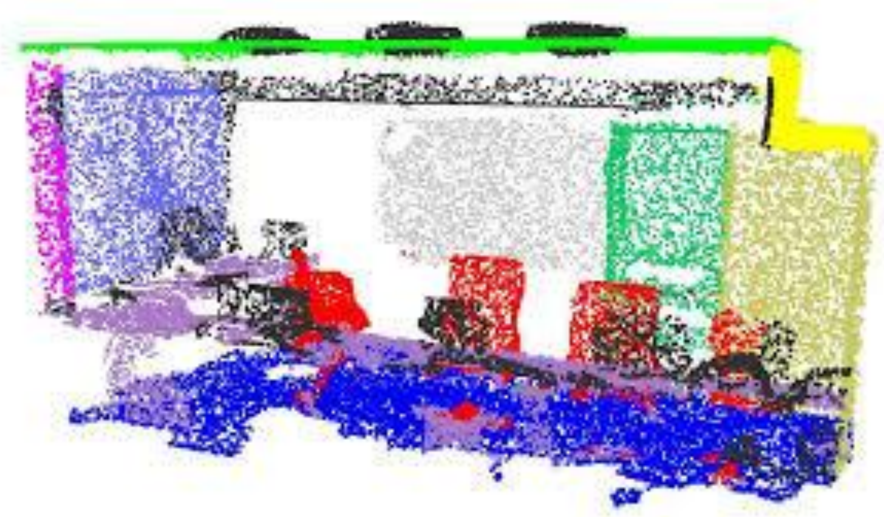
Set Models



Markus Chardonnet

SiDNN – 29.03.2022

Outline

- Sets and Multisets
- Motivations for set models
- Setup
    - Permutation Invariance
    - Permutation Equivariance
- Models
    - Equivariant models
    - Invariant models
- Conclusion

Sets and Multisets

Definition :

Example :

Set : Collection of different elements

$\{a, b, c\}$

Multiset : Collection of elements, where we allow for multiple instances of the same element

$\{a, a, b, a, b, c\}$

Sets and Multisets

Definition :

Set : Collection of different elements

Multiset : Collection of elements, where we allow for multiple instances of the same element

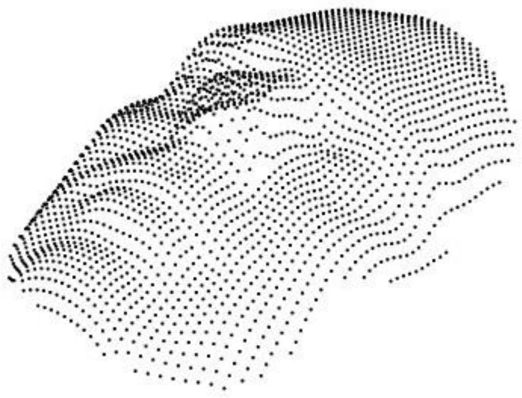
Example :

$\{a, b, c\}$

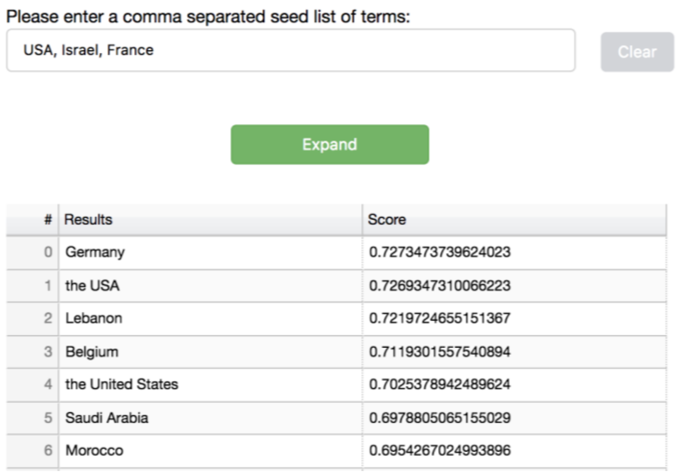
$\{a, a, b, a, b, c\}$

Motivation for set models

Point Clouds



Set expansion



Please enter a comma separated seed list of terms:

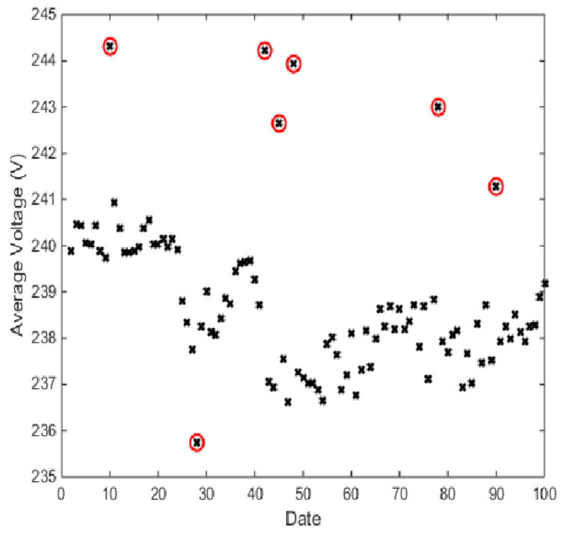
USA, Israel, France

Clear

Expand

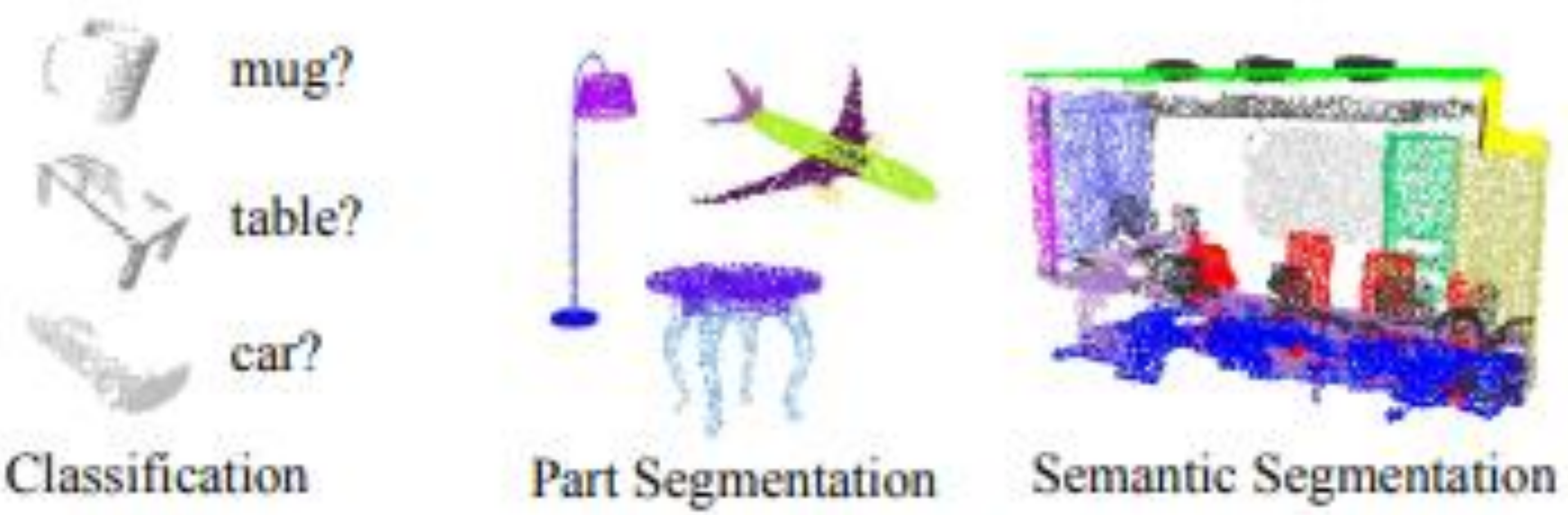
#	Results	Score
0	Germany	0.7273473739624023
1	the USA	0.7269347310066223
2	Lebanon	0.7219724655151367
3	Belgium	0.7119301557540894
4	the United States	0.7025378942489624
5	Saudi Arabia	0.6978805065155029
6	Morocco	0.6954267024993896

Outlier detection



Point Clouds

→ Set of point in space, representing 3D shape or object



Qi et al. – **PointNet : Deep Learning on Point Sets for 3D Classification and Segmentation -** CVPR 2017

Outlier Detection

Zaheer et al. – **Deep Sets -** NeurIPS 2017

What should a set model account for ?

