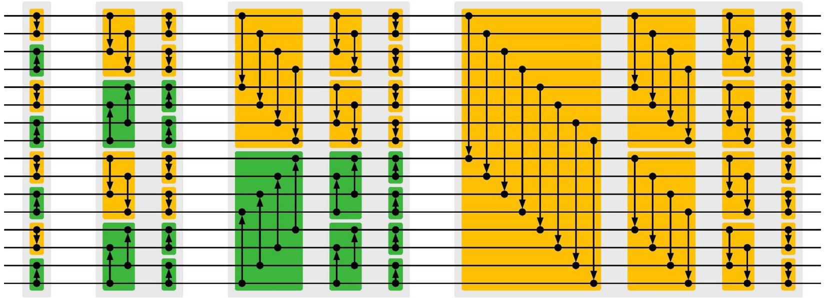


diffsort

Differentiable Sorting Networks for Scalable Sorting and Ranking Supervision

Felix Petersen, Christian Borgelt, Hilde Kuehne, Oliver Deussen

University of Konstanz, University of Salzburg,
University of Frankfurt, MIT-IBM Watson AI

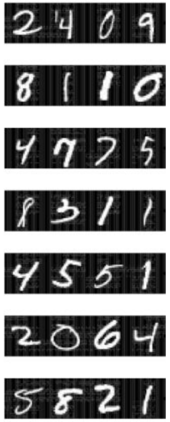
diffsort

Overview

- Sorting Supervision enables training NNs based on ordering constraints.
- Sorting Networks are sorting algorithms with a fixed execution structure requiring only *min* and *max* operations.
- We continuously relax sorting networks to Differentiable Sorting Networks via *softmin* and *softmax*.
- The Activation Replacement Trick (ART) that maps activations to regions with moderate gradients.
- Our method is scalable to differentially sorting 1024 elements.

diffsort

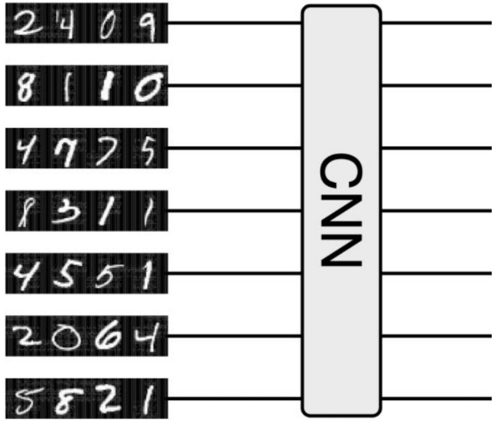
Sorting and Ranking Supervision



$$Q = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

- Images and their underlying order **Q** are given.
- The value of each image is predicted by a CNN.
- The predictions are differentially sorted.
- Supervision by enforcing that the differentiable permutation matrix **P** equals **Q**.

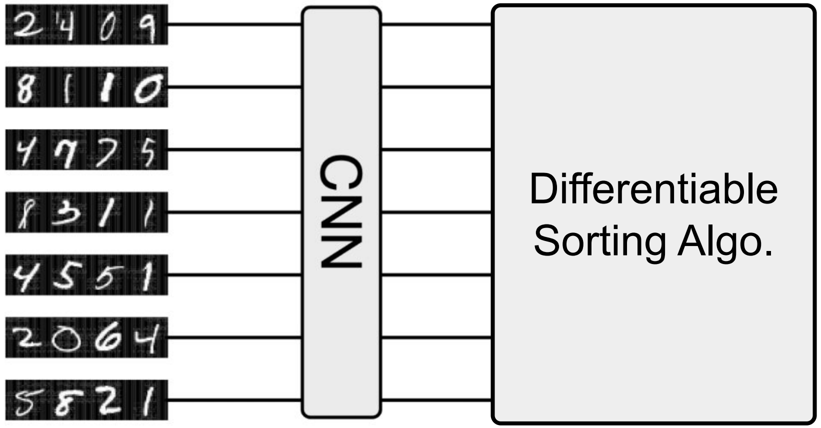
Sorting and Ranking Supervision



$$
\boldsymbol{Q} = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0
\end{pmatrix}
$$

- Images and their underlying order **Q** are given.
- The value of each image is predicted by a CNN.
- The predictions are differentially sorted.
- Supervision by enforcing that the differentiable permutation matrix **P** equals **Q**.

