Supplementary Material for Set Transformer

Juho Lee [1 2] **Yoonho Lee** [3] **Jungtaek Kim** [4] **Adam R. Kosiorek** [1 5] **Seungjin Choi** [4] **Yee Whye Teh** [1]

1. Proofs

Lemma 1. *The mean operator* $\mathrm{mean}(\{x_1, \dots, x_n\}) = \frac{1}{n}\sum_{i=1}^n x_i$ *is a special case of dot-product attention with softmax.*

Proof. Let $s = \mathbf{0} \in \mathbb{R}^d$ and $X \in \mathbb{R}^{n \times d}$.

$$\mathrm{Att}(s, X, X; \mathrm{softmax}) = \mathrm{softmax}\left(\frac{sX^\top}{\sqrt{d}}\right) X = \frac{1}{n}\sum_{i=1}^n x_i$$

□

Lemma 2. *The decoder of a Set Transformer, given enough nodes, can express any element-wise function of the form* $\left(\frac{1}{n}\sum_{i=1}^n z_i^p\right)^{\frac{1}{p}}$.

Proof. We first note that we can view the decoder as the composition of functions

$$\mathrm{Decoder}(Z) = \mathrm{rFF}(H) \tag{1}$$

$$\text{where } H = \mathrm{rFF}(\mathrm{MAB}(Z, \mathrm{rFF}(Z))) \tag{2}$$

We focus on H in (2). Since feed-forward networks are universal function approximators at the limit of infinite nodes, let the feed-forward layers in front and back of the MAB encode the element-wise functions $z \to z^p$ and $z \to z^{\frac{1}{p}}$, respectively. We let $h = d$, so the number of heads is the same as the dimensionality of the inputs, and each head is one-dimensional. Let the projection matrices in multi-head attention (W_j^Q, W_j^K, W_j^V) represent projections onto the jth dimension and the output matrix (W^O) the identity matrix. Since the mean operator is a special case of dot-product attention, by simple composition, we see that an MAB can express any dimension-wise function of the form

$$M_p(z_1, \cdots, z_n) = \left(\frac{1}{n}\sum_{i=1}^n z_i^p\right)^{\frac{1}{p}}. \tag{3}$$

□

Lemma 3. *A PMA, given enough nodes, can express sum pooling* $(\sum_{i=1}^n z_i)$.

Proof. We prove this by construction.

Set the seed s to a zero vector and let $\omega(\cdot) = 1 + f(\cdot)$, where f is any activation function such that $f(0) = 0$. The identiy, sigmoid, or relu functions are suitable choices for f. The output of the multihead attention is then simply a sum of the values, which is Z in this case. □

We additionally have the following universality theorem for pooling architectures:

Theorem 1. *Models of the form* $\mathrm{rFF}(\mathrm{sum}(\mathrm{rFF}(\cdot)))$ *are universal function approximators in the space of permutation invariant functions.*

Proof. See Appendix A of Zaheer et al. (2017). □

By Lemma 3, we know that $\mathrm{decoder}(Z)$ can express any function of the form $\mathrm{rFF}(\mathrm{sum}(Z))$. Using this fact along with Theorem 1, we can prove the universality of Set Transformers:

Proposition 1. *The Set Transformer is a universal function approximator in the space of permutation invariant functions.*

Proof. By setting the matrix W^O to a zero matrix in every SAB and ISAB, we can ignore all pairwise interaction terms in the encoder. Therefore, the $\mathrm{encoder}(X)$ can express any instance-wise feed-forward network ($Z = \mathrm{rFF}(X)$). Directly invoking Theorem 1 concludes this proof. □

While this proof required us to ignore the pairwise interaction terms inside the SABs and ISABs to prove that Set Transformers are universal function approximators, our experiments indicated that self-attention in the encoder was crucial for good performance.

2. Experiment Details

In all implementations, we omit the feed-forward layer in the beginning of the decoder ($\mathrm{rFF}(Z)$) because the end of the previous block contains a feed-forward layer. All MABs (inside SAB, ISAB and PMA) use fully-connected layers with ReLU activations for rFF layers.

In the architecture descriptions, $\mathrm{FC}(d, f)$ denotes the fully-connected layer with d units and activation function f. $\mathrm{SAB}(d, h)$ denotes the SAB with d units and h heads. $\mathrm{ISAB}_m(d, h)$ denotes the ISAB with d units, h heads and m inducing points. $\mathrm{PMA}_k(d, h)$ denotes the PMA with d units, h heads and k vectors. All MABs used in SAB and PMA uses FC layers with ReLU activations for FF layers.

