Learning Aggregation Functions

Giovanni Pellegrini[1*] , **Alessandro Tibo**[2*] ,
Paolo Frasconi[3] , **Andrea Passerini**[1,2] and **Manfred Jaeger**[2]

[1]DISI, University of Trento
[2]Computer Science Department, Aalborg University
[3]DINFO, Università di Firenze
{giovanni.pellegrini, andrea.passerini}@unitn.it, {alessandro, jaeger}@cs.aau.dk,
paolo.frasconi@pm.me

Abstract

Learning on sets is increasingly gaining attention in the machine learning community, due to its widespread applicability. Typically, representations over sets are computed by using fixed aggregation functions such as sum or maximum. However, recent results showed that universal function representation by sum- (or max-) decomposition requires either highly discontinuous (and thus poorly learnable) mappings, or a latent dimension equal to the maximum number of elements in the set. To mitigate this problem, we introduce a learnable aggregation function (LAF) for sets of arbitrary cardinality. LAF can approximate several extensively used aggregators (such as average, sum, maximum) as well as more complex functions (e.g., variance and skewness). We report experiments on semi-synthetic and real data showing that LAF outperforms state-of-the-art sum- (max-) decomposition architectures such as DeepSets and library-based architectures like Principal Neighborhood Aggregation, and can be effectively combined with attention-based architectures.

1 Introduction

The need to aggregate representations is ubiquitous in deep learning. Some recent examples include max-over-time pooling used in convolutional networks for sequence classification [Kim, 2014], average pooling of neighbors in graph convolutional networks [Kipf and Welling, 2017], max-pooling in Deep Sets [Zaheer et al., 2017], in (generalized) multi-instance learning [Tibo et al., 2020] and in Graph-SAGE [Hamilton et al., 2017]. In all the above cases (with the exception of LSTM-pooling in GraphSAGE) the aggregation function is predefined, i.e., not tunable, which may be in general a disadvantage [Ilse et al., 2018]. Sum-based aggregation has been advocated based on theoretical findings showing the permutation invariant functions can be sum-decomposed [Zaheer et al., 2017; Xu et al., 2019]. However, recent results [Wagstaff et al., 2019] showed that this universal function representation guarantee requires either highly

discontinuous (and thus poorly learnable) mappings, or a latent dimension equal to the maximum number of elements in the set. This suggests that learning set functions that are accurate on sets of large cardinality is difficult.

Inspired by previous work on learning uninorms [Melnikov and Hüllermeier, 2016], we propose a new parametric family of aggregation functions that we call LAF, for *learnable aggregation functions*. A single LAF unit can approximate standard aggregators like sum, max or mean as well as model intermediate behaviours (possibly different in different areas of the space). In addition, LAF layers with multiple aggregation units can approximate higher order moments of distributions like variance, skewness or kurtosis. In contrast, other authors [Corso et al., 2020] suggest to employ a predefined library of elementary aggregators to be combined. Since LAF can represent sums, it can be seen as a smooth version of the class of functions that are shown in [Zaheer et al., 2017] to enjoy universality results in representing set functions. The hope is that being smoother, LAF is more easily learnable. Our empirical findings show that this can be actually the case, especially when asking the model to generalize over large sets. In particular, we find that:

- LAF layers can learn a wide range of aggregators (including higher-order moments) on sets of scalars without background knowledge on the nature of the aggregation task.

- LAF layers on the top of traditional layers can learn the same wide range of aggregators on sets of high dimensional vectors (MNIST images).

- LAF outperforms state-of-the-art set learning methods such as DeepSets and PNA on real-world problems involving point clouds and text concept set retrieval.

- LAF performs comparably to PNA on random graph generation tasks, outperforming several graph neural networks architectures including GAT [Veličković et al., 2018] and GIN [Xu et al., 2019].

2 The LAF Framework

We use $\boldsymbol{x} = \{x_1, \ldots, x_N\}$ to denote finite multisets of real numbers $x_i \in \mathbb{R}$. Note that directly taking $\boldsymbol{x}$ to be a multiset, not a vector, means that there is no need to define properties like exchangeability or permutation equivariance for operations on $\boldsymbol{x}$. An aggregation function *agg* is any function that

---

[*]Equal Contribution. Contact Authors.

NAME	DEFINITION	a	b	c	d	e	f	g	h	α	β	γ	δ	LIMITS
CONSTANT	$\kappa \in \mathbb{R}$	0	1	-	-	0	1	-	-	κ	0	1	0	
MAX	$\max_i x_i$	$1/r$	r	-	-	0	1	-	-	1	0	1	0	$r \to \infty$
MIN	$\min_i x_i$	0	1	$1/r$	r	0	1	-	-	1	-1	1	0	$r \to \infty$
SUM	$\sum_i x_i$	1	1	-	-	0	1	-	-	1	0	1	0	
NONZERO COUNT	$\lvert\{i : x_i \neq 0\}\rvert$	1	0	-	-	0	1	-	-	1	0	1	0	
MEAN	$1/N \sum_i x_i$	1	1	-	-	1	0	-	-	1	0	1	0	
kTH MOMENT	$1/N \sum_i x_i^k$	1	k	-	-	1	0	-	-	1	0	1	0	
lTH POWER OF kTH MOMENT	$(1/N \sum_i x_i^k)^l$	l	k	-	-	l	0	-	-	1	0	1	0	
MIN/MAX	$\min_i x_i / \max_i x_i$	0	1	$1/r$	r	$1/s$	s	-	-	1	1	1	0	$r, s \to \infty$
MAX/MIN	$\max_i x_i / \min_i x_i$	$1/r$	r	-	-	0	1	$1/s$	s	1	0	1	1	$r, s \to \infty$

Table 1: Different functions achievable by varying the parameters in the formulation in Equation 2.

returns for any multiset $\boldsymbol{x}$ of arbitrary cardinality $N \in \mathbb{N}$ a value $agg(\boldsymbol{x}) \in \mathbb{R}$.

Standard aggregation functions like *mean* and *max* can be understood as (normalized) L_p-norms. We therefore build our parametric LAF aggregator around functions of a form that generalizes L_p-norms:

$$L_{a,b}(\boldsymbol{x}) := \left(\sum_i x_i^b \right)^a \qquad (a, b \geq 0). \tag{1}$$

$L_{a,b}$ is invariant under the addition of zeros: $L_{a,b}(\boldsymbol{x}) = L_{a,b}(\boldsymbol{x} \cup \mathbf{0})$ where $\mathbf{0}$ is a multiset of zeros of arbitrary cardinality. In order to also enable aggregations that can represent *conjunctive* behaviour such as *min*, we make symmetric use of aggregators of the multisets $\mathbf{1} - \boldsymbol{x} := \{1 - x_i | x_i \in \boldsymbol{x}\}$. For $L_{a,b}(\mathbf{1} - \boldsymbol{x})$ to be a well-behaved, dual version of $L_{a,b}(\boldsymbol{x})$, the values in $\boldsymbol{x}$ need to lie in the range $[0, 1]$. We therefore restrict the following definition of our *learnable aggregation function* to sets $\boldsymbol{x}$ whose elements are in $[0, 1]$:

$$\text{LAF}(\boldsymbol{x}) := \frac{\alpha L_{a,b}(\boldsymbol{x}) + \beta L_{c,d}(\mathbf{1} - \boldsymbol{x})}{\gamma L_{e,f}(\boldsymbol{x}) + \delta L_{g,h}(\mathbf{1} - \boldsymbol{x})} \tag{2}$$

defined by tunable parameters $a, \ldots, h \geq 0$, and $\alpha, \ldots, \delta \in \mathbb{R}$. In cases where sets need to be aggregated whose elements are not already bounded by $0, 1$, we apply a sigmoid function to the set elements prior to aggregation.

