Input simplifying as an approach for improving neural network efficiency

No Author Given

No Institute Given

Abstract. With the increasing popularity of smart phones and services, symbol recognition becomes a challenging task in terms of computational capacity. To our best knowledge, existing methods have focused on effective and fast neural networks architectures, including the ones which deal with the graph symbol representation. In this paper, we propose to optimize the neural networks input rather than the architecture. We compare the performance of several existing graph architectures in terms of accuracy, learning and training time using the advanced skeleton symbol representation. It comprises the inner symbol structure and strokes width patterns. We show the usefulness of this representation demonstrating significant reduction of 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

Text recognition in images is a fundamental computer vision problem The great success of smart phones and different vision systems have made character recognition a critical task in human computer interaction. This is due to its various applications such as scene understanding, visual assistance and image retrieval. In particular, these issues can imply letters and digits classification. Current approaches to character recognition [1, 2] are based on the Convolution Neural Networks (CNN) [5], bitmaps and combination of different heuristic to increase performance [6] and reduce learning and inference time.

Existing approaches allows to improve solution performance and efficiency by simplifying its architecture. However, state-of-the-art solutions are still time-consuming and resource-intensive. In this work we introduce a novel symbol classification method, which is based on the skeletonization [7] of the binarized images, the skeleton to graph translation and the further graph classification by the means of Graph Neural Networks (GNN).This work focuses on the classification task which is more challengeable than binarization and extraction of skeletons and its graph representations. The proposed technique simplifies neural network input rather than its architecture.

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 near the same classification accuracy as state-of-the-art GNNs trained on superpixels [8].

The rest of the paper is structured as follows. The second section describes related graph neural networks for the graph classification problems or tasks. The third section presents a proposed classification method. The fourth section is devoted to experiments. The fifth one concludes the results and contribution of this paper.

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. It should be noted that there are internal and external skeletons. In this work we consider the inner skeleton representation only. There are several different approaches to skeleton representation and further graph obtaining. For instance, in [3] 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 [7], 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 work [12], there are two types of Graph Convolution Neural Networks (GCNN): Spectral and Spatial GCNNs. Spectral approaches require input graphs to have the same size. While skeletons may have arbitrary size, that makes spectral GCNNs inapplicable for our task. So, in our work we mainly focus on spatial GCNNs, what in its turn, implies of recurrent and composition networks.

Message Passing Neural Networks (MPNNs) Gilmer and others [9] 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 [10] 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 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 a 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 [8] 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 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 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 Having been inspired by the work of MoNet, Matthias Fey, Jan Eric Lenssen and others proposed a new class of deep neural networks, based on the so-called spline convolution for non-constant geometric structures [4]. According to the authors, SplineCNN is the first deep architecture that can be trained end-to-end on geometric data.

It is assumed that the input data are arbitrary graph structures, each of the vertices of which is associated with a point in d-dimensional space and a feature vector. The core of the proposed convolution is a continuous function defined on a certain interval in the input data space. The parameters to be taught are the coefficients with which the previously selected basic function is summed, placed in the nodes of a uniform grid over the selected interval. Thus, the convolutional layer aggregates the features of the neighboring vertices, weighing them with the learned continuous function. As a pool of layers, this article uses the Graclus method, which clusters the vertices of the graph and declares clusters for the vertices of the squeezed graph. At the same time, either the maximum of cluster attributes (max pooling) or their average pooling (average pooling) is taken as signs of such vertices.

This method surpasses the quality of state-of-the-art architectures in such tasks as the classification of image graphs, the classification of graph nodes and the comparison of forms. At the same time, according to the authors, the proposed class of neural networks has a very good learning and prediction speed and is also optimized for computing on the GPU.

3 Proposed method

The motivation of this approach is to make classification neural networks lighter and near 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 a graph will be more efficient especially if a graph has a lot of times fewer nodes than pixels in a standard raw image for classification.

The proposed algorithm consists of four stages: binarization, skeletonization, skeleton to graph mapping and graph classification. The architecture of the proposed method is presented at the figure 1. In the next paragraphs we will consider them more precisely.

Binarization The binary image is required by all the standard skeleton algorithms, except neural networks skeleton generative approach proposed in [13].

Plenty of binarization algorithms was discussed in the previous works. For example, in [14] 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 a 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 [7]. 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 the algorithm effectively implemented on CPU as well as GPU.

