f(\cdot)$ is the corresponding final node feature. For the edge matching feature, we reuse the node affinity from Eq. (2): $\psi(\cdot,\cdot) = \hat{\psi}(\cdot,\cdot)$. For two interaction graphs G_i and G_j, we thus define the graph matching metric as:

$$\phi_{GM}(G_i, G_j) = ||\mathbf{T}(G_i) - \mathbf{T}(G_j)||^2, \quad (7)$$

the distance between the corresponding graph tensors. Here we overload the notation $\mathbf{T}$, where now $\mathbf{T}(G)$ is the graph tensor of graph G. One implicit assumption of our method is that we assume the availability of the node type classifier $c(\cdot)$ for aggregating and matching the nodes of the same type. Node type in this case can be human joint or object class. This resolves the node correspondence and simplifies the graph matching problem.

In the few-shot setting, we hope to learn a deep graph matching metric between a query graph generated from query and support graphs generated from the support examples in each action class. We follow the prototypical networks [14], and define the prototypical graph tensor of a class k as $\mathbf{T}_k = \frac{1}{N}\sum_{c(i)=k} \mathbf{T}_i$ the average of graph tensor of all the support graphs.

To predict the action class for a given query example x_i, we compare the query's graph tensor with the prototypical graph tensor of each class k:

$$p(y=k|x_i) = \frac{\exp(-||\mathbf{T}(g(x_i)), \mathbf{T}_k||^2)}{\sum_{k'} \exp(-||\mathbf{T}(g(x_i)), \mathbf{T}_{k'}||^2} = \frac{\exp(-\phi_{GM}(x_i, \mathbf{x}_k))}{\sum_{k'} \exp(-\phi_{GM}(x_i, \mathbf{x}_{k'}))}, \quad (8)$$

where $\mathbf{x}_k$ is the synthetic prototype for type k for interpretation.

4.3 Learning and Optimization

Learning is performed by minimizing the negative log-probability of the true class k via stochastic gradient descent:

$$J = -\log p(y=k|x_i) = \frac{\exp(-\phi_{GM}(x_i, \mathbf{x}_k))}{\sum_{k'} \exp(-\phi_{GM}(x_i, \mathbf{x}_{k'}))} \quad (9)$$

We use *episode-based* training [12] to simulate the few-shot setting at training to directly optimize for generalization to unseen novel classes.

Note that the proposed Neural Graph Matching Networks is end-to-end trainable from the input x. We have defined neural graph matching as:

$$\phi_{GM}(x_i, x_j) = ||\mathbf{T}(g(x_i)) - \mathbf{T}(g(x_j))||^2. \tag{10}$$

From Eq. (3) and Eq. (4), we can see that both the output feature and adjacency matrix are differentiable for graph generator $g(\cdot)$. From Eq. (5), both the node matching feature $\hat{f}(\cdot)$ and the edge matching feature $\hat{\psi}(\cdot)$ are differentiable. In this case, we are able to train the loss in Eq. (9) directly from the input with episode-based training. This allows us to jointly learn the optimized graph and the corresponding graph matching metric for few-shot learning.

5 Experiments

In this work, our goal is few-shot 3D action recognition, where the model is able to classify novel classes with only a few training examples. Instead of directly applying a holistic approach as used in the image space, we propose to use an interaction graph as the intermediate representation to explicitly leverage the inherent spatial information in 3D data. Our experiments aim to answer the following questions: (1) How does NGM's graphical representation approach compare with holistic methods such as PointNet [18] for few-shot 3D action learning? (2) How important are learnable edges for capturing node interaction beyond heuristics (*e.g.,* distance)? (3) How does the proposed *graph tensor* representation for learning graph matching function compare with alternatives such inexact graph matching and graph embedding? We answer the questions by comparing NGM with state-of-the-art 3D representations [10, 18], and conduct extensive ablation studies on the design choices of our model.