→ The set can have varying number of elements

→ The order of the elements doesn’t count

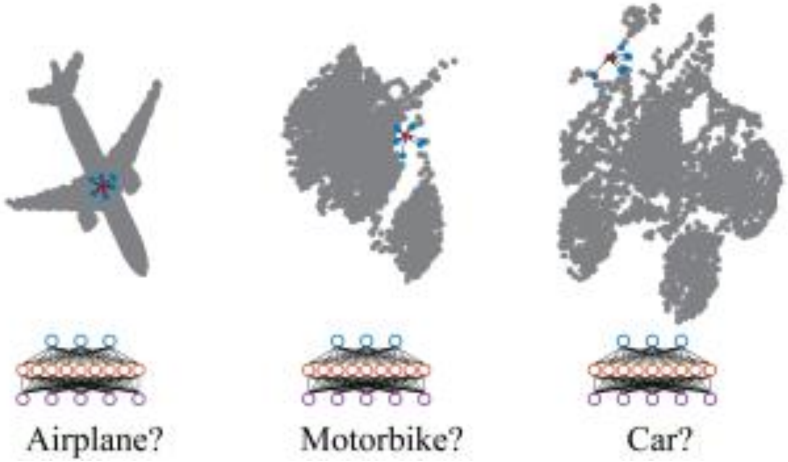
What should a set model account for ?

→ The set can have varying number of elements

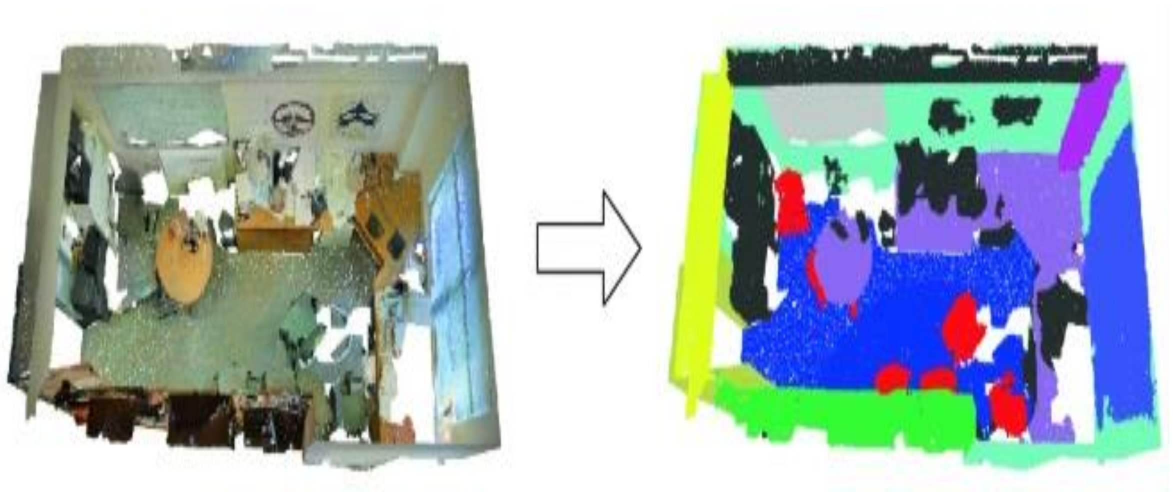
→ The order of the elements doesn’t count

Two types of tasks

Entire set

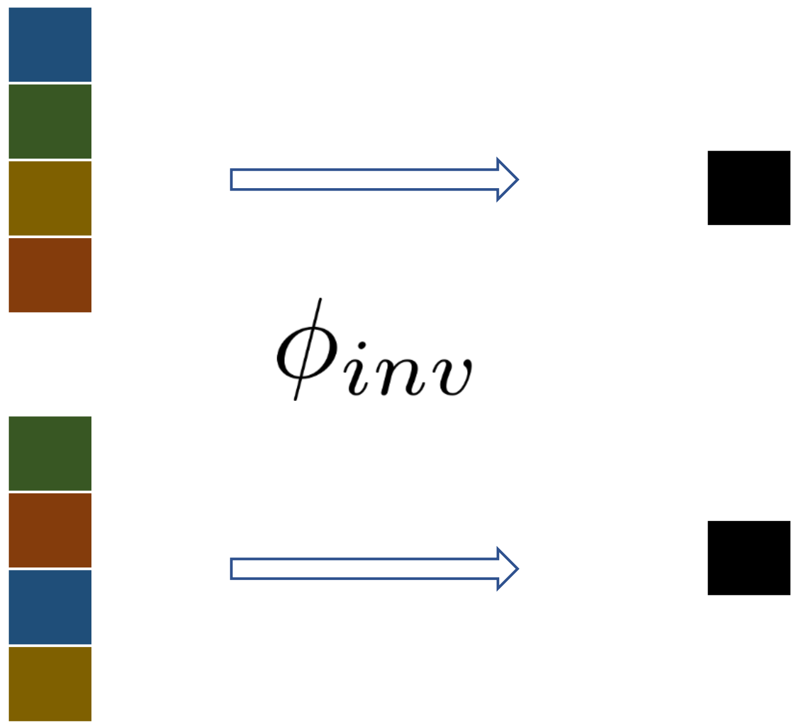


Elementwise

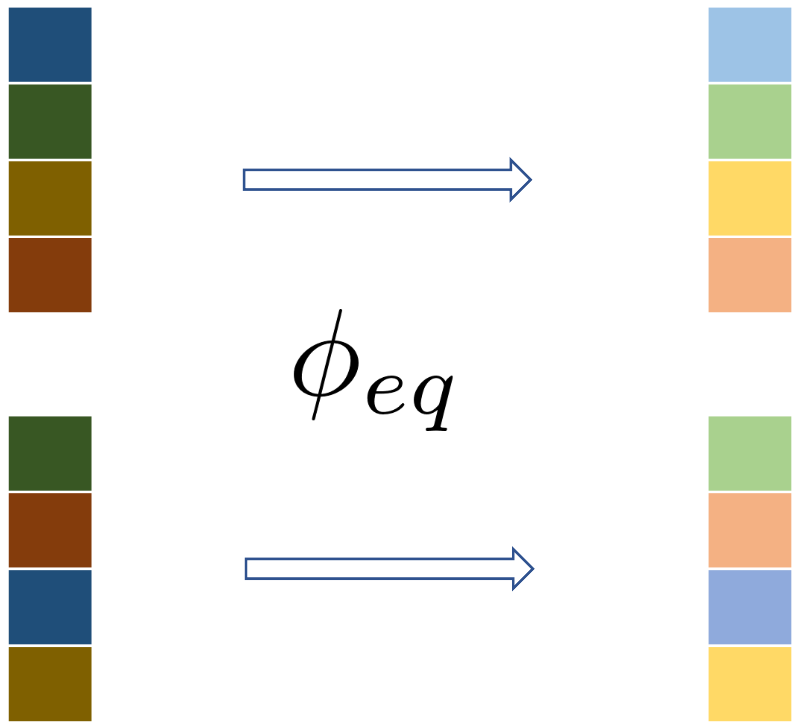


Two notions arise

Permutation Invariance



Permutation Equivariance



Group Invariance/Equivariance

Let $\phi : X \to Y$ be a map and let G be a group acting on X. This means that G is identied by a sub-group of the bijections of X.