Skeleton to Graph mapping Mapping from the obtained skeleton to the graph is performed by pruning of the skeleton. The proposed in [7] algorithm is to computation Hausdorff distance between the silhouette and the figure constructed from the skeleton. Skeleton is pruned till Hausdorff distance is greater than the predefined 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 the 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.

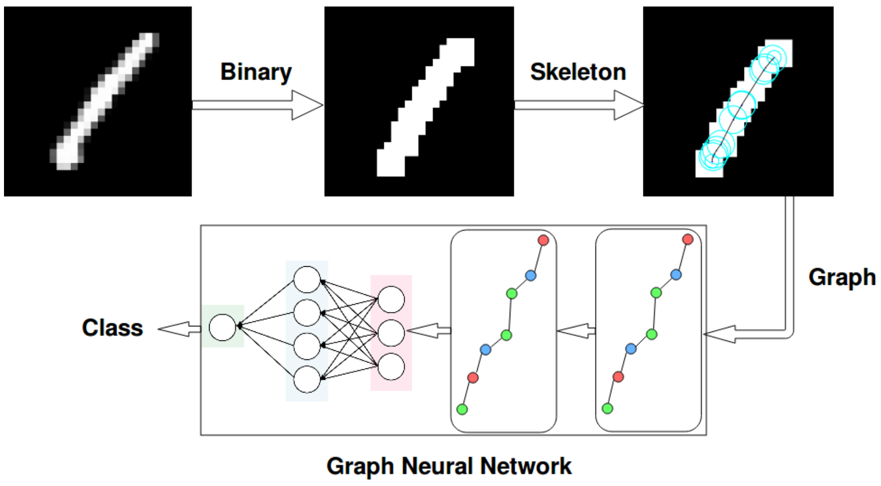

Fig. 1. Pipeline of the proposed approach

4 Experiments

4.1 MNIST Skeleton Dataset

We used MNIST Skeleton Dataset, which was collected by us with the help of the algorithm suggested in [7]. Since MNIST dataset is presented by gray images we introduced the binarization function of pixel (i, j) $B_{i,j}$ as the following:

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

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 shown on the picture 3.

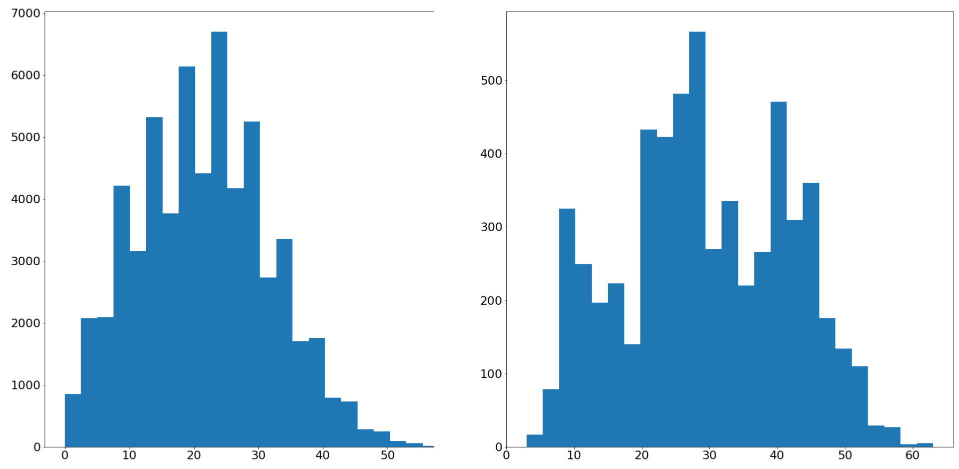

Fig. 2. Left: Distribution of graph nodes number for the MNIST Skeleton dataset. **Right:** Distribution of nodes number in "8" character in particular.

5 Experimental setup

In our experiment we used two datasets: MNIST Skeleton Dataset and MNIST Superpixel 75, which was described in [8]. We divided all of the datasets into three parts: Train set 55K, Test set 10K and Validation set 5K.