Table 1 shows how a number of important aggregation functions are special cases of LAF (for values in $[0, 1]$). We make repeated use of the fact that $L_{0,1}$ returns the constant 1. For max and min LAF only provides an asymptotic approximation in the limit of specific function parameters (as indicated in the last column of Table 1). In most cases, the parameterization of LAF for the functions in Table 1 will not be unique. Being able to encode the powers of moments implies that e.g. the variance of $\boldsymbol{x}$ can be expressed as the difference $1/N \sum_i x_i^2 - (1/N \sum_i x_i)^2$ of two LAF aggregators.

Since LAF includes sum-aggregation, we can adapt the results of [Zaheer *et al.*, 2017] and [Wagstaff *et al.*, 2019] on the theoretical universality of sum-aggregation as follows.

Proposition 1. *Let $\mathcal{X} \subset \mathbb{R}$ be countable, and f a function defined on finite multisets with elements from $\mathcal{X}$. Then there exist functions $\phi : \mathcal{X} \to [0, 1]$, $\rho : \mathbb{R} \to \mathbb{R}$, and a parameterization of* LAF, *such that $f(\boldsymbol{x}) = \rho(LAF(\phi\boldsymbol{x}); \alpha, \beta, \gamma, \delta, a, b, c, d)$, where $\phi\boldsymbol{x}$ is the multiset $\{\phi(x) | x \in \boldsymbol{x}\}$.*

A proof in [Wagstaff *et al.*, 2019] for a very similar proposition used a mapping from $\mathcal{X}$ into the reals. Our requirement that LAF inputs must be in $[0, 1]$ requires a modification of the proof (contained in the supplementary material[1]), which for the definition of ϕ relies on a randomized construction. Proposition 1 shows that we retain the theoretical universality guarantees of [Zaheer *et al.*, 2017], while enabling a wider range of solutions based on continuous encoding and decoding functions.

LAF enables a continuum of intermediate and hybrid aggregators. In Figure 1 we plot 4 different randomly generated LAF functions over $[0, 1] \times [0, 1]$, i.e., evaluated over sets of size 2. Parameters $\alpha, \ldots, \gamma$ were randomly sampled in $[0, 1]$, b, d, f, h in $\{0, \ldots, 5\}$, and a, c, e, g obtained as $1/i$ with i a random integer from $\{0, \ldots, 5\}$. The figure illustrates the rich repertoire of aggregation functions with different qualitative behaviours already for non-extreme parameter values.

Learning the functions depicted in Table 1 can in principle be done by a single LAF unit. However, learning complex aggregation functions might require a larger number of independent units, in that the final aggregation is the result of the combination of simpler aggregations. Moreover, a LAF layer should be able to approximate the behaviour of simpler functions also when multiple units are used. Therefore, we analyzed the application of multiple LAF units to some of the known functions in Table 1. The details and the visual representation of this analysis is shown in the supplementary material. Although using only one function is sometimes sufficient to greatly approximate the target function, the error variance among different runs is quite high, indicating that the optimization sometimes fails to converge to a good set of parameters. Multiple units provide more stability while performing better than a single unit aggregation in some cases. We therefore construct the LAF architecture for the experimental section by using multiple aggregators, computing the final aggregation as a linear combination of the units aggregations. Several LAF units can be combined as shown in Figure 1, to capture different aspects of the input set, which can be in general a multiset of vectors $\boldsymbol{x} = \{x_1, \ldots, x_N\}$, where now $x_i \in \mathbb{R}^d$. Note that multiple aggregators are also used in related frameworks such as DeepSets [Zaheer *et al.*, 2017] or Graph Neural Networks [Veličković *et al.*, 2018;
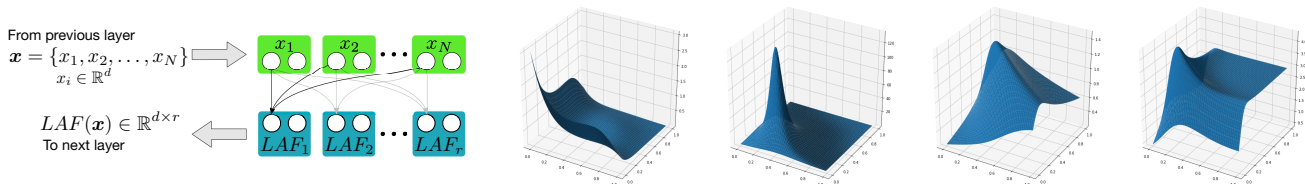
Figure 1: Left: End-to-end LAF architecture. Right: LAF functions with randomly generated parameters.

Corso *et al.*, 2020]. A module with r LAF units takes as input d-dimensional vectors and produces a vector of size $r \times d$ as output. Each LAF unit performs an *element-wise* aggregation of the vectors in the set such that $L_{k,j} = \mathrm{LAF}(\{x_{i,j}, \ldots, x_{N,j}\}; \alpha_k, \beta_k, \gamma_k, \delta_k, a_k, b_k, c_k, d_k)$ for $k = 1, \ldots, r$ and $j = 1, \ldots, d$. The output vector can be then fed into the next layer.

3 Related Work

Several studies address the problem of aggregating data over sets. Sum-decomposition strategies have been used in [Zaheer *et al.*, 2017] for points cloud classification and set expansion tasks and in [Santoro *et al.*, 2017] for question answering and dynamic physical systems computation. Max, sum and average are standard aggregation functions for node neighborhoods in graph neural networks [Hamilton *et al.*, 2017; Kipf and Welling, 2017; Xu *et al.*, 2019; Veličković *et al.*, 2018]. [Zaheer *et al.*, 2017] first proved universal representation results for these standard aggregators when combined with learned mappings over inputs and results of the aggregation. However, [Wagstaff *et al.*, 2019] showed that these universality results are of little practical use, as they either require highly discontinuous mappings that would be extremely difficult to learn, or a latent dimension that is at least the size of the maximum number of input elements.

Uninorms [Yager and Rybalov, 1996] are a class of aggregation functions in fuzzy logic that can behave in a *conjunctive*, *disjunctive* or *averaging* manner depending on a parameter called *neutral element*. [Melnikov and Hüllermeier, 2016] proposed to learn fuzzy aggregators by adjusting these learnable parameters, showing promising results on combining reviewers scores on papers into an overall decision of acceptance or reject. Despite the advantage of incorporating different behaviours in one single function, uninorms present discontinuities in the regions between aggregators, making them not amenable to be utilized in fully differentiable frameworks. Furthermore the range of possible behaviours is restricted to those commonly used in the context of fuzzy-logic.

