Rep the Set: Neural Networks for Learning Set Representations

Konstantinos Skianis
École Polytechnique

Giannis Nikolentzos
École Polytechnique &
AUEB

Stratis Limnios
École Polytechnique

Michalis Vazirgiannis
École Polytechnique &
AUEB

Abstract

In several domains, data objects can be decomposed into sets of simpler objects. It is then natural to represent each object as the set of its components or parts. Many conventional machine learning algorithms are unable to process this kind of representations, since sets may vary in cardinality and elements lack a meaningful ordering. In this paper, we present a new neural network architecture, called RepSet, that can handle examples that are represented as sets of vectors. The proposed model computes the correspondences between an input set and some hidden sets by solving a series of network flow problems. This representation is then fed to a standard neural network architecture to produce the output. The architecture allows end-to-end gradient-based learning. We demonstrate RepSet on classification tasks, including text categorization, and graph classification, and we show that the proposed neural network achieves performance better or comparable to state-of-the-art algorithms.

1 Introduction

In a variety of domains, complex data objects can be expressed as compositions of other, simpler objects. These simpler objects naturally correspond to the parts or components of the complex objects. For instance, in natural language processing, documents may be represented by sets of word embeddings. Likewise, in graph mining, a graph may be viewed as a set of vectors where these vectors correspond to the embeddings of its nodes. In computer vision, images

may be described by local features extracted from different regions of the image. In such scenarios, one set of feature vectors denotes a single instance of a particular class of interest (an object, document, graph, etc.). Performing machine learning tasks on such types of objects (e. g., set classification, set regression, etc.) is very challenging. While typical machine learning algorithms are designed for fixed dimensional data instances, the cardinalities of these sets are not fixed, but they are allowed to vary. Furthermore, the elements of the sets usually do not have an inherent ordering. Hence, machine learning algorithms have to be invariant to permutations of these elements.

Traditionally, the most common approach to the problem is to define a distance/similarity measure or kernel that finds a correspondence between each pair of sets, and to combine it with an instance-based machine learning algorithm such as k-nn or SVM. This approach has dominated the field, and has achieved state-of-the-art results on many datasets. However, its main disadvantage is that it is a two-step approach. Data representation and learning are independent from each other. Ideally, we would like to have an end-to-end approach. Besides the above problem, these methods usually suffer from high computational and memory complexity since they compare all sets to each other.

In recent years, neural network architectures have proven extremely successful on a wide variety of tasks, notably in computer vision, in natural language processing, and in graph mining (LeCun et al., 2015). One of the main reasons of the success of neural networks is that the representation of data is adapted to the task at hand. Specifically, recent neural network models are end-to-end trainable, and they generate features that are suitable for the task at hand. Most models that operate on sets usually update the representations of the elements of the set using some architectures, typically a stack of fully-connected layers. And then, they apply a permutation invariant function to the updated element representations to obtain a representation for the set. Although these networks have proven successful in several tasks, they usually employ very simple per-

mutation invariant functions such as the sum or the average of the representations of the elements. This may potentially limit the expressive power of these architectures.

In this paper, we propose a novel neural network architecture for performing machine learning tasks on sets of vectors, named RepSet. The network is capable of generating representations for unordered, variable-sized feature sets. Interestingly, the proposed model produces exactly the same output for all possible permutations of a set of vectors. To achieve that, it generates a number of hidden sets and it compares the input set with these sets using a network flow algorithm such as bipartite matching. The outputs of the network flow algorithm form the penultimate layer and are passed on to a fully-connected layer which produces the output. Since the objective functions of the employed network flow algorithms are differentiable, we can update these hidden sets during training with backpropagation. Hence, the proposed neural network is end-to-end trainable, while the hidden sets are different for each problem considered. Deeper models can be obtained by stacking more fully-connected layers one after another. When the size of the sets is large, solving the flow problems can become prohibitive. Hence, we also propose a relaxed formulation, ApproxRepSet, which is also permutation invariant and which scales to very large datasets. We demonstrate the proposed architecture in two classification tasks: text categorization from sets of word embeddings, and graph classification from bags of node embeddings. The results show that the proposed model produces better or competitive results with state-of-the-art techniques. Our main contributions are summarized as follows:

- We propose RepSet a novel architecture for performing machine learning on sets which, in contrast to traditional approaches, is capable of adapting data representation to the task at hand.

- We also propose ApproxRepSet, a simplified architecture which can be interpreted as an approximation of the proposed model, and which can handle very large datasets.

- We evaluate the proposed architecture on several benchmark datasets, and achieve state-of-the-art performance.

The rest of this paper is organized as follows. Section 2 provides an overview of the related work. Section 3 provides a description of the proposed neural network for sets. Section 4 evaluates the proposed architecture in two different tasks. Finally, Section 5 concludes.

2 Related Work

The past years witnessed a surge of interest in the area of neural networks for sets. These networks served mainly as the answer to computer vision problems such as the automated classification of point clouds. Although conceptually simple, the proposed architectures have achieved state-of-the-art results on many tasks. PointNet (Qi et al., 2017a) and DeepSets (Zaheer et al., 2017) transform the vectors of the sets (i. e., using several layers) into new representations. They then apply some permutation-invariant function to the emerging vectors to generate representations for the sets. PointNet uses max-pooling to aggregate information across vectors, while DeepSets adds up the representations of the vectors. The representation of the set is then passed on to a standard architecture (e. g., fully-connected layers, nonlinearities, etc). PointNet++ (Qi et al., 2017b) and SO-Net (Li et al., 2018) apply PointNet hierarchically in order to better capture local structures. Two other recent works employ neural networks to learn the parameters of the likelihood of each set (Rezatofighi et al., 2017, 2018). Vinyals et al. (2015) treat unordered sets as ordered sequences and apply RNN models to them. However, they show that the output of the network is highly dependent on the order of the elements of the set. More recently, Lee et al. (2019) proposed Set Transformer, a neural network that uses self-attention to model interactions among the elements of the input set.

Besides neural network architectures, several kernels between sets of vectors have been proposed in the past to enable kernel methods (e. g., SVMs) to handle unordered sets. Most of these kernels estimate a probability distribution on each set of vectors, and then derive their similarity using distribution-based comparison measures such as Fisher kernels (Jaakkola and Haussler, 1999), probability product kernels (Jebara et al., 2004; Lyu, 2005) and the classical Bhattacharyya similarity measure (Kondor and Jebara, 2003). Furthermore, there are also kernels that map the vectors of each set to multi-resolution histograms, and then in order to find an approximate correspondence between the two sets of vectors, they compare the histograms with a weighted histogram intersection measure (Grauman and Darrell, 2007b,a). Such kernels have been applied to different tasks such as to the problem of graph classification (Nikolentzos et al., 2017). Although very effective in several tasks, these kernel-based approaches suffer from high computational complexity. In most cases, the complexity of computing kernels between sets is quadratic in the number of their elements, while in classification problems, the complexity of optimizing the SVM classifier is quadratic in the number of training samples.
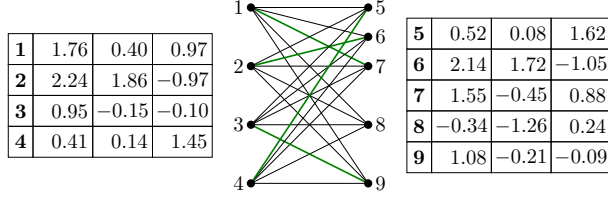
1	1.76	0.40	0.97
2	2.24	1.86	−0.97
3	0.95	−0.15	−0.10
4	0.41	0.14	1.45

5	0.52	0.08	1.62
6	2.14	1.72	−1.05
7	1.55	−0.45	0.88
8	−0.34	−1.26	0.24
9	1.08	−0.21	−0.09

Figure 1: Example of a bipartite graph generated from 2 sets of 3-dimensional vectors, and of its maximum matching M. Green color indicates an edge belongs to M. The weight of matching M is equal to 16.05.

