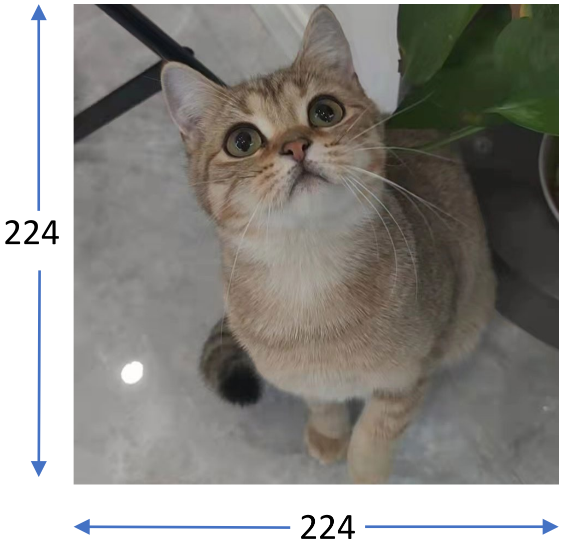
Deep Learning of Sets

Presenter: Minjie Cheng

Motivation

Traditionally, mechine learning handles the data of the form

- Fixed dimensional vectors
- Ordered sequences



Classification

Cat

Motivation

Traditionally, mechine learning handles the data of the form

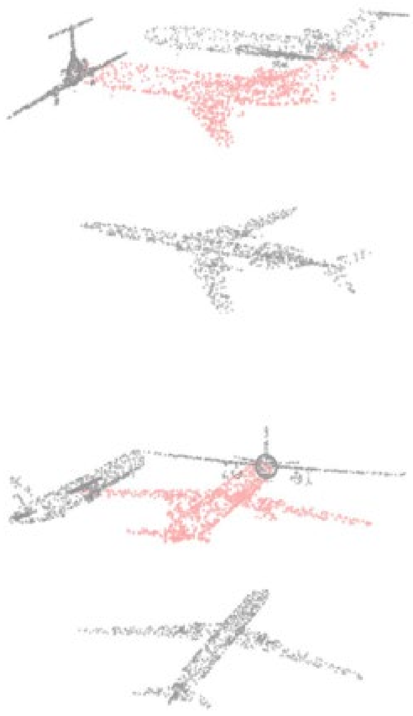
- Fixed dimensional vectors
- Ordered sequences

Sentiment

Motivation

What happens if input are sets

- Unordered collections of objects
- The number of objects can vary



Classification

Airplane

A point-cloud is a set of 3D coordinates of an underlying sampled surface.

Motivation

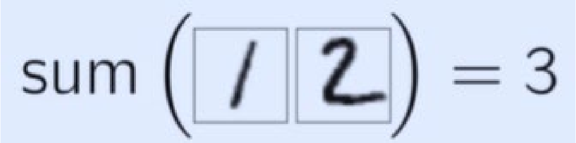
What happens if input are sets

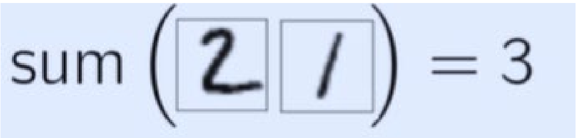
- Unordered collections of objects
- The number of objects can vary