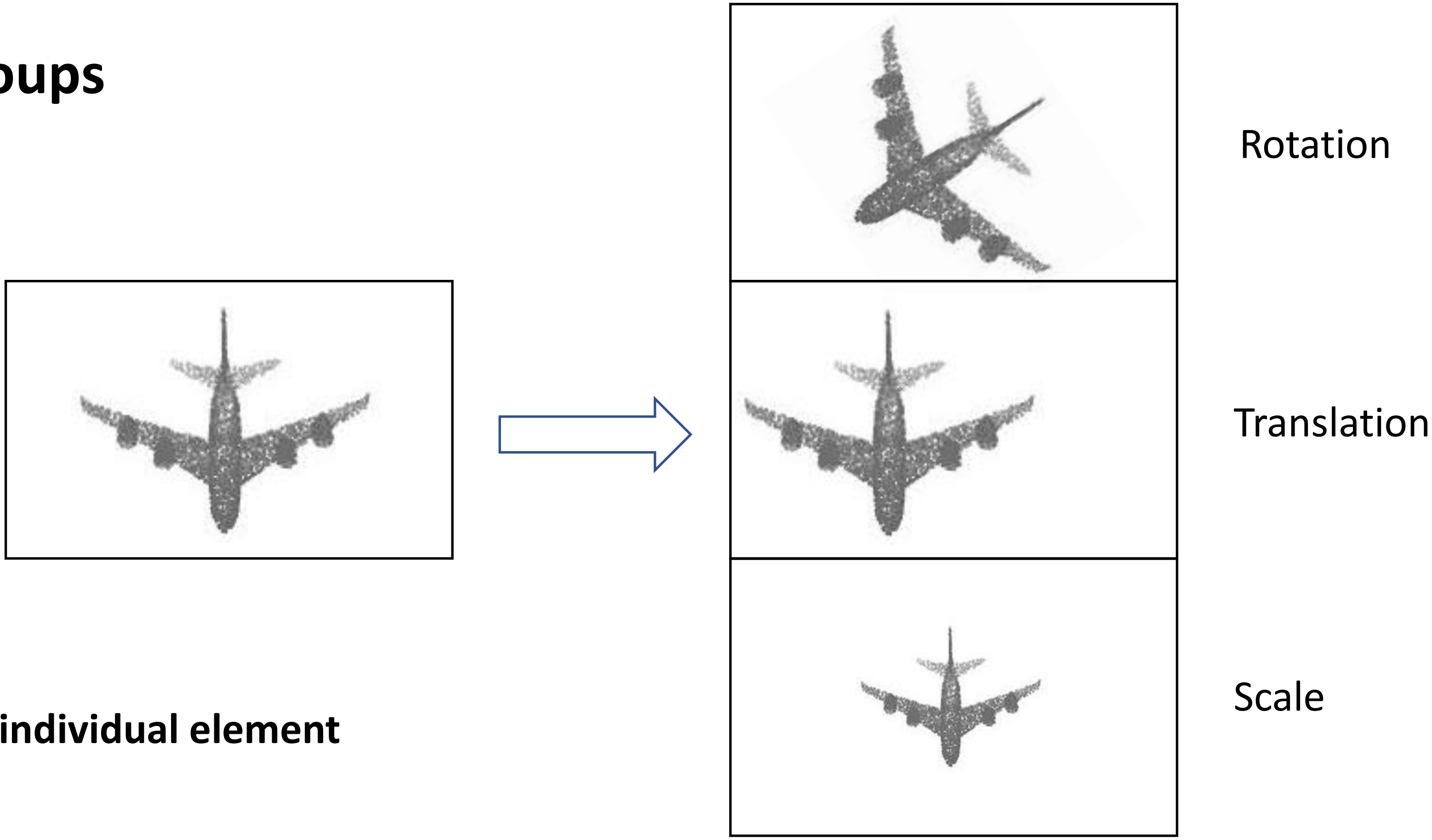
<u>Invariance</u> : we say ϕ is G-invariant if $\forall g \in G, x \in X$, $\phi(g.x) = \phi(x)$.

<u>Equivariance</u> : If G acts also on Y, we say ϕ $\mathcal{G}$-equivariant if $\forall g \in G, x \in X$, $\phi(g.x) = g.\phi(x)$

Group Invariance/Equivariance

→ **there are other relevant groups**

<u>Example</u> :

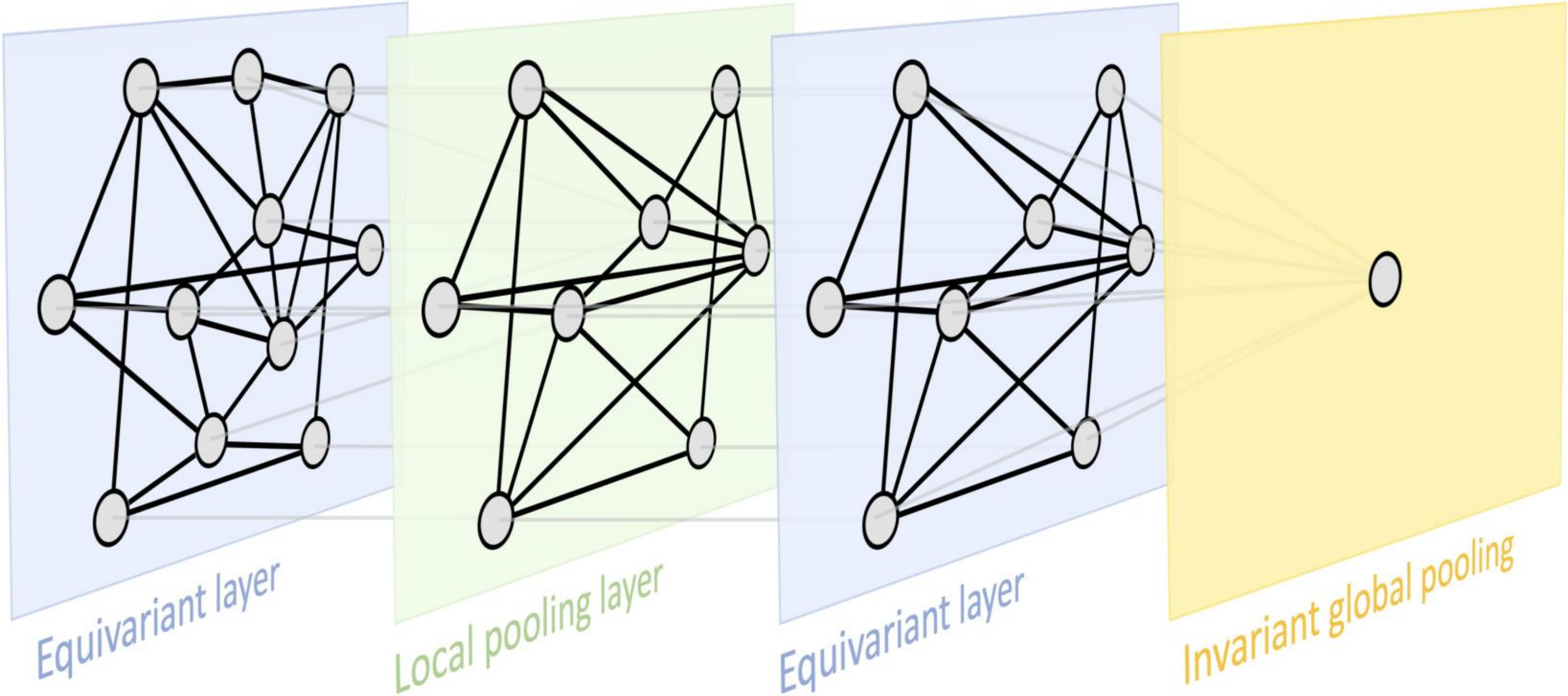


→ In this example, the group acts on each individual element

→ For permutation inv-/equivariance
the group acts between elements of the set

Permutation Invariance / Equivariance

→ two connected notions, often used side by side

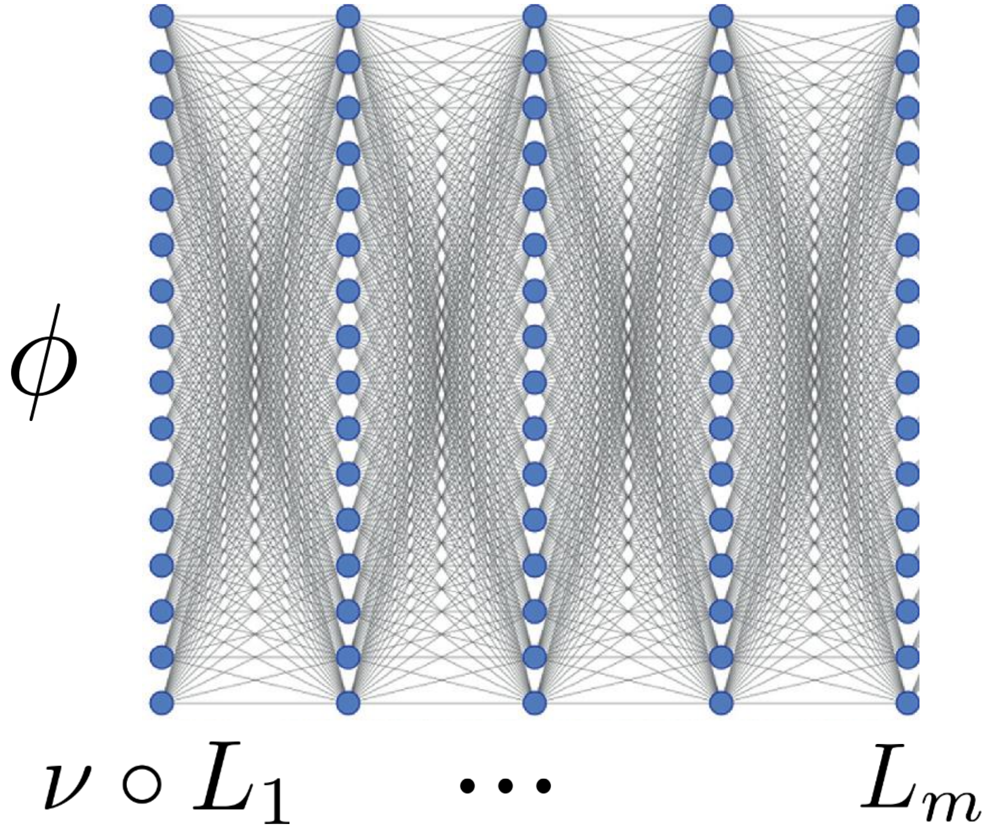


Framework

We will consider a set of n elements $x_1, .., x_n$, each having features in $\mathbb{R}^k$, and group them into one matrix $\mathbb{R}^{n \times k} \ni X = (x_1, .., x_n)^T$.