2.1. Max Regression

Given a set of real numbers $\{x_1, \ldots, x_n\}$, the goal of this task is to return the maximum value in the set $\max(x_1, \cdots, x_n)$. We construct training data as follows. We first sample a dataset size n uniformly from the set of integers $\{1, \cdots, 10\}$. We then sample real numbers x_i independently from the interval $[0, 100]$. Given the network's prediction p, we use the actual maximum value $\max(x_1, \cdots, x_n)$ to compute the mean absolute error $|p - \max(x_1, \cdots, x_n)|$. We don't explicitly consider splits of train and test data, since we sample a new set $\{x_1, \ldots, x_n\}$ at each time step.

Table 1. Detailed architectures used in the max regression experiments.

Encoder		Decoder	
FF	SAB	Pooling	PMA
$\mathrm{FC}(64, \mathrm{ReLU})$	$\mathrm{SAB}(64, 4)$	$\mathrm{mean}, \mathrm{sum}, \max$	$\mathrm{PMA}_1(64, 4)$
$\mathrm{FC}(64, \mathrm{ReLU})$	$\mathrm{SAB}(64, 4)$	$\mathrm{FC}(64, \mathrm{ReLU})$	$\mathrm{FC}(1, -)$
$\mathrm{FC}(64, \mathrm{ReLU})$		$\mathrm{FC}(1, -)$	
$\mathrm{FC}(64, -)$			

We show the detailed architectures used for the experiments in Table 1. We trained all networks using the Adam optimizer (Kingma & Ba, 2015) with a constant learning rate of 10^{-3} and a batch size of 128 for 20,000 batches, after which loss converged for all architectures.

2.2. Counting Unique Characters

The task generation procedure is as follows. We first sample a set size n uniformly from the set of integers $\{6, \ldots, 10\}$. We then sample the number of characters c uniformly from $\{1, \ldots, n\}$. We sample c characters from the training set of characters, and randomly sample instances of each character so that the total number of instances sums to n and each set of characters has at least one instance in the resulting set.

We show the detailed architectures used for the experiments in Table 3. For both architectures, the resulting 1-dimensional output is passed through a $\mathrm{softplus}$ activation to produce the Poisson parameter γ. The role of $\mathrm{softplus}$ is to ensure that γ is always positive.

Table 2. Detailed results for the unique character counting experiment.

Architecture	Accuracy
rFF + Pooling	0.4366 ± 0.0071
rFF + PMA	0.4617 ± 0.0073
rFFp-mean + Pooling	0.4617 ± 0.0076
rFFp-max + Pooling	0.4359 ± 0.0077
rFF + Dotprod	0.4471 ± 0.0076
SAB + Pooling	0.5659 ± 0.0067
SAB + Dotprod	0.5888 ± 0.0072
SAB + PMA (1)	$\mathbf{0.6037 \pm 0.0072}$
SAB + PMA (2)	0.5806 ± 0.0075
SAB + PMA (4)	0.5945 ± 0.0072
SAB + PMA (8)	0.6001 ± 0.0078

Table 3. Detailed architectures used in the unique character counting experiments.

Encoder		Decoder	
rFF	SAB	Pooling	PMA
$\mathrm{Conv}(64, 3, 2, \mathrm{BN}, \mathrm{ReLU})$	$\mathrm{Conv}(64, 3, 2, \mathrm{BN}, \mathrm{ReLU})$	mean	$\mathrm{PMA}_1(8, 8)$
$\mathrm{Conv}(64, 3, 2, \mathrm{BN}, \mathrm{ReLU})$	$\mathrm{Conv}(64, 3, 2, \mathrm{BN}, \mathrm{ReLU})$	$\mathrm{FC}(64, \mathrm{ReLU})$	$\mathrm{FC}(1, \mathrm{softplus})$
$\mathrm{Conv}(64, 3, 2, \mathrm{BN}, \mathrm{ReLU})$	$\mathrm{Conv}(64, 3, 2, \mathrm{BN}, \mathrm{ReLU})$	$\mathrm{FC}(1, \mathrm{softplus})$	
$\mathrm{Conv}(64, 3, 2, \mathrm{BN}, \mathrm{ReLU})$	$\mathrm{Conv}(64, 3, 2, \mathrm{BN}, \mathrm{ReLU})$		
$\mathrm{FC}(64, \mathrm{ReLU})$	$\mathrm{SAB}(64, 4)$		
$\mathrm{FC}(64, \mathrm{ReLU})$	$\mathrm{SAB}(64, 4)$		
$\mathrm{FC}(64, \mathrm{ReLU})$			
$\mathrm{FC}(64, -)$			