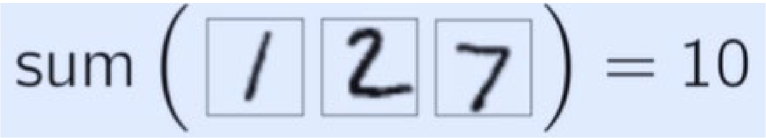
Anomaly Detection

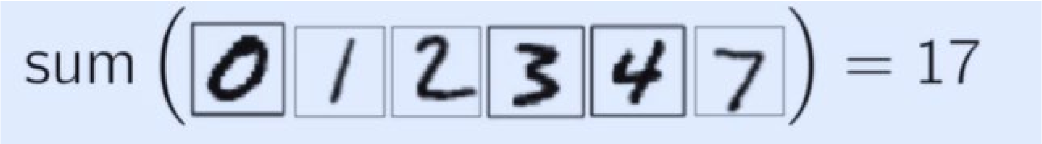
Motivation

- Unordered collections of objects
- The number of objects can vary





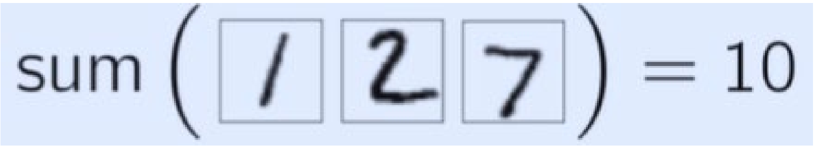


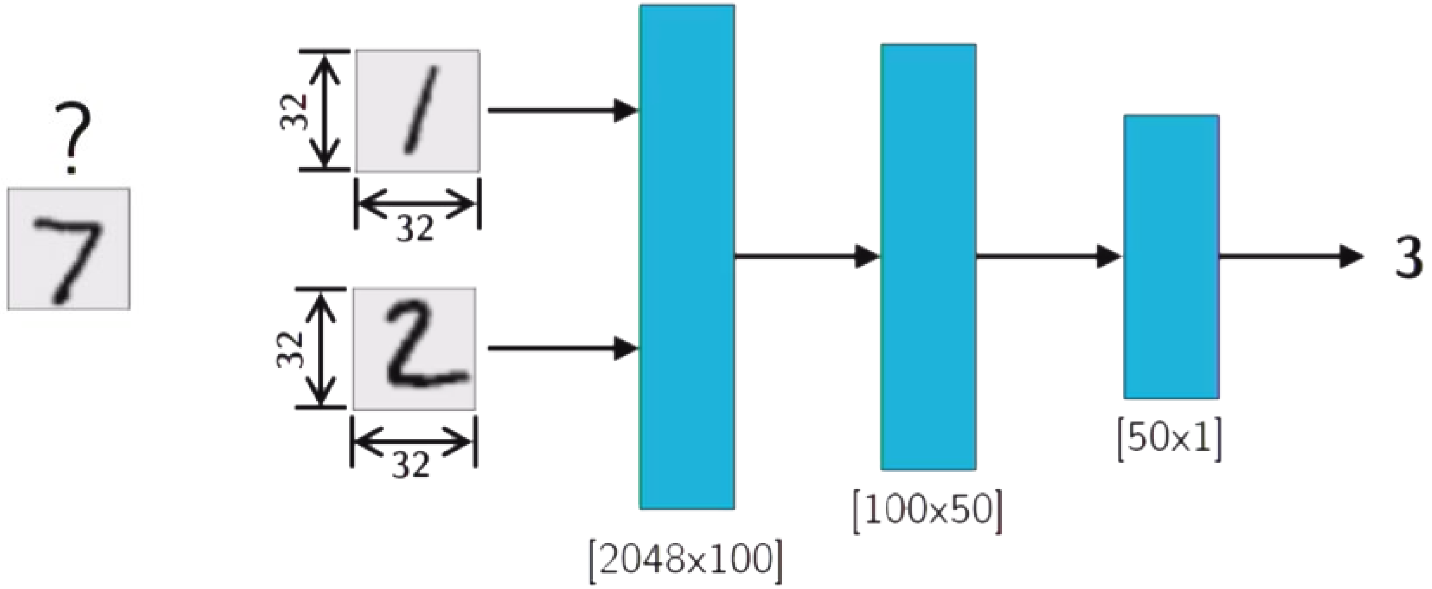


How do we feed this to a neural network?

Motivation-Task: Find Sum of a set of Numbers

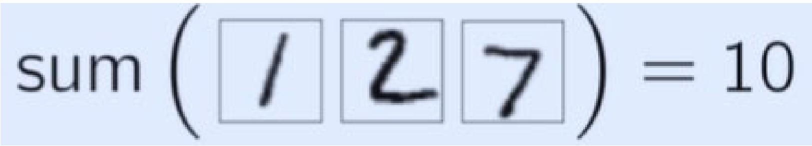
How do we feed this to a neural network?



