Set Aggregation Network as a Trainable Pooling Layer

Łukasz Maziarka[1,2], Marek Śmieja[1], Aleksandra Nowak[1], Jacek Tabor[1], Łukasz Struski[1], and Przemysław Spurek[1]

[1] Faculty of Mathematics and Computer Science
Jagiellonian University
Łojasiewicza 6, 30-348, Kraków, Poland
marek.smieja@ii.uj.edu.pl
[2] Ardigen
Podole 76, 30-394, Kraków, Poland

Abstract. Global pooling, such as max- or sum-pooling, is one of the key ingredients in deep neural networks used for processing images, texts, graphs and other types of structured data. Based on the recent DeepSets architecture proposed by Zaheer et al. (NIPS 2017), we introduce a Set Aggregation Network (SAN) as an alternative global pooling layer. In contrast to typical pooling operators, SAN allows to embed a given set of features to a vector representation of arbitrary size. We show that by adjusting the size of embedding, SAN is capable of preserving the whole information from the input. In experiments, we demonstrate that replacing global pooling layer by SAN leads to the improvement of classification accuracy. Moreover, it is less prone to overfitting and can be used as a regularizer.

Keywords: Global pooling · Structured data · Representation learning · Convolutional neural networks · Set processing · Image processing.

1 Introduction

Deep neural networks are one of the most powerful machine learning tools for processing structured data such as images, texts or graphs [9, 18]. While convolutional or recurrent neural networks allow to extract a set of meaningful features, it is not straightforward how to vectorize their output and pass it to the fully connected layers.

One typical approach to this problem relies on flattening the given tensor. However, the flattened vector may contain a lot of redundant information, which in turn may lead to overfitting. Moreover, flattening cannot be followed by a dense layer (e.g. in classification models), when the input has varied size [2, 11]. This situation often appears in graphs and texts classification, but is also common in learning from images of different resolutions [1, 4, 7].

In order to produce a fixed-length vector, a maximum or sum function may be applied to the learned data representations. This operation is commonly known as the global pooling. In image recognition the data is frequently aggregated by computing the average value over the channels of the feature map tensor obtained by the backbone network. Such vector is then passed to the predictor head. This is the case in numerous large scale networks such as ResNet [6], DenseNet [8] or, more recent, Amoeba-

Net [17]. In Graph Neural Networks the representation of a given node is usually computed by recursively mean-pooling the representations of its neighbours [12, 22]. Despite its wide applicability, the global pooling layer is not able to fully exploit the information from the input data, because it does not contain trainable parameters. Moreover, the global pooling cannot adjust the dimensionality of the representation, because the size of its result is solely determined by the number of input channels.

An additional requirement often imposed on the aggregation function is the invariance to the permutation of the input. This constraint arises as a consequence of set processing, and is present, for instance, in 3D point cloud recognition [15] and graph analysis [22]. The issue of efficiently learning a representation of permutation invariant inputs was recently studied by Zaheer et al. [23], who proposed a unified methodology for the processing of sets by neural networks. Their model, called DeepSets, is based on the idea of summing the representations of the sets elements. This concept was also further developed by [14] who define the pooling function as the average of permutation-sensitive maps of all reorderings of the sequence of the set members. Those permutation-sensitive functions may be modeled by recurrent networks and allow for learning representations that utilize high order dependencies between the set points already at the aggregation time. Neural networks capable of processing sets where also analyzed by [19], who prove that the studied permutation invariant/equivariant deep models are universal approximators of permutation invariant/equivariant functions.

In this paper, we propose Set Aggregation Network (SAN), an alternative to the global pooling operation, which guarantees no information loss. For this purpose, we adapt the DeepSets architecture to embed a set of features retrieved from structured data into a vector of fixed length. Contrary to pooling operation, the parameters of SAN are trainable and we can adjust the size of its representation. In addition to Zaheer et al. [23], we prove that for a sufficiently large latent space, SAN learns a unique representation of every input set, which justifies this approach from a theoretical perspective (Theorem 1).

Our experimental results confirm that replacing global pooling by SAN leads to the improvement of classification accuracy of convolutional neural networks used in classification (Section 4). Moreover, SAN is less prone to overfitting, which allows to use it as a regularizer. The experiments were carried out on a small network architecture as well as on the large-scale ResNet model.