5.1 Datasets.

We use two 3D action dataset with varieties of human-object interactions, where there exists challenging fine-grained actions to recognize. This is ideal for evaluating few-shot 3D action recognition. Because most existing few-shot approaches rely on the principle of transferring knowledge from seen classes to unseen classes, it is important for the seen and unseen classes to be related. At the same time, the actions should still be fine-grained to have challenges for proper evaluation.
CAD-120. We use CAD-120, a RGB-D video dataset containing over 60,000 video frames of activity performed by 4 subjects (Figure 2). We focus on evaluating the sub-activity labels (*e.g.,* reaching, moving, placing) and their combination with objects in the scene (*e.g.,* bowl, milk, and microwave). These fine-grained interactions with the objects make the classification challenging in few-shot setting. In addition, as the subjects are only given high-level instruction of the action, there can be real-world execution varieties in the videos. For our experiments, we split the dataset into 20 training and 10 testing classes.
PiGraphs. We use the PiGraphs dataset [22], which uses RGB-D sensors to capture common activities, annotated as sets of verb-noun pairs such as `use-laptop`,

Table 1. Few-Shot Action Recognition Results. We compare our method to baseline holistic methods and baseline part-based methods.

Model	CAD-120		PiGraphs	
	1-shot	5-shot	1-shot	5-shot
PointNet [18]	57.2	69.1	60.6	82.5
P-LSTM [10]	60.5	68.1	66.6	71.7
S-RNN [51]	65.4	85.4	–	–
NGM w/o Edges	66.1	85.0	75.9	71.1
NGM	**78.5**	**91.1**	**80.2**	**88.3**

`lie-bed`, and `look-whiteboard`. Annotations also include verb-noun pair compositions, resulting in action classes such as `stand-floor+write-whiteboard` and `sit-chair+look-monitor+type-keyboard`. The dataset contains reconstructed 3D indoor environments, which is ideal for understanding 3D human-object interaction. In addition, the dataset comes with voxel annotation that is not available in the CAD-120 dataset. We utilize the *iGraphs* from the original PiGraphs dataset as our heuristic-derived baseline. We used 32 training and 10 testing classes for our experiments.

For both datasets, nodes were derived from object locations in the dataset. We note that node locations can easily be extracted using a state-of-the-art object detector, but our work is primarily focused on the problem of generating and learning node *relationships*.

5.2 Evaluating 3D Action Representation for Few-Shot Learning

We now evaluate the representations for few-shot 3D action learning and analyze the importance of explicit graphical structure learning. We compare our method to three baselines:

PointNet. PointNet [18] utilizes permutation invariant operators and directly consumes the point cloud as input. This approach has achieved state-of-the-art results on 3D classification and semantic segmentation. We select this baseline as the representative holistic approach without explicitly leveraging the spatial information, and aim to capture the action classification by learning from the whole scene. For a fair comparison, in addition to the point coordinates and RGB values, we also concatenate the detected object type of each point as input to the PointNet.

P-LSTM. Part-aware LSTM (P-LSTM) [10] is an important skeleton based 3D action recognition approach that has been widely used. Unlike PointNet, P-LSTM implicity allows the emergence of structure in the LSTM cell. However, this structure is not explicitly required as in our Neural Graph Matching. In addition to the human joint location, we also feed in the object locations to P-LSTM for a fair comparison.

NGM w/o Edges. We compare our own ablation model without edges as a baseline. In this case, neither the graph learning in graph generation (Section