There are also some very popular permutation-invariant metrics for comparing unordered sets of vectors such as the Earth Mover's distance which corresponds to the solution of an optimization problem that transforms one set to another. This distance was first introduced by Gaspard Monge in the context of transportation theory, and is often used in computer vision (Rubner et al., 2000) and in natural language processing (Kusner et al., 2015).

3   Neural Networks for Learning Representations of Sets

Conventional machine learning algorithms are designed to operate on fixed-size feature vectors, and they are thus unable to handle sets. In our setting, each example is represented as a collection $X = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$ of d-dimensional vectors, $\mathbf{v}_i \in \mathbb{R}^d$. Note that examples are allowed to vary in the number of elements. Hence, it is not necessary that all examples comprise of exactly n components. Since our input is a set $X = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$, $\mathbf{v}_i \in \mathbb{R}^d$, our input domain is the power set $\mathcal{X} = 2^{\mathbb{R}^d}$, and we would like to design an architecture whose output would be the same regardless of the ordering of the elements of X. Clearly, to achieve that, a permutation invariant function is necessary to be introduced at some layer of the architecture. In fact, the model that we propose consists simply of standard fully-connected layers along with a permutation invariant layer. We next present the proposed permutation invariant layer in detail.

Permutation invariant layer. In this paper, we propose a novel permutation invariant layer which capitalizes on well-established concepts from flows and matchings in graphs. The proposed layer contains m "hidden sets" $Y_1, Y_2, \ldots, Y_m$ of d-dimensional vectors. These sets may have different cardinalities and their components are trainable, i.e., the elements of a hidden set Y_i correspond to the columns of a trainable matrix $\mathbf{H}^{(i)}$. Therefore, each column of matrix $\mathbf{H}^{(i)}$ is a vector $\mathbf{u} \in Y_i$.

A natural way to measure the similarity between the input set and each one of the hidden sets is by comparing their building blocks, i.e., their elements. To achieve that, we capitalize on network flow algorithms. Specifically, we use the bipartite matching algorithm to compute a correspondence between the elements of the input set X and the elements of each hidden set Y_i. The bipartite matching problem is one of the most well-studied problems in combinatorial optimization. The input of the problem is a weighted bipartite graph $G = (V, E)$. The set of nodes V of a bipartite graph can be decomposed into two disjoint sets V_1 and V_2, i.e., $V = V_1 \cup V_2$. Every edge $e \in E$ connects a vertex in V_1 to one in V_2. A matching M is a subset of edges such that each node in V appears in at most one edge in M. The optimal solution to the problem can be interpreted as the similarity between the two node sets V_1 and V_2. A bipartite graph has a natural representation as a rectangular $|V_1| \times |V_2|$ matrix where the ij^{th} component is equal to the weight of the edge between the ith element of V_1 and the jth element of V_2 if that edge exists, otherwise equal to 0. Figure 1 illustrates a weighted bipartite graph along with the optimal matching M. The weight of each edge (not shown in the Figure) is equal to the inner product of the representations of its endpoints. Green edges belong to the optimal matching M. In our setting, the input set X corresponds to set V_1, the hidden set Y_i corresponds to set V_2, and the bipartite graph is complete, i.e., every element of V_1 is connected to all the elements of V_2. The weight of each edge is the result of a differentiable function f on the representations of the edge's two endpoints. Formally, given an input set of vectors, $X = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{|X|}\}$ and a hidden set $Y = \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{|Y|}\}$, we can obtain the maximum matching between the elements of the two sets by solving the following linear program:

$$\max \sum_{i=1}^{|X|} \sum_{j=1}^{|Y|} z_{ij} f(\mathbf{v}_i, \mathbf{u}_j)$$

subject to:

$$\sum_{i=1}^{|X|} z_{ij} \leq 1 \quad \forall j \in \{1, \ldots, |Y|\} \tag{1}$$

$$\sum_{j=1}^{|Y|} z_{ij} \leq 1 \quad \forall i \in \{1, \ldots, |X|\}$$

$$z_{ij} \geq 0 \quad \forall i \in \{1, \ldots, |X|\}, \forall j \in \{1, \ldots, |Y|\}$$

where $|X|, |Y|$ are the cardinalities of X and Y, $f(\mathbf{v}_i, \mathbf{u}_j)$ is, as mentioned above, a differentiable function, and $z_{ij} = 1$ if component i of X is assigned to component j of Y_i and 0 otherwise. In our experiments, we have defined $f(\mathbf{v}_i, \mathbf{u}_j)$ as the inner product between the two vectors $\mathbf{v}_i$, and $\mathbf{u}_j$ followed by
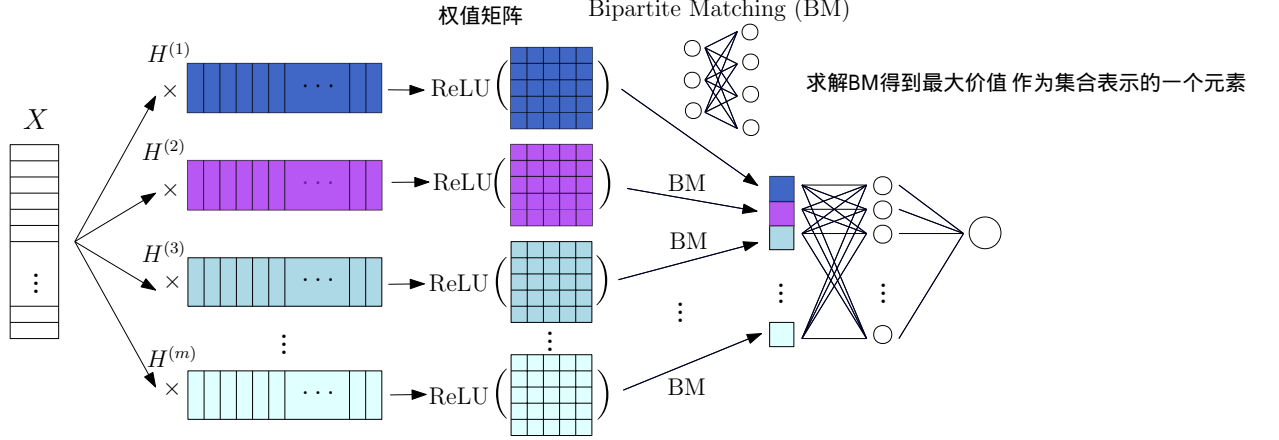
Bipartite Matching (BM)

Figure 2: Illustration of the proposed model for learning representations of sets. Each element of the input set is compared with the elements of all "hidden sets", and the emerging matrices serve as the input to the bipartite matching algorithm. The objective values of the matching problems correspond to the representation of the input set and are passed on to a standard neural network architecture.

the ReLU activation function. Hence, $f(\mathbf{v}_i, \mathbf{u}_j) = \text{ReLU}(\mathbf{v}_i^\top \mathbf{u}_j)$.