2 Set Aggregation Network

Suppose that we want to process structured data $X = (x_i)_i \subset \mathbb{R}^D$ by the use of a neural network. Some practical examples of X may include a sequence of word embeddings, image represented as a set of pixels or a graph structure. In this paper, we study one of the typical architectures used for processing such data. It consists of two networks combined with an intermediate pooling layer:

$$X = (x_i)_i \xrightarrow{\Psi} (\Psi x_i)_i \xrightarrow{\text{Pool}} \text{Pool}\{\Psi(x_i) : i\} \xrightarrow{\Phi} \mathbb{R}^N. \tag{1}$$

The first network $\Psi : \mathbb{R}^{aD} \to \mathbb{R}^K$, where $a \in \mathbb{N}$, is responsible for extracting meaningful features of structured data. In the case of images it can be a convolutional

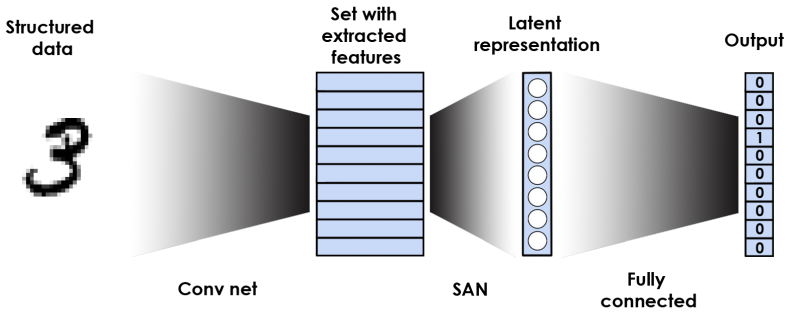

Fig. 1: SAN is an intermediate network which is responsible for learning a vector representation using a set of features extracted from of structured data.

network, while for a sequential data, such as texts, it may be a recurrent neural network. This network transforms elements of X sequentially and produces a set (or sequence) of K-dimensional vectors. Depending on the size of X (length of a sentence or image resolution), the number of vectors returned by Ψ may vary. To make the response of Ψ equally-sized, a global pooling is applied, which returns a single K-dimensional vector. A pooling layer, commonly implemented as a sum or maximum operation over the set of K-dimensional vectors, gives a vector representation of structured object. Finally, a network $\Phi : \mathbb{R}^K \to \mathbb{R}^N$ maps the resulting representation to the final output.

The basic problem with the above pipeline lies in the pooling layer. Global pooling "squashes" a set of K-dimensional vectors into a single output with K attributes. A single K-dimensional output vector may be insufficient to preserve the whole information contained in the input set (especially for large sets and small K), which makes a pooling operation the main bottleneck of the above architecture. In this paper, we would like to address this problem. We focus on defining more suitable aggregation network, which will be able to learn a sufficiently rich latent representation of structured data.

To replace a pooling layer, we extend DeepSets architecture introduced in [23] to the case of structured data. In consequence, we define an aggregation network, which embeds a set of extracted features to a fixed-length representation. First, we recall a basic idea behind pioneering work of Zaheer et al. and explain its use as an alternative to the classical pooling layer. In the next section, we prove that this framework is able to preserve the whole information contained in the set structure.

Let $f : \mathbb{R}^D \supset X \to y \in Y$ be a function, which maps sets $X = (x_i)_i$ to some target values $y \in Y$. Since f deals with sets, then the response of f should be invariant to the ordering of set elements. Zaheer et al. [23] showed that to realize this requirement f has to be decomposed in the form:

$$f(X) = \rho(\sum_i \tau(x_i)), \tag{2}$$

for suitable transformations ρ, τ. In the case of neural networks, f can be implemented by constructing two networks τ and ρ. The first network processes elements of a given

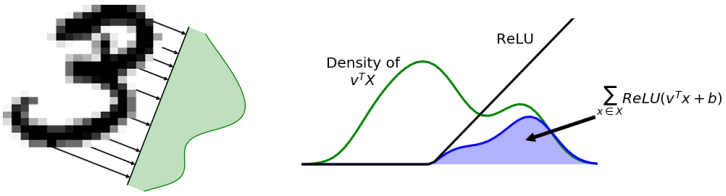

Fig. 2: The idea of our approach is to aggregate information from projections of a set onto several one-dimensional subspaces (left). Next non-linear activation function is applied to every set element and the results are aggregated (right).

set X sequentially. Next, the response of τ is summarized over the whole elements of X and a single vector is returned. Finally, a network ρ maps the resulting representation to the final output.