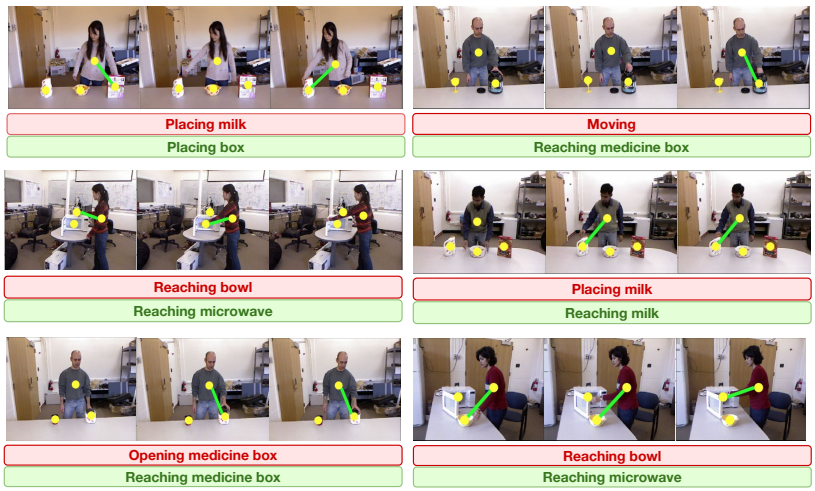

Fig. 4. We show the prediction of our model (green label) and P-LSTM [10] (red label) on the CAD-120 dataset. Each action example is represented by three frames (start, middle, end) of the action clip. The graph is overlaid on each frame, where yellow dots are nodes, and green lines are edges. Our graph based approach is able to correctly predict the action class while raw input data can be confusing the PLSTM.

4.1) nor the graph tensor in graph matching (Section 4.2) would be possible. In this case, the model is reduced a graph embedding model without passing messages between nodes. We choose this baseline to show the importance of learning both the edges and the graph matching tensor.

Results. The 1-shot and 5-shot action recognition results on both the CAD-120 and the PiGraphs datasets are shown in Table 1. It can be seen that NGM significantly outperforms the baselines on both datasets. We can see that, without enough training data in few-shot learning, holistic representation like PointNet cannot learn effective features for classification. On the other hand, while P-LSTM and S-RNN are effective for supervised action recognition, without enough data, the hidden states of these recurrent neural networks are unable to capture the structure of the video. In contrast to the baseline, NGM explicitly leverages the interaction graph as the graphical representation, and uses graph tensor in the graph matching stage to compare not only the vector representation of nodes, but also the structure of the graph through edge matching feature. It is important to note that the performance of "NGM w/o Edges" is significantly lower than our full model. This shows that learning the structure and relationship between objects/nodes in the scene plays an important role for generalizing few-shot learning to novel classes. For comparison, fully supervised results for PiGraphs and CAD-120 are 94.7% and 93.7%. The higher performance in the fully supervised setting demonstrates few-shot learning is more challenging than

Table 2. Graph Learning Ablation Study. We assess the effect of our edge learning. We compare having no edges, heuristic-defined edges, and our learned edges.

Edges	CAD-120		PiGraphs	
	1-shot	5-shot	1-shot	5-shot
None	66.1	85.0	75.9	82.5
Human-Object	74.9	89.7	75.5	71.3
Proximity	77.2	88.1	–	–
Learned	**78.5**	**91.1**	**80.2**	**88.3**

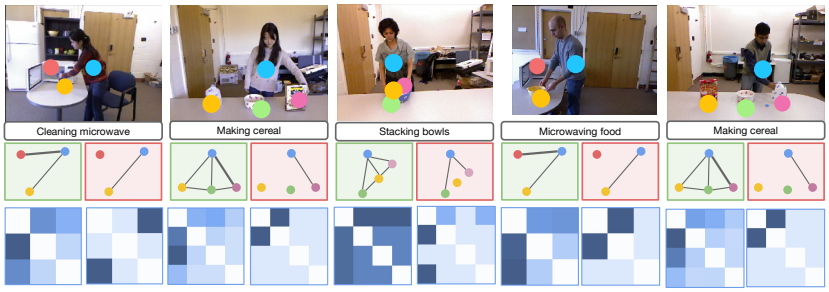

Fig. 5. CAD-120 Graph Learning Results. We show five examples of generated graph (green) compared with the heuristic-defined graph (red) in the middle row. Top row shows the corresponding frame, and bottom row shows the adjacency matrix. It can be seen that NGM generates node relationships that are important to understand the action but not captured in the heuristic edges. For example, NGM automatically generates the edge between human and the microwave in the *cleaning microwave* action.

fully supervised learning on these datasets. In the following sections, we will discuss more thorough analysis of each component of our model.

Qualitative Results. We show qualitative results comparing P-LSTM model (red label) to NGM (green label) in Figure 4. In particular, P-LSTM has difficulty capturing the specific interaction with an object (e.g. placing vs. reaching milk, opening vs. reaching medicine box). In addition, for action sequences where the human is interacting with multiple objects, P-LSTM does not always correctly predict the correct object relevant for the interaction (e.g., placing milk vs. box, reaching bowl vs. microwave). From the graphs shown in Figure 4, we can see that the explicit modeling of the evolution of graphs over time is a useful signal for predicting the correct action. For instance, in the case of *reaching medicine box*, the graph begins with no edges, then creates an edge between the human and the medicine box at later timesteps to represent "reaching".

