Input simplifying as an approach for improving neural network efficiency

Alexey Grigorev, Nikita Korobov, Polina Kutsevol, Artem Lukoyanov, and Ilya Zharikov

Moscow Institute of Physics and Technologies,
Dolgoprudny, Institutsky per. 9, 141700, Russia
{korobov.ns,grigorev.ad,kutsevol.pn,
lukoyanov.as,ilya.zharikov}@phystech.edu
http://www.springer.com/lncs

Abstract. The symbol recognition is becoming the important part of a lot of cutting edge services which should be feasible to be implemented into devices with the constrained computational capacity. Plenty of current studies are targeted at the development of effective and fast CNN architectures, including the ones which deal with the graph symbol representation. In our work we propose to optimize the neural networks input rather than the architecture. We analyze the performance of several existing cnn architectures in terms of accuracy, learning and training time using the advanced skeleton symbol representation. It takes into account the inner symbol structure and strokes width patterns. We show that this representation allows achieving considerable reduction of learning and training time without noticeable accuracy degradation. This makes our approach the worthy replacement of conventional graph representations in symbol recognition tasks.

Keywords: Image classification, Skeleton, Graph Neural Network, Graph, Text recognition

1 Introduction

Recognition of the text on the image is one of the traditional computer vision tasks. In particular, this issue can imply letters and digits classification. Different solutions to this problem have been suggested in numerous of previous works [], which, as rule, are based on the Convolution Neural Networks (CNN) [], bitmaps and combination of different heuristic to increase performance [] and reduce learning and inference time.

The existing approaches allowing for the better performance and the faster work achievement rely on the optimization and simplifying of the classification architecture. However, state-of-the-art solutions still require days to learn. The inference time sometimes isn't well along with the fact that classification networks should be launched on the expensive hardware. In this work we introduce a novel symbol classification method, which is based on the skeletonization [] of the binarized images, the skeleton to graph translation and the further graph classification of the skeletons with the help of Graph Neural Networks. The proposed technique simplifies the input data rather than neural network architecture.

It should be mentioned that image binarization and skeleton obtaining algorithms are not the targets of this research and all the considered experiments are conducted with the fixed algorithms of the image binarization and skeletonization, which are discribed later.

As it is shown in our paper, the suggested method outperforms existed classification methods in training time required for the same quality as well as inference time and memory consumption. It is also seen that this approach shows better classification accuracy compared with graph superpixel as graph classification methods [].

The rest of the paper is structured as following. The second section presents proposed classification method. The third section describes related graph neural networks for the graph classification purposes. The fourth section is devoted to the experiments and discussions.

2 Related works

2.1 Skeletonization algorithms

The skeleton is the plain locus of points, which have at least two nearest points on the figure boundary. Note that there are internal and external skeletons. In this work we consider the inner skeleton representation only. There are several different approaches of skeleton representation and further graph obtaining. For instance, in [link] it is proposed to improve the discrete topological skeleton representation by graph simplifying and taking into account both topological and contour symbol properties. In this work we use the algorithm, similar to [link], where the skeleton representation is derived from the binary image. Thus, the problem of the contour obtaining is solved, since it is known in the binary image. The considered algorithm is also noise stable and computationally efficient compared with the previous ones.

2.2 Graph Neural Networks

It is necessary to learn state-of-the-art Graph Neural Networks to build Graph Classifier. According to the system of the Graph Neural Networks type suggested in [**?**], there are two types of Graph Convolution Neural Networks (GCNN): Spectral and Spatial GCNNs. There are some drawbacks of spectral NNs, comparing them with spatial architectures. Some of them are an impossibility to apply NN to the graphs with different size and sufficient increase in the complexity of the NN with the increasing of the graph size because of need computing eigenvalues of the matrix, which depends on the number of nodes. That is why spatial GCNN architectures are used in this work. Spatial GCNNs, in its turn, implies of recurrent and composition networks.

Message Passing Neural Networks (MPNNs) Gilmer and others [] have unified the existing approaches to the graph convolution neuron networks. The proposed method can be split into two stages: the message or hidden attributes passing through the graph several times in a row and then readout phases, where the hidden state is extracted from the graph. The authors have concluded that the variety of spatial convolution networks imply the same idea of information circulation through the graph with different message passing and data reading functions.

