Capsule neural nets for graph objects classification

K A Maikov¹, B N Smirnov², A N Pylkin³ and A A Bubnov³

¹ Department of computer Software and information technologies, BMSTU, Moscow, Russia
² MCS, Russia
³ Department of Computational and Applied Mathematics, RSREU, Ryazan, Russia

E-mail: serbubnov@inbox.ru

Abstract. A new way to solve the graph classification problem is addressed. The main method utilized is the application of a capsule neural network on graphs. The results achieved include, firstly, the enhancement of the base algorithm for training a capsule network with the possibility of using graphs as an input (a stage of training for permutation invariants of graph vertices' transformation matrices is included as well as a memory block for trained matrices), and secondly, a proposition of a training set of labeled graph objects, transformed from the MNIST dataset. This opens a perspective for a better classification of graph objects due to preserving of their structure and transformation invariance between layers.

1. Introduction
The research question of the study is to find a new way to solve the graph classification problem: given several graphs as an input classify them to several classes based on transformation and other types of data invariance. The relevance of the problem is determined by a wide range of important contemporary applications that can be solved on its basis. For instance it includes studying the similarity of various molecular structures in chemical informatics [1], determining compliance with a standard in model checking and software testing [2], classifying business processes in organizations [3], analyzing state chains in various fields (for example, text chains, chains of transactions in the database [4]).

In machine learning this problem is traditionally solved within the framework of graph neural networks. The main idea is that the input data is transformed from graphs into "flat" vectors containing the maximum of useful information about the graph structure. Then classical deep learning methods are applied. The method of such a transformation determines the type of graph networks: the basic architecture proposes multiple information aggregation from neighboring vertices with an increasing "radius" of coverage [5]. A number of publications use the convolution operation: spectral convolutions (Fourier transform in Laplacian matrices) [6], and spatial convolutions of graphs (decomposition of the Laplacian matrix into Chebyshev polynomials) [7], with a number of computational simplifications [8], [9], [10]. However, such architectures deal with scalar transformation results, which entails the loss of a significant part of important non-local information about the structure of the source graphs (when this data is transmitted between network layers).

The basic architecture of capsule neural networks preserves information about the structure of processed objects (primarily due to the transition from scalars to multidimensional structures – capsules in the inner layers of the network) and was largely created for this very purpose [11]. However, it was originally only applicable to input data in the form of images and is not trivially extended to other data
types, including hierarchical objects (largely because of the non-Euclidean nature of their relation spaces and metrics).

One can view capsule networks as a multidimensional generalization of classical convolutional neural networks, which preserve the relations of the part and the whole in the object under classification due to the multidimensionality of capsules (groups of neurons, each of which contains a separate property of the data of the previous layer) and the mechanism for routing information between layers [12]. Currently, algorithms for training such networks to solve some specific problems have been implemented, primarily in the field of pattern recognition and text classification (see [11], [13], [14], [15]). However, the issues of generalizing the use of such networks for processing information in the form of graphs have not yet been resolved.

2. Utilized methods and technical aspects of the implemented study

After analyzing a number of popular alternatives, the following tools were selected for software implementation of the below-mentioned additions to the capsule neural network training algorithm and used during the generation stage of the proposed training sample (though we emphasize that the software implementation of the considered additions to the training algorithm for a CapsNet can be accomplished in various ways): “Google Colab” cloud computing environment for machine learning, “PyTorch” machine learning and deep learning library (open sourced, created on the basis of a well-known machine learning library “Torch”, written in “Python” programming language), and the “NetworkX” library for working with network structures and graphs.

The first graph network layer of the ultimate graph-capsule network architecture (specified in the following paragraph) can be represented by any graph convolutional neural network of one of the known types (for example, “GCNConv”, “ChebConv” or “SplineConv”, which have a basic implementation in the “torch_geometric” library). The implementation of graph capsule neural network layers can be based on any framework for building networks in terms of classes and layers (i.e. in the object-oriented programming paradigm), like Keras and PyTorch, using classes and static methods of the “torch_geometric” and “dgl” libraries. As part of the algorithm for preprocessing and obtaining a set of graph objects from MNIST image pixels (that involves creating the so-called ”super pixels” based on images), one can use the following ready-made "Python" libraries for working with graphs and networks: "NetworkX" and "scikit-image".

To reproduce the results of the study the capsule network layers (utilizing basic Hinton’s dynamic routing mechanism, [11]) can be based on any open-sourced implementation like the one from the “DGL” library.

3. Main results: graph-based modification of the capsule network architecture

We propose a modification of the capsule neural network architecture to solve the graph classification problem. Firstly, at the preprocessing stage, it is proposed to use a separate training sample (either artificially created from the input data, or the one, proposed further by the authors) to adjust the transformation matrices (between the network capsules) for the types of graph invariance specified in this sample (for example, permutation invariance). In fact, it is a "transfer learning" based on pre-trained datasets.

Secondly, it is proposed to add a memory block to the basic CapsNet architecture to store the values of the transformation matrices (trained at the preprocessing stage) between the iterations of the main learning process.

In addition to the possibility of using this architecture with graph data type, immutability of values of the transformation matrix when trained on a core set of data virtually reduces the complexity of the training phase for capsule networks (estimated in [16], the complexity of the CapsNet training for the core architecture is a "Big O" of n to the power of 8, where n is the number of neurons in the network, whereas for convolutional networks the training complexity is " Big O " of n to the power of 5).
Thus, the general algorithm for training a capsule neural network may be as follows (taking into account the need for preprocessing graph information and based on the classical algorithm for error’s back propagation [17], see figure 1 below).

