A. Dataset Details

We perform comprehensive experiments on popular graph datasets including citation graph Cora, Citeseer, Pubmed, and ogbn-arXiv [28, 37]. Their statistics are listed in Table 8. Note that the dataset split is a little different from their original settings, as we can only test the final optimized model due to the requirement of lifelong learning, thus we don't need a validation set. For Cora, Citeseer, and Pubmed, the model consists of two feature broadcast layers (7) with $C_{(1)} = 1$ and $C_{(2)} = 2$ channels each. For obgn-arXiv, we found that node features can be easily smoothed due to multiple feature propagation, hence we use the feature transform layers to concatenate its input features. We take the one-hot vector as the target vector $\mathbf{z}_i$, adopt the cross-entropy loss, and use the softsign [11] function $\sigma(x) = x/(1+|x|)$ as the non-linear activation.

Table 8. The statistics of the datasets used for lifelong learning.

Dataset	Nodes	Edges	Classes	Features	Labels
Cora	2,708	5,429	7	1,433	0.421
Citeseer	3,327	4,732	6	3,703	0.337
Pubmed	19,717	44,338	3	500	0.054
obgn-arXiv	169,343	1,166,243	40	128	0.5

B. Proof of Sequence Invariant Sampling

Let P be the probability that the observed items are selected at time t, thus the probability that one item is still kept in the memory after k selection is P^k. This explains that earlier items have lower probability to be kept in the memory and this phenomenon was reported in the Section 4.2 of [2]. To compensate for such effect and

Proposition B.1. *To ensure that all items in the continuum have the same probability to be kept in the memory at any time t, we can set the probability that the n-th item is selected at time t as*

$$P_n(t) = \begin{cases} 1 & t \leqslant M \\ M/n & t > M,\ t = n \\ (n-1)/n & t > M,\ t > n \end{cases}, \tag{10}$$

where M denotes the memory size.

Proof. It is obvious for $t \leqslant M$ as we only need to keep all items in the continuum. For $t > M$, the probability that the n-th item is still kept in the memory at time t is

$$\begin{aligned} P_{n,t} &= P_n(n) \cdot P_n(n+1) \cdots P_n(t-1) \cdot P_n(t), \\ &= \frac{M}{n} \cdot \frac{n}{n+1} \cdot \cdots \frac{t-2}{t-1} \cdot \frac{t-1}{t}, \\ &= \frac{M}{t}. \end{aligned} \tag{11}$$

This means the probability $P_{n,t}$ is irrelevant to n and all items in the continuum share the same probability. In practice, we always keep M items and sample balanced items accross classes. □

C. Distributed Human Action Recognition

Implementation In practice, the temporal growing graph can only be learned sequentially, thus we take the first 80% of each sequence for training and the remaining 20% for testing. Specifically, we define the radius of neighborhood as the temporal distance. Therefore, all the nodes at the same instant are 1-hop neighbors of each other. For each feature graph we have $K = 2$ in the continuum (1). We construct FGN using two feature transform layers (8) with attention weights (9) and one fully connected layer to predict for the sub-graph classification. For fairness, we use $C_{(1)} = 5$, $F_{(1)} = 25$, $C_{(2)} = 32$, and $F_{(2)} = 12$ for all models. In the experiments, we find that GCN, APPNP obtain the best overall performance using the SGD optimizer, while MLP, GAT, and FGN performed the best using the Adam optimizer.

Running time We also report the average running time for the models in Table 9. Note that the efficiency of FGN is on par with other methods. Considering that FGN has a much better performance, we believe that FGN is more promising.

Table 9. Running time comparison on the action recognition.

	MLP	GCN	APPNP	GAT	**FGN**
Runtime (ms)	3.51	5.38	5.36	5.48	5.76

D. Image Feature Matching

Although many hand-crafted feature descriptors such as SIFT [27] and ORB [33] have been proposed decades ago, their performance is still unsatisfied for large view point changes. Due to the well generalization ability, deep learning-based feature detectors have received increasing attentions. For example, SuperPoint [9] introduced a self-supervised framework for extracting interest point detectors and descriptors. SuperGlue [36] introduced graph attention model into SuperPoint for feature matching.

Implementation In the experiments, we adopt $C = 1$, $K = 1$, $F = 256$ in both FGN and FGN for fairness. The training loss function is adapted from [29] which maximizes the likelihood of predicting similar node embeddings corresponding to their spatial location. We recommend the readers refer to SuperPoint [9], SuperGlue [36], and [29] for more details of the loss functions.

Dataset We perform training and evaluation on the TartanAir dataset [46]. TartanAir is a large (about 3TB) and

very challenging visual SLAM dataset consisting of binocular RGB-D video sequences together with additional per-frame information such as camera poses, optical flow, and semantic annotations. The sequences are rendered in AirSim [38], a photo-realistic simulator, which features modeled environments with various themes including urban, rural, nature, domestic, public, sci-fi, *etc*. Figure 4 contains several example video frames from TartanAir. The dataset is collected such that it covers challenging viewpoints and diverse motion patterns. In addition, the dataset also includes other traditionally challenging factors in SLAM tasks such as moving objects, changing lighting conditions, and extreme weather. We randomly select 80% of the sequences for training and take the remaining for testing. We recommend the readers refer to [46] for more details of the dataset.

E. Limitation

Although we have shown that FGN outperformed the state-of-the-art methods in node classification, sub-graph classification, and edge prediction, it also has several limitations. **First**, our current implementation is not vectorized for taking a varying number of neighbors, which is less computationally efficient. **Second**, in the experiments, we assumed scalar edge weights, while in the general case, the graph edge weights are represented by a vector, as defined in the feature graphs. **Third**, since our main contribution is the novel graph topology, *i.e*., feature graph, we mainly compared it with the the state-of-the-art graph models such as GAT by applying an off-the-shelf lifelong learning algorithm. However, it might also be applicable to other lifelong learning algorithms. In the future, we plan to fully optimize the codes, extend it to applications with vector edge weights, and apply more lifelong learning algorithms.