5.3 Evaluating Edge Learning

Graph Learning is Important. We have shown in Section 5.2 that explicitly learning the object/human relationships through edges plays an important role

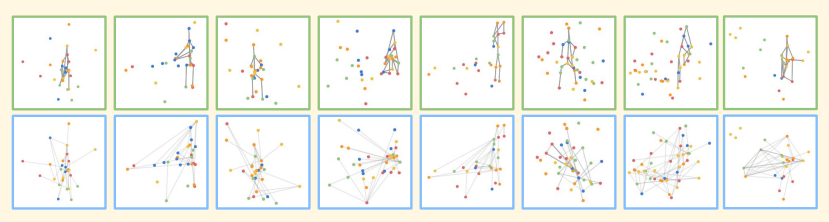

Fig. 6. PiGraphs Graph Learning Results. Comparison of our heuristic graph (top) and our generated graph (bottom). In contrast to the heuristic edges which only contain human joints and immediate objects in contact, our learned edges is able to incorporate further context of the action. While not hand-crafted, the learned edges is still able to capture that this is a human-centered problem and center the edges around the human.

to the success of our method. We now analyze the effect of different edge generation approaches. In addition to "NGM no Edge" (shown as "None" in this section), we consider two heuristics for generating the edges: The first is "Proximity", where we add an edge between two nodes if the 3D locations are close. The second is "Human-Object", which is similar to "Proximity", but we only add the edge if it is between a human joint and a object. The motivation is that it can focus the model on the human-object interaction that are important to understanding the action. The results are shown in Table 2. It can be seen that both of the heuristics can improve over the "None" baseline, but the proposed edge learning approach still performs the best. Inherently, heuristically-defined graphs are sensitive to noise in node location and constrained by hand-crafted rules that are challenging to generalize. For instance, if a human is close to an unrelated object, a proximity-based edge generator would naively create an undesired edge relationship between the human and the object. The forced mixing of unrelated node features such as in this case can affect performance.

Qualitative Results. In contrast to heuristic edge generation techniques, NGM automatically discovers the important node relationships by optimizing the graph structure for the few-shot learning objective, leading to better prediction performance. NGM has the freedom to learn edge semantics beyond specific hand-designed criteria (e.g., proximity) for node linking. We visualize learned edges in Figure 5 and Figure 6. NGM-learned graphs (green) contain node relationships that are not captured in heuristic-defined graphs (red).

Cleaning microwave in Figure 5 links the human node with a cleaning cloth, while NGM additionally links the human with the microwave, even placing stronger weight on the human-microwave than the human-cloth edge, showing that this relationship is critical to predicting the action class. In *making cereal*, we see that the heuristic graph links a naive edge between the human and the cereal. NGM also predicts this edge, but also learned the relevance of the milk jug and the bowl for the cereal-making action, despite the human not being in contact with these objects.

Table 3. Matching Ablation Study. We evaluate the performance of our features matching and our graph structure matching, as well as the combination of both.

Match	CAD-120		PiGraphs	
	1-shot	5-shot	1-shot	5-shot
Features	61.1	78.3	77.8	74.1
Adjacency	60.5	78.2	78.3	74.1
Adjacency & Features	**78.5**	**91.1**	**80.2**	**88.3**

We similarly visualize our learned node relationships on the PiGraphs dataset in Figure 6, in comparison to the heuristic graphs. The heuristic graph (Top) mainly captures the human skeleton and the immediate objects in contact with skeleton joints. Such a representation is akin to several existing methods on pose-based action recognition methods [29, 30]. In contrast, our learned graph captures far more complex relationships in the scene that are directly optimal for predicting the corresponding action class. Our learned edges tend to center towards the human, which is intuitive given that the PiGraphs dataset is focused on human-centric interactions. However, the edges do not contain the human skeleton edges, suggesting that edges between human joints may not actually be crucial to the classification of action scenes.