Equivariance

How do we model this with neural networks ?



$$\phi : \mathbb{R}^{n \times d_{in}} \to \mathbb{R}^{n \times d_{out}}$$

$$X \to L_m \circ \nu \circ \ldots \circ \nu \circ L_1(X)$$

$$L_i : \mathbb{R}^{n \times d_i} \to \mathbb{R}^{n \times d_{i+1}}$$

→ the activation ν is equivariant (elementwise operations)
→ need to pay attention to L_i

Equivariance

One Natural way :

→ apply an elementwise function :

$$x_1, .., x_n \longrightarrow (\pi(x_1), .., \pi(x_n))$$

Equivariance

One Natural way :

→ apply an elementwise function :

$$x_1, .., x_n \longrightarrow (\pi(x_1), .., \pi(x_n))$$

Leads to :

$$L_i(X) = \lambda_i I X + \mathbf{1} c_i$$

$I \in \mathbb{R}^{n \times n}$ is the identity
$\mathbf{1} = [1, .., 1]^T \in \mathbb{R}^n$
$\lambda_i \in \mathbb{R}^{d_i \times d_{i+1}}, \; c_i \in \mathbb{R}^{d_{i+1}}$

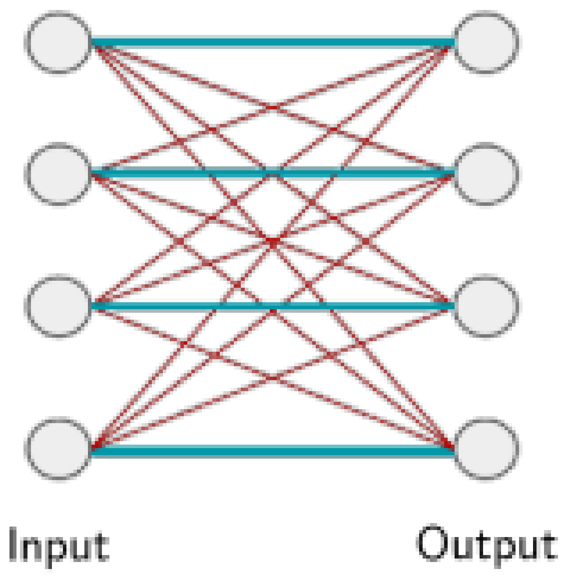
→ Drawback : no interaction between elements of the set

Equivariance

$$L_i(X) = \lambda_i I X + \mathbf{1}c_i$$

$$L_i'(X) = \lambda_i I X + \gamma_i(\mathbf{1}^T\mathbf{1}) + \mathbf{1}c_i$$

→ no interactions

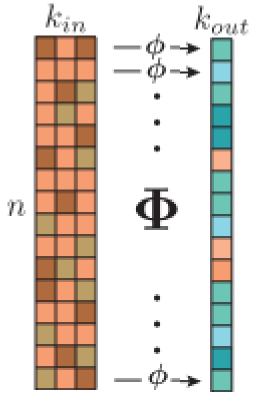


Zaheer et al. – **Deep Sets** - NeurIPS 2017

Equivariance

Deepsets :

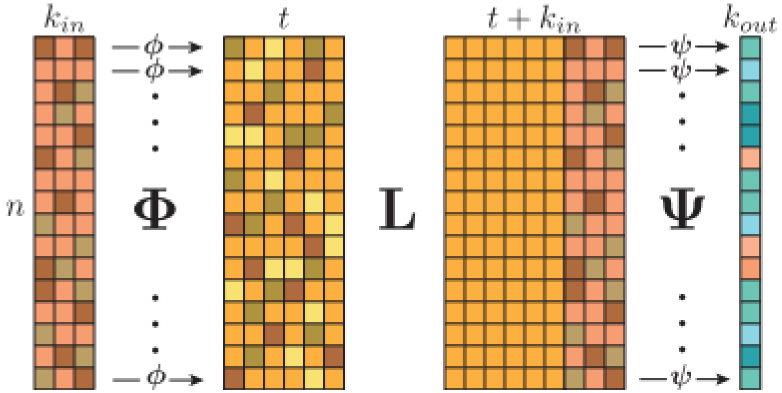
$$X \to L'_m \circ \nu \circ \ldots \circ \nu \circ L'_1(X)$$



$$L_i(X) = \lambda_i I X + \mathbf{1} c_i$$
$$L'_i(X) = \lambda_i I X + \gamma_i(\mathbf{1}^T \mathbf{1}) + \mathbf{1} c_i$$

PointNetST :

$$X \to L_m \circ \nu \circ \ldots \circ \nu \circ L'_j \circ \nu \circ \ldots \circ \nu \circ L_1(X)$$



→ both are universal approximator

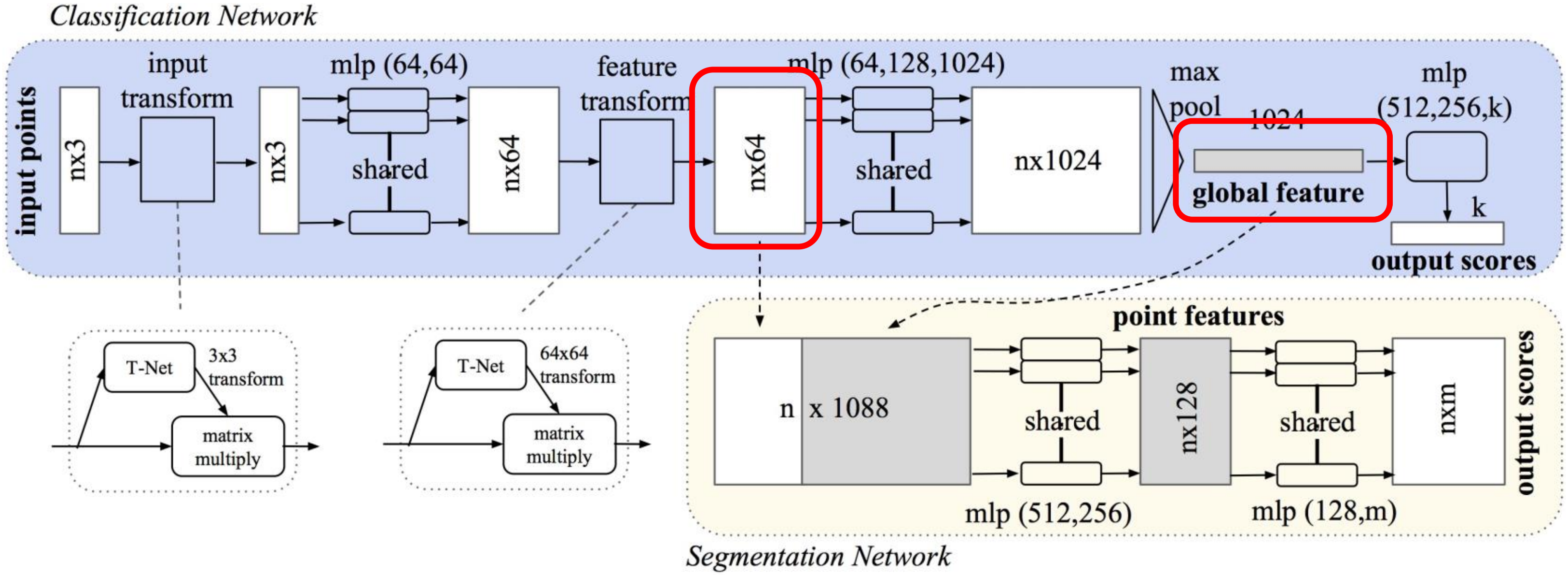
Segol et al. – **On Universal Equivariant SetNetworks-** 2020