Given an input set X and the m hidden sets $Y_1, Y_2, \ldots, Y_m$, we formulate m different bipartite matching problems, and by solving all m of them, we end up with an m-dimensional vector $\mathbf{x}$ which corresponds to the hidden representation of set X. This m-dimensional vector can be used as features for different machine learning tasks such as set regression or set classification. For instance, in the case of a set classification problem with $|\mathcal{C}|$ classes, the output is computed as follows:

$$\mathbf{p} = \text{softmax}(\mathbf{W}\,\mathbf{x} + \mathbf{b})$$

where $\mathbf{W} \in \mathbb{R}^{|\mathcal{C}| \times m}$ is a matrix of trainable parameters and $\mathbf{b} \in \mathbb{R}^{|\mathcal{C}|}$ is the bias term. Given a training set consisting of sets $X_1, X_2, \ldots, X_N$, we use the negative log likelihood of the correct labels as training loss:

$$L = -\sum_{i=1}^{N}\sum_{j=1}^{|\mathcal{C}|} \mathbf{y}_j^i \log \mathbf{p}_j^i$$

where $\mathbf{y}_j^i$ is equal to 1 if X_i belongs to the jth class, and 0 otherwise. Note that we can create a deeper architecture by adding more fully-connected layers. An overview of the proposed architecture is illustrated in Figure 2.

As mentioned above, a constraint that the neural network is necessary to satisfy is to be invariant under any permutation of the elements of the input set. Our next theoretical result shows that the proposed model generates the same output for all $n!$ permutations of the elements of an input set X.

Theorem 1. *Let X be a set having elements from a countable or uncountable universe. The proposed architecture is invariant to the permutation of elements in X.*

Computing the derivative with respect to the hidden sets. The objective function of the proposed architecture is differentiable, and we can find a minimum by using classical stochastic optimization techniques that have proven very successful for training deep neural networks. We next use the chain rule and show how the gradients are computed. Note that here we assume a multiclass classification setting. The gradients for different settings (e.g., regression) are computed in a similar fashion.

Let $\mathbf{r} = \mathbf{W}\mathbf{x} + \mathbf{b}$. We use the chain rule in order to compute the derivative with respect to the weight matrix $\mathbf{H}^{(k)}$ of hidden set Y_k:

$$\frac{\partial L}{\partial \mathbf{H}^{(k)}} = \frac{\partial L}{\partial \mathbf{r}} \frac{\partial \mathbf{r}}{\partial \mathbf{x}_k} \frac{\partial \mathbf{x}_k}{\partial \mathbf{H}^{(k)}} \tag{2}$$

where $\mathbf{x}_k$ is the kth component of vector $\mathbf{x}$. The gradient of the loss function with respect to $\mathbf{r}$ is:

$$\frac{\partial L}{\partial \mathbf{r}} = \mathbf{p} - \mathbf{y} \tag{3}$$

We also have that:

$$\frac{\partial \mathbf{r}}{\partial \mathbf{x}_k} = \mathbf{W}_k \tag{4}$$

where $\mathbf{W}_k$ denotes the kth column of matrix $\mathbf{W}$. The kth component of vector $\mathbf{x}$ is equal to:

$$\mathbf{x}_k = \sum_{i=1}^{|X|}\sum_{j=1}^{|Y_k|} \mathbf{D}_{ij}^{(k)} f(\mathbf{v}_i, \mathbf{u}_j^{(k)}) = \text{Tr}(\mathbf{D}^{(k)\top} \mathbf{F}^{(k)})$$

where $\mathbf{D}^{(k)}$ is the matrix that contains the optimal values of the variables of problem (1) for Y_k, i.e., $\mathbf{D}_{ij}^{(k)} = z_{ij}^{(k)}$, $\mathbf{u}_j^{(k)}$ is the jth column of matrix $\mathbf{H}^{(k)}$, and $\mathbf{F}^{(k)} \in \mathbb{R}^{|X| \times |Y_k|}$ is a matrix such that $\mathbf{F}_{ij}^{(k)} = f(\mathbf{v}_i, \mathbf{u}_j)$.

Let $\mathbf{V}$ be a matrix whose rows correspond to the elements of set X. Then, it holds that $\mathbf{F}^{(k)} = \text{ReLU}(\mathbf{VH}^{(k)})$. This yields:

$$\frac{\partial \mathbf{x}_k}{\partial \mathbf{H}^{(k)}} = \frac{\partial}{\partial \mathbf{H}^{(k)}} \text{Tr}\left(\mathbf{D}^{(k)^\top} \text{ReLU}(\mathbf{VH}^{(k)})\right) = \mathbf{V}^\top \mathbf{D}^{(k)} \tag{5}$$

From Equations (2), (3), (4) and (5), we finally have:

$$\frac{\partial L}{\partial \mathbf{H}^{(k)}} = \frac{\partial L}{\partial \mathbf{r}} \frac{\partial \mathbf{r}}{\partial \mathbf{x}_k} \frac{\partial \mathbf{x}_k}{\partial \mathbf{H}^{(k)}} = (\mathbf{p} - \mathbf{y})^\top \mathbf{W}_k (\mathbf{V}^\top \mathbf{D}^{(k)})$$

Relaxed problem. The major weakness of the above architecture is its computational complexity. Computing a maximum cardinality matching in a weighted bipartite graph with n vertices and m edges takes time $\mathcal{O}(mn + n^2 \log n)$, using the classical Hungarian algorithm. This prohibits the proposed model from being applied to very large datasets. To account for that, we next present ApproxRepSet, an approximation of the bipartite matching problem which involves operations that can be performed on a GPU, allowing thus efficient implementations. More specifically, given an input set of vectors, $X = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{|X|}\}$ and a hidden set $Y = \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{|Y|}\}$, first we identify which of the two sets has the highest cardinality. If $|X| \geq |Y|$, we solve the following linear program:

$$\max \sum_{i=1}^{|X|} \sum_{j=1}^{|Y|} z_{ij} f(\mathbf{v}_i, \mathbf{u}_j)$$
subject to:
$$\sum_{i=1}^{|X|} z_{ij} \leq 1 \quad \forall j \in \{1, \ldots, |Y|\} \tag{6}$$
$$z_{ij} \geq 0 \quad \forall i \in \{1, \ldots, |X|\}, \forall j \in \{1, \ldots, |Y|\}$$

Conversely, if $|X| < |Y|$, then we replace the first constraint with the following: $\sum_{j=1}^{|Y|} z_{ij} \leq 1 \quad \forall i \in \{1, \ldots, |X|\}$. This problem is clearly a relaxed formulation of the problem defined in Equation (1) where a constraint has been removed.

Proposition 1. *The optimization problem defined in Equation 6 is an upper bound to the bipartite matching problem defined in Equation 1.*

4 Experimental Evaluation

We next evaluate the proposed model in text categorization and graph classification, and compare it against strong baselines. Further experimental results are provided in the supplementary material.

4.1 Text Categorization

We first evaluate RepSet and ApproxRepSet in the task of text categorization. Given a document, the input to the model is the set of embeddings of its terms.

Baselines. We compare the proposed architecture against Word Mover's Distance (WMD) (Kusner et al., 2015) and its supervised version (S-WMD) (Huang et al., 2016), against DeepSets (Zaheer et al., 2017) and three variants of it, where we replaced the sum operator with mean (NN-mean) and max (NN-max) operators, and with an attention mechanism (NN-attention), and against Set-Transformer (Lee et al., 2019).