5.4 Evaluating Graph Matching Representation

In contrast to classical graph matching [47] and graph embeddings [42], our proposed *graph tensor* in Section 4.2 combines both the node representation along with the graph structure to our matching function. We now analyze the importance of combining the continuous feature with the graph structure. The results are shown in Table 3. It can be seen that having only the node representation or the graph structure cannot fully represent the graph for few-shot learning. This shows that there exists complementary information in the holistic graph embedding and the structural adjacency matrix, and the proposed graph tensor is able to leverage and combine both information.

6 Conclusion

We presented Neural Graph Matching (NGM) Networks, a novel few-shot learning framework that leverage the inherent spatial information in 3D through a graphical intermediate representation. NGM consists of two parts: a graph generator, and a graph matching metric, which can be jointly trained in an end-to-end fashion to directly optimize for the few-shot learning objective. We demonstrate that this leads to stronger generalization to unseen classes with only a few-example when compared to both holistic and heuristic-defined approaches.

References

1. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS. (2012)
2. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. (2016)
3. Fabian Caba Heilbron, Victor Escorcia, B.G., Niebles, J.C.: Activitynet: A large-scale video benchmark for human activity understanding. (2015) 961–970
4. Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei, L.: Large-scale video classification with convolutional neural networks. In: CVPR. (2014)
5. Luo, Z., Peng, B., Huang, D.A., Alahi, A., Fei-Fei, L.: Unsupervised learning of long-term motion dynamics for videos. In: CVPR. (2017)
6. Misra, I., Zitnick, C.L., Hebert, M.: Shuffle and learn: unsupervised learning using temporal order verification. In: ECCV. (2016)
7. Lai, K., Bo, L., Fox, D.: Unsupervised feature learning for 3d scene labeling. In: ICRA. (2014)
8. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012 (2015)
9. Koppula, H.S., Gupta, R., Saxena, A.: Learning human activities and object affordances from rgb-d videos. The International Journal of Robotics Research (2013)
10. Shahroudy, A., Liu, J., Ng, T.T., Wang, G.: Ntu rgb+ d: A large scale dataset for 3d human activity analysis. CVPR (2016)
11. Song, S., Lichtenberg, S.P., Xiao, J.: Sun rgb-d: A rgb-d scene understanding benchmark suite. In: CVPR. (2015)
12. Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al.: Matching networks for one shot learning. In: Advances in Neural Information Processing Systems. (2016) 3630–3638
13. Garcia, V., Bruna, J.: Few-shot learning with graph neural networks. In: ICLR. (2018)
14. Snell, J., Swersky, K., Zemel, R.S.: Prototypical networks for few-shot learning. arXiv preprint arXiv:1703.05175 (2017)
15. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. CVPR (2009)
16. Masci, J., Boscaini, D., Bronstein, M., Vandergheynst, P.: Geodesic convolutional neural networks on riemannian manifolds. In: ICCV Workshops. (2015)
17. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: A deep representation for volumetric shapes. In: CVPR. (2015)
18. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. arXiv preprint arXiv:1612.00593 (2016)
19. Andreas, J., Rohrbach, M., Darrell, T., Klein, D.: Neural module networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016) 39–48
20. Johnson, J., Hariharan, B., van der Maaten, L., Hoffman, J., Fei-Fei, L., Zitnick, C.L., Girshick, R.: Inferring and executing programs for visual reasoning. arXiv preprint arXiv:1705.03633 (2017)
21. Hu, R., Andreas, J., Rohrbach, M., Darrell, T., Saenko, K.: Learning to reason: End-to-end module networks for visual question answering. CoRR, abs/1704.05526 **3** (2017)

