# Multimodal Continual Graph Learning with Neural Architecture Search

Jie Cai[1], Xin Wang[1*], Chaoyu Guan[1], Yateng Tang[2], Jin Xu[2*], Bin Zhong[2], Wenwu Zhu[1*]
1. Tsinghua University, China
2. Data Quality Team, Wechat, Tencent Inc., China
{caij20,guancy19}@mails.tsinghua.edu.cn,{xin_wang,wwzhu}@tsinghua.edu.cn
{fredyttang,harryzhong}@tencent.com,cnjinxu@gmail.com

## ABSTRACT

Continual graph learning is rapidly emerging as an important role in a variety of real-world applications such as online product recommendation systems and social media. While achieving great success, existing works on continual graph learning ignore the information from multiple modalities (e.g., visual and textual features) as well as the rich dynamic structural information hidden in the ever-changing graph data and evolving tasks. However, considering multimodal continual graph learning with evolving topological structures poses great challenges: i) it is unclear how to incorporate the multimodal information into continual graph learning and ii) it is nontrivial to design models that can capture the structure-evolving dynamics in continual graph learning. To tackle these challenges, in this paper we propose a novel Multimodal Structure-evolving Continual Graph Learning (MSCGL) model, which continually learns both the model architecture and the corresponding parameters for Adaptive Multimodal Graph Neural Network (AdaMGNN). To be concrete, our proposed MSCGL model simultaneously takes social information and multimodal information into account to build the multimodal graphs. In order for continually adapting to new tasks without forgetting the old ones, our MSCGL model explores a new strategy with joint optimization of Neural Architecture Search (NAS) and Group Sparse Regularization (GSR) across different tasks. These two parts interact with each other reciprocally, where NAS is expected to explore more promising architectures and GSR is in charge of preserving important information from the previous tasks. We conduct extensive experiments over two real-world multimodal continual graph scenarios to demonstrate the superiority of the proposed MSCGL model. Empirical experiments indicate that both the architectures and weight sharing across different tasks play important roles in affecting the model performances.

## CCS CONCEPTS

• **Computing methodologies → Machine learning**; **Online learning settings**.

## KEYWORDS

continual learning, multimodal graph, neural architecture search

**ACM Reference Format:**
Jie Cai[1], Xin Wang[1*], Chaoyu Guan[1], Yateng Tang[2], Jin Xu[2*], Bin Zhong[2], Wenwu Zhu[1*]. 2022. Multimodal Continual Graph Learning with Neural Architecture Search. In *Proceedings of the ACM Web Conference 2022 (WWW '22), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3485447.3512176

## 1 INTRODUCTION

Continual graph learning has been an emerging research topic which learns from graph data with different tasks coming sequentially, aiming to gradually learn new knowledge without forgetting the old ones across sequentially coming tasks [17, 34, 38]. Nevertheless, existing continual graph learning methods ignore the information hidden in various modalities (e.g., visual and text etc.) as well as the rich dynamic structural information lying in the ever-changing graph data of sequentially coming tasks. On the one hand, many real-world applications such as chemical molecule discovery [13], social media [24, 26] and sentiment analysis [35] show increasing attentions to multimodal information in graphs [40]. On the other hand, we can always construct multimodal graphs with dynamic structural information (e.g., sequences of social media articles with users sharing, commenting, clicking and favoring behaviors) through extracting different modalities as well as social connections across items and users by tracking people who read the same article and people who buy the same product.

Specifically, the multimodal graphs with dynamic structural information and modality information in real-world have the following properties:

- Tasks come in a sequential order with data distribution dynamically changing due to some reasons (e.g., seasonal trends or emergencies);
- Contents contain elements of various modalities including texts, pictures and timestamps, etc.;
- The connections between different contents reflect the dissemination of information or the similarities between different contents, which is also dynamically changing in the course of time.

Given that the optimal model architecture may vary under different tasks with different data distributions [8, 41], when applying graph deep learning models such as graph neural networks (GNNs) to tasks coming sequentially, it is necessary to dynamically learn the

*Corresponding Authors

best architectures for different tasks in order to obtain satisfactory performances.

However, considering multimodal continual graph learning with evolving topological structures and modality information poses great challenges:

(1) It is unclear how to incorporate the multimodal information [39] into continual graph learning. Multimodal continual graphs result in more complex models to consider than single-modal graphs and defining the shared mutlimodal graph neural networks (MGNN) model in the process of continual learning is also not straightforward.
(2) It is nontrivial to design models that can capture the structure-evolving dynamics in continual graph learning. Existing continual graph learning methods can only be applied on the fixed GNN structures. How to share parameters and structures between different tasks while maintaining continual learning ability is important.

Existing continual graph methods [17, 27] do not take any multimodal information into consideration. They design a fixed architecture for every task and ignore the necessity of architecture adjustment to further improve the memory and expansion efficiency, causing the catastrophic forgetting problem of multimodal continual learning with structure-evolving dynamics.

To tackle these challenges, we propose a novel Multimodal Structure-evolving Continual Graph Learning (MSCGL) model, which continually learns both the model architectures and the corresponding parameters for Adaptive MGNN (AdaMGNN). Our MSCGL model is able to augment topology information, feature information and modality information over time without forgetting the information learned in the past.

To solve challenge (1), we design and maintain an expanding network called AdaMGNN to adaptively learn from streaming tasks. For each task, we allow the AdaMGNN to grow in complexity, change in architecture during training and warm-start from existing architectures and weights based on the idea of automated graph machine learning[30]. To handle challenge (2), we add constraints when searching for network architectures, so that the learned architectures have a better ability to learn new tasks without forgetting previous knowledge. We also use parameter sharing to avoid unnecessary parameter storage.

This paper is based on the motivation to use the historical information of multimodal graphs to solve graph learning problems of the new graph, without forgetting the knowledge learned in the past, so as to achieve continual learning of multimodal graph data. Our work is innovative from previous continual graph learning works because these methods ignore multimodal information from graph data and fix the network structure of GNNs, without adaptive changes to the network structure according to different tasks. To summarize, we make the following contributions:

- We present a Multimodal Structure-evolving Continual Graph Learning (MSCGL) model that adaptively explores model architectures without forgetting history information.
- We propose a adaptive MGNN (AdaMGNN) model cooperated with a sharing strategy. The sharing strategy avoids unnecessary architecture extension for similar tasks.
- We conduct sufficient experiments over two real-world multimodal continual graph scenarios to evaluate the proposed MSCGL framework. Empirical evidences indicate that both the evolving architectures and weight sharing across different tasks play important roles in affecting the model performances.