The loss function we optimize, as previously mentioned, is the log likelihood $\log p(x|\gamma) = x \log(\gamma) - \gamma - \log(x!)$. We chose this loss function over mean squared error or mean absolute error because it seemed like the more logical choice when trying to make a real number match a target integer. Early experiments showed that directly optimizing for mean absolute error had roughly the same result as optimizing γ in this way and measuring $|\gamma - x|$. We train using the Adam optimizer with a constant learning rate of 10^{-4} for 200,000 batches each with batch size 32.

2.3. Solving maximum likelihood problems for mixture of Gaussians

2.3.1. Details for 2D synthetic mixtures of Gaussians experiment

We generated the datasets according to the following generative process.

1. Generate the number of data points, $n \sim \mathrm{Unif}(100, 500)$.

2. Generate k centers.

$$\mu_{j,d} \sim \mathrm{Unif}(-4, 4), \quad j = 1, \ldots, 4, \;\; d = 1, 2. \tag{4}$$

3. Generate cluster labels.

$$\pi \sim \mathrm{Dir}([1, 1]^\top), \quad z_i \sim \mathrm{Categorical}(\pi), \;\; i = 1, \ldots, n. \tag{5}$$

4. Generate data from spherical Gaussian.

$$x_i \sim \mathcal{N}(\mu_{z_i}, (0.3)^2 I). \tag{6}$$

Table 4 summarizes the architectures used for the experiments. For all architectures, at each training step, we generate 10 random datasets according to the above generative process, and updated the parameters via Adam optimizer with initial learning rate 10^{-3}. We trained all the algorithms for $50k$ steps, and decayed the learning rate to 10^{-4} after $35k$ steps. Table 5 summarizes the detailed results with various number of inducing points in the ISAB. Figure 4 shows the actual clustering results based on the predicted parameters.

Table 4. Detailed architectures used in 2D synthetic experiments.

Encoder			Decoder	
rFF	SAB	ISAB	Pooling	PMA
$\mathrm{FC}(128, \mathrm{ReLU})$	$\mathrm{SAB}(128, 4)$	$\mathrm{ISAB}_m(128, 4)$	mean	$\mathrm{PMA}_4(128, 4)$
$\mathrm{FC}(128, \mathrm{ReLU})$	$\mathrm{SAB}(128, 4)$	$\mathrm{ISAB}_m(128, 4)$	$\mathrm{FC}(128, \mathrm{ReLU})$	$\mathrm{SAB}(128, 4)$
$\mathrm{FC}(128, \mathrm{ReLU})$			$\mathrm{FC}(128, \mathrm{ReLU})$	$\mathrm{FC}(4 \cdot (1 + 2 \cdot 2), -)$
$\mathrm{FC}(128, \mathrm{ReLU})$			$\mathrm{FC}(128, \mathrm{ReLU})$	
			$\mathrm{FC}(4 \cdot (1 + 2 \cdot 2), -)$	

Table 5. Average log-likelihood/data (LL0/data) and average log-likelihood/data after single EM iteration (LL1/data) the clustering experiment. The number inside parenthesis indicates the number of inducing points used in the SABs of encoder. For all PMAs, four seed vectors were used.

Architecture	LL0/data	LL1/data
Oracle	-1.4726	
rFF + Pooling	-2.0006 $\pm$ 0.0123	-1.6186 $\pm$ 0.0042
rFFp-mean + Pooling	-1.7606 $\pm$ 0.0213	-1.5191 $\pm$ 0.0026
rFFp-max + Pooling	-1.7692 $\pm$ 0.0130	-1.5103 $\pm$ 0.0035
rFF+Dotprod	-1.8549 $\pm$ 0.0128	-1.5621 $\pm$ 0.0046
SAB + Pooling	-1.6772 $\pm$ 0.0066	-1.5070 $\pm$ 0.0115
ISAB (16) + Pooling	-1.6955 $\pm$ 0.0730	-1.4742 $\pm$ 0.0158
ISAB (32) + Pooling	-1.6353 $\pm$ 0.0182	-1.4681 $\pm$ 0.0038
ISAB (64) + Pooling	-1.6349 $\pm$ 0.0429	-1.4664 $\pm$ 0.0080
rFF + PMA	-1.6680 $\pm$ 0.0040	-1.5409 $\pm$ 0.0037
SAB + PMA	-1.5145 $\pm$ 0.0046	-1.4619 $\pm$ 0.0048
ISAB (16) + PMA	-1.5009 $\pm$ 0.0068	-1.4530 $\pm$ 0.0037
ISAB (32) + PMA	**-1.4963 $\pm$ 0.0064**	**-1.4524 $\pm$ 0.0044**
ISAB (64) + PMA	-1.5042 $\pm$ 0.0158	-1.4535 $\pm$ 0.0053