The need for considering multiple candidate aggregators is advocated in a very recent work that was developed in parallel with our framework [Corso *et al.*, 2020]. The resulting architecture, termed *Principal Neighborhood Aggregation* (PNA) combines multiple standard aggregators, including most of the ones we consider in the LAF framework, adjusting their outputs with degree scalers. However, the underlying philosophy is rather different. PNA aims at learning to select the appropriate aggregator(s) from a pool of candidates, while LAF explores a continuous space of aggregators that includes

standard ones as extreme cases. Our experimental evaluation shows that PNA has troubles in learning aggregators that generalize over set sizes, despite having them in the pool of candidates, likely because of the quasi-combinatorial structure of its search space. On the other hand, LAF can successfully learn even the higher moment aggregators and consistently outperforms PNA.

Closely connected, but somewhat complementary to aggregation operators are *attention mechanisms* [Bahdanau *et al.*, 2015; Vaswani *et al.*, 2017]. They have been explored to manipulate set data in [Lee *et al.*, 2019] and in the context of multi-instance learning [Ilse *et al.*, 2018]. Attention operates at the level of set elements, and aims at a transformation (weighting) of their representations such as to optimize a subsequent weighted sum-aggregation. Although the objectives of attention-based frameworks and LAF are different in principle, our method can be used inside attention frameworks as the aggregation layer of the learned representation. We discuss the combination of LAF and attention in Section 5 showing the advantage of using LAF.

4 Experiments

In this section, we present and discuss experimental results showing the potential of the LAF framework on both synthetic and real-world tasks. Synthetic experiments are aimed at showing the ability of LAF to learn a wide range of aggregators and its ability to generalize over set sizes (i.e., having test-set sets whose cardinality exceeds the cardinality of the training-set sets), something that alternative architectures based on predefined aggregators fail to achieve. We use DeepSets, PNA, and LSTM as representatives of these architectures. The LSTM architecture corresponds to a version of DeepSets where the aggregation function is replaced by a LSTM layer. Experiments on diverse tasks including point cloud classification, text concept set retrieval and graph properties prediction are aimed at showing the potential of the framework on real-world applications.

4.1 Scalars Aggregation

This section shows the learning capacity of the LAF framework to learn simple and complex aggregation functions where constituents of the sets are simple numerical values. In this setting we consider sets made of scalar integer values. The training set is constructed as follows: for each set, we initially sample its cardinality K from a uniform distribution taking values in $\{2, \ldots, M\}$, and then we uniformly sample K integers in $\{0, \ldots, 9\}$. For the training set we use $M = 10$. We construct several test sets for different values
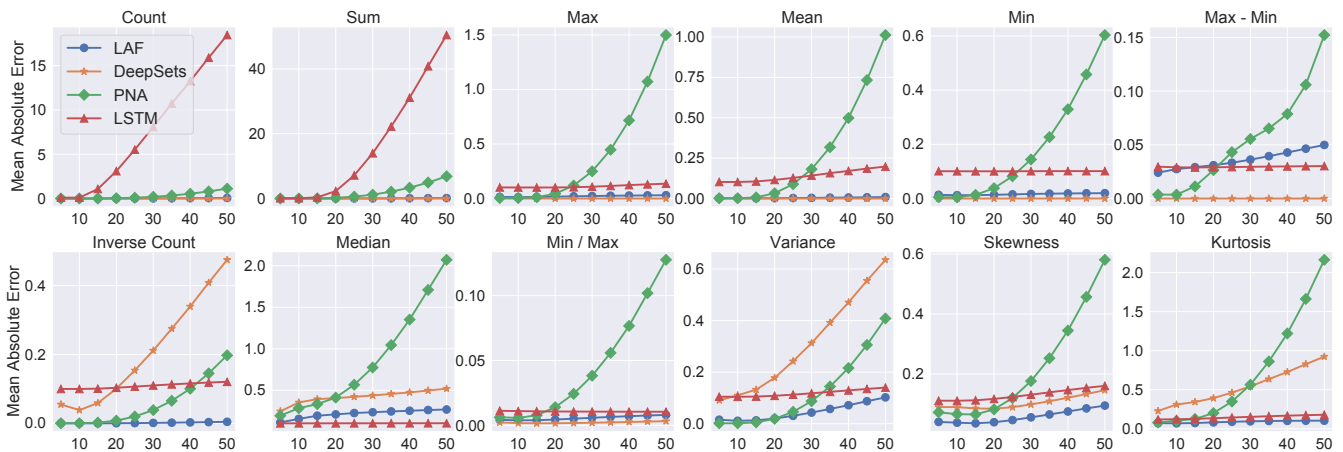
Figure 2: Test performances for the synthetic experiment with integer scalars on increasing test set size. The x axis represents the maximum test set cardinality, the y axis depicts the MAE. The dot, star, diamond and triangle symbols denote LAF, DeepSets, PNA, and LSTM respectively. Skewness: $1/N \sum_i ((x_i - \hat{\mu})/\hat{\sigma})^3$, Kurtosis: $1/N \sum_i ((x_i - \hat{\mu})/\hat{\sigma})^4$, where $\hat{\mu}$ and $\hat{\sigma}$ are the sample mean and standard deviation.

of M ($M = 5, 10, 15, 20, 25, 30, 35, 40, 45, 50$). This implies that models need to generalize to larger set sizes. Contrarily to the training set, each test set is constructed in order to diversify the target labels it contains, so as to avoid degenerate behaviours for large set sizes (e.g., maximum constantly equal to 9). Each synthetic dataset is composed of 100,000 sets for training, 20,000 set for validating and 100,000 for testing. The number of aggregation units is set as follows. The model contains nine LAF (Equation 2) units, whose parameters $\{a_k, \ldots, h_k\}$, $k = 1, \ldots, 9$ are initialized according to a uniform sampling in $[0, 1]$ as those parameters must be positive, whereas the coefficients $\{\alpha, \ldots, \delta\}$ are initialized with a Gaussian distribution with zero mean and standard deviation of 0.01 to cover also negative values. The positivity constraint for parameters $\{a, b, \ldots, h\}$ is enforced by projection during the optimization process. The remaining parameters can take on negative values. DeepSets also uses nine units: three max units, three sum units, and three mean units and PNA uses seven units: mean, max, sum, standard deviation, variance, skewness and kurtosis. Preliminary experiments showed that expanding the set of aggregators for PNA with higher order moments only leads to worse performance. Each set of integers is fed into an embedding layer (followed by a sigmoid) before performing the aggregation function. DeepSets and PNA do need an embedding layer (otherwise they would have no parameters to be tuned). Although LAF does not need an embedding layer, we used it in all models to make the comparison more uniform. The architecture details are reported in the supplementary material. We use the Mean Absolute Error (MAE) as a loss function to calculate the prediction error.