We discuss related work in Section 2, formulate the problem in Section 3, present our proposed model in Section 4, describe our experiments in Section 5 and conclude the paper in Section 6.

## 2 RELATED WORK

### 2.1 Multimodal Graph Neural Networks

Due to the success of graph learning in information aggregation, transmition and the maturity of multimodal learning, some scholars focus on multimodal graph learning for effective use of multimodal dependencies and information dissemination relationships. Multimodal graph neural network is a kind of deep graph learning model that aims to represent multimodal graph-structured data in an end-to-end way. [22, 33] We et al. [25] assign a graph convolution network for each modal to capture the representation of each node hidden in modal specific user-item bipartite graph. Tao et al. [31] not only use neighbor-ware attention to model the similarity between different users and different items, but also use gated attention mechanism to identify importance scores of different modalities to user preference. Gao et al. [7] dynamically update the representations of nodes by three attention-based aggregators which guide the message passing between modalities. Although these multimodal graph learning methods have made great success, the existing multimodal graph learning methods are designed for static graphs. That is, they cannot not be directly used in continual learning scenarios because of the catastrophic forgetting problem.

### 2.2 Continual Graph Learning

Continual graph learning aims to gradually extend the acquired knowledge when graph-structured data come in an infinite streaming way which successfully solve the catastrophic forgetting problem [2]. Existing continual graph learning methods can be divided into two categories: Replay-based methods that stores representative history data or well-designed representation of data; Regularization-based methods that append regular terms to the loss function to limit changes of the past knowledge.

GraphSAIL [34] stores the local structure, global structure, and self-information of each node explicitly using distillation mechanism. Liu et al. [17] propose topology-aware weight preserving (TWP) that explicitly learns the local structure of the input graph and the topology aggregation mechanism in an attempt to stabilize the parameters that play a key role in topology aggregation. Wang et al. [28] combine data replay and model regularization to preserve for existing patterns. Zhou et al. [38] select some important nodes in the past graphs as experience nodes and save them for playback when the new diagrams are trained. Galke et al. [6] systematically analyze the influence of knowledge stored explicitly as historical data or implicitly in model parameters. However, this method just uses warm restart or cold restart for each new task.

These articles represent two perspectives of continual graph learning – data perspective and model perspective. Our work is from model perspective, which can be also cast as a regularization-based method. One of the shortcomings of the above continual graph learning methods is that they fix the GNN structure, which is a notable difference between existing methods and our model. If the change of data distribution is dramatic, the history model architecture may not perform well for new data.

## 2.3 Continual Neural Architecture Search

Neural architecture search (NAS) has been a hot topic in recent years. It intends to automatically search for the most effective model architecture for a specific deep learning problems without human intervention. Different graph data vary greatly in structure, content, and task, and the befitting GNNs can vary greatly, so scientists also pay close attention to graph neural architecture search [9, 10, 15, 29, 36, 37]. Gao et al. [8] propose GraphNAS based on reinforcement learning. A recurrent network is used to generate the description of GNN, and is trained with reinforcement learning to maximize the expected accuracy of the generated architectures.

Pasunuru et al. [21] propose a new continual architecture search (CAS) method to evolve both model architecture and model parameters during continual training of multiple tasks without losing the performance of previously learned tasks. Hu et al. [12] propose Petridish to iteratively add shortcut connections to existing network layers, which can be used for warm-starting in lifelong-learning scenarios. With the idea of using NAS to find more competitive model for new tasks, Fu et al. [5] use weight sharing and knowledge distillation to shorten the training time and remember old classes. Niu et al. [20] propose Adaptive eXpert (AdaXpert) to adjust model architecture on the growing data. They adjust the model structure according to data distributions of different datasets. However, the above method are proposed for CNNs or RNNs. There is no continual neural architecture search methods for GNNs.

## 3 PROBLEM DEFINITION

In this section, we design the general formulation of the multimodal structure-evolving continual graph learning problem. In short, the problem is how to learn a multimodal GNN $f_\theta$ that can sequentially learn on coming multimodal graphs $G_1, \cdots, G_t$ successively.

Definition 1. *A* ***streaming multimodal graph*** *is a sequence of multimodal graphs $\mathcal{G} = \{G_1, G_2, \cdots, G_{t-1}, G_t, \cdots\}$, where each multimodal graph $G = \{V, E\}$. $V = \{v_1, ..., v_N\}$ denotes $N$ nodes, $E = \{\langle v_i, v_j \rangle | 1 \leq i, j \leq N\}$ denotes the set of edges. For each node $v_i \in V$, $v_i$ corresponds to multimodal node features $X_i$ and a category label $y_i \in Y_t = \{0, 1, 2, \cdots\}$.*

In our continual learning settings, different tasks correspond to different multimodal graphs $G$ and label sets $Y$. Each task is a two-category node classification task and each node has two modalities: vision modality and textual modality.

Definition 2. *Given a sequence of multimodal graphs $\mathcal{G} = \{G_1, G_2, \cdots, G_{t-1}, G_t, \cdots\}$, each graph $G_t$ corresponds to a task $\mathcal{T}_t$. Each task $\mathcal{T}_t$ contains a training node set $V_t^{tr}$ and a testing node set $V_t^{tr}$, with corresponding feature sets $X_t^{tr}$ and $X_t^{te}$.* ***Continual learning for streaming multimodal graph*** *aims to learn these tasks sequentially without catastrophic forgetting problem.*

**Table 1: Glossary of Notations**

| Notation | Description |
|---|---|
| $\mathcal{T}_t, G_t$ | the $t$-th task and corresponding multimodal graph |
| $\mathbf{X}_t, \mathbf{y}_t$ | feature matrix and label vector of the $t$-th task |
| $m$ | modality, i.e. vision and textual modality |
| $h_u^{(l)}(h_{u,m}^{(l)})$ | the $l$-th layer representation of node $u$ (of modality $m$) |
| $W_t$ | the parameters of the $t$-th MGNN model |
| $A_t$ | the architecture of the $t$-th MGNN model |

In order not to take up too many resources, we hope to solve the problem within the following two constraints: Firstly, at each snapshot, we don't have history graphs. Different from continual learning methods on data perspective, our method doesn't replay any experience nodes. However, our model is compatible with experience replay methods because we don't require additional assumption on data. Secondly, the model should not only perform well on the current task but also overcome catastrophic forgetting problem with respect to the previous tasks. That means we need to reduce unnecessary changes to parameters that are important to previous tasks. Thus we propose the following definition of multimodal structure-evolving continual graph learning.