22. Savva, M., Chang, A.X., Hanrahan, P., Fisher, M., Nießner, M.: Pigraphs: Learning interaction snapshots from observations. ACM Transactions on Graphics (TOG) **35**(4) (2016) 139
23. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. CVPR (2017)
24. Ravi, S., Larochelle, H.: Optimization as a model for few-shot learning. (2016)
25. Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., Lillicrap, T.: Meta-learning with memory-augmented neural networks. In: ICML. (2016)
26. Adam Santoro, David Raposo, D.G.B.M.M.R.P.P.B.T.L.: A simple neural network module for relational reasoning. arXiv preprint arXiv:1706.01427 (2017)
27. Oreifej, O., Liu, Z.: Hon4d: Histogram of oriented 4d normals for activity recognition from depth sequences. In: CVPR. (2013)
28. Rahmani, H., Mahmood, A., Huynh, D.Q., Mian, A.: Hopc: Histogram of oriented principal components of 3d pointclouds for action recognition. In: ECCV. (2014)
29. Wang, J., Liu, Z., Wu, Y., Yuan, J.: Mining actionlet ensemble for action recognition with depth cameras. In: CVPR. (2012)
30. Vemulapalli, R., Arrate, F., Chellappa, R.: Human action recognition by representing 3d skeletons as points in a lie group. In: CVPR. (2014)
31. Li, C., Wang, P., Wang, S., Hou, Y., Li, W.: Skeleton-based action recognition using lstm and cnn. arXiv preprint arXiv:1707.02356 (2017)
32. Liu, M., Chen, C., Meng, F.M., Liu, H.: 3d action recognition using multi-temporal skeleton visualization. CVPR CVPR.2017.391 (2017)
33. Yan, S., Xiong, Y., Lin, D.: Spatial temporal graph convolutional networks for skeleton-based action recognition. arXiv preprint arXiv:1801.07455 (2018)
34. Ikizler, N., Forsyth, D.A.: Searching for complex human activities with no visual examples. International Journal of Computer Vision (2008)
35. Chunhui Gu and Chen Sun and Sudheendra Vijayanarasimhan and Caroline Pantofaru and David A. Ross and George Toderici and Yeqing Li and Susanna Ricco and Rahul Sukthankar and Cordelia Schmid and Jitendra Malik: Ava: A video dataset of spatio-temporally localized atomic visual actions. CoRR, CoRR:1705.08421 (2017)
36. Johnson, J., Krishna, R., Stark, M., Li, L.J., Shamma, D., Bernstein, M., Fei-Fei, L.: Image retrieval using scene graphs. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2015) 3668–3678
37. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM (2014) 701–710
38. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line: Large-scale information network embedding. In: Proceedings of the 24th International Conference on World Wide Web, International World Wide Web Conferences Steering Committee (2015) 1067–1077
39. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, ACM (2016) 855–864
40. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. (2013) 3111–3119
41. Kearnes, S., McCloskey, K., Berndl, M., Pande, V., Riley, P.: Molecular graph convolutions: moving beyond fingerprints. Journal of computer-aided molecular design **30**(8) (2016) 595–608

42. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
43. Garcia, V., Bruna, J.: Few-shot learning with graph neural networks. arXiv preprint arXiv:1711.04043 (2017)
44. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: ICCV. (2017)
45. Lu, C., Krishna, R., Bernstein, M., Fei-Fei, L.: Visual relationship detection with language priors. In: ECCV. (2016)
46. Ullmann, J.R.: An algorithm for subgraph isomorphism. Journal of the ACM (JACM) **23**(1) (1976) 31–42
47. Riesen, K., Jiang, X., Bunke, H.: Exact and inexact graph matching: Methodology and applications. In: Managing and Mining Graph Data. Springer (2010) 217–247
48. Morrison, P., Zou, J.J.: Inexact graph matching using a hierarchy of matching processes. Computational Visual Media **1**(4) (2015) 291–307
49. Cai, H., Zheng, V.W., Chang, K.: A comprehensive survey of graph embedding: Problems, techniques and applications. IEEE Transactions on Knowledge and Data Engineering (2018)
50. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: NIPS. (2016) 3844–3852
51. Jain, A., Zamir, A.R., Savarese, S., Saxena, A.: Structural-rnn: Deep learning on spatio-temporal graphs. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016) 5308–5317
52. Johnson, J.: Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. (2017) CVPR17.

Supplemental Material: Neural Graph Matching Networks

A Importance of 3D

We emphasize that 3D is the key to the success of our method. In the past, spatial inference in 2D has been challenging due to issues such as occlusion and viewpoint variation [52]. In contrast, 3D data allows stronger spatial reasoning of objects and relationships because it provides absolute, rather than relative, locations of scene entities.