Equivariance

For each element, concatenate :

- **features obtained by applying elementwise function**
- **the output of an invariant function**

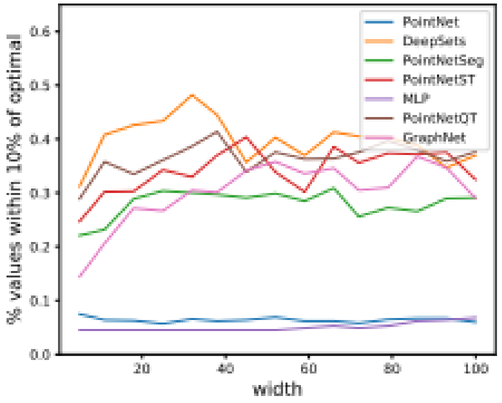
PointNetSeg :



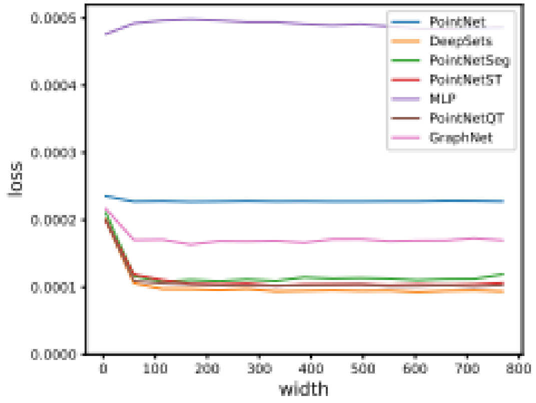
Qi et al. – **PointNet : Deep Learning on Point Sets for 3D Classification and Segmentation -** CVPR 2017

Comparison

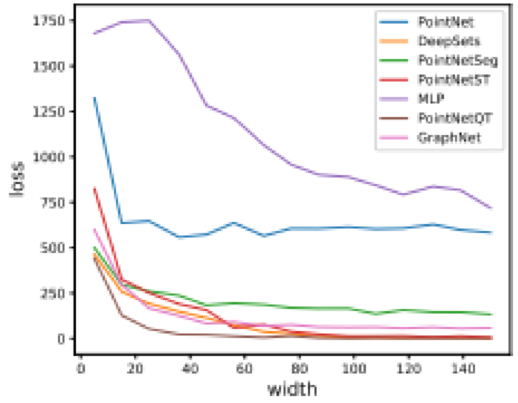
Knapsack test



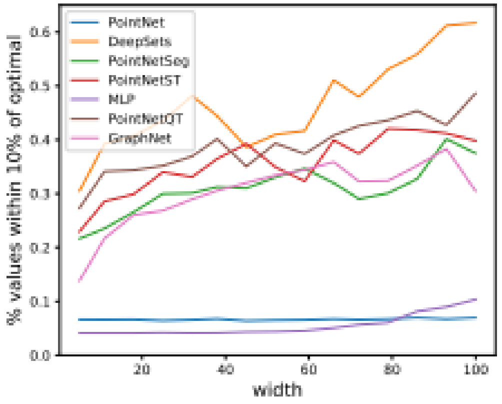
Fiedler test



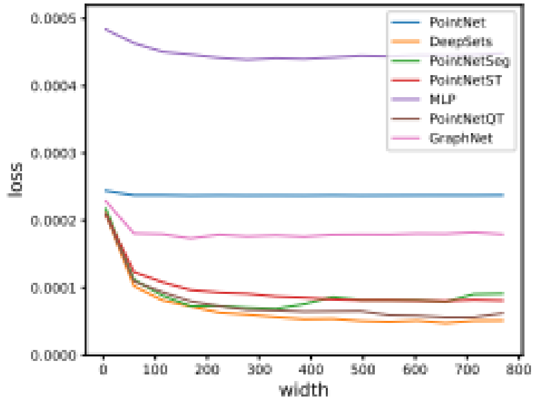
$\sum_{x \in \mathbf{X}} (x - \frac{1}{2})^2$ test



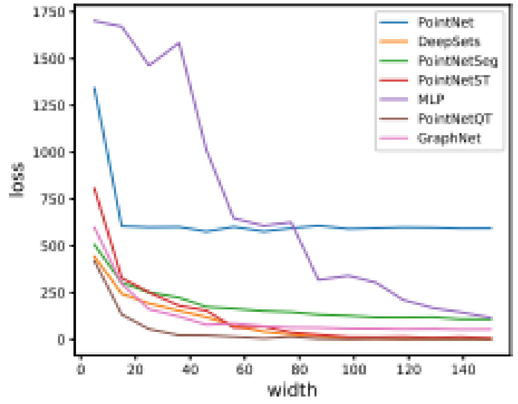
Knapsack train



Fiedler train



$\sum_{x \in \mathbf{X}} (x - \frac{1}{2})^2$ train



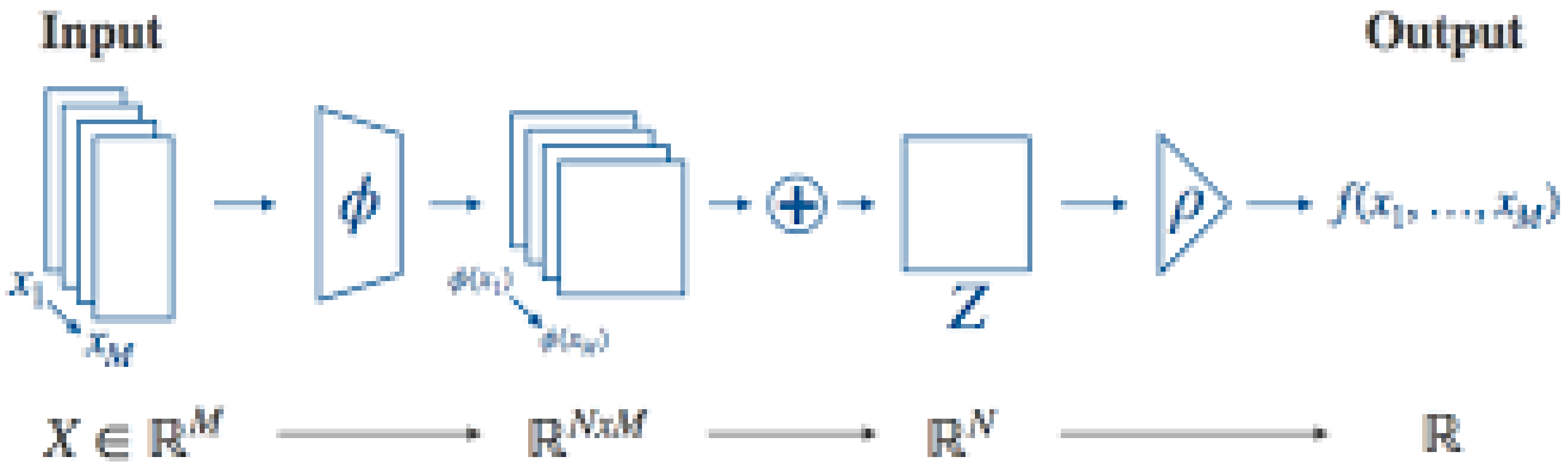
Segol et al. – **On Universal Equivariant SetNetworks-** 2020