Data and setup. We evaluate all approaches on 8 document classification datasets: (1) BBCSPORT, (2) TWITTER, (3) RECIPE, (4) OHSUMED, (5) CLASSIC, (6) REUTERS, (7) AMAZON, (8) 20NG. More details about the datasets are given in the supplementary material.

For the proposed models, we use a two-layer architecture: a permutation invariant layer followed by a fully-connected layer. Code is available at: `https://github.com/giannisnik/repset`. We choose the number of hidden sets from $\{20, 30, 50, 100\}$ and their cardinality from $\{10, 20, 50\}$ based on validation experiments. For DeepSets and its three variants, we use 3 fully-connected layers of sizes $\{300, 100, 30\}$ with tanh activations, followed by the aggregation operator, and then by 2 fully-connected layers of sizes $\{30, 10\}$ with tanh activations. For Set-Transformer, we use a stack of set attention blocks in encoder and a pooling by multihead attention module followed by a stack of set attention blocks in decoder. The dimensionality of all hidden layers is set to 64 and the number of attention heads to 4. For WMD and S-WMD, we provide the results reported in the original papers.

Results. Table 1 shows the average classification error of the proposed models and those of the baselines. On all datasets except two (OHSUMED, CLASSIC), the proposed model outperforms the baselines. In some cases, the gains in accuracy over the best performing competitors are considerable. For instance, on the 20NG, TWITTER, and RECIPE datasets, RepSet achieves respective absolute improvements of 3.82%, 2.08% and 0.63% in accuracy over the best competitor, the S-WMD method. Furthermore, the proposed model outperforms DeepSets and its variants, and Set-Transformer on all datasets, and on most of them by wide margins. Overall, it is clear from Table 1 that RepSet is superior to the other methods in text categorization. Regarding the two variants of the proposed architecture, the model that solves exactly the bipartite matching problem (RepSet) outperforms the

Table 1: Classification test error of the proposed architecture and the baselines on 8 text categorization datasets.

	BBCSPORT	TWITTER	RECIPE	OHSUMED	CLASSIC	REUTERS	AMAZON	20NG
WMD	4.60 ± 0.70	28.70 ± 0.60	42.60 ± 0.30	44.50	**2.88** $\pm$ 0.10	3.50	7.40 ± 0.30	26.80
S-WMD	2.10 ± 0.50	27.50 ± 0.50	39.20 ± 0.30	34.30	3.20 ± 0.20	3.20	5.80 ± 0.10	26.80
DeepSets	25.45 ± 20.1	29.66 ± 1.62	70.25 ± 0.00	71.53	5.95 ± 1.50	10.00	8.58 ± 0.67	38.88
NN-mean	10.09 ± 2.62	31.56 ± 1.53	64.30 ± 7.30	45.37	5.35 ± 0.75	11.37	13.66 ± 3.16	38.40
NN-max	2.18 ± 1.75	30.27 ± 1.26	43.47 ± 1.05	35.88	4.21 ± 0.11	4.33	7.55 ± 0.63	32.15
NN-attention	4.72 ± 0.97	29.09 ± 0.62	43.18 ± 1.22	**31.36**	4.42 ± 0.73	3.97	6.92 ± 0.51	28.73
Set-Transformer	4.18 ± 1.23	27.79 ± 0.47	42.54 ± 1.35	35.68	5.23 ± 0.52	4.52	7.18 ± 0.44	30.01
RepSet	**2.00** $\pm$ 0.89	**25.42** $\pm$ 1.10	**38.57** $\pm$ 0.83	33.88	3.38 ± 0.50	3.15	**5.29** $\pm$ 0.28	**22.98**
ApproxRepSet	4.27 ± 1.73	27.40 ± 1.95	40.94 ± 0.40	35.94	3.76 ± 0.45	**2.83**	5.69 ± 0.40	23.82

Table 2: Terms of the employed pre-trained model that are most similar to the elements and centroids of 5 hidden sets.

Hidden set	Terms similar to elements of hidden sets	Terms similar to centroids of hidden sets
1	chelsea, football, striker, club, champions	footballing
2	qualify, madrid, arsenal, striker, united, france	ARSENAL_Wenger
3	olympic, athlete, olympics, sport, pentathlon	Olympic_Medalist
4	penalty, cup, rugby, coach, goal	rugby
5	match, playing, batsman, batting, striker	batsman

model that approximates it (ApproxRepSet) on all datasets except from REUTERS. However, on most datasets the difference in performance is not large. Hence, although less powerful, ApproxRepSet is still capable of learning expressive representations of sets.

We should mention at this point that besides effective, the proposed model is also highly interpretable. For instance, in the text categorization setting, the elements of each hidden set can be regarded as the terms of hidden documents which are likely to be related to the topics of the different classes. To experimentally verify that, we trained a model with 50 hidden sets on the BBCSPORT dataset. Each hidden set consisted of 20 vectors. For 5 hidden sets, we found the terms that are closer to their elements and we also computed their centroids. Table 2 shows the terms of the pre-trained model that were found to be most similar (using cosine similarity), with respect to the vectors and centroids. Clearly, the centroids of these 5 hidden sets are close to terms that are related to sports. Interestingly, some of these terms correspond to cricket positions, while others to names of famous soccer teams.

4.2 Graph Classification

We also evaluate the proposed architecture in the graph classification task. We represent each graph as a set of vectors (i.e., the embeddings of its nodes), and pass them on to the proposed models.

Baselines. We compare the proposed models against

several recent state-of-the-art approaches: (1) PSCN, a model that extracts neighborhood subgraphs of specific size, defines an ordering on the nodes of these subgraphs and feeds the emerging adjacency matrices into a convolutional neural network (Niepert et al., 2016), (2) Deep GR, an approach that improves the graphlet kernel by using the Skip-gram model (Yanardag and Vishwanathan, 2015), (3) EMD, a method that represents each graph as a set of vectors and computes the distance between each pair of graphs using the Earth Mover's Distance algorithm (Nikolentzos et al., 2017), (4) DGCNN, a model that applies a message passing architecture followed by a pooling operator based on sorting to create a fixed-sized graph representation which is then passed on to a convolutional architecture (Zhang et al., 2018a), (5) SAEN, an algorithm that decomposes graphs in a hierarchical fashion, then uses shift, aggregate and extract operations, and finally applies a standard neural network to the emerging representations (Orsini et al., 2018), (6) RetGK, a graph kernel that capitalizes on the isomorphism-invariance property of the return probabilities of random walks (Zhang et al., 2018b), and (7) DiffPool, a message passing architecture which applies at each layer a differentiable graph pooling module that clusters the nodes of the previous layer (Ying et al., 2018). We also compare the proposed models against DeepSets (Zaheer et al., 2017) and the three variants of it that were presented above (NN-mean, NN-max, and NN-attention), and against Set-Transformer (Lee et al., 2019).

Table 3: Classification accuracy (± standard deviation) of the proposed architecture and the baselines on the 5 graph classification datasets.