Definition 3. *The goal of* ***multimodal structure-evolving continual graph learning*** *is to find the optimal multimodal GNN architecture and model parameters that satisfies:*

$$(A_t^*, W_t^*) = argmin_{(A_t, W_t) \in (\mathcal{A}, \mathcal{W})} \mathcal{L}_M(f_{(A_t, W_t)}(G_t | A_{t-1}, W_{t-1})) \tag{1}$$

*where $A_t^*$ and $W_t^*$ are the best architecture and model parameters for task $\mathcal{T}_t$, $\mathcal{A}$ and $\mathcal{W}$ is the search space of model architectures and parameters.*

What is different from structure-fixed continual learning methods is that the model structure changes while the new task comes. It's easy to understand that if the data distribution changes, the history model architecture may not be the optimized one for the new data. We only save the model from last task because multimodal graphs lead to more complex models to consider than ordinary graph learning problems. The model cannot be too complex and we cannot save the model for every task because of massive parameters.

## 4 METHODS

Here the question arises: how to avoid catastrophic forgetting problem caused by saving specific sub-models from the last task? In this section, we introduce the details of our multimodal and structure-evolving continual graph learning (MSCGL) method.

## 4.1 Overall MSCGL framework

The life-cycle of MSCGL framework comprises four stages, i.e., data processing, neural architecture search, searched model training and maintenance as summarized in Figure 1. In the data processing stage, we process raw data and construct a multimodal graph neural network named AdaMGNN (detailed in Section 4.2). In the neural architecture search stage, we search for the architecture and parameters jointly that not only remember past knowledge but

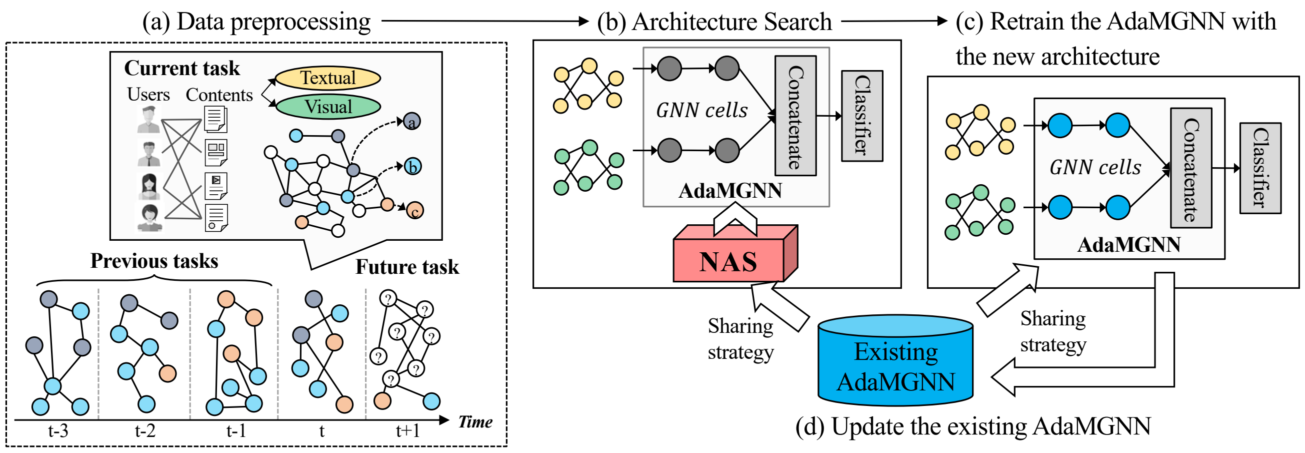


**Figure 1: The overall framework of the proposed MSCGL model.**
**(a)** Data preprocessing stage. A sequential of multimodal graphs contains textual and visual modalities is received. In this step, we extract multimodal features using ViT and Bert. **(b)** Architecture search stage. Yellow graph and green graph denote graph of textual and visual modalities, respectively. In this stage, we use NAS to search for the best architecture of the current task. **(c)** Retraining stage. We retrain the model found in the architecture search stage (blue color) with regularization for shared parameters. **(d)** Maintenance stage. We merge the retrained model with the current AdaMGNN model by sharing strategy.

also perform well on new tasks. In the shared model training stage, we retrain the best architecture found in the neural architecture search stage. In the maintenance stage, we explicitly remove obsolete blocks of old architectures and equips the model with new blocks that save the knowledge from both the new task and history tasks.

## 4.2 Multimodal Graph Neural Network

In this section, we introduce our AdaMGNN, which serves as a shared model through all stage of our MSCGL framework.

Since AdaMGNN needs to be shared and evolved between different tasks and different architectures, the design of our AdaMGNN is totally different from previous multimodal GNNs. In order to better collaborate with continual learning for structure evolving multimodal graphs, AdaMGNN need to 1) fully leverage the connections between different nodes; 2) organize different modalities in a unified manner for further node classification; 3) the architecture can be adjusted adaptively according to the coming tasks. To achieve these demands, we inherit and extend the framework of GraphNAS. Our AdaMGNN model consists of two components - GNN cells and prediction layer.

*4.2.1 GNN cells.* In GNN cells, we model the information propagation and aggregation under different modals. We represent the data from each modality as a graph $G_m = (V_m, E)$, where $V_m = \{X^m | X^m \in \mathcal{I}_m\}$ and $E = \{(i,j)\}$. $\mathcal{I}^m$ denote the set of features from a specific modality $m$. $m \in \mathcal{M} = \{v, t\}$ denote the visual and textual features, respectively.

