Learning Graph Convolutional Network for Skeleton-Based Human Action Recognition by Neural Searching

Wei Peng,[1] **Xiaopeng Hong,**[3,4,1] **Haoyu Chen,**[1] **Guoying Zhao**[1,2,*]
[1]CMVS, University of Oulu, Finland; [2]School of Information and Technology, Northwest University, PRC
[3]School of Cyber Science and Engineering, Faculty of Electronic and Information Engineering, Xi'an Jiaotong University, PRC
[4]Research Center for Artificial Intelligence, Peng Cheng Laboratory; [*]Corresponding author
{wei.peng, xiaopeng.hong, chen.haoyu, guoying.zhao}@oulu.fi

Abstract

Human action recognition from skeleton data, fuelled by the Graph Convolutional Network (GCN) with its powerful capability of modeling non-Euclidean data, has attracted lots of attention. However, many existing GCNs provide a pre-defined graph structure and share it through the entire network, which can loss implicit joint correlations especially for the higher-level features. Besides, the mainstream spectral GCN is approximated by one-order hop such that higher-order connections are not well involved. All of these require huge efforts to design a better GCN architecture. To address these problems, we turn to Neural Architecture Search (NAS) and propose the first automatically designed GCN for this task. Specifically, we explore the spatial-temporal correlations between nodes and build a search space with multiple dynamic graph modules. Besides, we introduce multiple-hop modules and expect to break the limitation of representational capacity caused by one-order approximation. Moreover, a corresponding sampling- and memory-efficient evolution strategy is proposed to search in this space. The resulted architecture proves the effectiveness of the higher-order approximation and the layer-wise dynamic graph modules. To evaluate the performance of the searched model, we conduct extensive experiments on two very large scale skeleton-based action recognition datasets. The results show that our model gets the state-of-the-art results in term of given metrics.

Introduction

Human action recognition is a valuable but challenging research area with widespread potential applications, say human computer interaction and autonomous driving. Nowadays, skeleton data is popularly used in action recognition. One important reason is that skeleton data conveys compact information of body movement, thus it is robust to the complex circumstances like the variations of the viewpoints, occlusion and self-occlusion. Previous works reorganize the skeleton data into a grid-shape structure so that the traditional recurrent neural networks (RNN) and convolutional neural networks (CNN) can be directly implemented. Though substantial improvements have been seen in action recognition, the capability of deep learning is constrained as there is no natural notion of locality in skeleton data. Currently, Graph Convolutional Networks (GCN) (Kipf and Welling 2016; Defferrard, Bresson, and Vandergheynst 2016) has been introduced to skeleton-based action recognition and achieved many encouraging results (Li et al. 2018; Yan, Xiong, and Lin 2018; Li et al. 2019a; Gao et al. 2019; Shi et al. 2019; Li et al. 2019b). Nonetheless, most GCN methods are based on a pre-defined graph with fixed topology constraint, which ignores implicit joint correlations. Work in (Shi et al. 2019) intends to replace the fixed graph with an adaptive one based on the node similarity. However, it provides a shared mechanism through the entire network. Besides, the spatial-temporal correlations are barely discussed. We argue that different layers contain different semantic information thus a layer-specific mechanism should be involved when construct a dynamic graph. Besides, mainstream GCN tends to one-order Chebyshev polynomials approximation (Kipf and Welling 2016) to reduce the computational expense, meanwhile high-order connections are not well involved so that the representational ability is limited. Current works, like (Gao et al. 2019), introduce high-order approximation to have GCN with a bigger receptive filed. Nonetheless, the contribution of the each component in the approximation is not discussed. It is apparent that designing such different function modules for different tasks requires exhausted try-and-error tests.

To address these problems, in this paper, we focus on reducing the manual efforts in designing better graph convolutional architecture. Specifically, the fixed graph structure is replaced with dynamic ones by Automatic Neural Architecture Search (NAS) (Zoph and Le 2016) and with NAS, we explore different graph generating mechanisms at different semantic levels. NAS is designed to obtain superior neural network structures with less or without human assists. However, it is not straightforward to apply NAS to GCN. Graph data like skeleton has no locality and order information as required by convolution operations, while current NAS methods focus on the design of neural operations. Unfortunately, GCN itself is an emerging research topic thus existing operations are very limited, e.g., GCN does not even have a general pooling operation. Therefore, we propose to search in a GCN space built with multiple graph

function modules. Moreover, a high sample-efficient deep neuro-evolution strategy (ES) (Angeline, Saunders, and Pollack 1994; Miller, Todd, and Hegde 1989) is provided to explore an optimal GCN architecture by estimating the architecture distribution. The search strategy can be conducted in both continuous and discrete search space such that one can just activate one function module at each iteration to search by a memory-efficient fashion. With our NAS for GCN, we automatically build a graph convolutional network for skeleton-based action recognition. To evaluate the proposed method, we perform comprehensive experiments on two large scale datasets, NTU RGB+D (Shahroudy et al. 2016) and Kinetcis-Skeleton (Kay et al. 2017; Yan, Xiong, and Lin 2018). Results show that our model is robust to the subject and view variations and achieves the state-of-the-art performance on both of the datasets. The contributions of this paper are manifold:

- We break the limitation of GCN caused by its fixed graph and, for the first time, determine the GCN architecture with NAS for skeleton-based action recognition.
- We enrich the search space for GCN from the following two aspects. Firstly, we provide multiple dynamic graph substructures on the basis of various spatial-temporal graphs modules. Secondly, we enlarge the receptive field of GCN convolution by building higher-order connections with Chebyshev polynomial approximation.
- To improve the search efficiency, we devise a novel evolution based NAS search strategy, which is both sampling- and memory-efficient.