	MUTAG	PROTEINS	IMDB BINARY	IMDB MULTI	REDDIT BINARY
PSCN $k = 10$	88.95 (± 4.37)	75.00 (± 2.51)	71.00 (± 2.29)	45.23 (± 2.84)	86.30 (± 1.58)
Deep GR	82.66 (± 1.45)	71.68 (± 0.50)	66.96 (± 0.56)	44.55 (± 0.52)	78.04 (± 0.39)
EMD	86.11 (± 0.84)	-	-	-	-
DGCNN	85.80 (± 1.70)	75.50 (± 0.90)	70.03 (± 0.86)	47.83 (± 0.85)	-
SAEN	84.99 (± 1.82)	75.31 (± 0.70)	71.59 (± 1.20)	48.53 (± 0.76)	87.22 (± 0.80)
RetGK	**90.30** (± 1.10)	76.20 (± 0.50)	72.30 (± 0.60)	48.70 (± 0.60)	**92.60** (± 0.30)
DiffPool	-	**76.25**	-	-	-
DeepSets	86.26 (± 1.09)	60.82 (± 0.79)	69.84 (± 0.64)	47.62 (± 1.18)	52.01 (± 1.47)
NN-mean	87.55 (± 0.98)	73.00 (± 1.21)	71.48 (± 0.48)	49.92 (± 0.82)	84.57 (± 0.84)
NN-max	85.84 (± 0.99)	71.05 (± 0.54)	69.56 (± 0.91)	48.28 (± 0.43)	80.98 (± 0.79)
NN-attention	85.92 (± 1.16)	74.48 (± 0.22)	**72.40** (± 0.45)	49.56 (± 0.47)	88.74 (± 0.53)
Set-Transformer	87.71 (± 1.14)	59.62 (± 1.42)	71.21 (± 1.28)	**50.25** (± 0.74)	83.79 (±) 0.83
RepSet	88.63 (± 0.86)	73.04 (± 0.42)	**72.40** (± 0.73)	49.93 (± 0.60)	87.45 (± 0.86)
ApproxRepSet	86.33 (± 1.48)	70.74 (± 0.85)	71.46 (± 0.91)	48.92 (± 0.28)	80.30 (± 0.56)

Data and setup. We evaluate the competing methods on standard graph classification datasets derived from bioinformatics (MUTAG, PROTEINS) and social networks (IMDB-BINARY, IMDB-MULTI, REDDIT-BINARY) (Kersting et al., 2016). Note that the social network graphs are unlabeled, while the bioinformatics graphs contain node labels. However, in our experiments, we did not take these node labels into account. More details about the datasets are given in the supplementary material. Since the datasets do not come with standard training/test splits, we performed 10-fold cross validation procedure where we randomly sampled 10% of each training fold to serve as a validation set. Furthermore, since some datasets are small, we repeated the whole experiment 10 times.

We generated embeddings for the nodes as follows: we create a single graph from each dataset by computing the disjoint union of all the graphs contained in it. In other words, for each dataset, we generate a disconnected graph whose components correspond to the graphs contained in the dataset. We then employ struc2vec, an algorithm that learns structural node representations (Ribeiro et al., 2017). Nodes with structurally similar neighborhoods are close to each other in the embeddings space. We set the dimensionality of the learned embeddings to 20. For RepSet, ApproxRepSet, DeepSets and its variants, and Set-Transformer, we use the same configuration as in the case of text categorization. Note that all the above models are trained on the same input data (i. e., sets of node embeddings). For the remaining baseline methods, we provide the results reported in the original papers.

Results. Table 3 shows the average classification accuracy of the proposed models and those of the baselines. We observe that the proposed models are on par with the state-of-the-art algorithms. Specifically, RepSet obtains the highest average performance among all methods on the IMDB-BINARY dataset, while it is the second best method on IMDB-MULTI, and the third best method on REDDIT-BINARY and on MUTAG. DeepSets and its variants achieve comparable accuracies to the proposed models. Specifically, the best-performing variant, NN-attention, outperforms the proposed models on PROTEINS and REDDIT-BINARY. Set-Transformer, on the other hand, outperforms the proposed models only on IMDB-MULTI. Interestingly, even though the embeddings which the proposed model utilizes do not incorporate information about the node labels, on the MUTAG dataset (whose graphs contain node labels), it remains competitive with the baselines which take these node labels into account. On the other hand, on the second dataset that contains node labels (PROTEINS), our proposed models, RepSet and ApproxRepSet, are outperformed by all the baselines except DeepSets, NN-max and Set-Transformer. Last, our approximate model, ApproxRepSet, achieves comparable to state-of-the-art results in almost all datasets, while being considerably faster than RepSet.

4.3 Runtime Analysis

To evaluate the runtime performance and scalability of the proposed models, we created a series of synthetic datasets and measured how the average running time per epoch varies with respect to the parameters of the model. Figure 3 illustrates the running time of RepSet and ApproxRepSet as a function of the size of the hidden sets $|Y_i|, i = 1, \ldots, m$ (top left) and as a function of the number of hidden sets m (bottom left). Note that for RepSet, training was performed on an Intel Xeon $E5-1607$ CPU (4 threads), while for ApproxRepSet, it
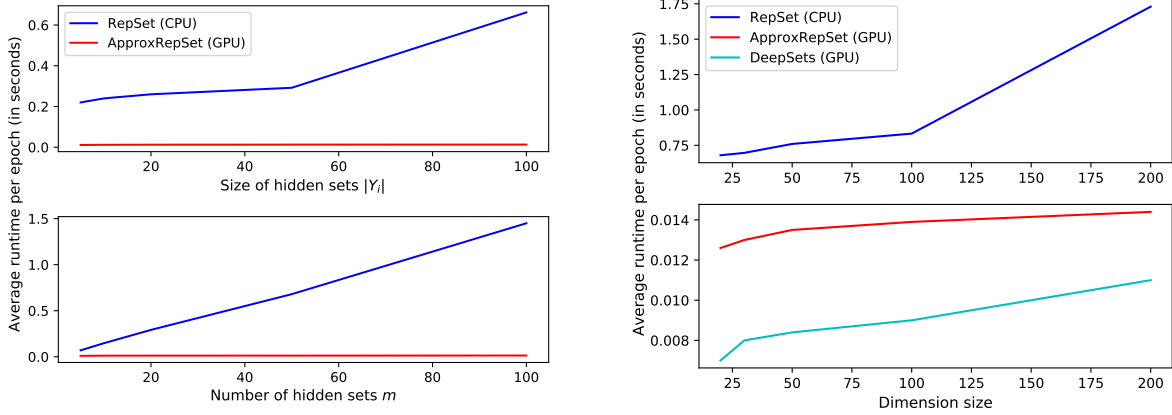
Figure 3: Runtimes with respect to the number of hidden sets m, the size of the hidden sets $|Y_i|$ (left) and embeddings with different dimensions (right).
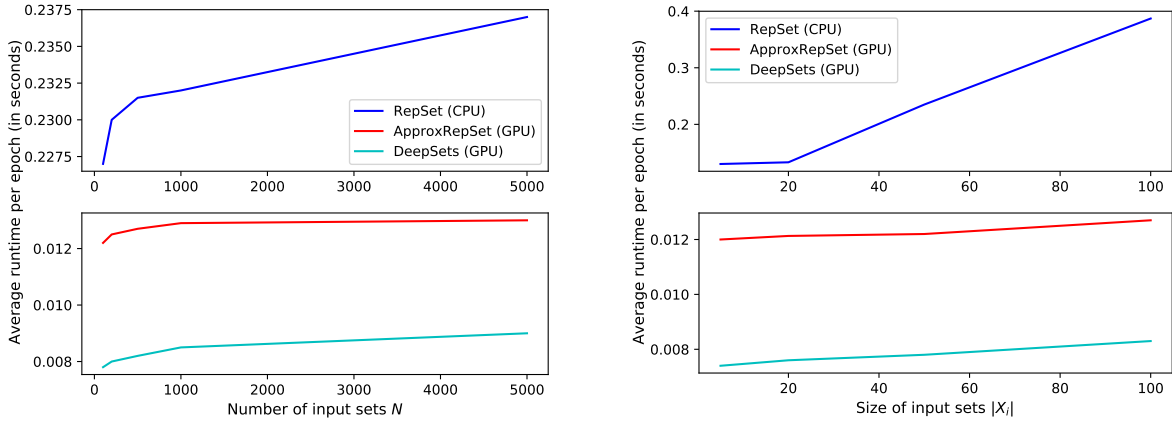