The GNN cell in the $l$-th layer of modality $m$ updates node feature $h_{v,m}$ for each node $v$ by aggregating its neighborhoods as

$$h_{u,m}^{(l)} = \sigma(W_m^{(l)} \cdot \Phi_{l,m}(\{h_{v,m}^{(l-1)} : v \in N(u)\})), \quad (2)$$

where $\Phi_{l,m}$ is the aggregation function including correlation computation and aggregation operation, $\sigma$ is the activation function. $W_m^{(l)}$ is the network weight, $h_{v,m}^{(l-1)}$ is the output of the last layer or the input feature for $l - 1 = 0$, $N(u)$ is the receptive field of the node $u$.

For $\Phi_{l,m}$, we firstly calculate correlation coefficient $e_{u,v,m}^{(l)}$ for each node $v \in N(u)$, then we aggregate information from neighborhoods as listed in Table 2. Formally,

$$\Phi_{l,m}(\{h_{v,m}^{(l-1)} : v \in N(u)\}) = Agg(\{e_{u,v,m}^{(l)} h_{v,m}^{(l-1)} : v \in N(u)\}). \quad (3)$$

*4.2.2 Prediction Layer.* After updating the representations of nodes in a particular modality $m$, we combine the representations of different modality into a new representation, which can be expressed as:

$$h'_u = h_{u,v} || h_{u,t} \quad (4)$$

We let the concagation of the output of final GNN layers $h'_u$ to be the input of the classification layer, and use a linear layer to predict the label of each node:

$$\hat{y}_u = softmax(tanh(W h_u^{(l)} + b)), \quad (5)$$

where $W$ is the trainable weight matrix and $b$ is the bias vector. We use softmax function to obtain the final prediction score $\hat{y}_u$.

## 4.3 Multimodal Graph Neural Architecture Search

One challenge is that existing continual learning methods need to store a large number of network parameters, so the spatial complexity is high. Thus, we use NAS to determine the new network architecture, and the new network architecture shares part of the network structure and weights with the history network architectures.

Table 2: Operators of search space $\mathcal{M}$

| Action | Operator | Value |
|---|---|---|
| aggregation | sum | $\sum_{j\in N_u} h_u$ |
| | mean | $1/\lvert N_u\rvert \sum_{j\in N_u} h_u$ |
| | max | $\max_{j\in N_u} h_u$ |
| | mlp | $MLP((1+\epsilon)h_u + \sum_{v\in N(u)} h_v)$ |
| activation | / | tanh, relu, identity, softplus, leaky_relu, relu6, elu |
| correlation | const | $e^{con}_{uv} = 1$ |
| | gcn | $e^{gcn}_{uv} = 1/\sqrt{d_u d_v}$ |
| | gat | $e^{gat}_{uv} = LeakyReLU(W_l * h_u + W_r * h_v)$ |
| | sym-gat | $e^{sgat}_{uv} = e^{gat}_{uv} + e^{gat}_{vu}$ |
| | mgat | $e^{mgat}_{uv} = e^{gat}_{uv} * \sigma(h_u * h_v/\sqrt{d_u d_v})$ |
| | cos | $e^{cos}_{uv} = \langle W_l * h_u, W_r * h_u\rangle$ |
| | linear | $e^{lin}_{uv} = tanh(sum(W_r * h_v))$ |

Our method is based on GraphNAS, which enables automatic design of the best graph neural architecture based on reinforcement learning. Given the search space $\mathcal{M}$ of AdaMGNN, we aim to find the best architecture $m^* \in \mathcal{M}$ that not only maximizes the accuracy $\mathcal{R}$ of the network on a validation set $D$, but also remember the learned knowledge from the past tasks.

In the remaining part of this section, we introduce a novel search space for AdaMGNN. Then we formulate the search process as an optimization problem and train the controller to search for the model architecture and weights jointly. At the last part, we show more details of controller training loss that helps to search for the model with best memory ability.

**Search Space.** We define the search space $\mathcal{M}$ as follows: the correlation measure dimension $Att$, the aggregation dimension $Agg$ and the activation function $Act$. We generate the architecture descriptions as a sequence of tokens, while each token corresponds to the operation of each AdaMGNN layer. Similar to GraphNAS, we design the search space of the components of the AdaMGNN layers in Table 2.

**Train the controller.** The training of controller is similar with GraphNAS. Letting $P(a;\theta)$ be the distribution of architecture $a$ parameterized by the choice of controller $\theta$, the aim is to maximize the expected accuracy $\mathcal{E}_{P(a;\theta)}[\mathcal{R}(a(w^*, G))]$ with minimizing the training loss $\mathcal{L}_{train}(a(w, G))$, which can be represented as a bi-level optimization problem listed below:

$$\max_{w} \mathcal{E}[\mathcal{R}(a(w^*, G))], \tag{6}$$

$$s.t. \;\; w^* = \arg\min_{w} \mathcal{L}_{train}(a(w, G)). \tag{7}$$

Here $w^*$ is the shared weights according to the sharing strategy in Section 4.4. Different from GraphNAS, our training loss is defined as follows:

$$\mathcal{L}_{train}(y_u, \hat{y}_u) = -\frac{1}{M}\sum_{i=0}^{N}(y_{u_i}\log\hat{y}_{u_i} - (1-y_{u_i})\log(1-\hat{y}_{u_i})) + \mathcal{L}_c, \tag{8}$$

where $\mathcal{L}_c$ is the regularization item aiming to find architectures that is qualified for continual learning. $y_{u_i}$ is the actual label, $\hat{y}_{u_i}$ is the predictive scores, $N$ is the batch size.

**Regularization for shared parameters.** We define $\mathcal{L}_c$ as a combination of two parts:

$$\mathcal{L}_c = \lambda_b \mathcal{L}_b + \lambda_o \mathcal{L}_o, \tag{9}$$

where

$$\mathcal{L}_b = \sum_{W}\sum_{i} ||W[i,:]||_2, \tag{10}$$

$$\mathcal{L}_o = \sum_{W_{ref}, W\in\mathcal{W}} ||W^T_{ref}\cdot\psi_{ref}||^2_2, \tag{11}$$

where $\lambda_b$ is the block sparse coefficient, $\lambda_o$ is the orthogonal coefficient. $\mathcal{W}$ is the space of shared parameters. $\psi_{ref} = W - W_{ref}$ is a learnable parameter which is block sparse and orthogonal to $W^T_{ref,k}$. These two regularizations can be seen as constraints for the search space of AdaMGNN parameters. We want to search for the parameters that are both block sparse and orthogonal to previous shared parameters.