Figure 2 shows the trend of the MAE for the three methods for increasing test set sizes, for different types of target aggregators. As expected, DeepSets manages to learn the identity function and thus correctly models aggregators like sum, max and mean. Even if LAF needs to adjust its parameters in order to properly aggregate the data, its performance are competitive with those of DeepSets. When moving to more

complex aggregators like inverse count, median or moments of different orders, DeepSets fails to learn the latent representation. One the other hand, the performance of LAF is very stable for growing set sizes. While having in principle at its disposal most of the target aggregators (including higher order moment) PNA badly overfits over the cardinality of sets in the training set in all cases (remember that the training set contains sets of cardinality at most 10). The reason why LAF substantially outperforms PNA on large set sizes could be explained in terms of a greater flexibility to adapt to the learnt representation. Indeed, LAF parameters can adjust the *laf* function to be compliant with the latent representation even if the input mapping fails to learn the identity. On the other hand, having a bunch of fixed, hard-coded aggregators, PNA needs to be able to both learn the identity mapping and select the correct aggregator among the candidates. Finally, LSTM results are generally poor when compared to the other methods, particularly in the case of the count and the sum.

4.2 MNIST Digits

We performed an additional set of experiments aiming to demonstrate the ability of LAF to learn from more complex representations of the data by plugging it into end-to-end differentiable architectures. In these experiments, we thus replaced numbers by visual representations obtained from MNIST digits. Unlike the model proposed in the previous section, here we use three dense layers for learning picture representations before performing the aggregation function. Results obtained in this way are very similar to those obtained with numerical inputs and due to space limitations we report them along with other architectural and experimental details in the supplementary material.

4.3 Point Clouds

In order to evaluate LAF on real-world dataset, we consider point cloud classification, a prototype task for set-wise prediction. Therefore, we run experimental comparisons on

Method	P100	P1000
DEEPSETS	82.0±2.0%	**87.0±1.0%**
PNA	82.9±0.7%	86.4±0.6%
LSTM	78.7±1.1%	82.2±1.7%
LAF	**84.0±0.6%**	**87.0±0.5%**

Table 2: Results on the Point Clouds classification task. Accuracies with standard deviations (over 5 runs) for the ModelNet40 dataset.

the ModelNet40 [Wu *et al.*, 2015] dataset, which consists of 9,843 training and 2,468 test point clouds of objects distributed over 40 classes. The dataset is preprocessed following the same procedure described by [Zaheer *et al.*, 2017]. We create point clouds of 100 and 1,000 three-dimensional points by adopting the point-cloud library's sampling routine developed by [Rusu and Cousins, 2011] and normalizing each set of points to have zero mean (along each axis) and unit (global) variance. We refer with P100 and P1000 to the two datasets. For all the settings, we consider the same architecture and hyper-parameters of the DeepSets permutation invariant model described by [Zaheer *et al.*, 2017]. For LAF, we replace the original aggregation function (max) used in DeepSets with 10 LAF units, while for PNA we use the concatenation of max, min, mean, and standard deviation, as proposed by the authors. For PNA we do not consider any scaler, as the cardinalities of the sets are fixed. Results in Table 2 show that LAF produces an advantage in the lower resolution dataset (i.e., on P100), while it obtains comparable (and slightly more stable) performances in the higher resolution one (i.e., on P1000). These results suggest that having predefined aggregators is not necessarily an optimal choice in real world cases, and that the flexibility of LAF in modeling diverse aggregation functions can boost performance and stability.

4.4 Set Expansion

Following the experimental setup of DeepSets, we also considered the *Set Expansion* task. In this task the aim is to augment a set of objects of the same class with other similar objects, as explained in [Zaheer *et al.*, 2017]. The model learns to predict a score for an object given a query set and decide whether to add the object to the existing set. Specifically, [Zaheer *et al.*, 2017] consider the specific application of set expansion to text concept retrieval. The idea is to retrieve words that belong to a particular concept, giving as input set a set of words having the same concept. We employ the same model and hyper-parameters of the original publication, where we replace the sum-decomposition aggregation with LAF units for our methods and the min, max, mean, and standard deviation aggregators for PNA.

We trained our model on sets constructed from a vocabulary of different size, namely *LDA-1K*, *LDA-3K* and *LDA-5K*. Table 3 shows the results of LAF, DeepSets and PNA on different evaluation metrics. We report the retrieval metrics recall@K, median rank and mean reciprocal rank. We also report the results on other methods the authors compared to in the original paper. More details on the other methods in the table can be found in the original publication. Briefly, *Random* samples a word uniformly from the vocabulary; *Bayes Set* [Ghahramani and Heller, 2006]; *w2v-Near* computes the nearest neighbors in the word2vec [Mikolov *et al.*, 2013] space; *NN-max* uses a similar architecture as our DeepSets but uses max pooling to compute the set feature, as opposed to sum pooling; *NN-max-con* uses max pooling on set elements but concatenates this pooled representation with that of query for a final set feature; *NN-sum-con* is similar to NN-max-con but uses sum pooling followed by concatenation with query representation. For the sake of fairness, we have rerun DeepSets using the current implementation from the authors (indicated as DeepSet* in Table 3), exhibiting better results than the ones reported in the original paper. Nonetheless, LAF outperforms all other methods in most cases, especially on *LDA-3K* and *LDA-5K*.

4.5 Multi-task Graph Properties

[Corso *et al.*, 2020] defines a benchmark consisting of 6 classical graph theory tasks on artificially generated graphs from a wide range of popular graph types like Erdos-Renyi, Barabasi-Albert or star-shaped graphs. Three of the tasks are defined for nodes, while the other three for whole graphs. The node tasks are the single-source shortest-path lengths (N1), the eccentricity (N2) and the Laplacian features (N3). The

Method	LDA-1k (Vocab = 17k)					LDA-3k (Vocab = 38k)					LDA-5k (Vocab = 61k)				
	Recall(%)			MRR	Med.	Recall(%)			MRR	Med.	Recall(%)			MRR	Med.
	@10	@100	@1k			@10	@100	@1k			@10	@100	@1k		
RANDOM	0.06	0.6	5.9	0.001	8520	0.02	0.2	2.6	0.000	28635	0.01	0.2	1.6	0.000	30600
BAYES SET	1.69	11.9	37.2	0.007	2848	2.01	14.5	36.5	0.008	3234	1.75	12.5	34.5	0.007	3590
W2V NEAR	6.00	**28.1**	54.7	0.021	641	4.80	21.2	43.2	0.016	2054	4.03	16.7	35.2	0.013	6900
NN-MAX	4.78	22.5	53.1	0.023	779	5.30	24.9	54.8	0.025	672	4.72	21.4	47.0	0.022	1320
NN-SUM-CON	4.58	19.8	48.5	0.021	1110	5.81	27.2	60.0	0.027	453	4.87	23.5	53.9	0.022	731
NN-MAX-CON	3.36	16.9	46.6	0.018	1250	5.61	25.7	57.5	0.026	570	4.72	22.0	51.8	0.022	877
DEEPSETS	5.53	24.2	54.3	0.025	696	6.04	28.5	60.7	0.027	426	5.54	26.1	55.5	0.026	616
DEEPSETS*	5.89	26.0	**55.3**	0.026	**619**	7.56	28.5	**64.0**	0.035	349	6.49	27.9	**56.9**	0.030	536
PNA	5.56	24.7	53.2	0.027	753	7.04	27.2	58.7	0.028	502	5.47	23.8	52.4	0.025	807
LSTM	4.29	21.5	52.6	0.022	690	5.56	25.7	58.8	0.026	830	4.87	23.8	55.0	0.022	672
LAF	**6.51**	26.6	54.5	**0.030**	650	**8.14**	**32.3**	62.8	**0.037**	339	**6.71**	**28.3**	**56.9**	**0.031**	**523**

Table 3: Results on Text Concept Set Retrieval on LDA-1k, LDA-3k, and LDA-5k. Bold values denote the best performance for each metric.