Sorting and Ranking Supervision

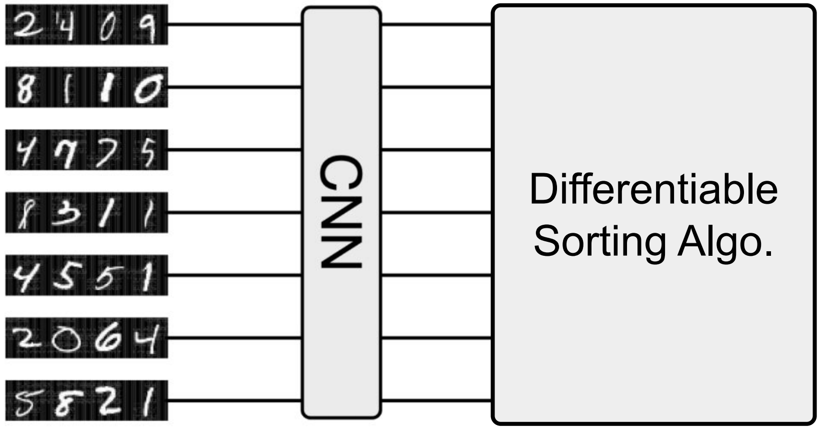


$$P = \begin{pmatrix} .4 & .0 & .1 & .0 & .0 & .4 & .0 \\ .5 & .0 & .1 & .0 & .0 & .4 & .0 \\ .1 & .0 & .4 & .0 & .3 & .1 & .1 \\ .0 & .0 & .3 & .0 & .3 & .1 & .2 \\ .0 & .1 & .1 & .1 & .2 & .0 & .5 \\ .0 & .5 & .0 & .4 & .0 & .0 & .1 \\ .0 & .4 & .0 & .5 & .0 & .0 & .1 \end{pmatrix}$$

$$Q = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

- Images and their underlying order **Q** are given.
- The value of each image is predicted by a CNN.
- The predictions are differentially sorted.
- Supervision by enforcing that the differentiable permutation matrix **P** equals **Q**.

Sorting and Ranking Supervision



$$P = \begin{pmatrix} .4 & .0 & .1 & .0 & .0 & .4 & .0 \\ .5 & .0 & .1 & .0 & .0 & .4 & .0 \\ .1 & .0 & .4 & .0 & .3 & .1 & .1 \\ .0 & .0 & .3 & .0 & .3 & .1 & .2 \\ .0 & .1 & .1 & .1 & .2 & .0 & .5 \\ .0 & .5 & .0 & .4 & .0 & .0 & .1 \\ .0 & .4 & .0 & .5 & .0 & .0 & .1 \end{pmatrix}$$

$$Q = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

- Images and their underlying order **Q** are given.
- The value of each image is predicted by a CNN.
- The predictions are differentially sorted.
- Supervision by enforcing that the differentiable permutation matrix **P** equals **Q**.

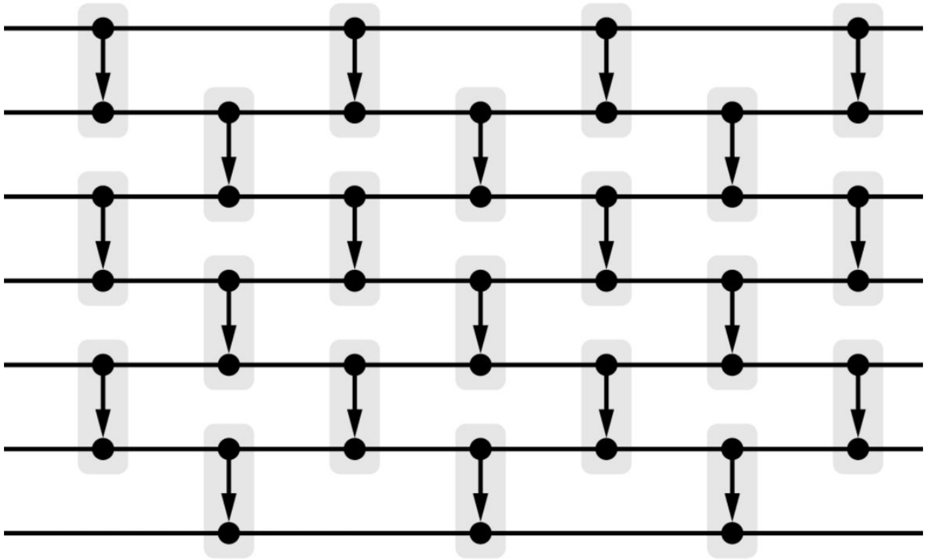
$$\mathcal{L} = \mathrm{CrossEntropy}(\boldsymbol{P}, \boldsymbol{Q})$$