One can directly adapt the above architecture to the pipeline considered in this paper. Namely, instead of taking the maximum or sum pooling over the set of extracted features, we define a separate neural network τ to compute the summarized activation for all set elements (the role of ρ is played by a network Φ in our framework). We refer to this network as set aggregation network (SAN), see Figure 1. If τ contains M output neurons, then we get M-dimensional vector representation of the structured data. In contrast to pooling operation, which always returns K-dimensional vector (K is a dimension of input feature vectors), the size of representation produced by SAN may be adjusted to a given problem. Moreover, their parameters are trainable and thus we allow for learning the most convenient representation of structured data. Although SAN is designed to process permutation invariant structures (sets), one may add special attributes to indicate the ordering of extracted features. One way is to use the normalization of the elements index or its trigonometric transformation [20, Section 3.5].

The following remark shows that max-pooling is a special case of SAN.

Remark 1. Observe that max-pooling is a special case of SAN. Clearly, for non-negative scalar data $X = (X_i) \subset \mathbb{R}$ and function $\tau_p(x) = x^p$, we have:

$$\tau^{-1}(\sum_i \tau(x_i)) \to \max_i(x_i) \text{ , as } p \to \infty. \tag{3}$$

To obtain a maximum, we use τ as the activity function in aggregative neuron, which is followed by a layer with its inverse. By extending this scheme, we can get a maximum value for every coordinate. Additionally, to deal with negative numbers, we first take the exponent followed by logarithm after the aggregation.

3 Theoretical Analysis

Although Zaheer et al. theoretically derived the form of f as the only permutation invariant transformation operating on sets, there is no guarantees that this network is capable of learning a unique representation for every set. In this section we address this

question and show that if τ is a universal approximator, then $\sum_{x\in X}\tau(x)$ gives a unique embedding of every set X in a vector space.

Before a formal proof, we first give an illustrative argument for this fact. A typical approach used in computer tomography applies Radon transform [5,16] to reconstruct a function (in practice the 2D or 3D image) from the knowledge of its integration over all one-dimensional lines. A similar statement is given by the Cramer-Wold Theorem [3], which says that for every two distinct measures one can find their one-dimensional projection which discriminates them. This implies that without loss of information we can process the set $X \subset \mathbb{R}^K$ through its all one-dimensional projections $v^T X \subset \mathbb{R}$, where $v \in \mathbb{R}^K$.

In consequence, we reduce the question of representing a multidimensional set to the characterization of one-dimensional sets. Next, one can easily see that the one-dimensional set $S \subset \mathbb{R}$ can be retrieved from the knowledge of aggregated ReLU on its translations: $b \to \sum_i \mathrm{ReLU}(s_i + b)$, see Figure 2. Summarizing the above reasoning, we obtain that the full knowledge of a set $X \subset \mathbb{R}^K$ is given by the scalar function

$$\mathbb{R}^K \times \mathbb{R} \ni (v, b) \to \sum_i \mathrm{ReLU}(v^T x_i + b). \tag{4}$$

Now, given M vectors $v_i \in \mathbb{R}^K$ and biases $b_i \in \mathbb{R}$, we obtain the fixed-size representation of the set $X \subset \mathbb{R}^K$ as a point in $\mathbb{R}^M$ given by

$$[\sum_i \mathrm{ReLU}(v_1^T x_i + b_1), \ldots, \sum_i \mathrm{ReLU}(v_M^T x_i + b_M)] \in \mathbb{R}^M. \tag{5}$$

The above transformation directly coincides with a single layer SAN parametrized by ReLU function. Thus for a sufficiently large number of neurons, SAN allows to uniquely identify every input set.