METHOD	N1	N2	N3	G1	G2	G3
BASELINE	-1.87	-1.50	-1.60	-0.62	-1.30	-1.41
GIN	-2.00	-1.90	-1.60	-1.61	-2.17	-2.66
GCN	-2.16	-1.89	-1.60	-1.69	-2.14	-2.79
GAT	-2.34	-2.09	-1.60	-2.44	-2.40	-2.70
MPNN (MAX)	-2.33	-2.26	-2.37	-1.82	-2.69	-3.52
MPNN (SUM)	-2.36	-2.16	-2.59	-2.54	-2.67	-2.87
PNA*	-2.54	-2.42	-2.94	**-2.61**	-2.82	-3.29
PNA	**-2.89**	**-2.89**	**-3.77**	**-2.61**	**-3.04**	-3.57
LAF	-2.13	-2.20	-1.67	-2.35	-2.77	**-3.63**

Table 4: Results on the Multi-task graph properties prediction benchmark, expressed in $\log 10$ of mean squared error.

graph tasks are graph connectivity (G1), diameter (G2), and the spectral radius (G3). We compare LAF against PNA by simply replacing the original PNA aggregators and scalers with 100 LAF units (see Equation 2). Table 4 shows that albeit these datasets were designed to highlight the features of the PNA architecture, that outperforms a wide range of alternative graph neural network approaches LAF produces competitive results, outperforming state-of-the-art GNN approaches like GIN [Xu *et al.*, 2019], GCN [Kipf and Welling, 2017] and GAT [Veličković *et al.*, 2018] and even improving over PNA on spectral radius prediction. PNA* is the variant without scalers of PNA still proposed by [Corso *et al.*, 2020].

5 SetTransformer With LAF Aggregation

In this section we investigate the combination of LAF aggregation with attention mechanisms on sets as proposed in the SetTransformer framework [Lee *et al.*, 2019]. Briefly, SetTransformer consists of an *encoder* and a *decoder*. The encoder maps a set of input vectors into a set of feature vectors by leveraging attention blocks. The decoder employs a *pooling multihead attention* (PMA) layer, which aggregates the set of feature vectors produced by the encoder. In the following experiment we replace PMA by a LAF layer. Inspired by one of the tasks described in [Lee *et al.*, 2019], we propose here to approximate the average of the unique numbers in a set of MNIST images. Solving the task requires to learn a cascade of two processing steps, one that detects unique elements in a set (which can be done by the transformer encoder, as shown in the experiment by [Lee *et al.*, 2019]) and one that aggregates the results by averaging (which LAF is supposed to do well). The set cardinalities are uniformly sampled from $\{2, 3, 4, 5\}$ and each set label is calculated as the average of the unique digits contained in the set. We trained two SetTransformer models: one with PMA (ST-PMA) and the other with LAF (ST-LAF). The full implementation details are reported in the supplementary material. Quantitative and qualitative results of the evaluation are shown in Figure 3, where we report the MAE for both methods[2]. ST-LAF exhibits a nice improvement over ST-PMA for this particular task. Note

---

[2]We run several experiments by changing the number of seeds k of PMA. All of them exhibited the same behaviour. For this experiment we used $k = 1$.
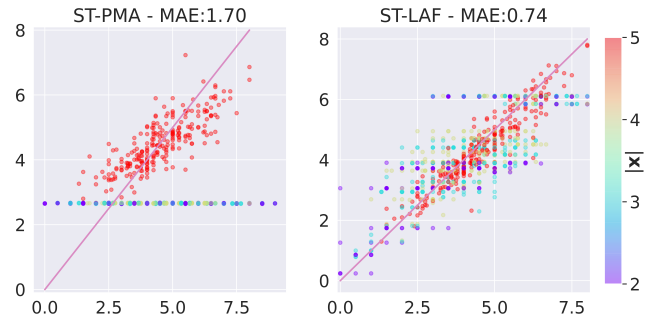

Figure 3: Distribution of the predicted values for ST-PMA and ST-LAF by set cardinalities. On the x-axis the true labels of the sets, on the y-axis the predicted ones. Different colors represent the sets' cardinalities $|\boldsymbol{x}|$.

that for ST-PMA only 25% of the sets (red points in the scatter plot), corresponding to sets with maximum cardinality, approximates well the average, while for all other cardinalities (the remaining 75% of the sets) ST-PMA predicts a constant value equal to the average label in the training set. ST-LAF instead clearly captures the distribution of the labels, generalizing better with respect to the set sizes. A similar behaviour was observed when learning to predict the sum rather than the average of the unique digits in a set (see supplementary material for the results).

6 Conclusions

The theoretical underpinnings for sum aggregation as a universal framework for defining set functions do not necessarily provide a template for practical solutions. Therefore we introduced a framework for learning aggregation functions that makes use of a parametric aggregator to effectively explore a rich space of possible aggregations. LAF defines a new class of aggregation functions, which include as special cases widely used aggregators, and also has the ability to learn complex functions such as higher-order moments. We empirically showed the generalization ability of our method on synthetic settings as well as real-world datasets, providing comparisons with state-of-the-art sum-decomposition approaches and recently introduced techniques. The flexibility of our model is a crucial aspect for potential practical use in many deep learning architectures, due to its ability to be easily plugged into larger architectures, as shown in our experiments with the SetTransformer. The portability of LAF opens a new range of possible applications for aggregation functions in machine learning methods, and future research in this direction can enhance the expressivity of many architectures and models that deal with unstructured data.

References

[Bahdanau *et al.*, 2015] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[Corso *et al.*, 2020] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Velickovic. Principal neighbourhood aggregation for graph nets. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[Ghahramani and Heller, 2006] Zoubin Ghahramani and Katherine A Heller. Bayesian sets. In *Advances in neural information processing systems*, pages 435–442, 2006.

[Hamilton *et al.*, 2017] William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 1024–1034, 2017.

[Ilse *et al.*, 2018] Maximilian Ilse, Jakub M. Tomczak, and Max Welling. Attention-based deep multiple instance learning. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 2132–2141, 2018.

[Kim, 2014] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751, 2014.

[Kipf and Welling, 2017] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[Lee *et al.*, 2019] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 3744–3753, 2019.

[Melnikov and Hüllermeier, 2016] Vitalik Melnikov and Eyke Hüllermeier. Learning to aggregate using uninorms. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part II*, pages 756–771, 2016.