- **Step 1.** First, the network initialization parameters are set.
- **Step 2.** Next, random weights are set for all transitions between network layers.
- **Step 3.** Next, random initial values for the transformation matrices are determined.
- **Step 4.** Then the transformation matrices are trained (independent to the rest of the network layers) on a separate training sample for the selected type of invariance of the graph adjacency matrices.
- **Step 5.** Based on the trained transformation matrices, the signal is directly propagated from the inputs of the network to its output; and the values of the inputs and outputs of the training layer are saved (at the first iteration of the algorithm, this is done at the very last layer of the network).
- **Step 6.** The error is calculated between the outputs predicted by the network and between the ones known from the training sample, or (in the next iterations of the algorithm after the first iteration) estimated when the signal is propagated back from the outputs to the inputs.
- **Step 7.** If the error is less than the specified level, or the number of training epochs is above a given threshold, the training stops and the algorithm terminates.
- **Step 8.** Otherwise, the input weights of the trained network layer are updated based on the error value and known formulas for the classical error back propagation algorithm.

![Flowchart of the backpropagation algorithm for training graph objects classification task with CapsNet.](image)

**Figure 1.** Flowchart of the backpropagation algorithm for training graph objects classification task with CapsNet.

With the technical implementation of the ideas outlined, the problem of data absence arises: the absence of labeled graph data for training a neural network that is simple in its structure and significant in its volume. Therefore, we propose a variant of generating a training sample in the form of labeled graph objects, simple in structure (up to 30 vertices in each graph) and significant in volume (10
thousand objects), which can be used, among other proposals, for the preprocessing stage in the algorithm of training a capsule neural network. This variant of a training sample was created by transforming the MNIST dataset (popular in the field of image recognition) into graphs by considering the coordinate matrix of image pixels as the adjacency matrix of a graph so that the Laplacian matrix for such a graph can be determined based on it. Then the maximum spanning tree of the resulting graph is determined.

4. Conclusion: discussion and future research perspectives
During an experimental verification based on this training sample and utilizing the proposed modification of the CapsNet architecture, after only 30 learning epochs a level of classification accuracy of 62% was achieved compared to 43% of accuracy based on "ChebConv" graph neural network alone.

The purposed architecture of CapsNet implies the possibility of obtaining the invariance of input data to permutations of their parts in the course of its training (which is vital for graph classification tasks and is provided in one form or another by graph neural networks).

This opens a perspective for a better classification of graph objects compared to graph neural network architecture alone due to preserving of their structure and transformation invariance between layers.

References
[1] Kwon Y et al. 2019 Efficient learning of non-autoregressive graph variational autoencoders for molecular graph generation Journal of Cheminformatics 11(1) 70
[2] Hlineny P, Filip P and Bodhayan R 2019 First-order model checking on geometric graphs Computational Geometry (Amsterdam, Netherland: Elsevier)
[3] Doumeingts G and Browne J 2016 Modelling techniques for Business Process Re-engineering and Benchmarking (US: Springer)
[4] Bismita S J, Khan C and Sunderraman R 2019 High performance frequent subgraph mining on transaction datasets: A survey and performance comparison Big Data Mining and Analytics 2(3) 159-80
[5] Gori M, Monfardini G and Scarselli F 2005 A new model for learning in graph domains (Montreal, Canada: IEEE)
[6] Bruna J, Zaremba W, Szlam A and LeCun Y 2014 Spectral networks and locally connected networks on graphs Proc. Int. Conf. on Learning Representations (ICLR2014)
[7] Henaff M, Bruna J and LeCun Y 2015 Deep convolutional networks on graph-structured data arXiv 1506.05163
[8] Kipf T N and Welling M 2017 Semi-supervised classification with graph convolutional networks Proc. Int. Conf. on Learning Representations
[9] Atwood J and Towsley D 2016 Diffusion-convolutional neural networks Proc. of NIPS
[10] Velikovi P, Cucurull G, Casanova A, Romero A, Li P and Bengio Y 2018 Graph attention networks arXiv 1710.10903
[11] Hinton G E, Sabour S and Frosst N 2017 Dynamic routing between capsules Proc. 31st Conf. on Neural Information Processing Systems (NIPS 2017) (Long Beach, CA, US)
[12] Bahadori M T 2018 Spectral capsule networks Proc. Int. Conf. Learn. Representations
[13] Chauhan A, Babu M, Kandru N and Lokegaonkar S 2018 Empirical Study on convergence of Capsule Networks with various hyperparameters (Blacksburg, VA, US: Virginia Polytechnic Institute and State University)
[14] Chen Z H and Crandall D 2018 Generalized capsule networks with trainable routing procedure arXiv 1808.08692
[15] Hinton G E, Krizhevsky A and Wang S D 2011 Transforming Auto-Encoders Lecture Notes in Computer Science (Berlin, Germany: Springer)
[16] Maikov K A and Smirnov B N 2019 Comparative analysis of capsule neural network architectures (Moscow, Russia: System administrator)
[17] Fan F, Cong W and Wang G 2017 General Backpropagation Algorithm for Training Second-order Neural Networks (New York, US: Biomedical Imaging Center)