## 4.4 Multimodal Structure-evolving Continual Graph Learning

Without loss of generality, we assume to have a finite sequence of $t$ tasks $\mathcal{T}_1, \cdots, \mathcal{T}_t$ and we maintain an AdaMGNN model with shared parameters $\theta_{shared}$.

Close to the idea of ENAS[12] that all the architectures which NAS ends up iterating over can be viewed as sub-graphs of a larger architecture graph, we represent the architecture of AdaMGNN as a graph that the topology is fixed. Each node in the graph represents the local computations and the edges represent the flow of information. Local computations on each node have their own parameters, which are only used when a particular computation is active. Therefore, MSCGL allows parameter sharing, orthogonal learning and warm-starting.

In the following, we facilitate the discussion of AdaMGNN with an example that illustrates our sharing strategy and continual learning strategy for a single GNN cell. We illustrate the AdaMGNN via a simple example with layers number $L = 2$ and modality number $M = 2$. To create an AdaMGNN architecture, the controller samples 4 blocks of decisions, each decision is used to create a GNN cell.

Step 1 Assume the best correlation function is found to be *mgat*, then the node *mgat* should be considered in block sparse regularization. The parameters of *mgat* are saved in node *mgat*.

Step 2 The best correlation function is found to be *mgat* again, which means that this node should be considered in both block sparse regularization and orthogonal regularization. After retraining of shared model, the parameters of node *mgat* in Step 1 are cover by the new parameters.

Step 3 The best correlation function is found to be *gat*, thus node *gat* only need be computed in block sparse regularization. The parameters of node *gat* are saved.

**Sharing Strategy.** We have explored different sharing strategies including: 1) Share parameters between the same GNN cells; 2) Share parameters between the same aggregations; 3) Share parameters between all aggregations. However, only the second strategy works well. In this way, when training AdaMGNN for a new task $T_t$, MSCGL will check for each GNN cell whether the aggregation action has parameters in the last GNN cells. If so, we will add two constructions on the training of this parameter. If not, we will only add block sparse construction.

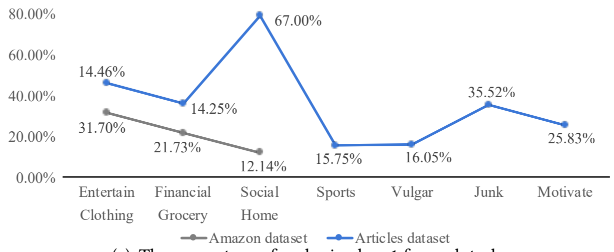
**Algorithm 1:** Multimodal and Structure-evolving Continual Graph Learning

```
Input: Sequences of node classification tasks,
       $\mathcal{T}_0, T_1, \cdots, \mathcal{T}_N$; Shared model $f$ with parameters
       $\theta_{share}$;
Output: Predicted labels for nodes of each task $\mathcal{T}_t$;
1 for $t \leftarrow 1$ to $N$ do
2     while not converge do
3         Controller samples architectures set $\mathcal{M}$ from search
            space;
4         for $M$ in $\mathcal{M}$ do
5             $\theta' \leftarrow$ sharing_strategy($\theta_{share}$);
6             Train AdaMGNN($\theta', M, G_t, \mathbf{X}_t, \mathbf{y}_t$);
7         Train controller;
8     $\theta_{share} \leftarrow$ Train AdaMGNN($\theta', M_t, G_t, \mathbf{X}_t, \mathbf{y}_t$);
9     $\hat{\mathbf{y}}_{pred,t} \leftarrow$ Predict using
        AdaMGNN($\theta_{share}, M_t, G_t, \mathbf{X}_t$);
```

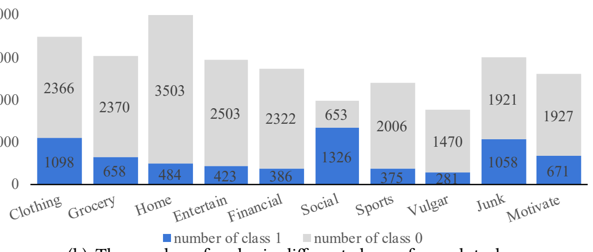
**Group sparse regularization.** Many scientists have conducted profound and extensive studies on the sparsity of neural networks. They design a series of network compression methods, such as low-rank approximation, network pruning, network quantization, knowledge distillation and so on[19]. Scardapane S et al. [23] propose group sparse regularization for deep neural networks. This method considers the following optimization tasks simultaneously: removing unnecessary weight of deep neural network, reducing the number of hidden layer neurons and doing input feature selection. Pasunuru et al. [21] propose two restrictions when searching for weights: the network parameters are block-sparse, and the changes of network parameters are block-sparse and orthogonal to the old network parameters. In this way, the new architecture and parameters have less impact on past tasks. Although these two restrictions are originally designed for DNN, our experiments also verify the effectiveness of this method for GNN.

## 5 EXPERIMENTS

In this section, we perform various experiments to verify the effectiveness of the proposed MSCGL method. We give detailed information about the datasets and the baselines we compare in Section 5.1 and 5.2 respectively. The metrics we use to evaluate models are introduced in Section 5.3. The implementation details are described in Section 5.4. The comparison results between MSCGL and other baselines are given in Section 5.5. At last, we verify the effectiveness of each part in MSCGL through ablation study in Section 5.6.



(a) The percentage of nodes in class 1 for each task.



(b) The number of nodes in different classes for each task.

**Figure 2: Statistics of Amazon dataset and Articles dataset.**

### 5.1 Datasets

We evaluate the proposed MSCGL together with several popular state-of-the-art baselines on two real-world multi-modal node classification datasets. Each dataset is split into several sub-tasks with no overlapped label space. We form these sub-tasks in a streaming manner to mimic the real-world scenarios that data always come with remarkably different new classes compared to the previous data. Continual learning is to handle the upcoming new tasks while preserving performances on previous tasks.

**Amazon Dataset**: We extract our Amazon Dataset from Amazon product data [11] which contains product data with product name, URL of the product image and related products from different product categories. Each node represents a product with two modality: visual image and textual name. Two nodes are connected if there exists one user viewing both of them. The labels of nodes are generated according to the fine-grained categories the products belong to. We construct 3 tasks for streaming setting: Clothing, Grocery, and Home respectively.