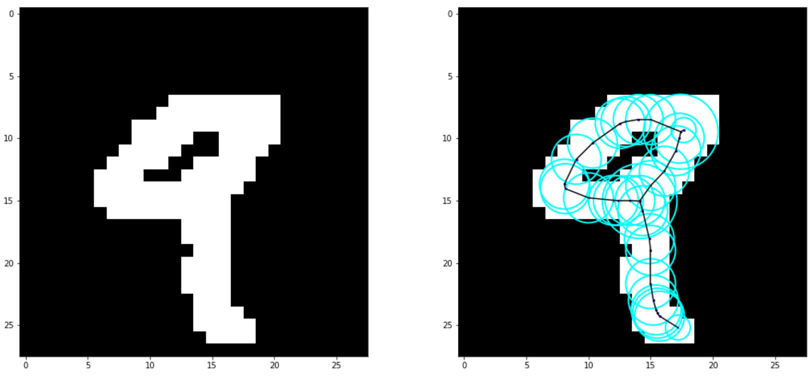

Fig. 3. Left: Binarized version of MNIST image with digit 9. **Right:** Skeleton version of the image.

The time required for training till convergence 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 was also evaluated. Inference time was measured for the whole pipeline of our approach including binarization, skeletonization, skeleton to graph mapping and graph classification. 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 hyperparameters 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 the help of Torch Geometric framework [11].

MPNN model was trained using a uniform random hyperparameter search. The number of passing steps was found as 4. Mean was used as an aggregate function. After message passing phase set2set [15] 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.

6 Results

The obtained training curves (dependencies between validation accuracy on each epoch and training time) are presented on the figure 4. It should be noted that for models trained on the skeleton dataset it is always needed much less training time to achieve the same quality, but on superpixels, in a long perspective, it achieves a higher quality.

There is the average inference time represented in the table 7. It should be noted that for skeletons we also included the prepossessing time needed for skeleton extraction but we did not include the prepossessing time for superpixel extraction. Even in such conditions skeletons show a much better performance for all the models, decreasing the inference time in 13.7, 3.1, 4.4 and 3.1 times for MPNN, k-GNN, MoNet and SplineCNN respectively. In the table 7 you can find the maximum GPU memory used during the training for all the models. It is shown that models trained on the skeleton dataset use significantly less memory.

Finally, classification accuracy is presented in the table 7. Mainly the number of errors is approximately 2 times higher for the skeleton dataset, except the MoNet architecture, which achieves a slightly higher accuracy on the skeletons.

7 Conclusion

In this work we have compared 4 Graph Neural Networks: MoNet, SplineCNN, MPNN and k-GNN in terms of inference time, training curves and memory usage. The comparison had been performed on the MNIST dataset with two different types of preprocessing: skeletonization and superpixel extraction. Both preprocessings represents a binarized image as graphs but skeletons have a fewer number of nodes, and thus less complex structures. The main result of our work is have shown that input simplification can significantly increase the efficiency of the neural network while sacrificing a little accuracy, what can be a desired trade-off for mobile and embedded systems. Models trained on skeleton dataset shows a least 3 times better inference time, at least 2 times less GPU memory usage and require significantly less time for training.

