Learning Aggregation Functions – Supplementary material

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

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

[*]Equal Contribution. Contact Authors.

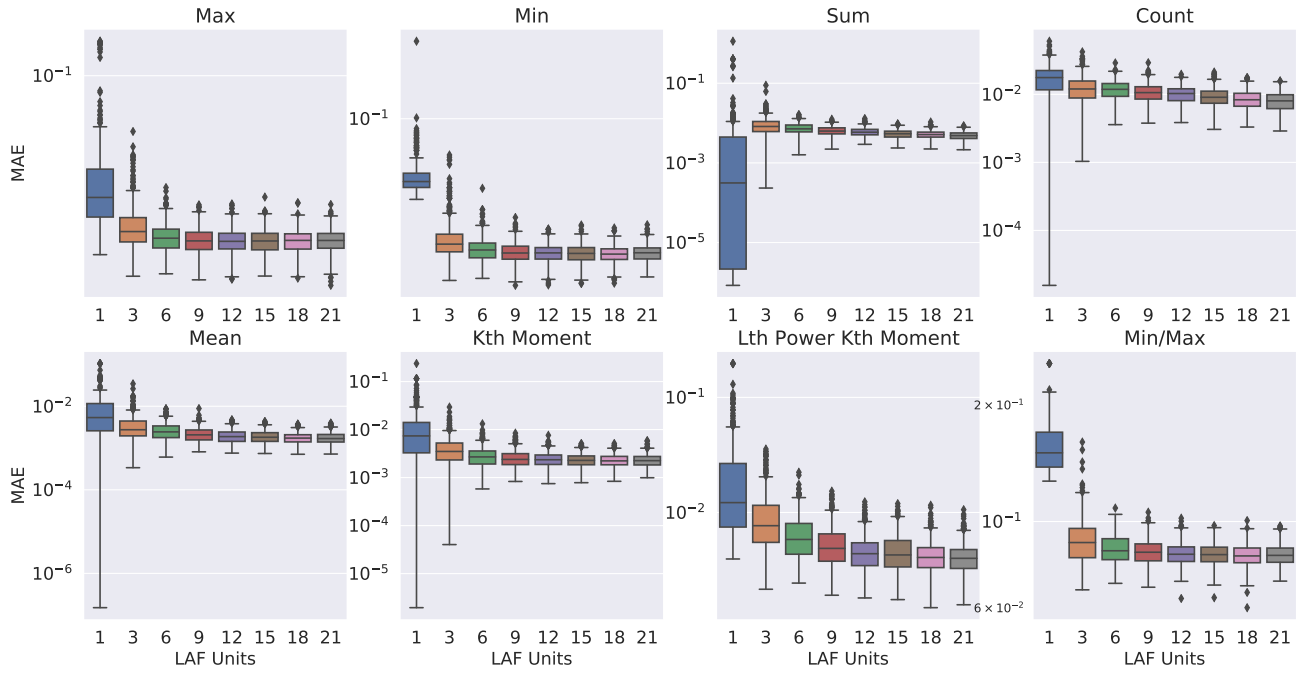

Figure B.1: Trend of the MAE obtained with an increasing number of LAF units for most of the functions reported in Table 1. The error distribution is obtained performing 500 runs with different random parameter initializations. A linear layer is stacked on top of the LAF layer with more than 1 unit. The y axis is plot in logaritmic scale.

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.

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:

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

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:

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

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.

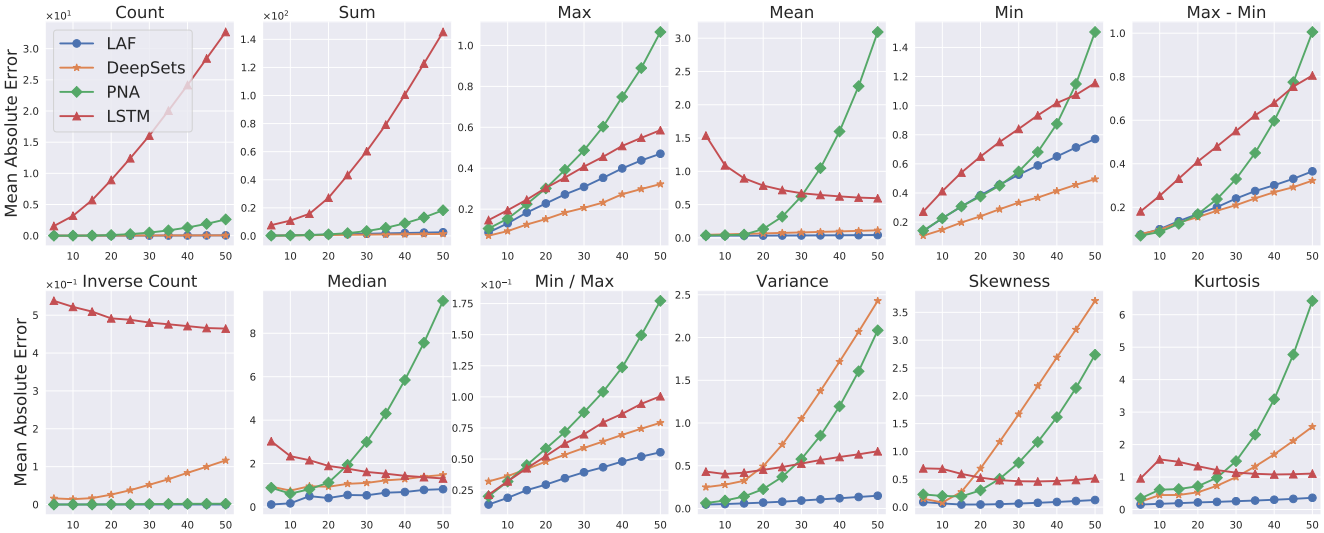

Figure D.1: Test performances for the synthetic experiment on MNIST digits on increasing test set size. The x axis of the figures represents the maximum test set cardinality, whereas the y axis depicts the MAE. The dot, star, diamond and traingle symbols denote LAF, DeepSets, PNA and LSTM respectively.