In [link] the authors have proposed a graph classification method with MPNN and Siamese networks. Note that the validation of this method has been conducted on the symbol graph representation database. It is seen that the proposed solution shows a 3% better quality compared with the conventional classification based on the MPNN architecture. One of the main disadvantages of MPNN is that the quality is severely reduced when the graph is widened. In particular, the authors claim that this approach can not be extended to the graphs of the size more than 30 vertexes without accuracy degradation. However, this problem can be avoided with the skeleton symbol representation, since the effective graph pruning algorithms are used.

***k*-GNN** k-GNN is a generalization of the traditional GNN architecture. The model is based on the k-dimensional WL algorithm. According to the authors, the key advantage of the proposed architecture is the ability to perform message passing directly between subgraphs instead of individual nodes. Such form of message passing makes it possible to take into account higher-order graph structures at different scales. Thus, the proposed architecture is more powerful than the classical GNN and outperforms it on a variety of graph classification and regression tasks. k-GNNs can also be stacked into hierarchical model which is capable of combining graph representations obtained at multiple scales. The implementation of the above-mentioned idea implies that the output of the lower-dimensional model is treated as the initial messages in the higher-dimensional k-GNN.

Mixture Model Network (MoNet) MoNet has extended the conventional CNN graph deep learning approach with the methods useful in non-euclidean metric spaces. For that Monti and others have introduced pseudo coordinates spaces, which are defined via the pairs of the coordinates of the vertexes and their neighbours, and the weight functions in these spaces.

Many of graphs CNN approaches such as GCNN, ACNN, Spline CNN, DCNN, GCN can be expressed via the proposed representation with different weight functions $w(u)$ and pseudo coordinates u. The authors propose to use the Gaussian kernel as a weight function, where the mean vector and covariance matrix are learnable parameters. With a MNIST database symbols as an example where every pixel is a graph node it has been shown that this architecture performs worse than the conventional CNN LeNet5. However, with 75 superpixels the accuracy of MoNet is higher than that of the spectral architecture ChebNet (91,11% against 75,62%).

SplineCNN

3 Proposed method

Motivation of this approach is to make classification neural networks lighter and near of state-of-the-art performance at the same time. Since the same symbols or doodles have the similar structures and the color, size of the symbol and other characteristics don't play a role it is more important to investigate this structure in a spatial way rather than visual way. It leads to the idea of using graph representation of the symbols. Processing of the graph will be more efficient especially if graph has a lot of times fewer nodes than pixels in standard raw image for classification. Architecture of the proposed method is presented at the figure 1. The proposed pipeline consist of 4 phases: binarization, skeletonization, skeleton to graph mapping and graph classification. Consider them more precisely.

Binarization The binary image is required by all the standard skeleton algorithms, except neural networks skeleton generative approach proposed in [].

The plenty of binarization algorithms was discussed in the previous works. For example, in [] binarization algorithms for participated in the ICFHR 2016 binarization competition is described. Generally speaking, binarization process looks in the following way: given image $I_{m \times n \times k} \in \mathbb{R}^{m \times n \times k}$, where m and n is the image size and k is the number of channels of the image. Binarization is the function $B(I) : \mathbb{R}^{m \times n \times k} \to \{0; 1\}^{m \times n \times 1}$.

Skeletonization Skeleton algorithm is based on the approximation of connected object by polygonal figure and attempting to find the sequence of circles of maximum radius inscribed in this figure. This algorithm is described by Mestetskiy and Semenov in []. Algorithm implies of two stages:

1. Boundary corridor construction, when the edge between white and black areas on the binary image is interpolated with the polygonal figure.
2. Construction of skeletons, which has complexity $O(n)$ for figures without holes.

It was shown that algorithm effectively implemented on CPU as well as GPU.

Skeleton to Graph mapping Mapping from obtained skeleton to graph is performed by pruning of skeleton. The proposed in [] algorithm is to computation Hausdorff distance between silhouette and figure constructed from skeleton. Skeleton is pruned till Hausdorff distance is greater than predefined threshold.