Invariance

<u>Permutation Invariance</u> : ϕ is permutation invariant if for all permutation π, we have :

$$\phi(x_{\pi(1)}, .., x_{\pi(n)}) = \phi(x_1, .., x_n)$$

Deep Sets



→ Universal Approximator of invariant functions

Zaheer et al. – **Deep Sets -** NeurIPS 2017

Deep Sets

$$f(X) = \boxed{\rho}\left(\boxed{\sum_{i=1}^{n}}\boxed{\phi}(x_i)\right)$$

Encoder

Aggregation / Pooling

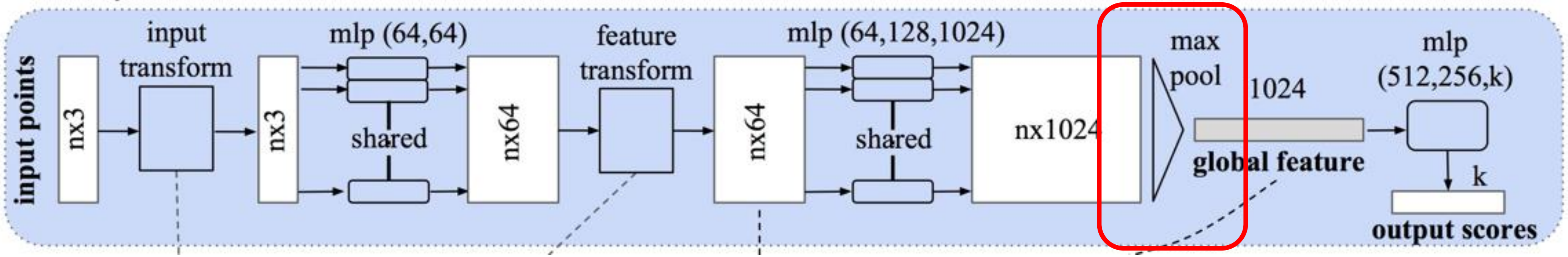
Decoder

Deep Sets

Other choices of aggregation functions :

- **Mean**
- **Max (ex : PointNet) :**

$$f(X) = \rho\left(\max_{i=1..n} \phi(x_i)\right)$$



→ advantage of better generalization on varying set sizes

Qi et al. – **PointNet : Deep Learning on Point Sets for 3D Classification and Segmentation -** CVPR 2017

Deep Sets

Some information on interactions is discarded during the encoding

$$f(X) = \rho\left(\sum_{i=1}^{n} \boxed{\phi}(x_i)\right)$$

→ Use previously discussed equivariant layers

$\boxed{\text{→ Higher order Janossy pooling}}$

Janossy pooling

Idea : consider each possible permutation on the elements and pass them into the same function, then average

$$f(X) = \boxed{\rho}\left(\boxed{\frac{1}{S_n}\sum_{\pi \in S_n}}\boxed{\phi}(\pi(X))\right)$$

- Encoder
- Aggregation / Pooling
- Decoder

Wagstaf et al. – Un**iversal Approximation of Functions on Sets -** 2021

Janossy pooling

Obvious drawback : computational complexity (number of permutations)

Solutions :

- **Sorting** the elements
- **Sampling** among the permutations
- **Restricting** permutations to k-tuples

Janossy pooling with k-tuples

$$f(X) = \rho\left(\frac{(n-k)!}{n!}\sum_{X_k}\phi(X_k)\right)$$

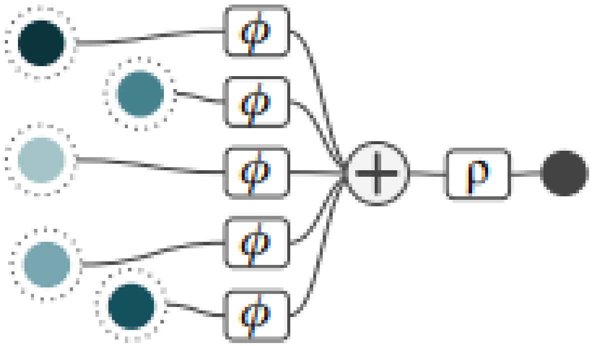
Sum over X_k formed by any k-tuple of elements in any order.

→ **Reduces complexity to** $\mathcal{O}(n^k)$

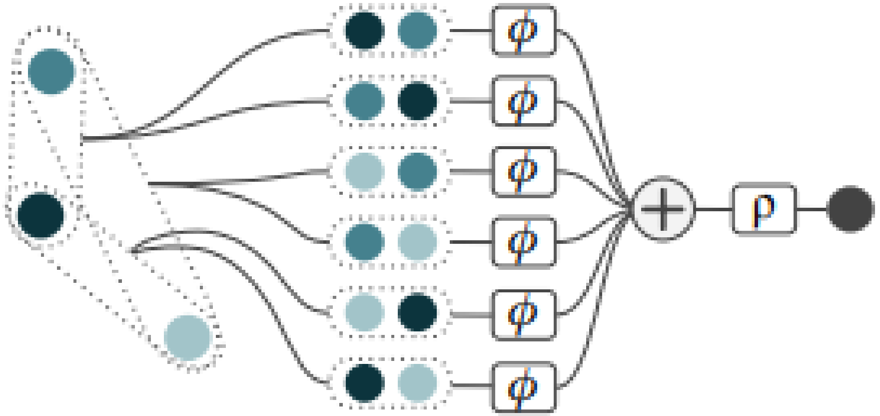
→ **Tuples incorporate interactions between elements**

→ **Deep Set :** $k = 1$

Janossy pooling



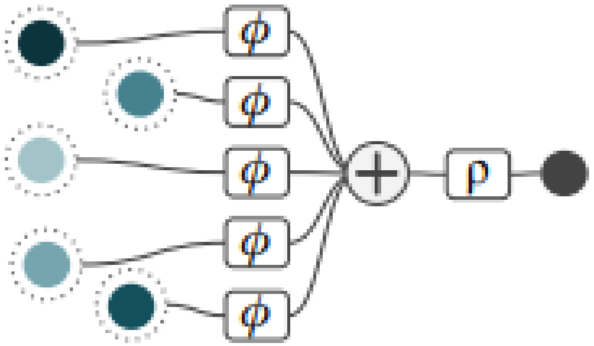
(a) Janossy pooling with $k = 1$ (*Deep Sets*)



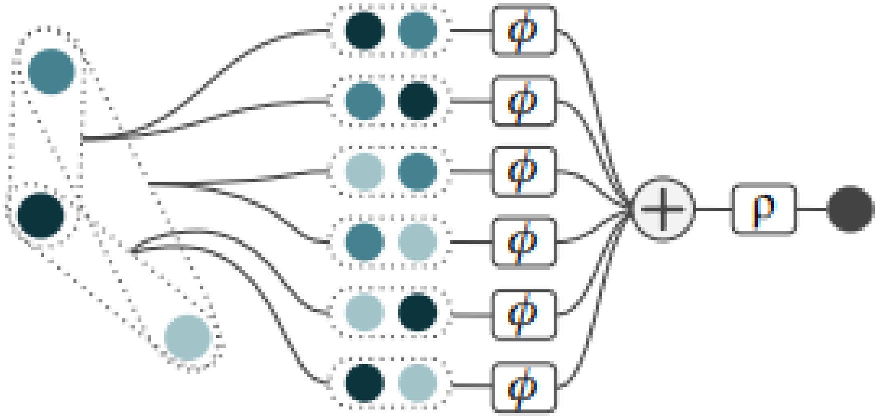
(b) Janossy pooling with $k = 2$

Wagstaf et al. – Un**iversal Approximation of Functions on Sets -** 2021

Janossy pooling



(a) Janossy pooling with $k = 1$ (*Deep Sets*)