Recent Differentiable Sorting Algorithms

- NeuralSort (Grover *et al.*, ICLR 2019)
- Optimal Transport Sort (Cuturi *et al.*, NeurIPS 2019)
- Fast Differentiable Sorting and Ranking (Blondel *et al.*, ICML 2020)

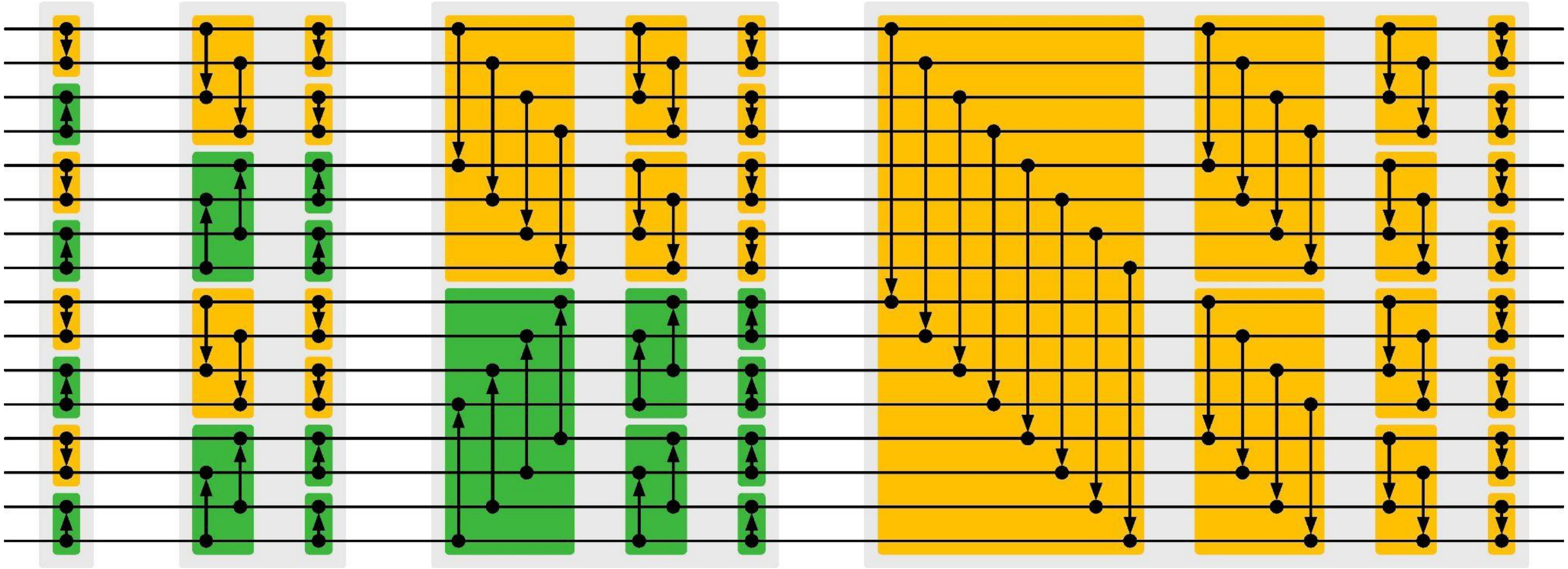
Sorting Networks

- Sorting via pairwise comparators
- Requires only *min* and *max* operations.

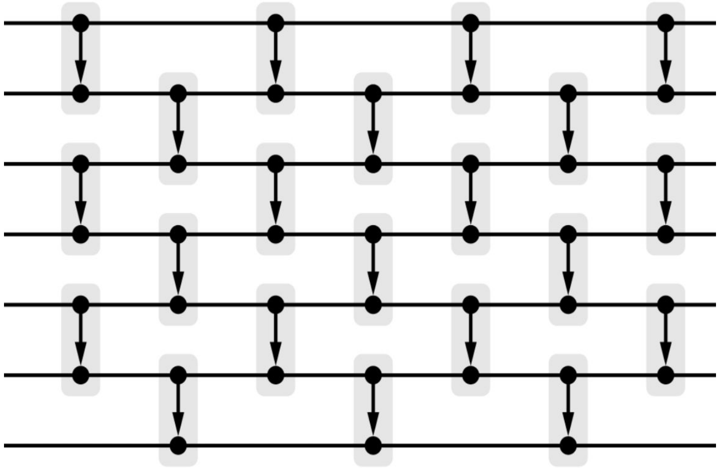


Sorting Networks

- Sorting via pairwise comparators
- Requires only *min* and *max* operations.