Related work

Skeleton-based Action Recognition In human action recognition, skeleton data increasingly attracts attention thanks to its robustness against changes in body scales, viewpoints and backgrounds. Different from the grid data, the graph constructed by skeleton lies in a non-Euclidean space. To benefit from the great representation ability of deep learning, conventional methods tend to rearrange the skeleton data into grid-shape structure and feed it directly into the classical RNN (Shahroudy et al. 2016; Song et al. 2017; Zhang et al. 2017) or CNN (Kim and Reiter 2017; Liu, Liu, and Chen 2017) architectures. However, as mentioned in (Monti et al. 2017), one can not express a meaningful operator in the vertex domain. Therefore, current works tend to GCNs since their operators are defined in the non-Euclidean space. Yan *et al.* and Li *et al.* are the first to use GCN for skeleton-based action recognition (Yan, Xiong, and Lin 2018; Li et al. 2018). Gao *et al.* proposed a sparsified graph regression based GCN (Gao et al. 2019) to exploit the dependencies of each joints. Shi *et al.* gave a two-stream GCN architecture, in which the joints and the second-order information (bones) are both used. With a score-level fusion strategy, it gets the current best result (Shi et al. 2019). Our method is also based on GCN and we will fully explore the influence of the graph topology for this task.

Neural Architecture Search As an important part of automated machine learning (AutoML), Neural Architecture Search (NAS) (Zoph and Le 2016) is to automatically build neural networks. Numerous approaches for NAS already exist in the literature, including black-box optimization based on reinforcement learning (Zoph and Le 2016), evolutionary search (Real et al. 2018), and gradient-based method (Liu, Simonyan, and Yang 2018). Besides, promising progresses are also seen in aspects such as searching space design (Liu, Simonyan, and Yang 2018), and architecture performance evaluation (Saxena and Verbeek 2016; Real et al. 2018). Automatically designed architectures have already got superior performances against the famous manual ones in the fields like image classification tasks (Zoph et al. 2018), and semantic image segmentation (Liu et al. 2019). There are also some attempts about NAS on action recognition (Peng, Hong, and Zhao 2019) from RGB data. However, little NAS-based methods provide a solution to the non-Euclidean data. In fact, currently Gao *et al.* transferred ENAS (Pham et al. 2018) to graph neural network for citation networks and inductive learning tasks. Compare to our task, it is totally different since it aims to find the transforming, propagating and aggregating functions for a network with only two or three layers.

GCN and Attention mechanism Graph neural network is widely used on irregular data like social networks, and biological data. Generally, there are two ways to define a GCN. The spectral-domain method (Defferrard, Bresson, and Vandergheynst 2016; Kipf and Welling 2016) models the representation in the Fourier domain based on eigen-decomposition, meanwhile it is time-consuming. The Nodal-domain method (Monti et al. 2017; Veličković et al. 2018) directly implements operators on the graph node and its neighbors. However, it is difficult to model the global structure. To further improve the performance of GCN, attention mechanisms, which select relatively critical information from all inputs, is introduced to GCN (Veličković et al. 2018; Vaswani et al. 2017). Velickovic *et al.*leveraged attention mechanism for graph node classification and achieved state-of-the-art performance (Veličković et al. 2018). Work in (Sankar et al. 2019) employs self-attention along both spatial and temporal dimensions and get superior results on link prediction tasks. Nonetheless, our work is different since we are to build a dynamic graph via the interaction between nodes based on various semantic information, while others are to compute the importance weights for different representations or frames.

Methodology

In this section, we detail our search-based GCN. To make the paper self-contained, we briefly review how to model a spatial graph with GCN first.

Consider an undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, A\}$ composed of $n = |\mathcal{V}|$ nodes, which are connected by $|\mathcal{E}|$ edges and the node connections are encoded in the adjacency matrix $A \in \mathcal{R}^{n \times n}$. Let $X \in \mathcal{R}^n$ be the input representation of $\mathcal{G}$ and $\{x_i, \forall i \in \mathcal{V}\}$ be its n elements. Then to model the representation of $\mathcal{G}$, a Fourier transform is conducted on the graph so that the transformed signal, as in the Euclidean space, could then be dealt with formulation of fundamental operations such as filtering. To this end, a graph Laplacian L, of

which the normalized definition is $L = I_n - D^{-1/2}AD^{-1/2}$ and $D_{ii} = \sum_j A_{ij}$, is used for Fourier transform. Then a graph filtered by operator g_θ, parameterized by θ, can be formulated as

$$Y = g_\theta(L)X = Ug_\theta(\Lambda)U^T X, \tag{1}$$

where Y is the extracted graph feature. U is the Fourier basis and it is a set of orthonormal eigenvectors for L so that $L = U\Lambda U^T$ with the Λ as its corresponding eigenvalues. However, multiplication with the eigenvectors matrix is expensive. The computational burden of this non-parametric filter is $\mathcal{O}(n^2)$ (Defferrard, Bresson, and Vandergheynst 2016). Suggested by (Hammond, Vandergheynst, and Gribonval 2011), the filter g_θ can be well-approximated by a Chebyshev polynomials with R-th order.

$$Y = \sum_{r=0}^{R} \theta_r^{'} T_r(\hat{L})X, \tag{2}$$

of which $\theta_r^{'}$ denotes Chebyshev coefficients. Chebyshev polynomial $T_r(\hat{L})$ is recursively defined as

$$T_r(\hat{L}) = 2\hat{L}T_{r-1}(\hat{L}) - T_{r-2}(\hat{L}) \tag{3}$$

with $T_0 = 1$ and $T_1 = \hat{L}$. Here $\hat{L} = 2L/\lambda_{max} - I_n$ is normalized to [-1,1]. For Eq. (2), work in (Kipf and Welling 2016) sets $R = 1$, $\lambda_{max} = 2$ and makes the network adapt to this change. In this way, a first-order approximation of spectral graph convolutions is formed. Therefore,

$$Y = \theta_0^{'}X + \theta_1^{'}(L - I_n)X = \theta_0^{'}X - \theta_1^{'}(D^{-1/2}AD^{-1/2})X. \tag{4}$$