[Mikolov *et al.*, 2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[Rusu and Cousins, 2011] Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *2011 IEEE international conference on robotics and automation*, pages 1–4. IEEE, 2011.

[Santoro *et al.*, 2017] Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, pages 4967–4976, 2017.

[Tibo *et al.*, 2020] Alessandro Tibo, Manfred Jaeger, and Paolo Frasconi. Learning and interpreting multi-multi-instance learning networks. *Journal of Machine Learning Research*, 21(193):1–60, 2020.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[Veličković *et al.*, 2018] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR'18*, 2018.

[Wagstaff *et al.*, 2019] Edward Wagstaff, Fabian B Fuchs, Martin Engelcke, Ingmar Posner, and Michael Osborne. On the limitations of representing functions on sets. *arXiv preprint arXiv:1901.09006*, 2019.

[Wu *et al.*, 2015] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[Xu *et al.*, 2019] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

[Yager and Rybalov, 1996] Ronald R Yager and Alexander Rybalov. Uninorm aggregation operators. *Fuzzy sets and systems*, 80(1):111–120, 1996.

[Zaheer *et al.*, 2017] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3391–3401. Curran Associates, Inc., 2017.

A Proof of Proposition 1

Let $\mathcal{X} = \{x_0, x_1, \ldots\}$. For $i \geq 0$ let r_i be a random number sampled uniformly from the interval $[0, 1]$. Define $\phi(x_i) := r_i$. Let $\boldsymbol{x} = \{a_i : x_i | i \in J\}$, $\boldsymbol{x}' = \{a'_h : x_h | h \in J'\}$ be two finite multisets with elements from $\mathcal{X}$, where J, J' are finite index sets, and a_i, a'_h denote the multiplicity with which elements x_i, x_h appear in $\boldsymbol{x}$, respectively $\boldsymbol{x}'$. Now assume that $\boldsymbol{x} \neq \boldsymbol{x}'$, but

$$\sum_{i \in J} a_i \phi(x_i) = \sum_{h \in J'} a'_h \phi(x_h), \tag{A.1}$$

i.e.,

$$\sum_{j \in J \cup J'} (a_j - a'_j) r_j = 0, \tag{A.2}$$

where now a_j, respectively a'_j is defined as 0 if $j \in J' \setminus J$, respectively $j \in J \setminus J'$. Since $\boldsymbol{x} \neq \boldsymbol{x}'$, the left side of this equation is not identical zero. Without loss of generality, we may actually assume that all coefficients $a_j - a'_j$ are nonzero. The event that the randomly sampled values $\{r_j | j \in J \cup J'\}$ satisfy the linear constraint (A.2) has probability zero. Since the set of pairs of finite multisets over $\mathcal{X}$ is countable, also the probability that there exists any pair $\boldsymbol{x} \neq \boldsymbol{x}'$ for which (A.1) holds is zero. Thus, with probability one, the mapping from multisets $\boldsymbol{x}$ to their sum-aggregation $\sum_{x \in \boldsymbol{x}} \phi(x)$ is injective. In particular, there exists a set of fixed values $r_0, r_1, \ldots$, such that the (deterministic) mapping $x_i \mapsto r_i$ has the desired properties. The existence of the "decoding" function ρ is now guaranteed as in the proofs of [Zaheer *et al.*, 2017; Wagstaff *et al.*, 2019].

Clearly, due to the randomized construction, the theorem and its proof have limited implications in practice. This however, already is true for previous results along these lines, where at least for the decoding function ρ, not much more than pure existence could be demonstrated.

B Learning

We study here the difficulty of solving the optimization problem when varying the number of LAF units, aiming to show that the use of multiple units helps finding a better solution. We formulate as learning tasks some of the target functions described in Table 1. Additionally, we inspect the parameters of the learned model. We construct a simple architecture similar to the aggregation layer presented in Section 4, in which the aggregation is performed using one or more LAF units and, in the case of multiple aggregators, their outputs are combined together using a linear layer. We also discard any non-linear activation function prior to the aggregation because the input sets are composed of real numbers in the range $[0, 1]$, with a maximum of 10 elements for each set. We consider 1,3,6,9,12,15,18 and 21 LAF units in this setting. For each function and for each number of units we performed 500 random restarts. The results are shown in Figure B.1, where we report the MAE distributions. Let's initially consider the cases when a single unit performs the aggregation. Note first that the functions listed in Table 1 can be parametrized in an infinite number of alternative ways. For instance, consider the *sum* function. A possible solution is obtained if $L_{a,b}$ learns

the *sum*, $L_{e,f} = 1$ and $\alpha = \gamma$. If instead $L_{a,b} = $ *sum* and $L_{e,f} = L_{g,h} = 1$, it is sufficient that $\gamma + \delta = \alpha$ to still obtain the sum. This is indeed what we found when inspecting the best performing models among the various restarts, as shown in the following:

$$sum : \frac{1.75(\sum x^{1.00})^{1.00} + 0.00(\sum (1-x)^{0.00})^{0.56}}{0.91(\sum x^{0.24})^{0.00} + 0.84(\sum (1-x)^{0.36})^{0.00}}$$

$$count : \frac{1.01(\sum x^{0.00})^{0.99} + 0.94(\sum (1-x)^{0.00})^{1.01}}{1.08(\sum x^{0.47})^{0.00} + 0.88(\sum (1-x)^{1.02})^{0.00}}$$

$$mean : \frac{1.51(\sum x^{1.00})^{1.00} + 0.00(\sum (1-x)^{0.62})^{0.00}}{0.00(\sum x^{0.30})^{0.00} + 1.51(\sum (1-x)^{0.00})^{1.00}}$$

A detailed overview of the parameters' values learned using one LAF unit is depicted in Table B.1. For each function in Figure B.1, we report the values of the random restart that obtained the lowest error. The evaluation clearly shows that learning a function with just one LAF unit is not trivial. In some cases LAF was able to almost perfectly match the target function, but to be reasonably confident to learn a good representation many random restarts are needed, since the variance among different runs is quite large. The error variance reduces when more than one LAF unit is adopted, drastically dropping when six units are used in parallel, still maintaining a reasonable average error. Jointly learning multiple LAF units and combining their outputs can lead to two possible behaviours giving rise to an accurate approximation of the underlying function: in the first case, it is possible that one "lucky" unit learns a parametrization close to the target function, leaving the linear layer after the aggregation to learn to choose that unit or to rescale its output. In the second case the target function representation is "distributed" among the different units, here the linear layer is responsible to obtain the function by combining the LAF aggregation outputs. In the following we show another example of a learnt model, for a setting with three LAF units. Here the target function is the *count*.