Figure 4: Runtimes with respect to the number of input sets N (left) and the size of the input sets $|X_i|$ (right).

was performed on an NVIDIA Titan Xp GPU. As expected, we observe that RepSet is more computationally expensive than ApproxRepSet, while its running time increases significantly as the size and the number of hidden sets increase. On the other hand, the values of these parameters do not have a large impact on the running time of ApproxRepSet. We also evaluate (Figure 3 (right)) how the proposed models scale as the dimensionality of the vectors contained in the sets increases and compare them against the DeepSets model. DeepSets is the fastest model, followed by ApproxRepSet. The running times of both these models are much smaller than that of RepSet, while they also grow very slowly as the dimensionality of vectors increases. Conversely, the running time of RepSet increases notably especially for dimensionalities larger than 100. We also examine in Figure 4 how the running time of the three models varies with respect to the number of input samples N (left) and to their cardinality $|X_i|, i = 1, \ldots, N$ (right). Surprisingly, we find that the running time of RepSet grows slowly as the number of samples increases. On the other hand, it in-

creases significantly as the cardinality of these samples increases. The running times of DeepSets and ApproxRepSet are again much lower than that of RepSet, and grow only slightly as the number of samples and their cardinality increase.

5 Conclusion

In this paper, we proposed RepSet, a neural network for learning set representations. RepSet computes the correspondences between the input sets and some hidden sets by solving a series of matching/network flow problems. We also introduced a relaxed version which involves fast matrix operations and scales to large datasets. Experiments in two tasks show that our architecture is competitive with the state-of-the-art.

Acknowledgments

References

Borgwardt, K. M., Ong, C. S., Schönauer, S., Vishwanathan, S., Smola, A. J., and Kriegel, H.-P. (2005). Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl 1):i47–i56.

Cachopo, A. M. d. J. C. (2007). Improving methods for single-label text categorization. *Instituto Superior Técnico, Portugal*.

Debnath, A. K., Lopez de Compadre, R. L., Debnath, G., Shusterman, A. J., and Hansch, C. (1991). Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of Medicinal Chemistry*, 34(2):786–797.

Grauman, K. and Darrell, T. (2007a). Approximate correspondences in high dimensions. In *Advances in Neural Information Processing Systems*, pages 505–512.

Grauman, K. and Darrell, T. (2007b). The pyramid match kernel: Efficient learning with sets of features. *Journal of Machine Learning Research*, 8(Apr):725–760.

Huang, G., Guo, C., Kusner, M. J., Sun, Y., Sha, F., and Weinberger, K. Q. (2016). Supervised word mover's distance. In *Advances in Neural Information Processing Systems*, pages 4862–4870.

Jaakkola, T. and Haussler, D. (1999). Exploiting generative models in discriminative classifiers. In *Advances in Neural Information Processing Systems*, pages 487–493.

Jebara, T., Kondor, R., and Howard, A. (2004). Probability product kernels. *Journal of Machine Learning Research*, 5(Jul):819–844.

Kersting, K., Kriege, N. M., Morris, C., Mutzel, P., and Neumann, M. (2016). Benchmark data sets for graph kernels.

Kondor, R. and Jebara, T. (2003). A kernel between sets of vectors. In *Proceedings of the 20th International Conference on Machine Learning*, pages 361–368.

Kusner, M., Sun, Y., Kolkin, N., and Weinberger, K. (2015). From word embeddings to document distances. In *International Conference on Machine Learning*, pages 957–966.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436.

Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., and Teh, Y. W. (2019). Set transformer: A framework for attention-based permutation-invariant neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, pages 3744–3753.

Li, J., Chen, B. M., and Hee Lee, G. (2018). So-net: Self-organizing network for point cloud analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9397–9406.

Lyu, S. (2005). A kernel between unordered sets of data: The gaussian mixture approach. In *European Conference on Machine Learning*, pages 255–267. Springer.

Niepert, M., Ahmed, M., and Kutzkov, K. (2016). Learning convolutional neural networks for graphs. In *International conference on machine learning*, pages 2014–2023.

Nikolentzos, G., Meladianos, P., and Vazirgiannis, M. (2017). Matching node embeddings for graph similarity. In *AAAI*, pages 2429–2435.

Orsini, F., Baracchi, D., and Frasconi, P. (2018). Shift Aggregate Extract Networks. *Frontiers in Robotics and AI*, 5:42.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017a). Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660.

Qi, C. R., Yi, L., Su, H., and Guibas, L. J. (2017b). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5099–5108.

Rezatofighi, S. H., Milan, A., Abbasnejad, E., Dick, A., Reid, I., et al. (2017). Deepsetnet: Predicting sets with deep neural networks. In *Proceedings of the 2017 IEEE International Conference on Computer Vision*, pages 5257–5266.

Rezatofighi, S. H., Milan, A., Shi, Q., Dick, A., and Reid, I. (2018). Joint learning of set cardinality and state distribution. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*.

Ribeiro, L. F., Saverese, P. H., and Figueiredo, D. R. (2017). struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 385–394.

Rubner, Y., Tomasi, C., and Guibas, L. J. (2000). The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99–121.

Salton, G. and Buckley, C. (1971). The smart information retrieval system.

Vinyals, O., Bengio, S., and Kudlur, M. (2015). Order matters: Sequence to sequence for sets. In *International Conference on Learning Representations*.

Yanardag, P. and Vishwanathan, S. (2015). Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1365–1374. ACM.

Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., and Leskovec, J. (2018). Hierarchical Graph Representation Learning with Differentiable Pooling. In *Advances in Neural Information Processing Systems*, pages 4801–4811.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. (2017). Deep sets. In *Advances in Neural Information Processing Systems*, pages 3391–3401.

Zhang, M., Cui, Z., Neumann, M., and Chen, Y. (2018a). An End-to-End Deep Learning Architecture for Graph Classification. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 4438–4445.

Zhang, Z., Wang, M., Xiang, Y., Huang, Y., and Nehorai, A. (2018b). RetGK: Graph Kernels based on Return Probabilities of Random Walks. In *Advances in Neural Information Processing Systems*, pages 3964–3974.

Supplementary Material

A Overview

This document is supplementary material for the paper "Rep the Set: Neural Networks for Learning Set Representations". It is organized as follows. We will prove in Section B the Theorem 1. In Section C, we will present the proof of Proposition 1. In Section D, we will give details about the datasets we used in our experiments. Finally, in Section E, we perform a sensitivity analysis, and we present the features that the model learns on a simple synthetic dataset.

B Proof of Theorem 1

For the reader's convenience we will restate Theorem 1 from the article.