Likewise, $\theta_r^{'}$ can also be approximated with an unified parameter θ, which means $\theta = \theta_0 = -\theta_1$, and let the training process adapt the approximation error, then

$$Y = \theta(I_n + D^{-1/2}AD^{-1/2})X. \tag{5}$$

The computational expense is $\mathcal{O}(|\mathcal{E}|)$. One can stack multiple GCN layers to get high-level graph feature. To make it simple, in the following sections, we set $L = I_n + D^{-1/2}AD^{-1/2}$, and generally, $X \in \mathcal{R}^{n\times C}$ is with multi-channels. Thus

$$Y = LX\theta. \tag{6}$$

Work in (Yan, Xiong, and Lin 2018) presents a ST-GCN block, which takes skeleton data and a fixed matrix L as inputs, to extract the spatial-temporal representation of nodes. Our GCN-block is also a spatial-temporal block, while instead of providing a pre-defined graph, we generate dynamic graphs based on the node correlations captured by different function modules.

Searched Graph Convolutional Network

We consider the human action recognition problem from skeleton data as a graph classification task from a sequence of graphs $\mathbb{G} = \{\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_T\}$. Each graph denotes a skeleton at a certain time step and its nodes and edges represent the skeleton joints and their connections, respectively. Then, this task can be framed as supervised learning problem on graph data, in which the goal is to learn a robust representation of $\mathbb{G}$ with GCN and thus to give a better prediction of action classes. To this end, we propose to construct this GCN with NAS, which automatically assembles graph generating modules for layers at different semantic levels.

Firstly, we will detail the GCN search space built with different graph modules. Then, we present a sampling- and memory-efficient search strategy.

GCN search space In NAS, a neural search space determines what and how neural operations a searching strategy could take to build a neural network. Here we search in space built with multiple GCN modules to explore the optimal module combination for dynamic graph at different representation levels. There are mainly two kinds of correlations being captured to construct the dynamic graph.

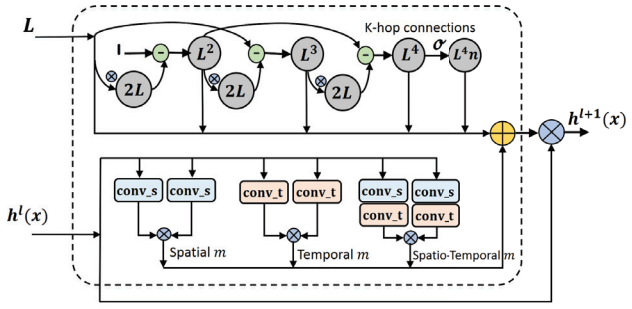

Figure 1: Illustration of the search space. Here, $\bigotimes$ denotes matrix multiplication. $\bigoplus$ is the element-wise summation. There are eight function modules for generating graphs. The top part is a implementation of Chebyshev polynomial based on Eq. (3). We also add its separate components to the graphs and let the network choose the final ones. The bottom part contains three dynamic graph modules. All the graphs are added together according to Eq. (8). The contribution of each module works as the architecture parameters. Note that there is a softmax function before the summation operation for dynamic graphs.

Structure representation Correlation. Structure correlation is to model graph topology based on current node connections. To determine how strong the connection is between two nodes, like in (Shi et al. 2019), a normalized Gaussian function is applied on the graph nodes and the similarity score works as the correlation. That is

$$\forall i, j \in \mathcal{V}, A_D(i,j) = \frac{e^{\phi(h(x_i)) \bigotimes \psi(h(x_j))}}{\sum_{j=1}^{n} e^{\phi(h(x_i)) \bigotimes \psi(h(x_j))}}. \tag{7}$$

This module is named as 'Spatial m' in Figure 1. Here, we compute the correlation score $A_D(i,j)$ between node i and node j based on their representations $h(x_i)$ and $h(x_j)$. The $\bigotimes$ represents matrix multiplication. The $\phi(\cdot)$ and $\psi(\cdot)$ are two projection functions, referred as $conv\_s$ in Figure 1 and can be implemented by channel-wise convolution filters. In this way, the similarity between nodes are captured to build the dynamic graph.

Temporal representation Correlation. Structure correlation definitely contains most intuitive cues for the topology

Algorithm 1 CEIM algorithm

procedure
  epochs $\leftarrow$ Max steps of iterations
  $N \leftarrow$ Size of populations in CEIM
  $j \leftarrow$ Initial iteration step with *0*
  $i \leftarrow$ Initial sample index with *1*
  $\Sigma \leftarrow$ Initial covariance matrix of CEIM
  $\mu \leftarrow$ Initial mean of CEIM
  $\alpha \leftarrow$ Initial architecture with μ

  while $j < epochs$ **do**
    Update network weights Θ with α fixed.
    if S_{old} is not $\emptyset$ **then**
      Uniformly get $r1$ and $r2$ in the range [0,1].
      while $i <= N$ **do**
        Take a sample α_o^i from S_{old}.
        if Ineq (10) satisfied **then**
          $S_{new} \leftarrow \alpha_o^i$.
        Draw a sample α_n^i with $\pi \sim \mathcal{N}(\mu, \Sigma)$.
        if Ineq (11) satisfied **then**
          $S_{new} \leftarrow \alpha_n^i$.
        $i = i + 1$.
    while $|S_{new}| > N$ **do**
      Randomly remove a sample from S_{new}.
    while $|S_{new}| < N$ **do**
      $S_{new} \leftarrow$ Draw a sample with π.
    Evaluate every $\alpha \in S_{new}$ on current network Θ.
    Sort samples by their performances.
    Compute importance weight λ by Eq. (12).
    $\pi_{old} \leftarrow \pi$
    $S_{old} \leftarrow S_{new}$
    Update μ and Σ by Eq. (13) and Eq. (14).
    $S_{new} \leftarrow \emptyset$
    $\pi_{new} \leftarrow \pi$
    $\alpha \leftarrow \mu$
    $j = j + 1$
  if $j ==$ *epochs* **then return** best $\alpha \in S_{new}$.