Now, we show formally that the above reasoning can be extended to a wide range of activity functions. For this purpose, we will use the UAP (universal approximation property). We say that a family of neurons $\mathcal{N}$ has UAP if for every compact set $K \subset \mathbb{R}^D$ and a continuous function $f : K \to \mathbb{R}$ the function f can be arbitrarily close approximated with respect to supremum norm by $\mathrm{span}(\mathcal{N})$ (linear combinations of elements of $\mathcal{N}$). We show that if a given family of neurons satisfies UAP, then the corresponding SAN allows to distinguish any two sets:

Theorem 1. *Let X, Y be two sets in $\mathbb{R}^D$. Let $\mathcal{N}$ be a family of functions having UAP. If*

$$\tau(X) = \tau(Y) \text{ , for every } \tau \in \mathcal{N}, \tag{6}$$

then $X = Y$.

Proof. Let μ and ν be two measures representing sets X and Y, respectively, i.e. $\mu = \mathbf{1}_X$ and $\nu = \mathbf{1}_Y$. We show that if $\tau(X) = \tau(Y)$ then μ and ν are equal.

Let $R > 1$ be such that $X \cup Y \subset B(0, R-1)$, where $B(a, r)$ denotes the closed ball centered at a and with radius r. To prove that measures μ, ν are equal it is sufficient to prove that they coincide on each ball $B(a, r)$ with arbitrary $a \in B(0, R-1)$ and radius $r < 1$.

Let ϕ_n be defined by

$$\phi_n(x) = 1 - n \cdot d(x, B(a,r)) \text{ for } x \in \mathbb{R}^D, \tag{7}$$

where $d(x,U)$ denotes the distance of point x from the set U. Observe that ϕ_n is a continuous function which is one on $B(a,r)$ an and zero on $\mathbb{R}^D \setminus B(a, r+1/n)$, and therefore ϕ_n is a uniformly bounded sequence of functions which converges pointwise to the characteristic funtion $\mathbf{1}_{B(a,r)}$ of the set $B(a,r)$.

By the UAP property we choose $\psi_n \in \text{span}(\mathcal{N})$ such that

$$\sup_{x \in B(0,R)} |\phi_n(x) - \psi_n(x)| \leq 1/n. \tag{8}$$

Thus ψ_n restricted to $B(0,R)$ is also a uniformly bounded sequence of functions which converges pointwise to $\mathbf{1}_{B(a,r)}$. Since $\psi_n \in \mathcal{N}$, by (6) we get

$$\sum_{x \in X} \mu(x)\psi_n(x) = \sum_{y \in Y} \nu(y)\psi_n(y). \tag{9}$$

Now by the Lebesgue dominated convergence theorem we trivially get

$$\begin{aligned} \sum_{x \in X} \mu(x)\psi_n(x) &= \int_{B(0,R)} \psi_n(x) d\mu(x) \to \mu(B(a,r)), \\ \sum_{y \in Y} \nu(y)\psi_n(y) &= \int_{B(0,R)} \psi_n(x) d\nu(x) \to \nu(B(a,r)), \end{aligned} \tag{10}$$

which completes proof.

4 Experiments

We apply SAN to typical architectures used for processing images. Our primary goal is to compare SAN with global pooling in various settings. First, we focus on classifying images of the same sizes using a small convolutional neural network. Next, we extend this experiment to the case of images with varied resolutions. Finally, we consider a large scale ResNet architecture and show that replacing a global pooling layer by SAN leads to the improvement of classification performance. Our implementation is available at github[3].

4.1 Small Classification Network

We considered a classification task on CIFAR-10 [13], which consists of 60 000 color images of the size 32x32 and the Fashion-MNIST, composed of 70 000 gray-scale pictures of size 28x28 [21]. We used small neural networks with 3 convolutional layers with ReLU activity function and a max-pooling between them, see Table 1 for details.

To aggregate resulted tensor we considered the following variants:

[3] https://github.com/gmum/set-aggregation.

Table 1: Architecture summary of a small neural network (N is the size of the input to the layer, and D is the number of output neurons from the SAN layer).