Theorem 2. *Let X be a set having elements from a countable or uncountable universe. The proposed architecture is invariant to the permutation of elements in X.*

Proof. Let Π_n be the set of all permutations of the integers from 1 to n. Let $\pi \in \Pi_{|X|}$ be an arbitrary permutation. We will apply π to the input set X. The bipartite matching problem then becomes:

$$\max \sum_{i=1}^{|X|} \sum_{j=1}^{|Y|} z_{ij} f(\mathbf{v}_{\pi(i)}, \mathbf{u}_j)$$

subject to:

$$\sum_{i=1}^{|X|} z_{ij} \leq 1 \quad \forall j \in \{1, \ldots, |Y|\} \qquad (7)$$

$$\sum_{j=1}^{|Y|} z_{ij} \leq 1 \quad \forall i \in \{1, \ldots, |X|\}$$

$$z_{ij} \geq 0 \quad \forall i \in \{1, \ldots, |X|\}, \forall j \in \{1, \ldots, |Y|\}$$

The constraints of the optimization problem remain intact since summing the elements of a set is a permutation invariant function. Moreover, it holds that:

$$\sum_{i=1}^{|X|} \sum_{j=1}^{|Y|} z_{ij} f(\mathbf{v}_{\pi(i)}, \mathbf{u}_j) = \sum_{i=1}^{|X|} \sum_{j=1}^{|Y|} z_{\pi^{-1}(i)j} f(\mathbf{v}_i, \mathbf{u}_j) \quad (8)$$

The sets of variables that lead to the optimal solutions of problems (1) in the main paper and (7) above are identical, i.e., $z^*_{\pi^{-1}(i)j} = z^*_{ij}$, $\forall i \in \{1, \ldots, |X|\}, \forall j \in \{1, \ldots, |Y|\}$. Hence, the optimal value of the bipartite matching problem is the same for all $n!$ permutations of the input, and therefore, the proposed model maps all permutations of the input into the same representation. $\square$

C Proof of Proposition 1

For the reader's convenience we will restate Proposition 1 from the article.

Proposition 2. *The optimization problem defined in Equation 6 is an upper bound to the bipartite matching problem defined in Equation 1.*

Proof. We assume without loss of generality that $|X| \geq |Y|$. Let $\mathbf{D}^*$ be the optimal solution to the relaxed problem, i.e., $\mathbf{D}^*_{ij} = z_{ij}$. Therefore, it holds that:

$$\mathbf{D}^*_{ij} = \begin{cases} 1 & \text{if } i = \arg\max_k f(\mathbf{v}_k, \mathbf{u}_j) \wedge \max_k f(\mathbf{v}_k, \mathbf{u}_j) > 0 \\ 0 & \text{otherwise} \end{cases}$$

The relaxed problem is allowed to match multiple elements of Y with the same element of X. Specifically, the optimal solution of the relaxed problem matches an element of Y with an element of X if their inner product is positive and is the highest among the inner products between that element of Y and all the elements of X. Then, for every j, let $i^* = \arg\max_k f(\mathbf{v}_k, \mathbf{u}_j)$. For any feasible solution $\mathbf{D}$ of the exact problem, and for any j, we have:

$$\begin{aligned} \sum_{i=1}^{|X|} \mathbf{D}_{ij} f(\mathbf{v}_i, \mathbf{u}_j) &\leq \sum_{i=1}^{|X|} \mathbf{D}_{ij} f(\mathbf{v}_{i^*}, \mathbf{u}_j) \\ &= f(\mathbf{v}_{i^*}, \mathbf{u}_j) \sum_{i=1}^{|X|} \mathbf{D}_{ij} \\ &\leq f(\mathbf{v}_{i^*}, \mathbf{u}_j) \\ &= \sum_{i=1}^{|X|} \mathbf{D}^*_{ij} f(\mathbf{v}_i, \mathbf{u}_j) \end{aligned} \quad (9)$$

Therefore, the objective value of the relaxed problem (obtained by $\mathbf{D}^*$) gives an upper to the exact problem. $\square$

D Datasets

D.1 Text Categorization Datasets

We evaluated all approaches on 8 supervised document datasets: (1) BBCSPORT: BBC sports articles between 2004-2005, (2) TWITTER: a set of tweets labeled with sentiments positive, negative, or neutral (the set is reduced due to the unavailability of some tweets), (3) RECIPE: a set of recipe procedure descriptions labeled by their region of origin, (4) OHSUMED: a collection of medical abstracts categorized by different cardiovascular disease groups (for computational efficiency we subsample the dataset, using the first 10 classes), (5) CLASSIC: sets of sentences from academic

Table 4: Datasets used in text categorization experiments.

Dataset	n	Voc	Unique Words(avg)	y
BBCSPORT	517	13243	117	5
TWITTER	2176	6344	9.9	3
RECIPE	3059	5708	48.5	15
OHSUMED	3999	31789	59.2	10
CLASSIC	4965	24277	38.6	4
REUTERS	5485	22425	37.1	8
AMAZON	5600	42063	45.0	4
20NG	11293	29671	72	20

Table 5: Datasets used in graph classification.

Dataset	#Graphs	y	Nodes(avg)	Edges(avg)
MUTAG	188	2	17.93	19.79
PROTEINS	1113	2	39.06	72.82
IMDB BINARY	1000	2	19.77	96.53
IMDB MULTI	1500	3	13.00	65.94
REDDIT BINARY	2000	2	429.63	497.75

papers, labeled by publisher name, (6) REUTERS: a classic news dataset labeled by news topics (we use the 8-class version with train/test split as described in Cachopo (2007), (7) AMAZON: a set of Amazon reviews which are labeled by category product in books, dvd, electronics, kitchen (as opposed to by sentiment), (8) 20NG: news articles classified into 20 different categories (we use the bydate train/test split by Cachopo (2007)). We preprocess all datasets by removing all words in the SMART stop word list (Salton and Buckley, 1971). Table 4 shows statistics of the 8 datasets that were used for the evaluation. We obtained a distributed representation for each word from a publicly available set of pre-trained vectors[1]. For datasets that do not come with a predefined train/test split, we report the average accuracy over five 70/30 train/test splits as well as the standard deviation.

D.2 Graph Classification Datasets

We evaluated the proposed architecture on the following 5 datasets: (1) MUTAG, (2) PROTEINS, (3) IMDB-BINARY, (4) IMDB-MULTI, and (5) REDDIT-BINARY.

MUTAG contains mutagenic aromatic and heteroaromatic nitro compounds. Each chemical compound is labeled according to whether or not it has mutagenic effect on the Gram-negative bacterium Salmonella typhimurium (Debnath et al., 1991). PROTEINS consists of proteins represented as graphs where vertices are secondary structure elements and there is an edge between two vertices if they are neighbors in the amino-acid sequence or in 3D space. The task is to classify proteins into enzymes and non-enzymes (Borgwardt et al., 2005). IMDB-BINARY and IMDB-MULTI contain movie collaboration graphs. The vertices of each graph represent actors/actresses and two vertices are connected by an edge if the corresponding actors/actresses appear in the same movie. Each

---

[1] https://code.google.com/archive/p/word2vec/

graph is the ego-network of an actor/actress, and the task is to predict which genre an ego-network belongs to (Yanardag and Vishwanathan, 2015). REDDIT-BINARY consists of online discussion threads represented as graphs. Each vertex corresponds to a user, and two users are connected by an edge if one of them responded to at least one of the other's comments. The task is to classify graphs into either communities (Yanardag and Vishwanathan, 2015). A summary of the datasets is given in Table 5.