of the graph. However, ignoring the temporal correlation can loss implicit joint correlations. We take an example from NTU RGB+D dataset. Without the temporal information, it is hard to tell a person is to touch his head or just wave his hand, since from the structure perspective, little connection could be captured from head node and hand node during the action of "touch head". But including the temporal information will make it much easier. Therefore, we introduce two temporal convolutions to extract the temporal information of each node before computing node correlations with Eq. (7). In this way, the node interactions between neighbor frames are involved when we calculate the node connections. Note that, this is different from temporal attention mechanism since we capture temporal information for a better generation of the spatial graph. To this regard, we also introduce a Gaussian function, as in Eq. (7), to compute the node correlation. This module is referred as 'Temporal m', in which the functions $\phi(\cdot)$ and $\psi(\cdot)$ are implemented by temporal convolutions, referred as $conv\_t$ in Figure 1. It is also worth mentioning that, for the structure correlation, even the T frames of representation are all involved when compute the graph, the interaction is limited to the features from same dimension at the same time step. While the temporal module could involve interactions beyond frames.

With our 'Spatial m' and 'Temporal m', it is straightforward to build a spatial-temporal function module for dynamic graph. Thus we build the 'Spatio-Temporal m' as shown in Figure 1. In total, there are three kinds of modules for dynamic graphs.

Furthermore, we want to explore the contribution of each component in the Chebyshev polynomials, and thus benefit from high-order hop connections. As we know, the work in (Kipf and Welling 2016) gives a well-approximation of the spectral filter with the order-one Chebyshev polynomials. Instead, as illustrated in Figure 1, in our search space, we build Chebyshev polynomials functions with different orders at different layers and let the network determine which order and polynomial components each layer prefers. The function module can be constructed by Eq. (3), and here the biggest order is $R = 4$. Since all the dynamic graphs are normalized, here we also add a normalized one for the order-4 approximation. Therefore, there are totally eight function modules, as illustrated in the Figure 1, in this search space.

With these eight modules, we could search for the best architecture. Previous NAS methods would search a single block to reduce the computational burden. However, we argue that different feature layers contain different level of semantic content and thus a layer-specific mechanism is preferred to build a graph. So we search for an entire GCN network instead of a single block. To improve the efficiency, a high computation- and memory-efficient search strategy will be provided.

Let us formally define the search space first. Here we redefine X as a sequence of graphs. Given a fixed graph L and the feature $h^k(X)$ from the k-th layer, we extract the output representation $h^{k+1}(X)$ at $k+1$ layer, with the function modules we choose. Inspired by one-shot NAS and DARTS (Liu, Simonyan, and Yang 2018), all the function modules are paralleled and the weighted sum of their outputs are the output $h^{k+1}(X)$, that is

$$h^{k+1}(X) = \sum_{i=1}^{M} \frac{\alpha_{k+1,i}}{\sum_j^M \alpha_{k+1,j}} \mathcal{M}_i(h^k(X), L) h^k(X) \Theta_k. \tag{8}$$

Here, Θ_k is the network weights for the k-th layer. $\mathcal{M}_i$ denotes the i-th function module, and $\alpha_{k+1,i}$, which works as the architecture parameter, is its corresponding parameter at the $k+1$ layer. Then the problem here is to search a set of parameters $\alpha \in \mathrm{R}^{K \times M}$ for a network with K layers so that α minimizes the loss L_{valid} on the validation data. That is

$$\alpha^* = \underset{\alpha}{\operatorname{argmin}} L_{valid}(\Theta(\alpha), \alpha) \tag{9}$$

Here, Θ is the network parameters shared by all subnetworks and it will be learned on the training dataset. Previous works search on a small proxy dataset to evade the computational burden. Instead, we search directly on the target dataset to avoid extra domain adaption problem.

GCN search strategy Inspired by (Pou. and S. 2019), we propose to search with a high sampling-efficient ES-based method, denoted CEIM. This method explore an optimal architecture by estimating the architecture distribution. Thus it is not limited in a differentiable search space. One could improve the memory-efficiency by only activating one function module at each searching step.

Specifically, this search strategy combines **C**ross-**E**ntropy method (Larrañaga and Lozano 2001) with **I**mportance-**M**ixing (CEIM), in which architecture parameters α is treated as a population and the distribution of architecture is modeled by a Gaussian distribution. Then CEIM samples a group of architectures and with their performances, important samples are selected to update the architecture distribution. Thus an optimal architecture could be finally sampled from the architecture distribution. In total, there are three steps in our CEIM algorithm, sampling populations, selecting populations, and updating architecture distribution. Firstly, we model the architecture distribution with a Gaussian distribution $\pi \sim \mathcal{N}(\mu, \Sigma)$ and sample N architecture samples $\mathcal{S}_{new} = \{\alpha_n^i\}_{i=1}^N$ as the populations for CEIM. Secondly, combining $\mathcal{S}_{new}$ with historical selected populations $\mathcal{S}_{old} = \{\alpha_o^i\}_{i=1}^N$, we employ an importance mixing method on all these populations to choose architecture samples. Finally, the newly selected samples are used to update the architecture distribution π.