Another graph pruning approach is described in []. It is based on the deletion of edges less than threshold and nodes which is coresponded to the intensity of grey pixel on the original image less than threshold.

The obtained graph is presented in the following way: $G = (E, V)$, where V is the set of vertices and E is the set of edges. The graph is matched with the $F \in \mathbf{R}^{N \times D}$ - feature matrix, where N - is a number of vertices and D is the dimension of the feature vector F_i of the v_i vertex. Moreover, each vertex has own coordinates u_i. Hidden representation of the feature vector is the D dimensional vector and $D = 2$ in our case. $F_i = [r_i, d_i]$, where r_i is the scale of the skeleton in this vertex and d_i is the degree of the vertex.

Graph Neural Network The last phase is graph classification algorithm. We are using graph classification neural networks. The short survey of the graph convolution neural networks will be given in the next section.

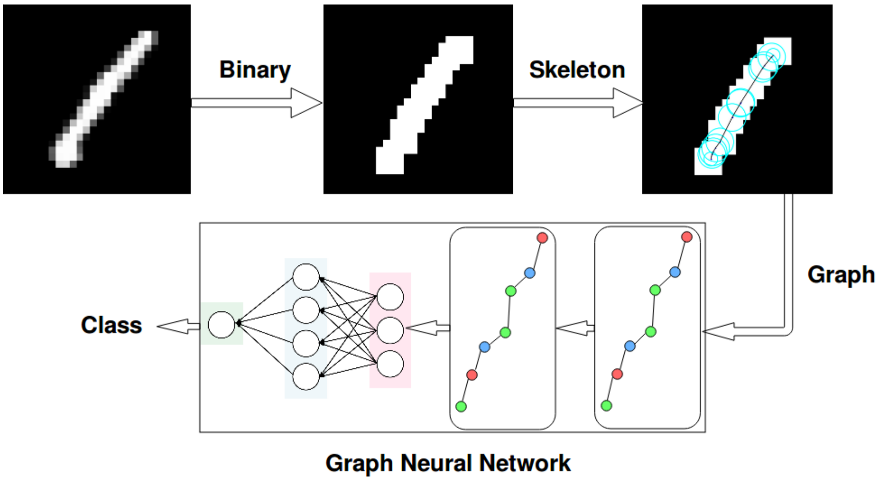

Fig. 1. Proposed method classification architecture

4 Experiments

4.1 MNIST Skeleton Dataset

We solemnly present MNIST Skeleton Dataset [link to dataset], which was collected by us with help of algorithm suggested in [**?**]. Since MNIST dataset is presented by gray images we introduced the binarization function of pixel (i, j) $B_{i,j}$ as the following:

$$B(I)_{i,j} = \begin{cases} 1, & \text{if } I_(i,j) > 0. \\ 0, & \text{otherwise.} \end{cases}, \qquad (1)$$

where $I(i, j)$ return the pixel value in the one-channel image.

On the image 2 is shown the distribution of the number of vertices in the obtained dataset. It is clear that graphs have not big size and the sensitive to the size of the graph in terms of performance NNs can be efficiently applied to skeletons. The representative example of the dataset is show on the picture 3.