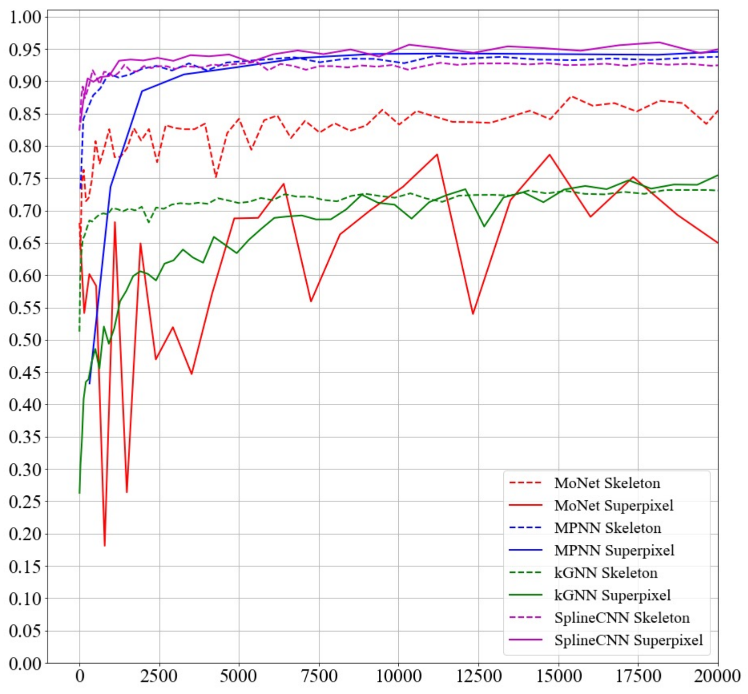

Fig. 4. Training curve for the GNNs on both skeletos and superpixels datasets

References

1. Palvanov A., Im Cho Y.: Comparisons of deep learning algorithms for MNIST in real-time environment. International Journal of Fuzzy Logic and Intelligent Systems. 18(2), 126–134 (2018)
2. Zou, X., Duan, S., Wang, L., Zhang, J.: Fast Convergent Capsule Network with Applications in MNIST. International Symposium on Neural Networks. pp. 3–10. Springer, Cham (2018)
3. Bai, X., Latecki, L.J., Liu, W.-Y, 2007. Skeleton pruning by contour partitioning with discrete curve evolution. IEEE transactions on pattern analysis and machine intelligence, vol. 29, No. 3, March 2007

Table 1. Average inference time in mileseconds

	MPNN	***k*-GNN**	**MoNet**	**SplineCNN**
Skeleton	**4.4**	**3.6**	**3.6**	**3.5**
Superpixel	60.5	11.2	15.8	10.9

Table 2. Maximum GPU memory usage during training process presented in gigabytes

	MPNN	***k*-GNN**	**MoNet**	**SplineCNN**
Skeleton	**1.50**	**0.34**	**0.32**	**0.53**
Superpixel	10.76	0.85	1.54	1.04

4. Fey M. et al.: SplineCNN: Fast geometric deep learning with continuous B-spline kernels. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 869–877 (2018)
5. LeCun, Y., Bengio, Y.: Convolutional networks for images, speech, and time series. The handbook of brain theory and neural networks, 3361(10) (1995)
6. Ciresan, D. C., Meier, U., Gambardella, L. M., Schmidhuber, J.: Convolutional neural network committees for handwritten character classification. In 2011 International Conference on Document Analysis and Recognition. IEEE, pp. 1135–1139 (2011)
7. Mestetskiy, L., Semenov, A.: Binary Image Skeleton-Continuous Approach. VISAPP, 1, pp. 251–258 (2008)
8. Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., Bronstein, M. M.: Geometric deep learning on graphs and manifolds using mixture model cnns. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5115–5124. (2017)
9. Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., Dahl, G. E.: . Neural message passing for quantum chemistry. In Proceedings of the 34th International Conference on Machine Learning. 70 1263–1272 (2017)
10. Riba, P., Fischer, A., Llads, J., Forns, A.: Learning graph distances with message passing neural networks. In 2018 24th International Conference on Pattern Recognition (ICPR). IEEE, pp. 2239–2244 (2018)
11. Fey, M., Lenssen, J. E.: Fast Graph Representation Learning with PyTorch Geometric. arXiv preprint arXiv:1903.02428 (2019)
12. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Yu, P. S.: A comprehensive survey on graph neural networks. arXiv preprint arXiv:1901.00596 (2019)
13. Shen, W., Zhao, K., Jiang, Y., Wang, Y., Bai, X., Yuille, A.: Deepskeleton: Learning multi-task scale-associated deep side outputs for object skeleton extraction in natural images. IEEE Transactions on Image Processing, 26(11), 5298–5311 (2017)
14. Pratikakis, I., Zagoris, K., Barlas, G., Gatos, B.: ICFHR2016 handwritten document image binarization contest (H-DIBCO 2016). In 2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR). IEEE, pp. 619–623 (2016)
15. Vinyals, O., Bengio, S., Kudlur, M.: Order matters: Sequence to sequence for sets. arXiv preprint arXiv:1511.06391. (2015)
16. Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Adam, H. et al.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)

Table 3. Classification accuracy in percents

	MPNN	***k*-GNN**	**MoNet**	**SplineCNN**
Skeleton	94.5	74.4	**91.7**	93.4
Superpixel	**97.3**	**81.5**	89.7	**97.5**

17. Jongejan, J., Rowley, H., Kawashima, T., Kim, J., Fox-Gieg, N.: The quick, draw!-ai experiment. Mount View, CA, (2016)