Here, we detail the last two steps. In the selecting step, for each population in $\mathcal{S}_{old}$ and $\mathcal{S}_{new}$, we compare its probability density (pd) in both current (π_{new}) and old (π_{old}) probability density functions (pdf). Generally, for the old population α_o^i, we keep it once it is with a bigger pd in the new distribution than that in the old one. That is

$$min(1, \frac{p(\alpha_o^i; \pi_{new})}{p(\alpha_o^i; \pi_{old})}) > r1 \tag{10}$$

Here $r1$ is a threshold randomly got from rang $[0, 1]$ and $p(\cdot; \pi)$ is a pdf with specific distribution π. Likewise, for new sample α_n^i drawn from the current distribution, if its pd in the new pdf is bigger than that in the old one, we will also keep it. Therefore, when

$$max(0, 1 - \frac{p(\alpha_n^i; \pi_{old})}{p(\alpha_n^i; \pi_{new})}) > r2 \tag{11}$$

we save it. Here $r2$ is another threshold in $[0, 1]$.

For the updating step, the samples selected in previous step are used to update mean μ and convariance Σ. Before that, the Θ of the network is updated on the training data with current architecture $\alpha = \mu$. Then, the Θ is fixed and every selected sample is set as the current architecture. Its corresponding fitness is evaluated on the validation data. With their performances, all the selected samples are sorted. Based on the performance order, an importance weight λ_i is assigned to the i-th sample. That is

$$\lambda_i = \frac{log(1+N)/i}{\sum_{i=1}^N log(1+N)/i}. \tag{12}$$

In this way, the sample with better performance will be given a bigger weight, thus it contributes more to the updating of the distribution. Finally, the weighted samples is applied to update the architecture distribution. That is

$$\mu_{new} = \sum_{i=1}^N \lambda_i \alpha^i, \tag{13}$$

$$\Sigma_{new} = \sum_{i=1}^N \lambda_i (\alpha^i - \mu)^2 + \epsilon \mathcal{I}. \tag{14}$$

Here, $\epsilon\mathcal{I}$ is a noise term for better exploring of the neural architecture. Since, in practice, Σ is too large to compute and update, here we constrain it to be a diagonal one. Note that in Eq. (14), different with the original cross-entropy method, which updates Σ with the new mean μ_{new}, we use the mean of last iteration to update Σ since convariance matrix adaption evolution strategy (CMA-ES) shows it is more efficient (H. 2016). More details about CEIM please refer to Algorithm. 1.

One could improve the memory-efficiency by only activating one function module at each searching step. That means for the output $h^{k+1}(X)$, it can be a single output from the activated module.

$$h^{k+1}(X) = \begin{cases} \mathcal{M}_1(h^k(X), L) h^k(X) \Theta_{k+1}, \ p = \frac{\alpha_{k+1,1}}{\sum_j^M \alpha_{k+1,j}} \\ \cdots \\ \mathcal{M}_M(h^k(X), L) h^k(X) \Theta_{k+1}, \ p = \frac{\alpha_{k+1,M}}{\sum_j^M \alpha_{k+1,j}} \end{cases} \tag{15}$$

Here, each module is activated by a multinomial distribution with the probability $p \sim \alpha_{k+1,i}$ and Θ_{k+1} is the activated weight of the $(k+1)$-th layer. In the following section, we will evaluate the proposed method.

Experiments

To evaluate the performance of our model, we carry out comparative experiments on two large-scale skeleton datasets, NTU RGB+D (Shahroudy et al. 2016) and Kenitics-Skeleton (Kay et al. 2017; Yan, Xiong, and Lin 2018), for action recognition task.

Dataset & Evaluation Metrics

NTU RGB+D (Shahroudy et al. 2016) is currently the most widely used and the largest multi-modality indoor-captured action recognition dataset. There are RGB videos, depth sequences, infrared videos and 3D skeleton data in it. The skeleton data is the one we use. There are totally 56,880 video clips captured from three cameras at different heights with different horizontal angles. These actions cover 60 human action classes including single-actor actions, which are from class 1 to 49, and two-actor actions, which are from class 50 to 60. There are 25 3D joints coordinates for each actor. We follow the benchmark evaluations in the original work (Shahroudy et al. 2016), which are Cross-subject (CS) and the Cross-view (CV) evaluations. In the CS evaluation, the training set contains 40,320 videos from 20 subjects, and the rest 16,560 video clips are used for testing. In the CV evaluation, videos captured from camera two and three, containing 37,920 videos, are used in the training and the

videos from camera one, containing 18,960 videos, are used for testing. In the comparison, Top-1 accuracy is reported on both of the two benchmarks.

Kinetics-Skeleton is based on the very large scale action dataset Kinetics (Kay et al. 2017), in which there are approximately 300 000 video clips collected from YouTube. This dataset covers 400 kinds of human actions. However, the original Kinectics dataset has no skeleton data. Yan *et al.* employed the open source toolbox OpenPose (Cao et al. 2017) to estimate the 2D joints location of each frame and then built this huge dataset Kinetics-Skeleton (Yan, Xiong, and Lin 2018). For each person, coordinates (X, Y) form 18 joints are estimated. For the frames which contain more than two persons, only the top-2 persons are selected based on the average joint confidence. The released data pads every clips to 300 frames. During comparison, both the Top-1 and Top-5 recognition accuracy are reported since this task is much harder due to its great variety.