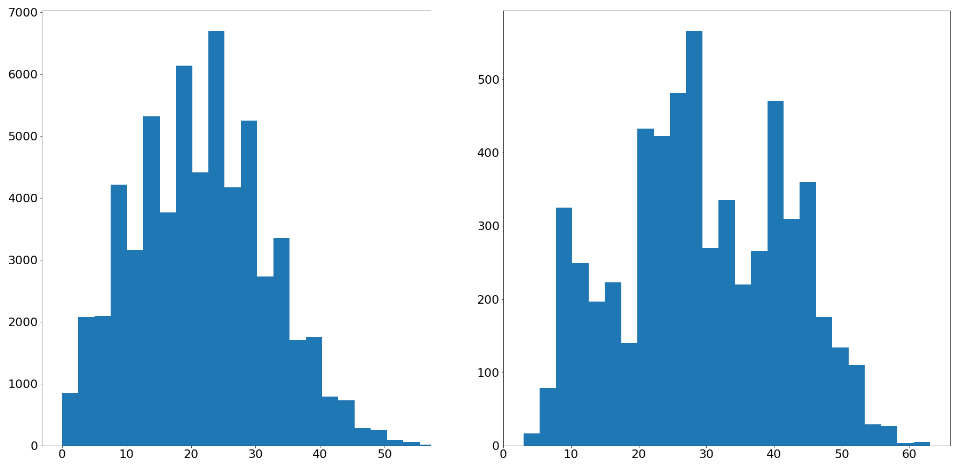

Fig. 2. Left: Distribution of number of nodes in graph in MNIST Skeleton dataset. **Right:** Distribution of number of nodes in number 8.

4.2 Experimental setup

In our experiment we used two datasets: MNIST Skeleton Dataset and MNIST Superpixel 75, which was discribed in [**?**]. We divided all of datasets into three parts: Train set 55K, Test set 10K and Validation set 5K.

Time required for training and accuracy of classification during training were measured for four architectures: k-GNN, MoNet, MPNN, SplineCNN for two datasets. The required memory size on training and inference time were also evaluated. For each architecture the following experiment was carried out: the suggested in the article architecture (see description below) was implemented. The batch size was taken as $min(k; 64)$, where k is maximum batch size, which can be placed in memory. The inference time was measured with batch size 1. The hyper-parameters of the Neural networks was searched as was described in original papers. The same set of hyper-parameters was used for both models: model on skeletons and model on superpixels. The experiments were performed with help of Torch Geometric framework [].

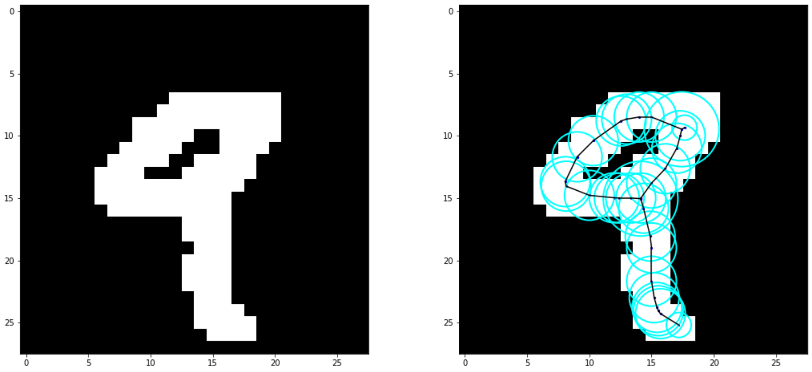

Fig. 3. Left: Binarized version of MNIST image with digit 9. **Right:** Skeleton version of the image.

MPNN model was trained using a uniform random hyper parameter search. The number of passing steps was found as 4. Mean was used as an aggregate function. After message passing phase set2set [] was used as readout function. 8 is the parameter of proceeding steps for set2set. Two full-connected layers and log soft-max proceed after set2set. Models were trained using SGD with the ADAM optimizer [] with learning rate $1e3$. We used a linear learning rate decay that began between $1e-3$ and $1e-5$ of the way through training. Model on skeletons dataset was trained with batch size 64, and model on superpixels dataset with batch size 25.

The k-GNN architecture was implemented as follows. The parameter k was taken equal to 1. The neural network consisted of 3 convolutional layers described in the considered article with a hidden-dimension size of 64. Three fully connected layers with the ELU activation functions and dropout $p = 0.5$ after the first layer were used afterwards. Lastly, the softmax function was applied to the output of the last layer. What is more, we employed the Adam optimizer with an initial learning rate of 10^{-2} and a rate decay based on validation results down to 10^{-5} given the patience parameter is equal to 5. The networks were trained for 100 epochs for both datasets.

The MoNet architecture was implemented in a different way comparing to the original work. It has been found that the graclus clustering and pooling layer interleaved with the convolutions as described in the original paper gives worse performance than three convolution layers with batch normalization and the ELU activation function were followed by the global mean polling layer and two full-connected layers. We used the Adam optimizer with initial learning rate 10^{-4}, dropout factor $p = 0.5$. The batch size 64 was used for both skeletons and superpixels dataset.