**Articles Dataset**: The dataset is constructed from the articles on Tencent Wechat. Each article corresponds to two modality: visual head images and textual titles. Two articles are connected if there exists one user viewing both of them. Each article is labeled according to the quality of the content. We construct 7 tasks: Junk, Motivate, Vulgar, Social, Financial, Entertain and Sports.

For each dataset, we use the open-source implementations [32] of pre-trained Bert [3] to extract the textual features and pre-trained Vision Transformer (ViT) [4] to extract the visual features. Each dataset is pre-processed following the transductive setting: Input graphs can be observed in all dataset segmentation (training, validation, and test sets). We split nodes and labels into training, validation, and test sets in a 4:2:2 ratio.

## 5.2 Comparative Methods

We accommodate several state-of-the-art continual graph learning baselines. To make them support multimodal settings, the visual and textual embedding of nodes are concatenated and the comparative methods are listed below.

- **MSCGL (ours)**: It is a continual learning method tailored for multimodal GNN. MSCGL explores a new strategy with joint optimization of NAS and GSR across sequential tasks.
- **Topology-aware Weight Preserving (TWP)**[17]: It explicitly studies the topology aggregation mechanism of different tasks, captures the topology information of graphs, and finds the key parameters that are important for both task-related and topology-related goals.
- **Learning Without Forgetting (LWF)**[16]: This method can be seen as the combination of knowledge distill technology and fine-tune. Different from fine-tune, LWF uses knowledge distill loss to encourage the output of the new network close to the output of the old network.
- **Elastic Weight Consolidation (EWC)**[14]: As a regularization based method, EWC maintains parameters that are important to previously learned tasks by limiting model parameters close to the old values when new task comes.
- **Memory aware synapses (MAS)**[1]: It computes the importance of the parameters of a neural network. When learning a new task, changes to important parameters are penalized, which effectively prevent important knowledge related to previous tasks from being overwritten.
- **Gradient Episodic Memory (GEM)**[18]: For each task, it uses episodic memory to store a subset of the observed examples. When learning for the new task, the loss for old task is restricted to not increasing.

We also compare MSCGL with fine-tune and joint-train. For the fine-tune method, we directly fine tune the trained model on the new tasks without considering the catastrophic forgetting problem. For the joint-train, we train all the tasks jointly and evaluate on each task. Note that joint-train is an approximate upper bound for the continual learning settings since no forgetting phenomenon exists. We leverage MGAT, a strong multimodal graph learning state-of-the-art as the base model in these baselines.

## 5.3 Metrics

Firstly, we use classification accuracy as the primary evaluation measure for each task. In order to measure the memorizing and learning abilities of our continual graph learning method, we use average performance (AP) and average forgetting (AF) as in [38] and [17]. AP is the average test performance of all tasks of the final model. AF is the average difference between the final model performance and the intermediary model performance of each task.

## 5.4 Implementation Details

**Details of AdaMGNN.** For each task, we empirically set the dimension number of the hidden states of AdaMGNN to be 256, and the dimension number of fusion layer to be 64. We fix the layer number of AdaMGNN to 2, and use concatenation and sum as our fusion and aggregation strategy. We use an initial learning rate of $1e-4$ and an early stopping strategy according to the cross-entropy loss and accuracy on the validation datasets, with a patience of 100 epochs.

**Table 3: Predictive performance of each step on Amazon dataset.**

| Method | Amazon | | Articles | |
|---|---|---|---|---|
| | AP | AF | AP | AF |
| Fine-tune | 75.61 | -19.98 | 61.01 | -39.1 |
| LWF | 78.78 | -14.14 | 86.69 | -2.12 |
| EWC | 86.38 | -3.10 | 82.17 | -6.41 |
| MAS | 85.20 | -0.85 | 86.37 | **0.17** |
| GEM | 77.93 | 2.48 | 80.68 | -10.39 |
| TWP | 83.27 | -0.22 | **86.70** | -0.26 |
| **MSCGL** | **89.44** | **0** | **86.70** | -0.89 |
| Joint Train | 90.04 | - | 91.93 | - |

**Table 4: Predictive performance on Amazon dataset and Articles dataset.**

| Method | Amazon | | | Articles | | |
|---|---|---|---|---|---|---|
| | Clothing | Grocery | Home | Entertain | Financial | Motivate |
| Step-1 | 83.42 | - | - | 88.40 | - | - |
| Step-2 | 83.42 | 88.14 | - | 87.71 | 86.37 | - |
| Step-3 | 83.42 | 88.14 | 96.74 | 87.88 | 85.27 | 86.95 |
| Joint Train | 83.42 | 89.46 | 97.24 | 83.96 | 94.29 | 87.52 |

**Details of architecture search.** Similar to GraphNAS, the controller is a one-layer LSTM with 100 hidden units. The controller is set to run 50 steps. Once the controller samples an architecture, the constructed child model is randomly initialized and trained for 300 epochs without parameter sharing. During training, we apply both block sparse loss term and orthogonal loss term with coefficients 1e-3 and 1e-3. After the controller trains 400 architectures, we select the best architecture and retrain it on new dataset with block sparse regularization and orthogonal regularization for 300 epochs.

**Details of baselines.** We use an open-source implementation of continual graph learning methods and the experimental settings are similar with [17]. The visual and textual features are concatenated directly to make them support multimodal settings.