$$unit1 : \frac{0.81(\sum x^{0.87})^{0.37} + 0.80(\sum (1-x)^{0.74})^{0.72}}{1.19(\sum x^{0.19})^{0.72} + 1.18(\sum (1-x)^{0.00})^{0.62}}$$

$$unit2 : \frac{1.43(\sum x^{0.00})^{1.10} + 1.31(\sum (1-x)^{0.01})^{0.74}}{0.64(\sum x^{0.85})^{0.00} + 0.62(\sum (1-x)^{0.46})^{0.00}}$$

$$unit3 : \frac{0.83(\sum x^{0.87})^{0.37} + 0.77(\sum (1-x)^{0.12})^{0.00}}{1.17(\sum x^{0.69})^{0.86} + 1.22(\sum (1-x)^{0.00})^{0.16}}$$

$$linear : \quad 0.02 + (-0.13 * unit1) + $$
$$+(0.50 * unit2) + (-0.07 * unit3)$$

In this case, the second unit learns a function that counts twice the elements of the set. The output of this unit is then halved by the linear layer, which gives very little weights to the outputs of the other units.
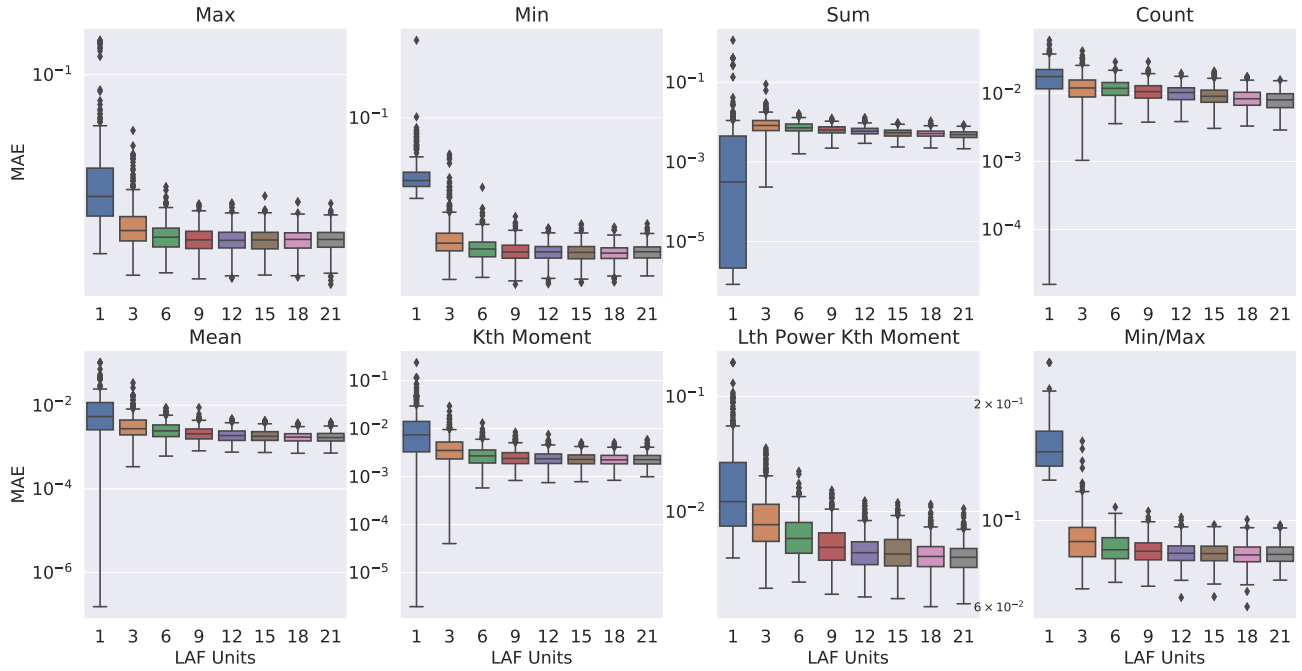
Figure B.1: Trend of the MAE obtained with an increasing number of LAF units for most of the functions reported in Table 1. The error distribution is obtained performing 500 runs with different random parameter initializations. A linear layer is stacked on top of the LAF layer with more than 1 unit. The y axis is plot in logaritmic scale.

NAME	a	b	c	d	e	f	g	h	α	β	γ	δ
MAX	0.28	4.74	0.00	0.57	0.33	1.74	0.00	0.48	1.68	0.00	0.90	0.75
MIN	0.28	0.28	0.27	1.13	0.30	0.35	0.87	3.69	0.51	0.00	0.45	1.91
SUM	1.00	1.00	0.56	0.00	0.00	0.24	0.00	0.36	1.75	0.00	0.91	0.84
COUNT	0.99	0.00	1.01	0.00	0.00	0.47	0.00	1.02	1.01	0.94	1.08	0.88
MEAN	1.00	1.00	0.00	0.62	0.00	0.30	1.00	0.00	1.51	0.00	0.00	1.51
kTH MOMENT	1.00	2.00	0.00	0.13	1.00	0.00	1.00	0.00	1.67	0.00	0.83	0.84
lTH POWER OF kTH MOMENT	2.87	2.15	0.00	0.91	2.94	0.00	1.71	0.00	1.65	0.01	1.44	0.24
MIN/MAX	0.06	0.00	1.52	2.36	0.18	4.40	0.64	7.25	0.23	0.10	0.27	2.26

Table B.1: Parameters' values learned with one LAF unit.

C Details of Sections 4.1 - Experiments on Scalars

We used mini-batches of 64 sets and trained the models for 100 epochs. We use Adam as parameter optimizer, setting the initial learning rate to $1e^{-3}$ and apply adaptive decay based on the validation loss.

Each element in the dataset is a set of scalars $\boldsymbol{x} = \{x_1, \ldots, x_N\}, x_i \in \mathbb{R}$.

Network architecture:

$$\boldsymbol{x} \to \text{EMBEDDING}(10,10) \to \text{SIGMOID}$$
$$\to \text{LAF}(9) \to \text{DENSE}(10 \times 9, 1)$$

D Details of Sections 4.2 - MNIST Digits

In this section, we modify the experimental setting in Section 4.1 for the integers scalars to process MNIST images of digits. The dataset is the same as in the experiment on scalars,

but integers are replaced by randomly sampling MNIST images for the same digits. Instances for the training and test sets are drawn from the MNIST training and test sets, respectively. We used mini-batches of 64 sets and trained the models for 100 epochs. We use Adam as parameter optimizer, setting the initial learning rate to $1e^{-3}$ and apply adaptive decay based on the validation loss. Each element in the dataset is a set of vectors $\boldsymbol{x} = \{x_1, \ldots, x_N\}, x_i \in \mathbb{R}^{784}$. Network architecture:

$$\boldsymbol{x} \to \text{DENSE}(784,300) \to \text{TANH}$$
$$\to \text{DENSE}(300,100) \to \text{TANH}$$
$$\to \text{DENSE}(100,30) \to \text{SIGMOD}$$
$$\to \text{LAF}(9) \to \text{DENSE}(30 \times 9, 1000) \to \text{TANH}$$
$$\to \text{DENSE}(1000,100) \to \text{TANH} \to \text{DENSE}(100,1)$$