4.3 Results

The obtained results of the dependencies between accuracy on the test dataset and training time presented on the image 4. As you can see the training time on superpixels dataset is always much longer than time on skeletons dataset.

Comparing two MPNN models, the skeleton model has a near 1% less of accuracy than superpixel one, but takes significantly less training time. It is worth mentioning that only part of the superpixels model training is shown to not explode the plot on the x axis. MoNet models have noisy accuracy during training despite the fact of using batch normalization []. The both models have shown almost the same accuracy with domination of superpixels model. But superpixel model has been trained two times longer than skeleton one. The best accuracy has been performed by k-GNN on the skeleton dataset and the worse performance.

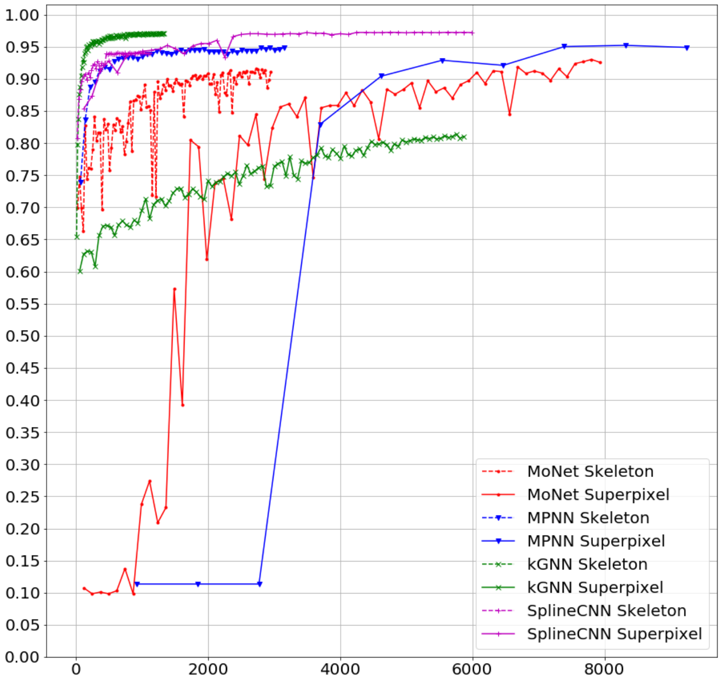

Fig. 4. Caption

Table 1. GPU memory usage during training process

	MPNN	k-GNN	MoNet	SplineCNN
Skeleton	**1.50**	**0.34**	**0.32**	**0.73**
Superpixel	10.76	0.85	1.54	1.04

Table 2. Test time with batch size 1 on the test dataset

	MPNN	k-GNN	MoNet	SplineCNN
Skeleton	**147.2**	47.48	**56.2**	38.8
Superpixel	211.3	**26.23**	140.79	**17.9**

5 Conclusion

6 Discussion

Acknowledgments. The heading should be treated as a subsubsection heading and should not be assigned a number.

References

1. Smith, T.F., Waterman, M.S.: Identification of Common Molecular Subsequences. J. Mol. Biol. 147, 195–197 (1981)
2. May, P., Ehrlich, H.C., Steinke, T.: ZIB Structure Prediction Pipeline: Composing a Complex Biological Workflow through Web Services. In: Nagel, W.E., Walter, W.V., Lehner, W. (eds.) Euro-Par 2006. LNCS, vol. 4128, pp. 1148–1158. Springer, Heidelberg (2006)
3. Foster, I., Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, San Francisco (1999)
4. Czajkowski, K., Fitzgerald, S., Foster, I., Kesselman, C.: Grid Information Services for Distributed Resource Sharing. In: 10th IEEE International Symposium on High Performance Distributed Computing, pp. 181–184. IEEE Press, New York (2001)
5. Foster, I., Kesselman, C., Nick, J., Tuecke, S.: The Physiology of the Grid: an Open Grid Services Architecture for Distributed Systems Integration. Technical report, Global Grid Forum (2002)
6. National Center for Biotechnology Information, `http://www.ncbi.nlm.nih.gov`