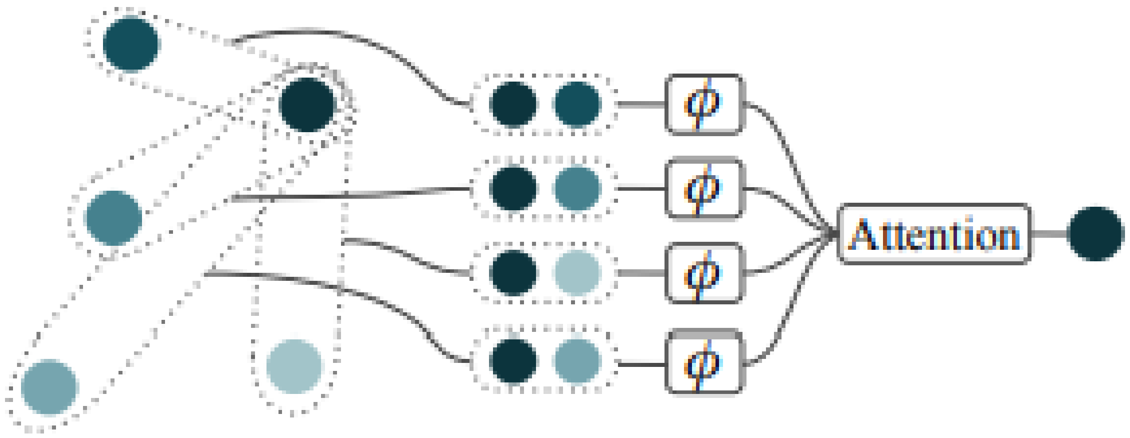


(b) Janossy pooling with $k = 2$

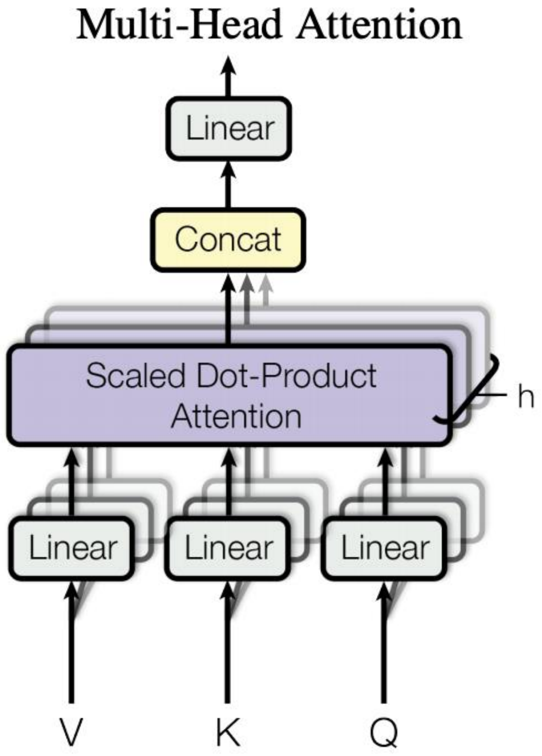
Wagstaf et al. – Un**iversal Approximation of Functions on Sets -** 2021

Set transformer

Deal with interaction ?
→ Using self-attention



(c) Self-attention

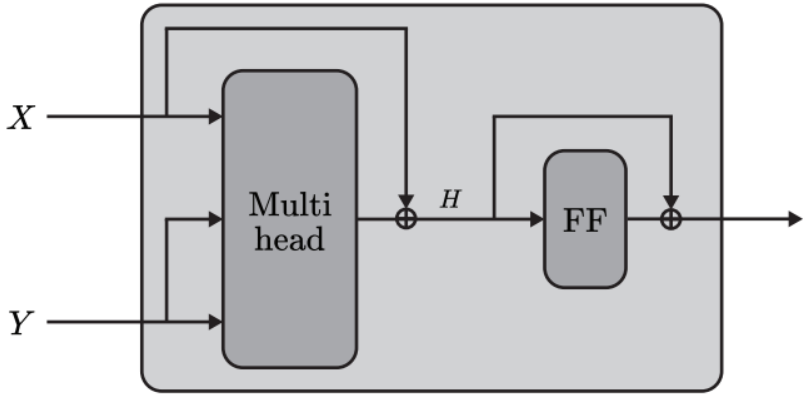


$$\mathrm{MultiHead}(\mathbf{Q},\mathbf{K},\mathbf{V}) = [\mathrm{head}_1;\ldots;\mathrm{head}_h]\mathbf{W}^O$$
$$\text{where } \mathrm{head}_i = \mathrm{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V)$$

Lee et al. – **Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks –** ICML 2019

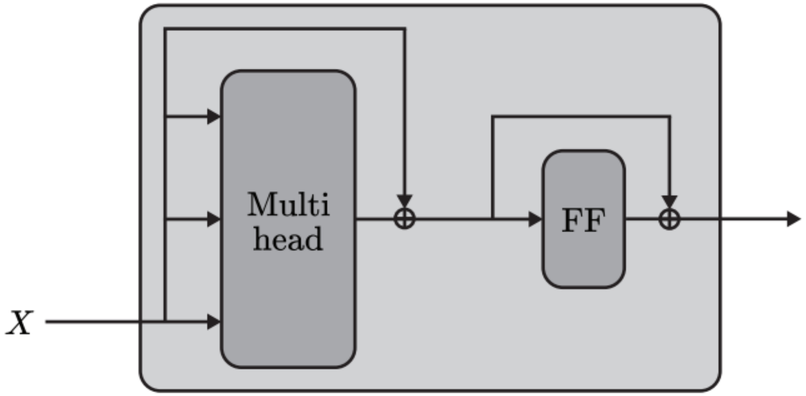
Set transformer

Deal with interaction ?
→ Using self-attention



(b) MAB

Multi Attention Block



(c) SAB

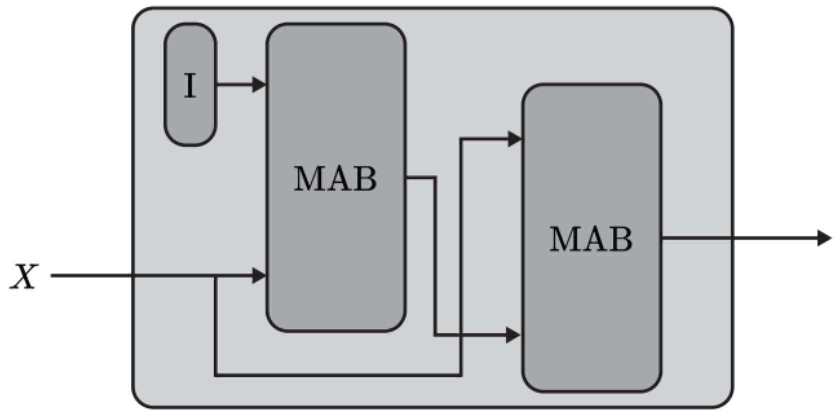
Set Attention Block

Complexity is $\mathcal{O}(n^2)$

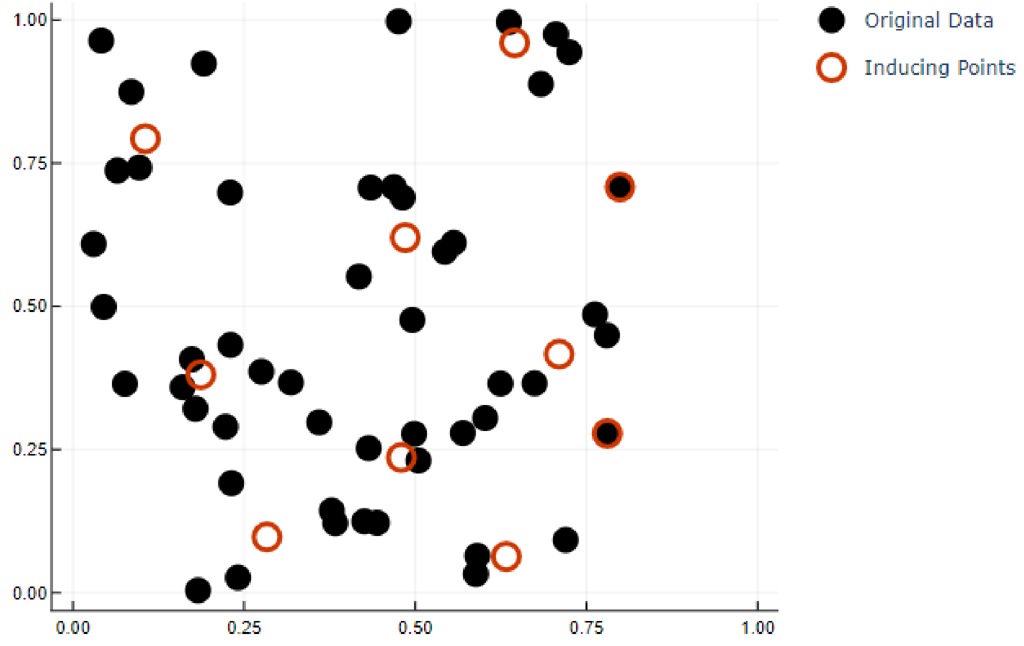
Lee et al. – **Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks –** ICML 2019

Set transformer

→ **Further reduce the complexity using inducing points**



(d) ISAB



Complexity is now : $\mathcal{O}(nm)$

Lee et al. – **Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks –** ICML 2019

Set transformer

Improve the aggregation function ?

→ differentiate influence of each instance

Set transformer

Idea :

- **Aggregate the encodings of each element, using attention to weight them according to their respective influence**
- **Output a set of k elements (usually k=1)**

<u>Pooling Multihead Attention :</u>

$$\mathrm{PMA}_k = \mathrm{MAB}(S, \mathrm{rFF}(Z))$$

encoder output $Z \in \mathbb{R}^{n \times d}$
k seed vectors $S \in \mathbb{R}^{k \times d}$

Lee et al. – **Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks –** ICML 2019

Recap of invariant models

Encoder

Aggregation / Pooling

Decoder

Deep Sets :

$$\boxed{\mathrm{FF}} \circ \boxed{\mathrm{Pooling}} \circ \boxed{\mathrm{rFF}}(X)$$

Set Transformer :

$$\boxed{\mathrm{FF}} \circ \boxed{\mathrm{SAB} \circ \mathrm{PMA}_1} \circ \boxed{\mathrm{ISAB}_m \circ \ldots \mathrm{ISAB}_m}(X)$$

→ **It is possible to mix those methods**

Comparison

Table 4. Test accuracy for the point cloud classification task using 100, 1000, 5000 points.