Implementation details

Our framework is implemented on the PyTorch (Paszke et al. 2017) and the code is released at here[1]. To keep consistent with the state-of-the-art GCN methods (Yan, Xiong, and Lin 2018; Shi et al. 2019)[2], we introduce ten GCN blocks into our network for both searching and training steps. Each of them is based on the block in Figure 1. Like the previous works, each block is followed with a temporal convolution with the kernel size 9×1 to capture the temporal information. The first GCN block projects the inputs into a feature space with the channel number of 64. Then there will be three layers outputting 64 channels for the outputs. After that, the following three layers double the output channels. The last three layers have 256 channels for outputs. Just like (Yan, Xiong, and Lin 2018), the resnet mechanism is applied on each GCN block. Finally, the extracted features are fed into a fully-connected layer for the final prediction.

For each GCN block, the spatial modules $conv\_s$ are channel-wise convolution filters and the temporal filters $conv\_t$ are convolution filters with kernel size 9×1 performing along the temporal dimension. During searching, we conduct the experiments on the NTU RGB+D Joint data to find the optimal architecture. We share the searched architecture for all the aforementioned datasets to keep consistent with the state-of-the-art methods.

For the training process, a stochastic gradient descent (SGD) with Nesterov momentum (0.9) is applied as the optimization algorithm for the network. The cross-entropy loss is selected as the loss function for the recognition task. The weight decay is set to 0.0001 and 0.0006 for searching and training, respectively. For the NTU RGB+D dataset, there are at most two people in each sample. If the number of bodies in the sample is less than 2, we pad the second body with 0. The max number of frames in each sample is 300. For samples with less than 300 frames, we repeat the samples until it reaches 300 frames. The learning rate is set as 0.1 and is divided by 10 at the 30th, 45th, and 60th epoch. The training process is ended at the 70th epoch.

[1]https://github.com/xiaoiker/GCN-NAS

[2]These two SOTA methods mentioned in the paper that there are nine GCN blocks while the released codes show there are ten.

Table 1: Searched modules at each layer. Here, each row refers to a block layer. There are eight modules, including dynamic graph modules ($\mathcal{M}(S)$, $\mathcal{M}(T)$, $\mathcal{M}(ST)$) with various spatial-temporal cues and Chebshev approximation with different orders (L, L^4n,L^4,L^3, L^2). The modules marked with ✓ represent modules selected by CEIM.

$\mathcal{M}$	L	L^4n	L^4	L^3	L^2	$\mathcal{M}(S)$	$\mathcal{M}(T)$	$\mathcal{M}(ST)$
K_1					✓	✓	✓	✓
K_2						✓	✓	✓
K_3						✓	✓	✓
K_4						✓	✓	✓
K_5					✓	✓	✓	
K_6					✓		✓	
K_7		✓			✓	✓	✓	✓
K_8					✓		✓	
K_9					✓		✓	
K_{10}							✓	

Architecture search analysis

We conduct 70 epochs for searching. For the first 20 epochs, we randomly update each module in the network without evaluating any architectures. After that, we sample $N = 50$ architectures and update the architecture distribution with our CEIM algorithm. When the searching is done, we choose the module $\mathcal{M}_i$ if its $\alpha > 0.1$. Like (Shi et al. 2019), a complementary dynamic graph is added, as we do not weight module with its α in the final architecture. The searched architecture is listed in Table 1. The result shows that different layers prefer different mechanisms to generate graphs, which is consistent with our expectation since high level representation contains more semantic information. Concretely, as in Table 1, the lower layers, like layer K_1 to K_4, count in all dynamic function modules to capture richer information. For higher layers, the temporal representation correlations are more preferred. It is interesting that temporal graph module $\mathcal{M}(T)$ is selected through the entire network while the spatial function module $\mathcal{M}(S)$ is only chose by the lower layers, which proves the effectiveness of the proposed temporal function module. For the higher-order connections, we found that 2-order hop connection is much welcomed than any other one. And surprisingly we found the L, which encodes the physical structure of the skeleton data, is not involved at any layer. This founding gives us a new inspiration about how to build a GCN.

Ablation Study

Here, we explore the effectiveness of the graph modules and also our searched GCN. Therefore, we perform the following experiments on NTU RGB+D with the benchmark of cross-view. Here we compare with six baselines, which are with different mechanism to build the dynamic graphs. Specifically, the modules used to generate graph are based on: 1) Structure representation correlations (S, here it is 2S-ACGN (Shi et al. 2019)); 2) Temporal representation correlations (Ours(T)); 3) Spatial-Temporal representation correlations(Ours(ST)); 4) Temporal correlations

Table 2: Ablation Study. Performance comparison on NTU RGB+D with CV evaluation.

Methods	Joint(%)	Bone(%)	Combine(%)
2S-AGCN (Shi et al. 2019)	93.7	93.2	95.1
Ours(T)	93.8	93.7	95.1
Ours(ST)	94.0	93.8	95.2
Ours(T+Cheb)	94.0	93.9	95.2
Ours(ST+Cheb)	94.2	93.9	95.3
Ours(S+T+ST+Cheb)	93.9	93.6	95.1
Ours(NAS)	**94.6**	**94.7**	**95.7**

with 4-order Chebyshev approximation (Ours(T+Cheb)); 5) Spatial-Temporal representation correlations with 4-order Chebyshev approximation (Ours(ST+Cheb)); and 6) Combine all aforementioned modules(Ours(S+T+ST+Cheb)). Inspired by (Shi et al. 2019), we evaluate these models on both joints and the bones (the second order information of skeleton joints) data, thus a fusion result from score-level is also reported. For these six methods, the same block is shared through the whole network. Instead, our searched method explores the best modules for different layers. The comparison results are listed in the Table 2. It shows that temporal information do helps for GCN (Ours(T) and Ours(ST)) and involving all modules can not make sure a better performance (Ours(S+T+ST+Cheb)). Besides, higher-order also helps for GCN (Ours(+Cheb)).The superior performance of the NAS-based GCN (Ours(NAS)) proves the effectiveness of our method. When compared to the current best result (Shi et al. 2019), shows in the first row, we improve the accuracy 0.9%, 1.5%, and 0.6% on the Joint, Bone and Combine, respectively. This verifies the effectiveness of our method.