Flatten			Max/Avg-pooling			SAN		
Type	Kernel	Outputs	Type	Kernel	Outputs	Type	Kernel	Outputs
Conv 2d	3x3	32	Conv 2d	3x3	32	Conv 2d	3x3	32
Max pooling	2x2		Max pooling	2x2		Max pooling	2x2	
Conv 2d	3x3	64	Conv 2d	3x3	64	Conv 2d	3x3	64
Max pooling	2x2		Max pooling	2x2		Max pooling	2x2	
Conv 2d	3x3	64	Conv 2d	3x3	64	Conv 2d	3x3	64
Flatten			Max/Avg pooling	NxN		SAN		D
Dense		128	Dense		D	Dense		10
Dense		10	Dense		10			

- **flatten**: We flattened a tensor to preserve the whole information from the previous network.
- **conv-1x1**: We applied 1x1 convolutions with one channel and flattened the output. This reduces the number of parameters in subsequent dense layer compared to the classical flatten approach.
- **max-pooling**: We applied max pooling along spatial dimensions (width and height of a tensor) to reduce the dimensionality. In consequence, we obtained a vector of the size equal the depth of the tensor.
- **avg-pooling**: We considered a similar procedure as above, but instead of max pooling we used average pooling.
- **SAN**: We used a single layer SAN as an alternative aggregation. The resulting tensor was treated as a set of vectors with sizes equal to the depth of the tensor. Moreover, the (normalized) indices were added to every vector to preserve the information about local structure.

SAN allows to select the number of output neurons. For the experiment, we considered the following numbers of output neurons: $\{128, 256, 512, 1024, 2048, 4096\}$. To use the same number of parameters for global pooling and conv1x1 approaches we followed them by a dense network with identical number of neurons to SAN. In the case of flatten, we obtained a comparable number of parameters to the other networks trained on the size 4 096. In each case we used additional two dense layers, except for the SAN model, where only one dense layer was used. All models were trained using adam optimizer with a learning rate 10^{-3} and batch size 256. We used 5 000 images for the validation set, 5 000 images for the test set for both CIFAR-10 and Fashion-MNIST. We trained every model on the remaining images.

It is evident from Table 2 that SAN outperformed all reported operations on CIFAR-10. In addition, it gave higher accuracy than flatten when both approaches have a comparable number of parameters (last row). We verified that lower results of flatten were caused by its slight overfitting to the training data. Adding dropout to flatten makes both approaches comparable. In the case of significantly simpler Fashion-MNIST dataset, the differences between all methods are smaller. SAN achieved an identical accuracy to flatten for the size 2048. Note however, that the flatten approach uses twice as much

Table 2: Classification accuracy on a small network (images with the same resolutions) for different number of parameters used in aggregation layer.

	Fashion MNIST					CIFAR 10				
size	flatten	avg-pool	max-pool	conv-1x1	SAN	flatten	avg-pool	max-pool	conv-1x1	SAN
128	–	0.852	0.901	**0.916**	0.912	–	0.624	0.690	0.731	**0.738**
256	–	0.877	0.908	**0.916**	0.911	–	0.649	0.697	0.738	**0.739**
512	–	0.879	0.906	**0.918**	0.910	–	0.671	0.701	0.722	**0.730**
1024	–	0.888	0.914	0.912	**0.915**	–	0.659	0.683	0.722	**0.756**
2048	–	0.889	0.912	0.914	**0.919**	–	0.697	0.686	0.707	**0.733**
4096	**0.919**	0.912	0.900	0.914	0.912	0.720	0.738	0.708	0.709	**0.762**

parameters as SAN. This demonstrates that by the use of SAN the network can be simplified without any loss in accuracy.

To get more insight to the results, we present learning curves for CIFAR-10 and Fashion-MNIST in Figure 3 and Figure 4, respectively. It is evident that max-pooling and conv-1x1 highly overfitted to training data, especially on the more demanding CIFAR-10 dataset. Although avg-pooling presented comparable accuracy on train and test sets, its overall performance was quite low, and matched that of the other methods only for high number of the parameters. In contrast, SAN provided high accuracy and did not suffer from high overfitting to training data. In addition, for the Fashion-MNIST experiment, SAN converged to high performance (over 90%) much faster than the corresponding avg-pooling approach.