E Experimental Evaluation

E.1 Sensitivity Analysis

The proposed RepSet and ApproxRepSet models involve two main parameters: (1) the number of hidden sets m, and (2) the cardinalities of the hidden sets $|Y_i|, i = 1, \ldots, m$. We next investigate how these two parameters influence the performance of the RepSet model. Specifically, in Figures 5 and 6, we examine how the different choices of these parameters affect the performance of RepSet on the TWITTER and RECIPE datasets, respectively. We measure the test error as a function of the two parameters. Note that each hidden set Y_i can have a different cardinality compared to the other sets. However, we set the cardinalities of all hidden sets to the same value. We observe that on TWITTER, the number of hidden sets m does not have a large impact on the performance, especially for small cardinalities of the hidden sets ($|Y_i| \leq 50$). For most cardinalities, the test error is within 1% to 3% when varying this parameter.

Furthermore, in most cases, the best performance is attained when the number of hidden sets is small ($m \leq 20$). Similar behavior is also observed for the second parameter on the TWITTER dataset. For most values of m, the test error changes only slightly when varying the cardinalities of the hidden sets. For $m \geq 50$, the model produces best results when the cardinalities of the hidden sets $|Y_i|$ are close to 20. On the other hand, for small values of m, the model yields good performance even when the cardinalities of the hidden sets $|Y_i|$ are large. On the RECIPE dataset, both parameters have a higher impact on the performance of the RepSet model. In general, small values
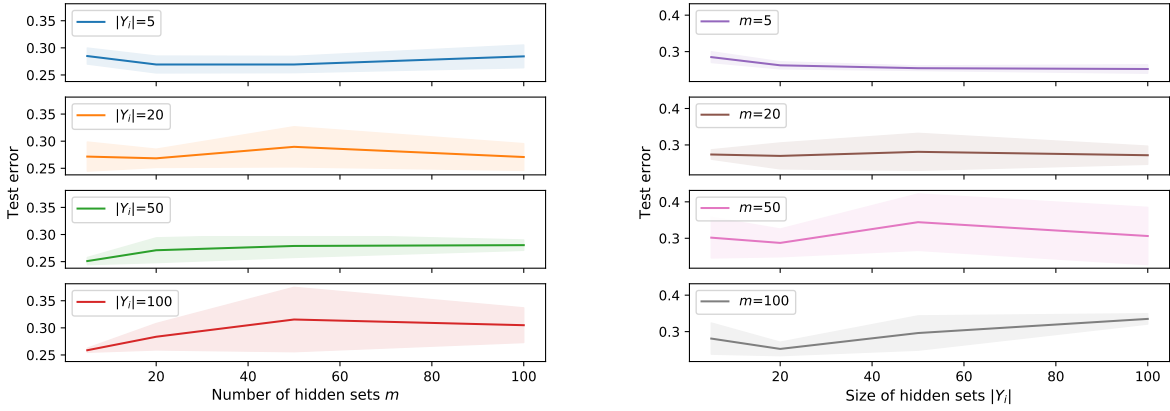
Figure 5: Average test error of the RepSet model with respect to the number of hidden sets m (left) and the size of the hidden sets $|Y_i|$ (right) on the TWITTER dataset.
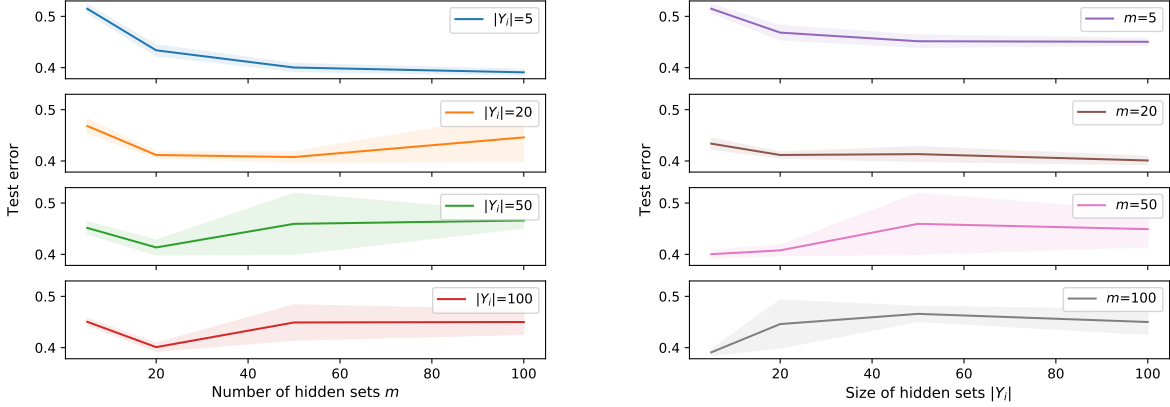

Figure 6: Average test error of the RepSet model with respect to the number of hidden sets m (left) and the size of the hidden sets $|Y_i|$ (right) on the RECIPE dataset.

of m lead to higher test error than larger values of m. For most values of $|Y_i|$, values of m between 20 and 50 result in the lowest test error. As regards the size of the hidden sets $|Y_i|$, there is no consistency in the obtained results. Specifically, for small values of m ($m \leq 20$), large cardinalities of the hidden sets result in better performace, while for large values of m ($m \geq 50$), small cardinalities lead to smaller error.

E.2 Synthetic Data

We first demonstrate the proposed RepSet architecture on a very simple dataset. The dataset consists of 4 sets of 2-dimensional vectors. The cardinality of all sets is equal to 2. The elements of the 4 sets are illustrated in Figure 7. Although seemingly simple, this dataset may prove challenging for several algorithms that apply aggregation mechanisms to the elements of

the sets since all 4 sets have identical centroids while the sum of their elements is also the same for all of them. To learn to classify these sets, we used an instance of the proposed model consisting of 2 hidden sets of cardinality equal to 2. The model managed easily to discriminate between the 4 sets and to achieve perfect accuracy. A question that arises at this point is what kind of features the hidden sets of the model learn during training. Hence, besides the input sets, Figure 7 also shows the vectors of the 2 hidden sets. The hidden sets learned very similar patterns, which indicates that less than 2 hidden sets may be required. In fact, we observed that even with one hidden set, the model can achieve perfect performance.
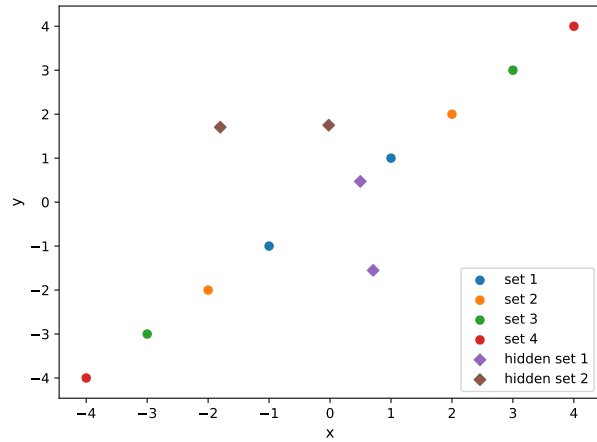
Figure 7: A very simple dataset consisting of 4 examples (i.e., sets). Each set contains a pair of 2-dimensional vectors (circles). The diamonds indicate the "hidden sets" that the proposed model learned during training.