Comparison with State-of-the-arts (SOTA)

To evaluate the performance of our searched model, we compared it with 14 SOTA skeleton-based action recognition approaches, including hand-crafted method (Hu et al. 2015), CNN-based methods (Kim and Reiter 2017; Liu, Liu, and Chen 2017), LSTM-based methods (Shahroudy et al. 2016; Song et al. 2017; Zhang et al. 2017; Si et al. 2018), reinforcement learning based method (Tang et al. 2018), and current promising GCN-based methods (Li et al. 2018; Yan, Xiong, and Lin 2018; Li et al. 2019a; Gao et al. 2019; Li et al. 2019b; Shi et al. 2019), on NTU RGB+D and Kinetics-Skeleton datasets. Tables 3 and 4 show the results on these two datasets, respectively. We report the best result after performing the score-level fusion on joints and bones. It can be seen from Table 3 and 4 that the searched model achieves the best performance on both of the two datasets in terms of all evaluation metrics. Specifically, for the NTU RGB+D dataset, under the CV protocol, the searched model improves the current best result 95.1% to 95.7%. Our method also overcomes current best result by 0.9% uncer the CS protocol. Besides, we also get the best Top-1(37.1%) and Top-5(60.1%) on Kinetics-Skeleton dataset. All these prove the effectiveness of our method.

Table 3: Performance comparison on NTU RGB+D with 14 state-of-the-art methods.

Methods	CS(%)	CV(%)	Conference
Joint (Hu et al. 2015)	60.2	65.2	CVPR2015
P-LSTM (Shahroudy et al. 2016)	62.9	70.3	CVPR2016
STA-LSTM (Song et al. 2017)	73.4	81.2	AAAI2017
TCN (Kim and Reiter 2017)	74.3	83.1	CVPRW2017
VA-LSTM (Zhang et al. 2017)	79.2	87.7	CVPR2017
SynCNN (Liu, Liu, and Chen 2017)	80.0	87.2	PR2017(Jou.)
Deep STGCK (Li et al. 2018)	74.9	86.3	AAAI2018
ST-GCN (Yan et al. 2018)	81.5	88.3	AAAI2018
DPRL (Tang et al. 2018)	83.5	89.8	CVPR2018
SR-TSL (Si et al. 2018)	84.8	92.4	ECCV2018
STGR-GCN (Li et al. 2019a)	86.9	92.3	AAAI2019
GR-GCN (Gao et al. 2019)	87.5	94.3	ACMM2019
AS-GCN (Li et al. 2019b)	86.8	94.2	CVPR2019
2S-AGCN (Shi et al. 2019)	88.5	95.1	CVPR2019
Ours(Joint+Bone)	**89.4**	**95.7**	-

Table 4: Performance comparison on Kinetics-skeleton with eight state-of-the-art methods.

Methods	Top-1(%)	Top-5(%)	Conference
Feature (Fernando et al. 2015)	14.9	25.8	CVPR2015
P-LSTM (Shahroudy et al. 2016)	16.4	35.3	CVPR2016
TCN (Kim and Reiter 2017)	20.3	40.0	CVPRW2017
ST-GCN (Yan et al. 2018)	30.7	52.8	AAAI2018
AS-GCN (Li et al. 2019b)	34.8	56.5	CVPR2019
2S-AGCN(**Joint**) (Shi et al. 2019)	35.1	57.1	CVPR2019
2S-AGCN(**Bone**) (Shi et al. 2019)	33.3	55.7	CVPR2019
2S-AGCN (Shi et al. 2019)	36.1	58.7	CVPR2019
Ours(Joint)	**35.5**	**57.9**	-
Ours(Bone)	**34.9**	**57.1**	-
Ours(Joint+Bone)	**37.1**	**60.1**	-

Conclusion

In this paper, we propose to build graph convolutional network for skeleton-based action recognition with NAS. To enrich the NAS search space, firstly, three dynamic graph generating modules are constructed on the basis of various spatial-temporal correlations of nodes. Secondly, modules with higher-order connections are introduced to enlarge the receptive field of GCN convolution. Besides, we devise a novel search strategy by combining cross-entropy evolution strategy with importance-mixing (CEIM), which is both sampling- and memory-efficient. Based on the proposed NAS method, we explore the optimal GCN architecture in this space for skeleton action recognition. The searched model proves the effectiveness of our function modules. Comprehensive experiments on two very large scale datasets show its overwhelming performance when compared to the state-of-the-art approaches.

Acknowledgements

This work was supported by the Academy of Finland ICT 2023 project (Grant No. 313600), Tekes Fidipro program (Grant No. 1849/31/2015) and Business Finland project (Grant No. 3116/31/2017), Infotech Oulu, and the National Natural Science Foundation of China (Grants No. 61772419). As well, the authors wish to acknowledge CSC-IT Center for Science, Finland, for computational resources.

References

Angeline, P. J.; Saunders, G. M.; and Pollack, J. B. 1994. An evolutionary algorithm that constructs recurrent neural networks. *IEEE transactions on Neural Networks* 5:54–65.

Cao, Z.; Simon, T.; Wei, S.-E.; and Sheikh, Y. 2017. Realtime multi-person 2d pose estimation using part affinity fields. *2017 IEEE CVPR* 7291–7299.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *NeurIPS*, 3844–3852.