This experiment partially confirms our theoretical result that for a sufficient number of neurons, SAN is able to preserve the whole information contained in the input. On the other hand, it shows that SAN can work as a regularizer, which prevents the model from overfitting.

4.2 Classifying Images with Varied Sizes

Most classification models assume that input images are of the same size. If this is not the case, we are forced to scale images at preprocessing stage or use pooling operation as an intermediate layer to apply fully connected layers afterwards. In this experiment, we compared SAN with max-pooling and avg-pooling in classifying images of different sizes. We used analogical architecture as in previous section. Note that we were unable to use flatten or conv-1x1, because the output from convolutional network had different sizes.

We again considered Fashion-MNIST and CIFAR-10 datasets. To create examples with different sizes we used bicubic interpolation on randomly selected images[4]. We examined two cases. In the first one, the network was trained only on images with

[4] For CIFAR-10, original images of size 32×32 were scaled to $16 \times 16, 24 \times 24, 32 \times 32, 40 \times 40, 48 \times 48$. For Fashion-MNIST, images of size 28×28 were scaled to $14 \times 14, 22 \times 22, 42 \times 42, 56 \times 56$.

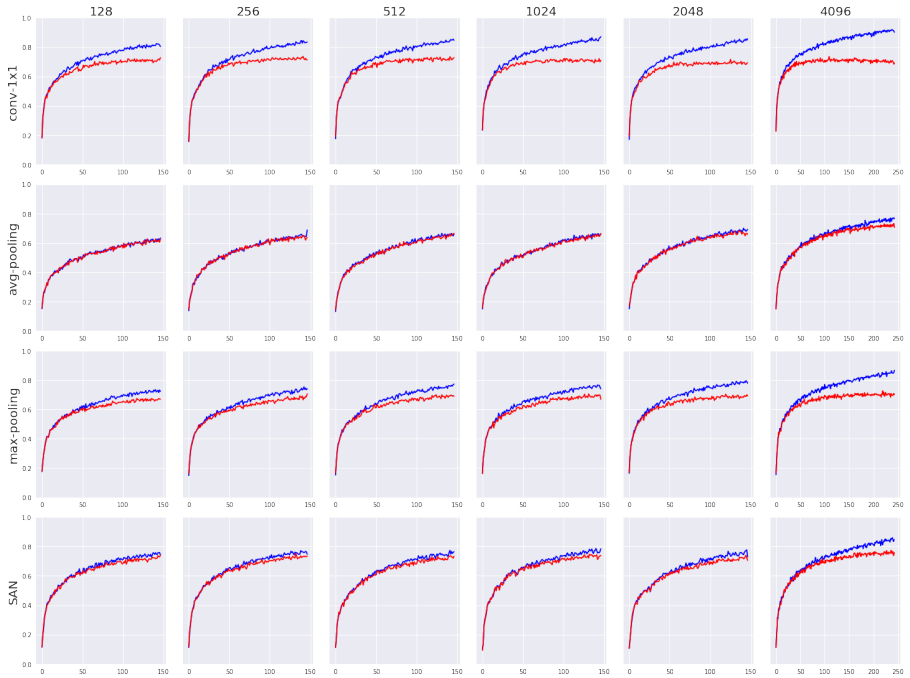

Fig. 3: Train (blue) and test accuracy (red) on CIFAR-10 (images with the same resolutions) for different number of parameters used in aggregation layer.

original resolution, but tested on images with different resolutions. In the second case, scaled images were used both in training and testing.

The results presented in Table 3 show that **SAN** produced more accurate results than both global pooling approaches for almost every image resolution. Observe that the results are worse when only images with 32×32 size were used in train set. It can be explained by the fact that convolutional filters were not trained to recognize relevant features from images with different scales. In this case, the differences are even higher.

4.3 Large Scale ResNet Experiment

In previous experiments we deliberately use a rather simple network in order the examine the effect of only alternating the aggregation method. This allows for the assessment of the methods performance in isolation from any additional layers which could further improve models regularization effect and which are necessary to efficiently train a vast network (such as, for instance, batch norm [10]). In this experiment, we tested the impact of using SAN in a large-scale network.