2.3.2. 2D Synthetic Mixtures of Gaussians Experiment on Large-scale Data

To show the scalability of the set transformer, we conducted additional experiments on large-scale 2D synthetic clustering dataset. We generated the synthetic data as before, except that we sample the number of data points n $\mathrm{Unif}(1000, 5000)$ and set $k = 6$. We report the clustering accuracy of a subset of comparing methods in Table 6. The set transformer with only 32 inducing points works extremely well, demonstrating its scalability and efficiency.

2.3.3. Details for CIFAR-100 amortized clutering experiment

We pretrained VGG net (Simonyan & Zisserman, 2014) with CIFAR-100, and obtained the test accuracy 68.54%. Then, we extracted feature vectors of 50k training images of CIFAR-100 from the 512-dimensional hidden layers of the VGG net (the layer just before the last layer). Given these feature vectors, the generative process of datasets is as follows.

1. Generate the number of data points, $n \sim \mathrm{Unif}(100, 500)$.
2. Uniformly sample four classes among 100 classes.
3. Uniformly sample n data points among four sampled classes.

Table 6. Average log-likelihood/data (LL0/data) and average log-likelihood/data after single EM iteration (LL1/data) the clustering experiment on large-scale data. The number inside parenthesis indicates the number of inducing points used in the SABs of encoder. For all PMAs, six seed vectors were used.

Architecture	LL0/data	LL1/data
Oracle	-1.8202	
rFF + Pooling	-2.5195 $\pm$ 0.0105	-2.0709 $\pm$ 0.0062
rFFp-mean + Pooling	-2.3126 $\pm$ 0.0154	-1.9749 $\pm$ 0.0062
rFF + PMA (6)	-2.0515 $\pm$ 0.0067	-1.9424 $\pm$ 0.0047
SAB (32) + PMA (6)	**-1.8928** $\pm$ **0.0076**	**-1.8549** $\pm$ **0.0024**

Table 7. Detailed architectures used in CIFAR-100 meta clustering experiments.

Encoder			Decoder	
rFF	SAB	ISAB	rFF	PMA
$\mathrm{FC}(256, \mathrm{ReLU})$	$\mathrm{SAB}(256, 4)$	$\mathrm{ISAB}_m(256, 4)$	mean	$\mathrm{PMA}_4(128, 4)$
$\mathrm{FC}(256, \mathrm{ReLU})$	$\mathrm{SAB}(256, 4)$	$\mathrm{ISAB}_m(256, 4)$	$\mathrm{FC}(256, \mathrm{ReLU})$	$\mathrm{SAB}(256, 4)$
$\mathrm{FC}(256, \mathrm{ReLU})$	$\mathrm{SAB}(256, 4)$	$\mathrm{ISAB}_m(256, 4)$	$\mathrm{FC}(256, \mathrm{ReLU})$	$\mathrm{SAB}(256, 4)$
$\mathrm{FC}(256, \mathrm{ReLU})$			$\mathrm{FC}(256, \mathrm{ReLU}))$	$\mathrm{FC}(4 \cdot (1 + 2 \cdot 512), -)$
$\mathrm{FC}(256, \mathrm{ReLU})$			$\mathrm{FC}(256, \mathrm{ReLU})$	
$\mathrm{FC}(256, -)$			$\mathrm{FC}(256, \mathrm{ReLU})$	
			$\mathrm{FC}(4 \cdot (1 + 2 \cdot 512), -)$	

Table 7 summarizes the architectures used for the experiments. For all architectures, at each training step, we generate 10 random datasets according to the above generative process, and updated the parameters via Adam optimizer with initial learning rate 10^{-4}. We trained all the algorithms for $50k$ steps, and decayed the learning rate to 10^{-5} after $35k$ steps. Table 8 summarizes the detailed results with various number of inducing points in the ISAB.