A.1 Additional Qualitative Results

We share additional qualitative results of our learned graphs in Figure 10. For each example video, we show the RGB point cloud of the start, middle, and end frames of the video, along with our learned graph projected onto the point cloud coordinates.

B Graph Evolution

B.1 Confusion Matrices

We show confusion matrices of CAD-120 classes in Figure 7 when ignoring graph evolution in our model. We can see that the model primarily confuses actions where the involved objects are similar (*reaching medicine box* vs. *opening medicine box*). The confusion is especially apparent in the case of temporally ambiguous classes (e.g. *reaching* vs. *moving* vs. *placing* plate).

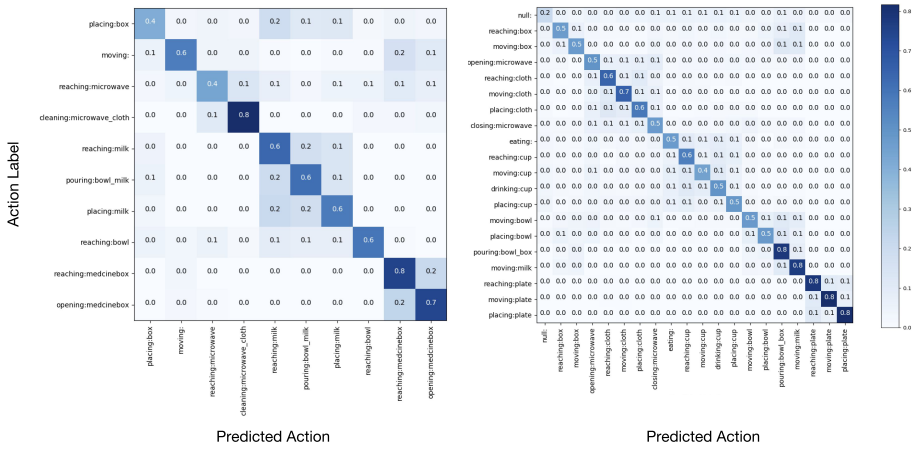

Fig. 7. Confusion matrices of CAD-120 classes without graph evolution (i.e. using only the middle frame of the action clip) on the train (right) and test (left) classes.

B.2 Ablation Experiment: Temporal Graphs

Table 4. Importance of graph evolution. We compare the effect of temporal graphs in our model. We evaluate temporal vs. non-temporal for different versions of our method: i) our method without edges, ii) our method with heuristic edges, iii) our full method. For non-temporal, we use only the middle frame of the video.

Temporal	**NGM (No Edges)**		**NGM (Heuristic)**		**NGM**	
	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
N	54.1	71.5	72.0	87.4	78.1	90.0
Y	**66.1**	**84.0**	**74.9**	**89.7**	**78.5**	**90.1**

From this observation we note the importance of *graph evolution* of 3D vidoes, where the temporal evolution of graph sequences can help disambiguate between temporally ambiguous classes. We evaluated the effect of learning temporal graph sequences in Table 4. We show that evolution of sequences is important in modeling 3D videos. However, we note that the necessity of modeling the temporal evolution is less important with our full model compared to our model without edges. One possible interpretation of this is that the edges learned by our model are able to capture complex interactions that pure node locations are not able to for the purposes of 3D action recognition.

C Additional Method Details

C.1 Episodic Few Shot Training

We show additional details on the process of episodic fewshot training of our NGM networks in Figure 8. The key to the success of our episodic NGM training is that the training and testing procedures are completely mirrored. After splitting classes into training and test classes (i.e., training and testing classes do not overlap), we sample the query and support examples from each class. For each example, we jointly learn to generate and match the graph between support and query examples.