For this purpose we chose the ResNet-56 model [6], which consists of 56 layers. The original ResNet uses the global average pooling approach followed by a dense layer, which projects the data into the output dimension. Our modification relies on replacing global pooling by SAN. As introduction of the SAN with 4096 vectors comes

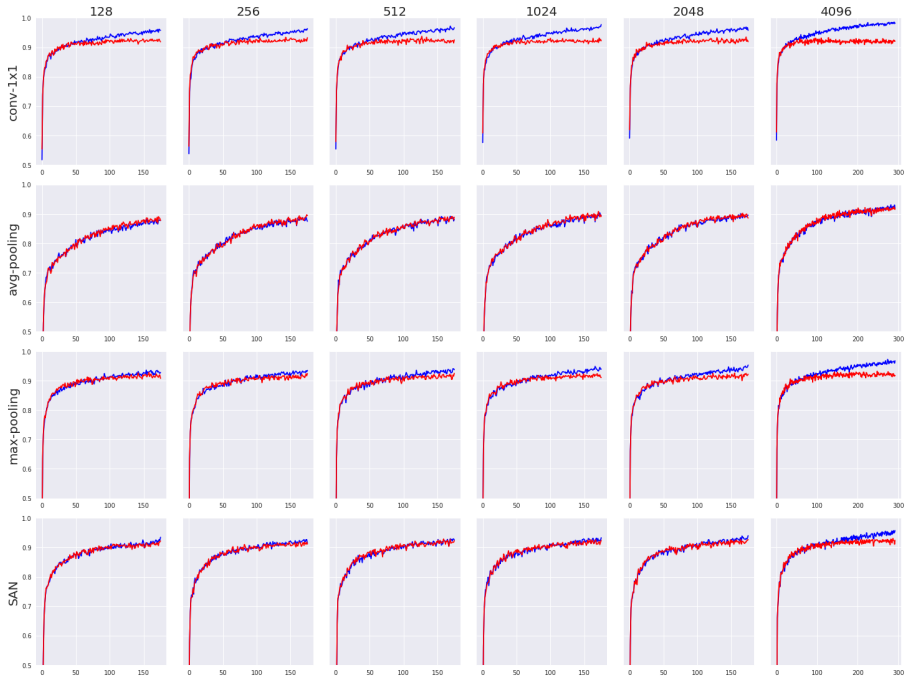

Fig. 4: Train (blue) and test accuracy (red) on Fashion-MNIST (images with the same resolutions) for different number of parameters used in aggregation layer.

at a cost of increased number of parameters, we added an additional, penultimate dense layer with hidden dimension 4096 to the ResNet for the average- and the max-pooling, and the conv-1x1, in order to allow for fair comparison.

The results for CIFAR-10 dataset are reported in Table 4. It is evident that the introduction of SAN to the original ResNet architecture led to the improvement of classification accuracy. Moreover, SAN outperformed other aggregation approaches.

5 Conclusion

In this paper, we proposed a novel aggregation network, SAN, for processing structured data. Our architecture is based on recent methodology used for learning from permutation invariant structures (sets) [23]. In addition, to Zaheer's work, we showed that for a sufficiently large number of neurons, SAN allows to preserve the whole information contained in the input. This theoretical result was experimentally confirmed applying convolutional network to image data. Conducted experiments demonstrated that the replacing of global pooling by SAN in typical neural networks used for processing images leads to higher performance of the model.

Table 3: Classification accuracy for images with varied resolutions.