Figure D.1 shows the comparison of LAF, DeepSets, PNA, and LSTM in this setting. Results are quite similar to those
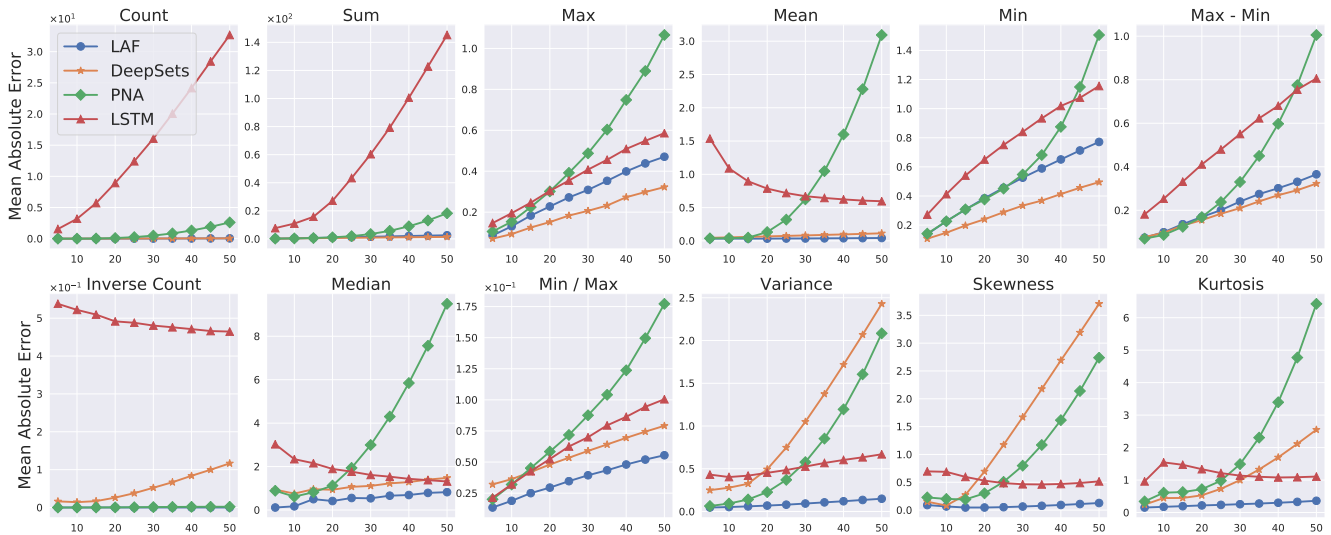
Figure D.1: Test performances for the synthetic experiment on MNIST digits on increasing test set size. The x axis of the figures represents the maximum test set cardinality, whereas the y axis depicts the MAE. The dot, star, diamond and traingle symbols denote LAF, DeepSets, PNA and LSTM respectively.
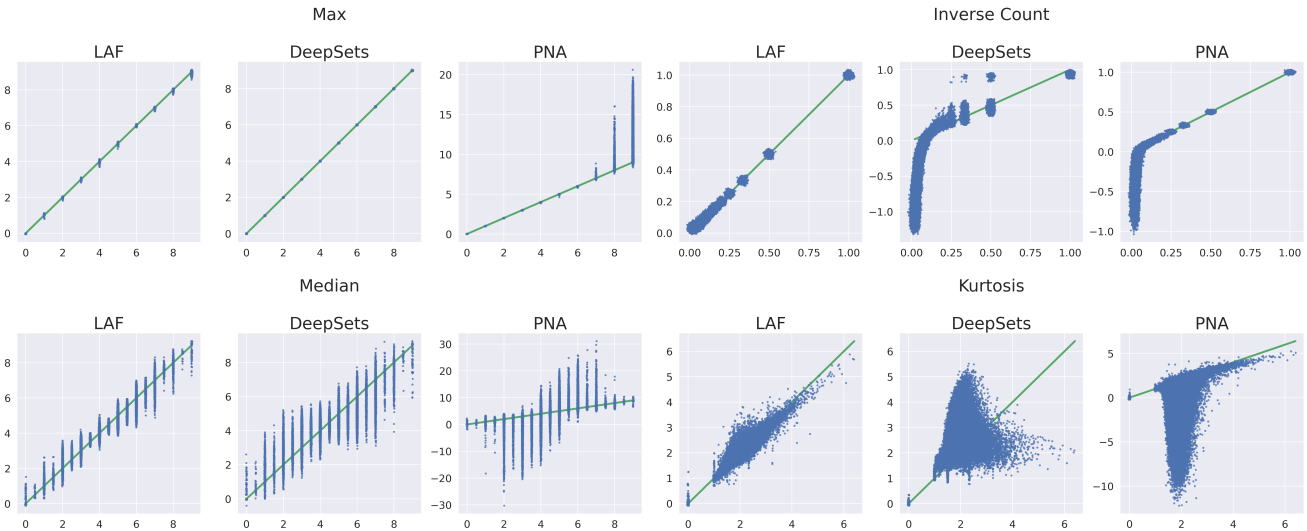

Figure D.2: Scatter plots of the MNIST experiment comparing true (x axis) and predicted (y axis) values with 50 as maximum test set size. The target aggregations are *max* (up-left), *inverse count* (up-right), *median* (bottom-left) and *kurtosis* (bottom-right).

achieved in the scalar setting, indicating that LAF is capable of effectively backpropagating information so as to drive the learning of an appropriate latent representation, while DeepSets, PNA, and LSTM suffer from the same problems seen in aggregating scalars.

Furthermore, Figure D.2 provides a qualitative evaluation of the predictions of the LAF, DeepSets, and PNA methods on a representative subset of the target aggregators. The images illustrate the correlation between the true labels and the predictions. LAF predictions are distributed over the diagonal line, with no clear bias. On the other hand, DeepSets and PNA perform generally worse than LAF, exhibiting higher variances. In particular, for inverse count and kurtosis,

DeepSets and PNA predictions are condensed in a specific area, suggesting an overfitting on the training set.

E   Details of Sections SetTransformer with LAF aggregation

We used mini-batches of 64 sets and trained the models for 1,000 epochs. We use Adam as parameter optimizer, setting the initial learning rate to $5e^{-4}$. Each element in the dataset is a set of vectors $\boldsymbol{x} = \{x_1, \ldots, x_N\}$, $x_i \in \mathbb{R}^{784}$. Network

Figure E.1: Distribution of the predicted values for ST-PMA and ST-LAF by set cardinalities. On the x-axis the true labels of the sets, on the y-axis the predicted ones. Different colors represent the sets' cardinalities $|\boldsymbol{x}|$.

architecture:

$$\boldsymbol{x} \to \text{DENSE}(784,300) \to \text{RELU}$$
$$\to \text{DENSE}(300,100) \to \text{RELU}$$
$$\to \text{DENSE}(100,30) \to \text{SIGMOD}$$
$$\to \text{SAB}(64,4) \to \text{SAB}(64,4)$$
$$\to \text{PMA}_k(64,4) \text{ OR } \text{LAF}(10)$$
$$\to \text{DENSE}(64 \times k \text{ OR } 9, 100) \to \text{RELU}$$
$$\to \text{DENSE}(100,1)$$

Please refer to [Lee *et al.*, 2019] for the SAB and PMA details. Figure E.1 shows the comparison of ST-PMA and ST-LAF for unique sum of MNIST images.