2.4. Set Anomaly Detection

Table 9 describes the architecture for meta set anomaly experiments. We trained all models via Adam optimizer with learning rate 10^{-4} and exponential decay of learning rate for 1,000 iterations. 1,000 datasets subsampled from CelebA dataset (see Figure 5) are used to train and test all the methods. We split 800 training datasets and 200 test datasets for the subsampled datasets.

2.5. Point Cloud Classification

We used the ModelNet40 dataset for our point cloud classification experiments. This dataset consists of a three-dimensional representation of 9,843 training and 2,468 test data which each belong to one of 40 object classes. As input to our architectures, we produce point clouds with $n = 100, 1000, 5000$ points each (each point is represented by (x, y, z) coordinates). For generalization, we randomly rotate and scale each set during training.

We show results our architectures in Table 10 and additional experiments which used $n = 100, 5000$ points in Table 4. We trained using the Adam optimizer with an initial learning rate of 10^{-3} which we decayed by a factor of 0.3 every 20,000 steps. For the experiment with 5,000 points (Table 4), we increased the dimension of the attention blocks ($\mathrm{ISAB}_{16}(512, 4)$ instead of $\mathrm{ISAB}_{16}(128, 4)$) and also decayed the weights by a factor of 10^{-7}. We also only used one ISAB block in the encoder because using two lead to overfitting in this setting.

3. Additional Experiments

3.1. Runtime of SAB and ISAB

We measured the runtime of SAB and ISAB on a simple benchmark (Figure 1). We used a single GPU (Tesla P40) for this experiment. The input data was a constant (zero) tensor of n three-dimensional vectors. We report the number of seconds it

Table 8. Average clustering accuracies measured by Adjusted Rand Index (ARI) for CIFAR100 clustering experiments. The number inside parenthesis indicates the number of inducing points used in the SABs of encoder. For all PMAs, four seed vectors were used.

Architecture	ARI0	ARI1
Oracle	0.9151	
rFF + Pooling	0.5593 ± 0.0149	0.5693 ± 0.0171
rFFp-mean + Pooling	0.5673 ± 0.0053	0.5798 ± 0.0058
rFFp-max + Pooling	0.5369 ± 0.0154	0.5536 ± 0.0186
rFF+Dotprod	0.5666 ± 0.0221	0.5763 ± 0.0212
SAB + Pooling	0.5831 ± 0.0341	0.5943 ± 0.0337
ISAB (16) + Pooling	0.5672 ± 0.0124	0.5805 ± 0.0122
ISAB (32) + Pooling	0.5587 ± 0.0104	0.5700 ± 0.0134
ISAB (64) + Pooling	0.5586 ± 0.0205	0.5708 ± 0.0183
rFF + PMA	0.7612 ± 0.0237	0.7670 ± 0.0231
SAB + PMA	0.9015 ± 0.0097	0.9024 ± 0.0097
ISAB (16) + PMA	$\mathbf{0.9210 \pm 0.0055}$	$\mathbf{0.9223 \pm 0.0056}$
ISAB (32) + PMA	0.9103 ± 0.0061	0.9119 ± 0.0052
ISAB (64) + PMA	0.9141 ± 0.0040	0.9153 ± 0.0041

Table 9. Detailed architectures used in CelebA meta set anomaly experiments. $\mathrm{Conv}(d, k, s, r, f)$ is a convolutional layer with d output channels, k kernel size, s stride size, r regularization method, and activation function f. If d is a list, each element in the list is distributed. $\mathrm{FC}(d, f, r)$ denotes a fully-connected layer with d units, activation function f and r regularization method. If d is a list, each element in the list is distributed. $\mathrm{SAB}(d, h)$ denotes the SAB with d units and h heads. $\mathrm{PMA}(d, h, n_{\mathrm{seed}})$ denotes the PMA with d units, h heads and n_{seed} vectors. All MABs used in SAB and PMA uses FC layers with ReLU activations for rFF layers.