Differentiable Sorting Networks



diffsort

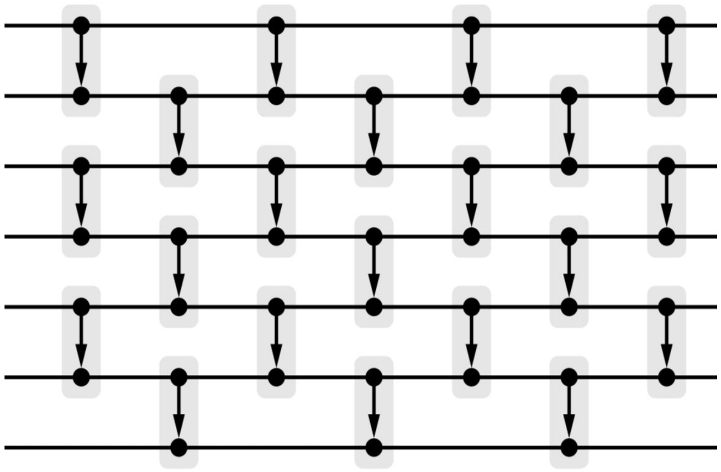
Differentiable Sorting Networks

- Relaxing the comparators

$$\mathrm{softmin}(a_i, a_j) := \alpha_{ij} \cdot a_i + (1 - \alpha_{ij}) \cdot a_j$$

$$\mathrm{softmax}(a_i, a_j) := (1 - \alpha_{ij}) \cdot a_i + \alpha_{ij} \cdot a_j$$

$$\alpha_{ij} := \sigma((a_j - a_i) \cdot s)$$

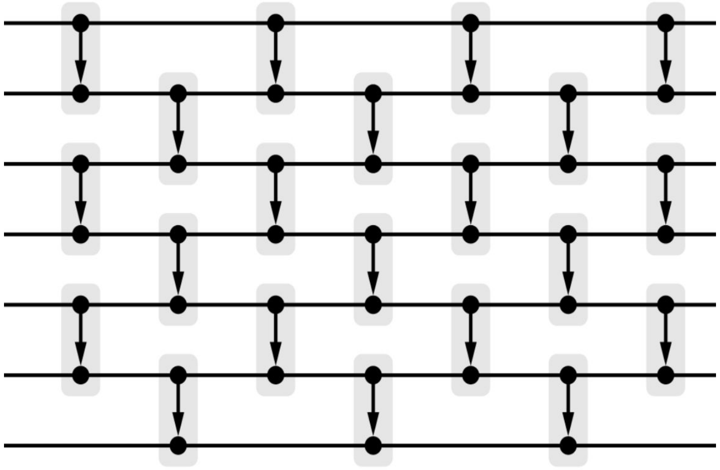


Differentiable Sorting Networks

- Relaxing the comparators

$$\mathrm{softmin}(a_i, a_j) := \alpha_{ij} \cdot a_i + (1 - \alpha_{ij}) \cdot a_j$$
$$\mathrm{softmax}(a_i, a_j) := (1 - \alpha_{ij}) \cdot a_i + \alpha_{ij} \cdot a_j$$
$$\alpha_{ij} := \sigma((a_j - a_i) \cdot s)$$

$$\boldsymbol{P} = P_n \cdot \ldots \cdot P_2 \cdot P_1 = \left(\prod_{l=1}^{n} P_l^\top \right)^\top$$



$P_1^\top \cdot P_2^\top \cdot P_3^\top \cdot P_4^\top \cdot P_5^\top \cdot P_6^\top \cdot P_7^\top$

Activation Replacement Trick

- For sorting large sets / very deep sorting networks:
    - Vanishing gradients
    - Extensive blurring