	Architecture	100 pts	1000 pts	5000 pts
Deep Sets	rFF + Pooling (Zaheer et al., 2017)	-	0.83 ± 0.01	-
	rFFp-max + Pooling (Zaheer et al., 2017)	0.82 ± 0.02	0.87 ± 0.01	$\mathbf{0.90 \pm 0.003}$
	rFF + Pooling	0.7951 ± 0.0166	0.8551 ± 0.0142	0.8933 ± 0.0156
Set Transformer	rFF + PMA (ours)	0.8076 ± 0.0160	0.8534 ± 0.0152	0.8628 ± 0.0136
	ISAB (16) + Pooling (ours)	0.8273 ± 0.0159	$\mathbf{0.8915 \pm 0.0144}$	$\mathbf{0.9040 \pm 0.0173}$
	ISAB (16) + PMA (ours)	$\mathbf{0.8454 \pm 0.0144}$	0.8662 ± 0.0149	0.8779 ± 0.0122

rFF : Row-wise Feed Forward layer
rFFp-max : rFF with permutation equivariant variants
ISAB : Inducing Set Attention Block
Pooling : Usual pooling without attention
PMA : Pooling by Multihead Attention

Lee et al. – **Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks –** ICML 2019

Comparison

Table 5. Meta set anomaly results. Each architecture is evaluated using average of test AUROC and test AUPR.

	Architecture	Test AUROC	Test AUPR
Deep Sets	Random guess	0.5	0.125
	rFF + Pooling	0.5643 ± 0.0139	0.4126 ± 0.0108
	rFFp-mean + Pooling	0.5687 ± 0.0061	0.4125 ± 0.0127
	rFFp-max + Pooling	0.5717 ± 0.0117	0.4135 ± 0.0162
	rFF + Dotprod	0.5671 ± 0.0139	0.4155 ± 0.0115
Set Transformer	SAB + Pooling (ours)	0.5757 ± 0.0143	0.4189 ± 0.0167
	rFF + PMA (ours)	0.5756 ± 0.0130	0.4227 ± 0.0127
	SAB + PMA (ours)	$\mathbf{0.5941} \pm \mathbf{0.0170}$	$\mathbf{0.4386} \pm \mathbf{0.0089}$

Lee et al. – **Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks –** ICML 2019

References

[1] 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.

[2] Charles R. Q., Su H., Kaichun M., and Guibas L. J. **PointNet: Deep learning on point sets for 3D classification and segmentation**. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[3] Segol N., Lipman Y. **On Universal Equivariant Set Networks**, 2020.

[4] Wagstaff E., Fuchs F. B., Engelcke M., Osborne M. A., Posner I. **Universal Approximation of Functions on Sets**, 2021.

[5] Lee J., Lee Y., Kim J., Kosiorek A. R., Choi S., Teh Y. W. **Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks**, In *International Conference on Machine Learning (ICML),* 2019.

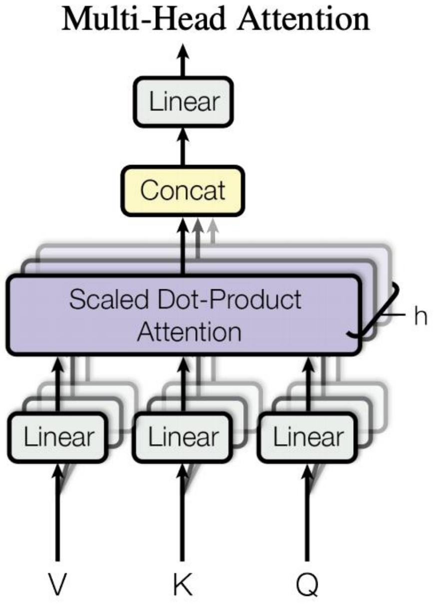
Conclusion

Questions ?

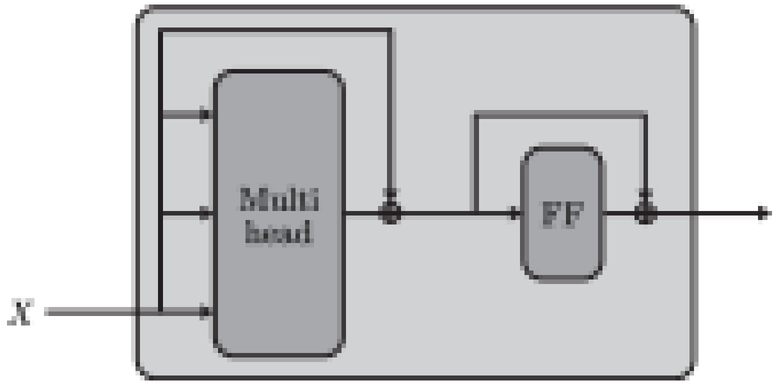
Set transformer

Multihead-Attention :

$$\mathrm{Multihead}(Q,K,V;\lambda,\omega) = \mathrm{concat}(O_1,..,O_h)W^O$$
$$O_j = \mathrm{Att}(QW_j^Q, KW_j^K, VW_j^V; w_j)$$
$$\mathrm{Att}(Q,K,V;\omega) = \omega(QK^T)V$$



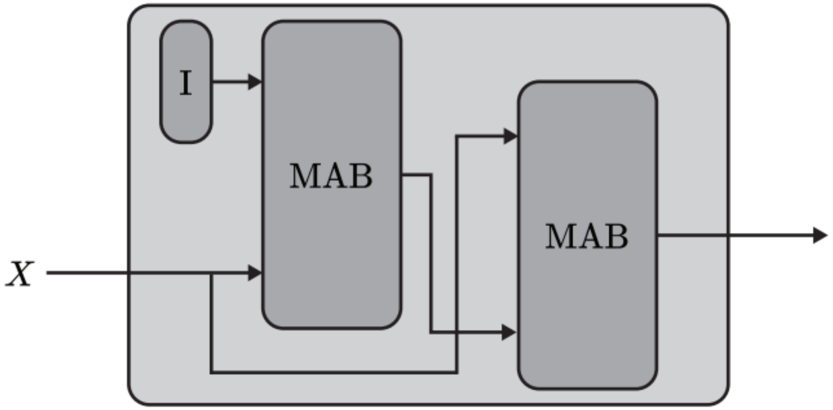
Attentions blocks :



(c) SAB

$$\mathrm{SAB}(X) = \mathrm{MAB}(X,X) \in \mathbb{R}^{n\times d}$$
$$\mathrm{MAB}(X,Y) := \mathrm{LayerNorm}(H + \mathrm{rFF}(H))$$
$$\text{where } H = \mathrm{LayerNorm}(X + \mathrm{Multihead}(X,Y,Y;\omega))$$

Set transformer



(d) ISAB

$$\mathrm{MAB}(X, Y) := \mathrm{LayerNorm}(H + \mathrm{rFF}(H))$$
$$\text{where } H = \mathrm{LayerNorm}(X + \mathrm{Multihead}(X, Y, Y; \omega))$$

$$ISAB_m(X) := \mathrm{MAB}(X, H) \in \mathbb{R}^{n \times d}$$
$$\text{where } H = \mathrm{MAB}(I, X) \in \mathbb{R}^{m \times d}$$

Lee et al. – **Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks –** ICML 2019

Set transformer

Improve the aggregation function ?

→ differentiate influence of each instance

<u>Pooling Multihead Attention :</u>

$$\mathrm{MAB}(X,Y) := \mathrm{LayerNorm}(H + \mathrm{rFF}(H))$$
$$\text{where } H = \mathrm{LayerNorm}(X + \mathrm{Multihead}(X,Y,Y;\omega))$$

Idea :

- **Aggregate the encodings of each element, using attention to weight them according to their respective influence**
- **Output a set of k elements**

$$\mathrm{PMA}_k = \mathrm{MAB}(S, \mathrm{rFF}(Z))$$
$$\text{encoder output } Z \in \mathbb{R}^{n\times d}$$
$$k \text{ seed vectors } S \in \mathbb{R}^{k\times d}$$

Lee et al. – **Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks –** ICML 2019