Fernando, B.; Gavves, E.; Oramas, J. M.; Ghodrati, A.; and Tuytelaars, T. 2015. Modeling video evolution for action recognition. In *IEEE CVPR*, 5378–5387.

Gao, X.; Hu, W.; Tang, J.; Liu, J.; and Guo, Z. 2019. Optimized skeleton-based action recognition via sparsified graph regression. In *Proceedings of the 2019 ACM International Conference on Multimedia*. ACM.

H., N. 2016. The cma evolution strategy: A tutorial. *arXiv*.

Hammond, D. K.; Vandergheynst, P.; and Gribonval, R. 2011. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis* 30(2):129–150.

Hu, J.-F.; Zheng, W.-S.; Lai, J.; and Zhang, J. 2015. Jointly learning heterogeneous features for rgb-d activity recognition. In *IEEE CVPR*, 5344–5352.

Kay, W.; C., J.; S., K.; Z., B.; H., C.; V., S.; Viola, F.; Green, T.; Back, T.; Natsev, P.; et al. 2017. The kinetics human action video dataset. *arXiv*.

Kim, T. S., and Reiter, A. 2017. Interpretable 3d human action analysis with temporal convolutional networks. In *2017 IEEE CVPR workshops*, 1623–1631.

Kipf, T. N., and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv*.

Larrañaga, P., and Lozano, J. A. 2001. *Estimation of distribution algorithms: A new tool for evolutionary computation*, volume 2. Springer Science & Business Media.

Li, C.; Cui, Z.; Zheng, W.; Xu, C.; and Yang, J. 2018. Spatio-temporal graph convolution for skeleton based action recognition. In *Thirty-Second AAAI*.

Li, B.; Li, X.; Zhang, Z.; and Wu, F. 2019a. Spatio-temporal graph routing for skeleton-based action recognition.

Li, M.; Chen, S.; Chen, X.; Zhang, Y.; Wang, Y.; and Tian, Q. 2019b. Actional-structural graph convolutional networks for skeleton-based action recognition. In *IEEE CVPR*.

Liu, C.; Chen, L.-C.; Schroff, F.; Adam, H.; Hua, W.; Yuille, A.; and Fei-Fei, L. 2019. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. *arXiv*.

Liu, M.; Liu, H.; and Chen, C. 2017. Enhanced skeleton visualization for view invariant human action recognition. *Pattern Recognition* 68:346–362.

Liu, H.; Simonyan, K.; and Yang, Y. 2018. Darts: Differentiable architecture search. *arXiv*.

Miller, G. F.; Todd, P. M.; and Hegde, S. U. 1989. Designing neural networks using genetic algorithms. In *ICGA*, volume 89, 379–384.

Monti, F.; Boscaini, D.; Masci, J.; Rodola, E.; Svoboda, J.; and Bronstein, M. M. 2017. Geometric deep learning on graphs and manifolds using mixture model cnns. In *IEEE CVPR*, 5115–5124.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in pytorch.

Peng, W.; Hong, X.; and Zhao, G. 2019. Video action recognition via neural architecture searching. *arXiv*.

Pham, H.; Guan, M.; Zoph, B.; Le, Q.; and Dean, J. 2018. Efficient neural architecture search via parameter sharing. In *ICML*, 4092–4101.

Pou., A., and S., O. 2019. Cem-rl: Combining evolutionary and gradient-based methods for policy search. *ICLR*.

Real, E.; Ag., A.; H., Y.; and Le, Q. V. 2018. Regularized evolution for image classifier architecture search. *arXiv*.

Sankar, A.; Wu, Y.; Gou, L.; Zhang, W.; and Yang, H. 2019. Dynamic graph representation learning via self-attention networks. *ICLRW*.

Saxena, S., and Verbeek, J. 2016. Convolutional neural fabrics. In *NeurIPS*, 4053–4061.

Shahroudy, A.; Liu, J.; Ng, T.-T.; and Wang, G. 2016. Ntu rgb+ d: A large scale dataset for 3d human activity analysis. In *IEEE CVPR*, 1010–1019.

Shi, L.; Zhang, Y.; Cheng, J.; and Lu, H. 2019. Two-stream adaptive graph convolutional networks for skeleton-based action recognition. In *IEEE CVPR*, 12026–12035.

Si, C.; Jing, Y.; Wang, W.; Wang, L.; and Tan, T. 2018. Skeleton-based action recognition with spatial reasoning and temporal stack learning. In *ECCV*, 103–118.

Song, S.; Lan, C.; Xing, J.; Zeng, W.; and Liu, J. 2017. An end-to-end spatio-temporal attention model for human action recognition from skeleton data. In *Thirty-first AAAI*.

Tang, Y.; Tian, Y.; Lu, J.; Li, P.; and Zhou, J. 2018. Deep progressive reinforcement learning for skeleton-based action recognition. In *IEEE CVPR*, 5323–5332.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *NeurIPS*, 5998–6008.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2018. Graph attention networks. *ICLR*.

Yan, S.; Xiong, Y.; and Lin, D. 2018. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Thirty-Second AAAI*.

Zhang, P.; Lan, C.; Xing, J.; Zeng, W.; Xue, J.; and Zheng, N. 2017. View adaptive recurrent neural networks for high performance human action recognition from skeleton data. In *IEEE ICCV*, 2117–2126.

Zoph, B., and Le, Q. V. 2016. Neural architecture search with reinforcement learning. *ArXiv*.

Zoph, B.; Vasudevan, V.; Shlens, J.; and Le, Q. V. 2018. Learning transferable architectures for scalable image recognition. In *IEEE CVPR*, 8697–8710.