		Trained on all resolutions			Trained only on original resolution		
Dataset	Image size	max-pool	avg-pool	SAN	max-pool	avg-pool	SAN
Fashion MNIST	14x14	0.8788	0.8753	**0.8810**	0.2519	0.270	**0.2884**
	22x22	0.8969	0.9002	**0.9064**	0.7380	0.801	**0.8247**
	28x28	0.9023	0.9078	**0.9111**	0.9062	0.904	**0.9150**
	42x42	0.9020	**0.9041**	0.9033	0.5548	0.6511	**0.6893**
	56x56	0.8913	0.8960	**0.8966**	0.3274	0.3809	**0.4515**
CIFAR-10	16x16	0.5593	0.5820	**0.6305**	0.3251	0.2714	**0.3808**
	24x24	0.6450	0.6935	**0.7317**	0.6409	0.6130	**0.6956**
	32x32	0.6729	0.7018	**0.7565**	0.7131	**0.7637**	0.7534
	40x40	0.6739	0.6914	**0.7430**	0.6512	0.6780	**0.7234**
	48x48	0.6770	0.6625	**0.7626**	0.5325	0.5366	**0.6264**

Table 4: Test accuracy on CIFAR-10 using the ResNets architecture. The first column corresponds to the original ResNet model. The ResNet-avg/max/conv-1x1 models come with an additional penultimate dense layer of size 4096, in order to match the number of parameters in SAN .

	original	**ResNet-avg**	**ResNet-max**	**ResNet-conv1x1**	**ResNet-SAN**
error	0.0735	0.0724	0.0782	0.0780	**0.0697**

Acknowledgements

This work was partially supported by the National Science Centre (Poland) grants numbers: 2018/31/B/ST6/00993, 2017/25/B/ST6/01271 and 2015/19/D/ST6/01472.

References

1. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. Computer networks and ISDN systems **30**(1-7), 107–117 (1998)
2. Ciresan, D.C., Meier, U., Masci, J., Maria Gambardella, L., Schmidhuber, J.: Flexible, high performance convolutional neural networks for image classification. In: IJCAI Proceedings-International Joint Conference on Artificial Intelligence. vol. 22, p. 1237. Barcelona, Spain (2011)
3. Cramér, H., Wold, H.: Some theorems on distribution functions. Journal of the London Mathematical Society **1**(4), 290–294 (1936)
4. Frasconi, P., Gori, M., Sperduti, A.: A general framework for adaptive processing of data structures. IEEE transactions on Neural Networks **9**(5), 768–786 (1998)
5. van Ginkel, M., Hendriks, C.L., van Vliet, L.J.: A short introduction to the radon and hough transforms and how they relate to each other. Delft University of Technology (2004)
6. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778 (June 2016). https://doi.org/10.1109/CVPR.2016.90

7. He, K., Zhang, X., Ren, S., Sun, J.: Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE transactions on pattern analysis and machine intelligence **37**(9), 1904–1916 (2015)
8. Huang, G., Liu, Z., Weinberger, K.Q.: Densely connected convolutional networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2261–2269 (2017)
9. Iizuka, S., Simo-Serra, E., Ishikawa, H.: Globally and locally consistent image completion. ACM Transactions on Graphics (TOG) **36**(4), 107 (2017)
10. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37. pp. 448–456. ICML'15, JMLR.org (2015)
11. Karpathy, A., Fei-Fei, L.: Deep visual-semantic alignments for generating image descriptions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3128–3137 (2015)
12. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
13. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Tech. rep., Citeseer (2009)
14. Murphy, R.L., Srinivasan, B., Rao, V., Ribeiro, B.: Janossy pooling: Learning deep permutation-invariant functions for variable-size inputs. In: International Conference on Learning Representations (2019), https://openreview.net/forum?id=BJluy2RcFm
15. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 77–85 (2017)
16. Radon, J.: On the determination of functions from their integral values along certain manifolds. IEEE transactions on medical imaging **5**(4), 170–176 (1986)
17. Real, E., Aggarwal, A., Huang, Y., Le, Q.V.: Regularized evolution for image classifier architecture search. arXiv preprint arXiv:1802.01548 (2018)
18. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention. pp. 234–241. Springer (2015)
19. Sannai, A., Takai, Y., Cordonnier, M.: Universal approximations of permutation invariant/equivariant functions by deep neural networks. arXiv preprint arXiv:1903.01939 (2019)
20. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems. pp. 5998–6008 (2017)
21. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747 (2017)
22. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? In: International Conference on Learning Representations (2019), https://openreview.net/forum?id=ryGs6iA5Km
23. Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R.R., Smola, A.J.: Deep sets. In: Advances in Neural Information Processing Systems. pp. 3391–3401 (2017)