## 5.5 Study on Multimodal Continual Graphs

In Table 3, we compare the baseline methods against our proposed method MSCGL on two real-world dataset for three steps. For Amazon dataset, we use Clothing, Grocery and Home to construct multimodal graphs. For Articles dataset, we use Entertain, Financial and Motivate to construct multimodal graphs. MSCGL significantly and consistently outperforms the baselines for both Amazon dataset and Articles dataset. For Amazon dataset, the AP of MSCGL is close to the AP of joint train model. The fine-tune model achieves the worst performance among all baselines, reflecting the catastrophic

**Table 5: Predictive performance of each step on Articles dataset.**

| Models | Articles | | | | | | |
|---|---|---|---|---|---|---|---|
| | Entertain | Financial | Social | Sports | Vulgar | Junk | Motivate |
| Step-1 | 90.10 | - | - | - | - | - | - |
| Step-2 | 90.10 | 91.16 | - | - | - | - | - |
| Step-3 | 90.10 | 89.69 | 77.58 | - | - | - | - |
| Step-4 | 89.59 | 88.03 | 72.54 | 88.89 | - | - | - |
| Step-5 | 90.10 | 89.69 | 76.32 | 88.89 | 86.61 | - | - |
| Step-6 | 90.10 | 89.69 | 76.32 | 88.89 | 86.61 | 73.53 | - |
| Step-7 | 90.10 | 80.29 | 73.05 | 86.58 | 86.61 | 73.20 | 85.22 |

**Table 6: Ablation experiment about our regularization methods. C1 is block sparse coefficient. C2 is orthogonal coefficient.**

| Models | AP | AF |
|---|---|---|
| None | 62.81 | -36.77 |
| C1 | 81.25 | -6.81 |
| C2 | 71.43 | -21.99 |
| **C1 + C2** | **86.70** | **-0.89** |

**Table 7: Predictive performance with different coefficients. C1 is block sparse coefficient. C2 is orthogonal coefficient.**

| Hyperparameter of C1 and C2 | AP | AF |
|---|---|---|
| 1e-2 | 78.47 | - 0.915 |
| 1e-3 | 91.20 | -2.10 |
| 1e-4 | 84.14 | -26.28 |

forgetting problem of multimodal GNNs. Another point of concern is that although TWP is originally designed for GNNs, its performance on multimodal graphs is close to that of MAS.

In Table 4, we show the performance for each task at each step. For Amazon dataset, the GNN cells searched by MSCGL are totally different between different tasks, that's why our model doesn't forget any history knowledge. It is worth mentioning that although the model architecture is different, the performance is competitive with the performance of joint train. For Articles dataset, the GNN cells searched by MSCGL are overlapped between different tasks, so there's a slight decrease in the accuracy of the model between different tasks.

On the contrary, the continual graph learning baselines cannot perform well simultaneously on both datasets. For example, TWP performs as well as MSCGL on Amazon dataset but performs far less than MSCGL on Articles dataset. That reflects another shortcomings of existing continual graph learning methods: they perform well on specific datasets and can not be generalized to datasets of various domains because their memory ability is single-faceted. In comparison, our MSCGL model leverages NAS to keep current most informative sub-model while automatically finding the best additional sub-model specialized for the new tasks.

## 5.6 Ablation Study

**Long-term memory ability.** In Table 5, we show the long-term memory ability of our MSCGL framework on Articles dataset. In this experiment, we use random search for better ablation study of our sharing strategy. The PM is 82.15% and the AF is 3.0%. For most tasks except Financial, the accuracy maintains in a stable range.

**Effectiveness of regularization terms.** We verify the effectiveness of the block sparse (C1) and orthogonal (C2) terms and show the result in Table 6. Compared with baseline with no regularization terms, both C1 and C2 can lead to significant improvement on AP and AF. Moreover, leveraging both C1 and C2 results in further improvement on both metrics.

**Hyperparameter sensitivity.** We also study the effect of changing block sparse regularization coefficient and orthogonal regularization coefficient. In this experiment, we fix the model architecture of different tasks to be GAT and the dataset is Amazon with 3 tasks. Table 7 compares the performance of MSCGL when adopting different coefficient values $1e-2$, $1e-3$ and $1e-4$. From the result, we can see well-designed hyperparameters are needed to balance the learning ability and memory ability of AdaMGNN. Specifically, a larger block sparse coefficient will leave more space for AdaMGNN to storage history information that enhance the effect of memory. However, it will also result in limited learning ability for new tasks because a more compact model is learned.

## 6 CONCLUSION

In this paper, we propose a novel MSCGL model, which continually learns both the model architecture and the corresponding parameters for sequential multimodal graph tasks. In order for continually adapting to new tasks without forgetting the old ones, our MSCGL model considers both the multimodal and structural information to acquire, preserve and extend knowledge. The two parts interact with each other reciprocally, where NAS explores more promising architectures and GSR preserves important history information. We empirically demonstrate the superiority of MSCGL compared with existing continual graph learning methods through extensive experiments. An interesting future research direction would be using information about past tasks as prior knowledge in the architecture search process.

## ACKNOWLEDGMENTS

This work is supported by the National Key Research and Development Program of China No.2020AAA0106300 and National Natural Science Foundation of China No. 62102222.

## REFERENCES

[1] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. 2018. Memory Aware Synapses: Learning what (not) to forget. arXiv:1711.09601 [cs.CV]

[2] Matthias Delange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Greg Slabaugh, and Tinne Tuytelaars. 2021. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805 (2018). arXiv:1810.04805 http://arxiv.org/abs/1810.04805

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2020. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *CoRR* abs/2010.11929 (2020). arXiv:2010.11929 https://arxiv.org/abs/2010.11929

[5] Xianya Fu, Wenrui Li, Qiurui Chen, Lianyi Zhang, Kai Yang, Duzheng Qing, and Rui Wang. 2020. NASIL: Neural Network Architecture Searching for Incremental Learning in Image Classification. In *International Symposium on Parallel Architectures, Algorithms and Programming*. Springer, 68–80.

[6] Lukas Galke, Benedikt Franke, Tobias Zielke, and Ansgar Scherp. 2021. Lifelong Learning of Graph Neural Networks for Open-World Node Classification. In *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.

[7] Difei Gao, Ke Li, Ruiping Wang, Shiguang Shan, and Xilin Chen. 2020. Multi-modal graph neural network for joint reasoning on vision and scene text. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12746–12756.

[8] Yang Gao, Hong Yang, Peng Zhang, Chuan Zhou, and Yue Hu. 2019. Graphnas: Graph neural architecture search with reinforcement learning. *arXiv preprint arXiv:1904.09981* (2019).

[9] Yang Gao, Hong Yang, Peng Zhang, Chuan Zhou, and Yue Hu. 2020. Graph Neural Architecture Search.. In *IJCAI*, Vol. 20. 1403–1409.

[10] Chaoyu Guan, Xin Wang, and Wenwu Zhu. 2021. Autoattend: Automated attention representation search. In *International Conference on Machine Learning*. PMLR, 3864–3874.

[11] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*. 507–517.