Encoder		Decoder	
rFF	SAB	Pooling	PMA
$\mathrm{Conv}([32, 64, 128], 3, 2, \mathrm{Dropout}, \mathrm{ReLU})$		mean	$\mathrm{PMA}_4(128, 4)$
$\mathrm{FC}([1024, 512, 256], -, \mathrm{Dropout})$		$\mathrm{FC}(128, \mathrm{ReLU}, -)$	$\mathrm{SAB}(128, 4)$
$\mathrm{FC}(256, -, -)$		$\mathrm{FC}(128, \mathrm{ReLU}, -)$	$\mathrm{FC}(256 \cdot 8, -, -)$
$\mathrm{FC}([128, 128, 128], \mathrm{ReLU}, -)$	$\mathrm{SAB}(128, 4)$	$\mathrm{FC}(128, \mathrm{ReLU}, -)$	
$\mathrm{FC}([128, 128, 128], \mathrm{ReLU}, -)$	$\mathrm{SAB}(128, 4)$	$\mathrm{FC}(256 \cdot 8, -, -)$	
$\mathrm{FC}(128, \mathrm{ReLU}, -)$	$\mathrm{SAB}(128, 4)$		
$\mathrm{FC}(128, -, -)$	$\mathrm{SAB}(128, 4)$		

took to process 10,000 sets of each size. The maximum set size we report for SAB is 2,000 because the computation graph of bigger sets could not fit on our GPU. The specific attention blocks used are $\mathrm{ISAB}_4(64, 8)$ and $\mathrm{SAB}(64, 8)$.

References

Kingma, D. P. and Ba, Jimmy, L. Adam: a method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv e-prints*, arXiv:1409.1556, 2014.

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

Table 10. Detailed architectures used in the point cloud classification experiments.

Encoder		Decoder	
rFF	ISAB	Pooling	PMA
$\mathrm{FC}(256, \mathrm{ReLU})$ $\mathrm{FC}(256, \mathrm{ReLU})$ $\mathrm{FC}(256, \mathrm{ReLU})$ $\mathrm{FC}(256, -)$	$\mathrm{ISAB}(256, 4)$ $\mathrm{ISAB}(256, 4)$	max $\mathrm{Dropout}(0.5)$ $\mathrm{FC}(256, \mathrm{ReLU})$ $\mathrm{Dropout}(0.5)$ $\mathrm{FC}(40, -)$	$\mathrm{Dropout}(0.5)$ $\mathrm{PMA}_1(256, 4)$ $\mathrm{Dropout}(0.5)$ $\mathrm{FC}(40, -)$

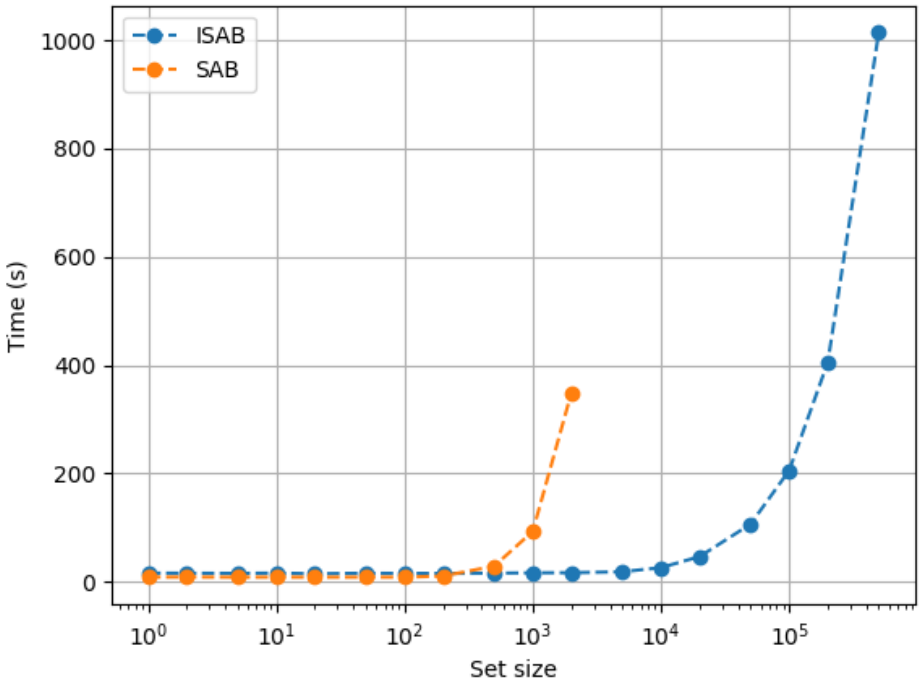

Figure 1. Runtime of a single SAB/ISAB block on dummy data. x axis is the size of the input set and y axis is time (seconds). Note that the x-axis is log-scale.