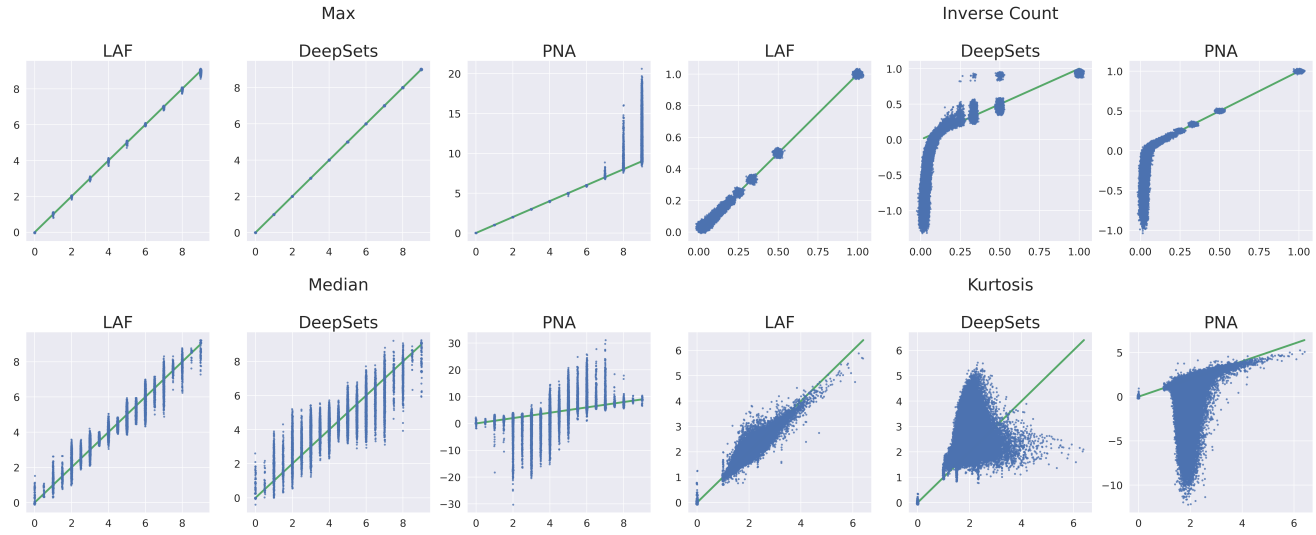

Figure D.2: Scatter plots of the MNIST experiment comparing true (x axis) and predicted (y axis) values with 50 as maximum test set size. The target aggregations are *max* (up-left), *inverse count* (up-right), *median* (bottom-left) and *kurtosis* (bottom-right).

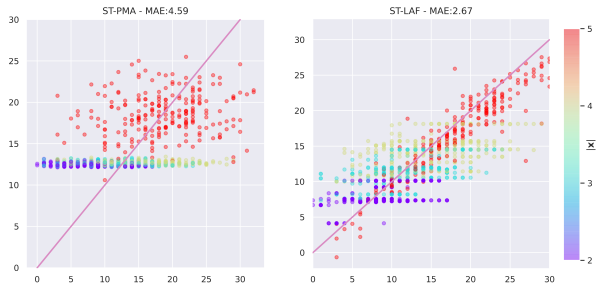

Figure E.1: Distribution of the predicted values for ST-PMA and ST-LAF by set cardinalities. On the x-axis the true labels of the sets, on the y-axis the predicted ones. Different colors represent the sets' cardinalities $|\boldsymbol{x}|$.

Figure D.1 shows the comparison of LAF, DeepSets, PNA, and LSTM in this setting. Results are quite similar to those 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 architecture:

$$
\begin{aligned}
\boldsymbol{x} &\rightarrow \textsc{Dense}(784{,}300) \rightarrow \textsc{ReLU} \\
&\rightarrow \textsc{Dense}(300{,}100) \rightarrow \textsc{ReLU} \\
&\rightarrow \textsc{Dense}(100{,}30) \rightarrow \textsc{Sigmod} \\
&\rightarrow \textsc{SAB}(64{,}4) \rightarrow \textsc{SAB}(64{,}4) \\
&\rightarrow \textsc{PMA}_k(64{,}4) \textsc{ or } \textsc{LAF}(10) \\
&\rightarrow \textsc{Dense}(64 \times k \textsc{ or } 9,\ 100) \rightarrow \textsc{ReLU} \\
&\rightarrow \textsc{Dense}(100{,}1)
\end{aligned}
$$

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.

References

[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.

[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.

[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.