[12] Hanzhang Hu, John Langford, Rich Caruana, Saurajit Mukherjee, Eric Horvitz, and Debadeepta Dey. 2019. Efficient forward architecture search. *arXiv preprint arXiv:1905.13360* (2019).

[13] Wengong Jin, Kevin Yang, Regina Barzilay, and Tommi S. Jaakkola. 2018. Learning Multimodal Graph-to-Graph Translation for Molecular Optimization. *CoRR* abs/1812.01070 (2018). arXiv:1812.01070 http://arxiv.org/abs/1812.01070

[14] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2017. Overcoming catastrophic forgetting in neural networks. arXiv:1612.00796 [cs.LG]

[15] Yanxi Li, Zean Wen, Yunhe Wang, and Chang Xu. 2021. One-shot graph neural architecture search with dynamic search space. In *Proc. AAAI Conf. Artif. Intell*, Vol. 35. 8510–8517.

[16] Zhizhong Li and Derek Hoiem. 2017. Learning without Forgetting. arXiv:1606.09282 [cs.CV]

[17] Huihui Liu, Yiding Yang, and Xinchao Wang. 2020. Overcoming catastrophic forgetting in graph neural networks. *arXiv preprint arXiv:2012.06002* (2020).

[18] David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. *Advances in neural information processing systems* 30 (2017), 6467–6476.

[19] Rahul Mishra, Hari Prabhat Gupta, and Tanima Dutta. 2020. A Survey on Deep Neural Network Compression: Challenges, Overview, and Solutions. arXiv:2010.03954 [cs.LG]

[20] Shuaicheng Niu, Jiaxiang Wu, Guanghui Xu, Yifan Zhang, Yong Guo, Peilin Zhao, Peng Wang, and Mingkui Tan. 2021. AdaXpert: Adapting Neural Architecture for Growing Data. In *International Conference on Machine Learning*. PMLR, 8184–8194.

[21] Ramakanth Pasunuru and Mohit Bansal. 2019. Continual and multi-task architecture search. *arXiv preprint arXiv:1906.05226* (2019).

[22] Yu-xin Peng, Wen-wu Zhu, Yao Zhao, Chang-sheng Xu, Qing-ming Huang, Han-qing Lu, Qing-hua Zheng, Tie-jun Huang, and Wen Gao. 2017. Cross-media analysis and reasoning: advances and directions. *Frontiers of Information Technology & Electronic Engineering* 18, 1 (2017), 44–57.

[23] Simone Scardapane, Danilo Comminiello, Amir Hussain, and Aurelio Uncini. 2017. Group sparse regularization for deep neural networks. *Neurocomputing* 241 (2017), 81–89.

[24] Manos Schinas, Symeon Papadopoulos, Georgios Petkos, Yiannis Kompatsiaris, and Pericles A Mitkas. 2015. Multimodal graph-based event detection and summarization in social media streams. In *Proceedings of the 23rd ACM international conference on Multimedia*. 189–192.

[25] Zhulin Tao, Yinwei Wei, Xiang Wang, Xiangnan He, Xianglin Huang, and Tat-Seng Chua. 2020. MGAT: multimodal graph attention network for recommendation. *Information Processing & Management* 57, 5 (2020), 102277.

[26] Jinguang Wang, Jun Hu, Shengsheng Qian, Quan Fang, and Changsheng Xu. 2020. Multimodal graph convolutional networks for high quality content recognition. *Neurocomputing* 412 (2020), 42–51.

[27] Junshan Wang, Guojie Song, Yi Wu, and Liang Wang. 2020. Streaming graph neural networks via continual learning. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 1515–1524.

[28] Junshan Wang, Guojie Song, Yi Wu, and Liang Wang. 2020. Streaming Graph Neural Networks via Continual Learning. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management* (Virtual Event, Ireland) *(CIKM '20)*. Association for Computing Machinery, New York, NY, USA, 1515–1524. https://doi.org/10.1145/3340531.3411963

[29] Xin Wang, Shuyi Fan, Kun Kuang, and Wenwu Zhu. 2021. Explainable automated graph representation learning with hyperparameter importance. In *International Conference on Machine Learning*. PMLR, 10727–10737.

[30] Xin Wang, Ziwei Zhang, and Wenwu Zhu. 2022. Automated Graph Machine Learning: Approaches, Libraries and Directions. arXiv:2201.01288 [cs.LG]

[31] Yinwei Wei, Xiang Wang, Liqiang Nie, Xiangnan He, Richang Hong, and Tat-Seng Chua. 2019. MMGCN: Multi-modal graph convolution network for personalized recommendation of micro-video. In *Proceedings of the 27th ACM International Conference on Multimedia*. 1437–1445.

[32] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, Online, 38–45. https://www.aclweb.org/anthology/2020.emnlp-demos.6

[33] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* 32, 1 (2020), 4–24.

[34] Yishi Xu, Yingxue Zhang, Wei Guo, Huifeng Guo, Ruiming Tang, and Mark Coates. 2020. Graphsail: Graph structure aware incremental learning for recommender systems.. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2861–2868.

[35] Jianing Yang, Yongxin Wang, Ruitao Yi, Yuying Zhu, Azaan Rehman, Amir Zadeh, Soujanya Poria, and Louis-Philippe Morency. 2021. MTAG: Modal-Temporal Attention Graph for Unaligned Human Multimodal Language Sequences. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 1009–1021.

[36] Qin Yijian, Wang Xin, Zhang Zeyang, and Zhu Wenwu. 2021. Graph Differentiable Architecture Search with Structure Learning. In *Thirty-Fifth Conference on Neural Information Processing Systems*.

[37] Huan Zhao, Lanning Wei, and Quanming Yao. 2020. Simplifying architecture search for graph neural network. *arXiv preprint arXiv:2008.11652* (2020).

[38] Fan Zhou and Chengtai Cao. 2021. Overcoming Catastrophic Forgetting in Graph Neural Networks with Experience Replay. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 4714–4722.

[39] W. Zhu, P. Cui, Z. Wang, and G. Hua. 2015. Multimedia Big Data Computing. *IEEE MultiMedia* 22, 03 (jul 2015), 96–c3. https://doi.org/10.1109/MMUL.2015.66

[40] Wenwu Zhu and Xin Wang. 2021. Automated Machine Learning and Meta-Learning for Multimedia.

[41] Barret Zoph and Quoc V Le. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578* (2016).