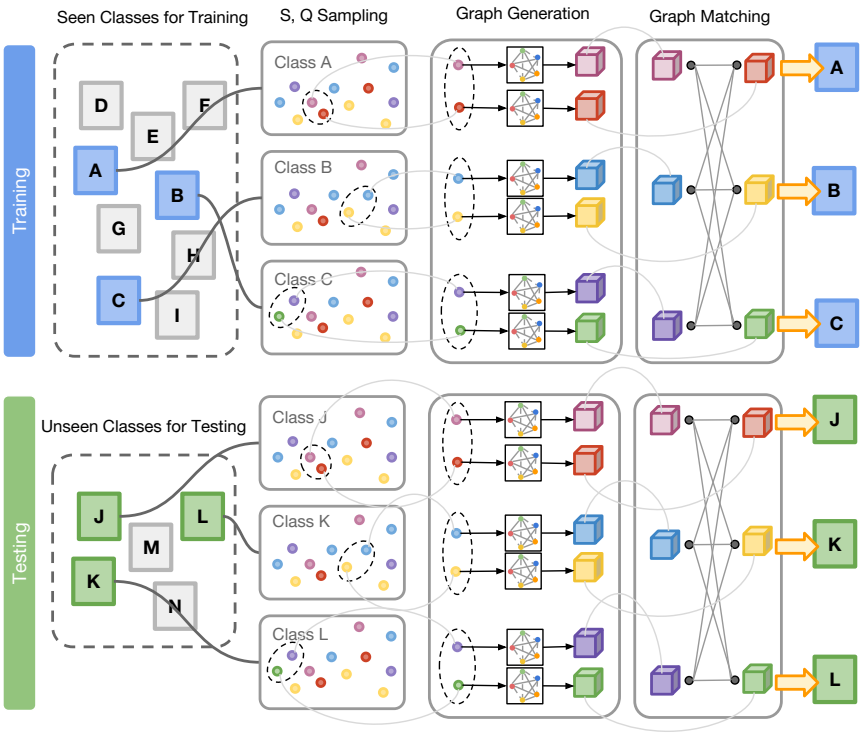

Fig. 8. An overview of the episodic training architecture for NGM networks for 3-way 1-shot 3D action recognition. We split classes into training and test classes, where testing classes are unseen during training. The episodic training procedure is mirrored during the episodic testing procedure. For each episode, we sample 3 classes, and sample one query and one support example from each class. For each example we generate the graph and associated graph tensor, and perform graph matching to find the best matching class.

C.2 Graph Tensor Generation

A more in-depth explanation of the *graph tensor* generation process is shown in Figure 9. Given input video frames, we jointly learn to generate the graph (represented as a graph tensor), with the graph matching function. The graph tensors learned for each video frame are concatenated to form a graph tensor across the entire video. Finally, we match this final video graph tensor to obtain the best matching class.

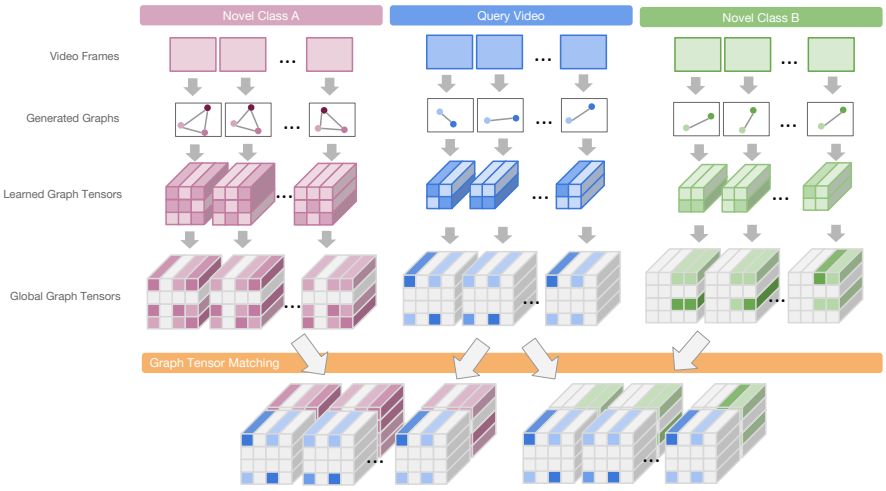

Fig. 9. A visualization of the training and matching process of our NGM *graph tensors* from input video frames. Given input video frames, we jointly learn the graph generation and the graph matching function.

Fig. 10. Qualitative results for CAD-120. For each video example, we visualize the start, middle and end frames, and visualize the i) RGB video, ii) point cloud video, and iii) our learned graph nodes and edges, overlaid on the point cloud. For each node we show the associated object label. Learned edge weights are not depicted in the graph for better visualization effect.