Activation Replacement Trick

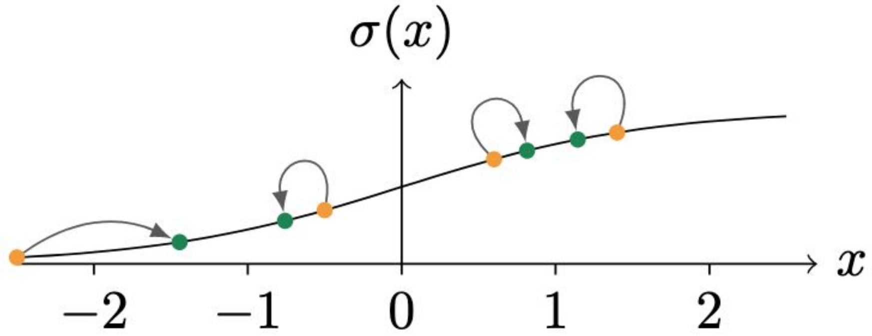
- For sorting large sets / very deep sorting networks:
  - Vanishing gradients
  - Extensive blurring
- To solve this, we map activations to regions with moderate gradients

$$\varphi : x \mapsto |x|^{1-\lambda} \cdot \mathrm{sgn}(x)$$

Activation Replacement Trick

- For sorting large sets / very deep sorting networks:
  - Vanishing gradients
  - Extensive blurring
- To solve this, we map activations to regions with moderate gradients

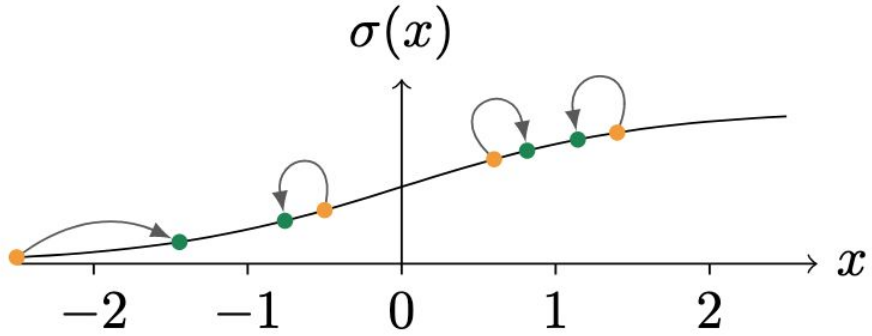
$$\varphi : x \mapsto |x|^{1-\lambda} \cdot \mathrm{sgn}(x)$$

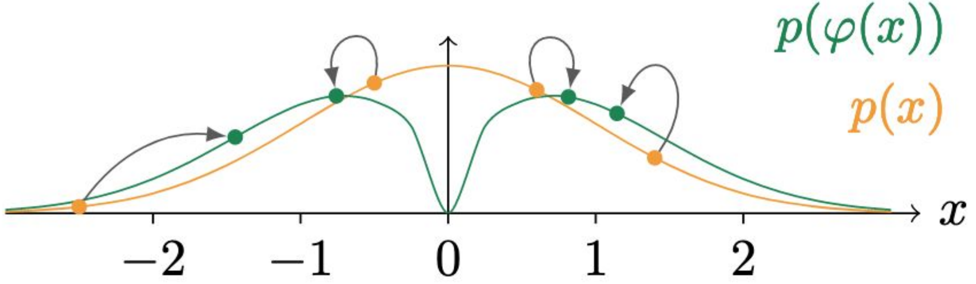


Activation Replacement Trick

- For sorting large sets / very deep sorting networks:
  - Vanishing gradients
  - Extensive blurring
- To solve this, we map activations to regions with moderate gradients

$$\varphi : x \mapsto |x|^{1-\lambda} \cdot \mathrm{sgn}(x)$$





Experimental Results

Experimental Results

Four-digit MNIST Sorting Benchmark

Method	$n=\mathbf{3}$			$n=\mathbf{5}$			$n=\mathbf{7}$			$n=\mathbf{9}$			$n=\mathbf{15}$		
Stoch. NeuralSort	92.0	94.6		79.0	90.7	79.0	63.6	87.3		45.2	82.9		12.2	73.4	
Det. NeuralSort	91.9	94.5		77.7	90.1	77.7	61.0	86.2		43.4	82.4		9.7	71.6	
Optimal Transport	92.8	95.0		81.1	91.7	81.1	65.6	88.2		49.7	84.7		12.6	74.2	
Fast Sort & Rank	90.6	93.5	73.5	71.5	87.2	71.5	49.7	81.3	70.5	29.0	75.2	69.2	2.8	60.9	67.4
Odd-Even	**95.2**	**96.7**	**86.1**	**86.3**	**93.8**	**86.3**	**75.4**	**91.2**	**86.4**	**64.3**	**89.0**	**86.7**	**35.4**	**83.7**	**87.6**

diffsort

Experimental Results

Four-digit MNIST Sorting Benchmark

Method	$n=\mathbf{3}$			$n=\mathbf{5}$			$n=\mathbf{7}$			$n=\mathbf{9}$			$n=\mathbf{15}$		
Stoch. NeuralSort	92.0	94.6		79.0	90.7	79.0	63.6	87.3		45.2	82.9		12.2	73.4	
Det. NeuralSort	91.9	94.5		77.7	90.1	77.7	61.0	86.2		43.4	82.4		9.7	71.6	
Optimal Transport	92.8	95.0		81.1	91.7	81.1	65.6	88.2		49.7	84.7		12.6	74.2	
Fast Sort & Rank	90.6	93.5	73.5	71.5	87.2	71.5	49.7	81.3	70.5	29.0	75.2	69.2	2.8	60.9	67.4
Odd-Even	**95.2**	**96.7**	**86.1**	**86.3**	**93.8**	**86.3**	**75.4**	**91.2**	**86.4**	**64.3**	**89.0**	**86.7**	**35.4**	**83.7**	**87.6**

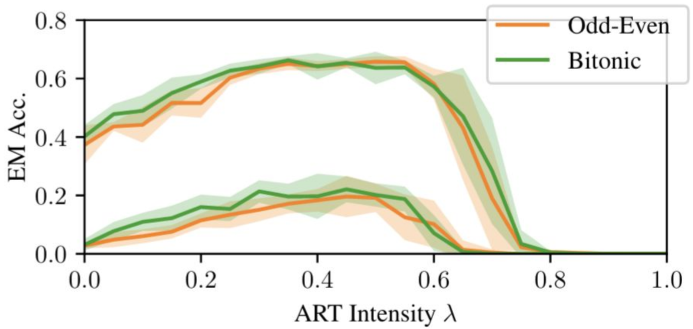
SVHN Sorting Benchmark

Method	$n=2$			$n=4$			$n=8$			$n=16$			$n=32$		
Det. NeuralSort	90.1	90.1	39.9	61.4	78.1	45.4	15.7	62.3	48.5	0.1	45.7	51.0	0.0	29.9	52.7
Optimal Transport	85.5	85.5	25.9	57.6	75.6	41.6	19.9	64.5	51.7	0.3	47.7	53.8	0.0	29.4	53.3
Fast Sort & Rank	93.4	93.4	57.6	58.0	75.8	41.5	8.6	52.7	34.4	0.3	36.5	41.6	0.0	14.0	27.5
Odd-Even	93.4	93.4	58.0	**74.8**	**85.5**	**62.6**	35.2	73.5	63.9	1.8	54.4	62.3	0.0	36.6	62.6
Bitonic	**93.8**	**93.8**	**58.6**	74.4	85.3	62.1	**38.3**	**75.1**	**66.8**	**3.9**	**59.6**	**66.8**	0.0	**42.4**	**67.7**

Experimental Results

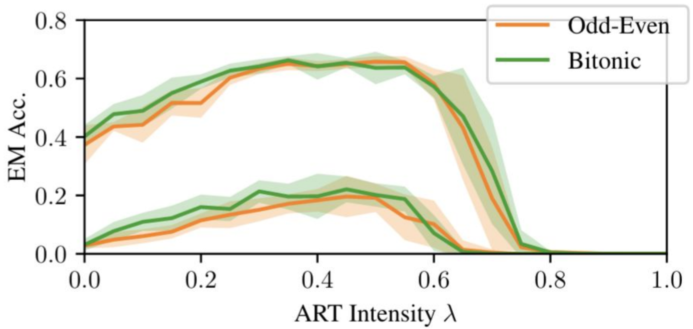
Experimental Results

The Activation Replacement Trick



Experimental Results

The Activation Replacement Trick



Large-Scale Four-digit MNIST Sorting Benchmark

n	32	64	128	256	512	1024
batch size	128	64	32	16	8	4
mean	80.29	80.89	81.28	81.03	82.24	81.36
best s	**80.97**	**81.66**	**82.50**	**82.05**	**82.67**	**82.80**
worst s	79.62	80.05	80.15	79.75	81.51	80.07

diffsort

Thank You!

Check out our code at https://github.com/Felix-Petersen/diffsort

Felix Petersen

University of Konstanz

@FHKPetersen

Petersen.ai

Universität Konstanz

MIT-IBM Watson AI Lab

PARIS LODRON UNIVERSITY SALZBURG

GOETHE UNIVERSITÄT FRANKFURT AM MAIN