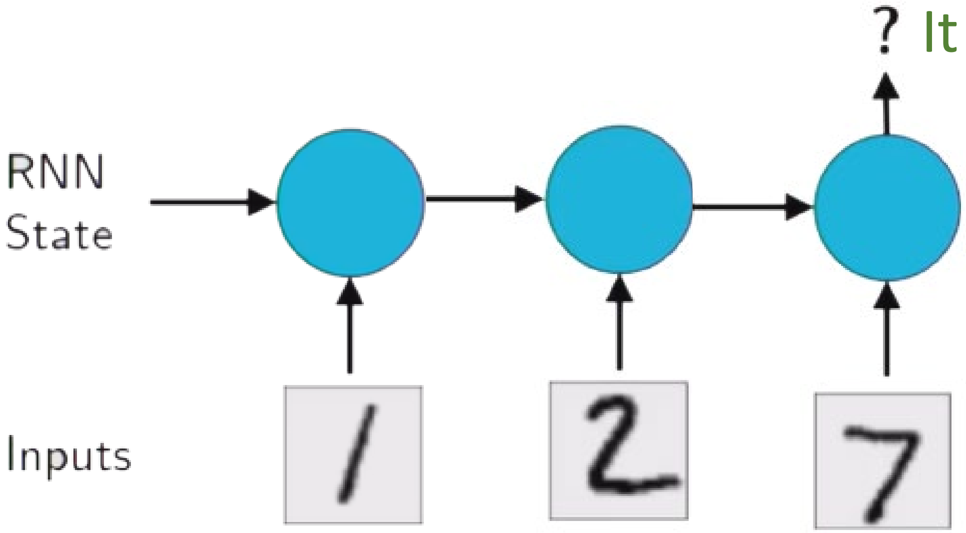


Motivation-Task: Find Sum of a set of Numbers

How do we feed this to a neural network?



How about treating as sequence?



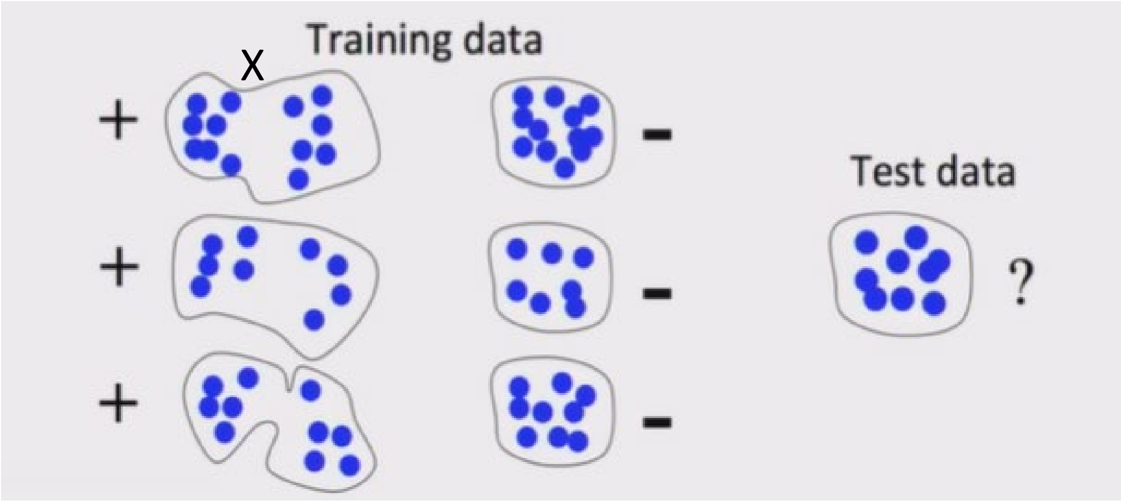
It may not guarantee same answer

Problem Definition

Perform ML tasks like classification or regression when inputs is a set

- Order does not matter
- Number of elements varies

That is to learn a function $f(X)$ operating on the set to produce output.



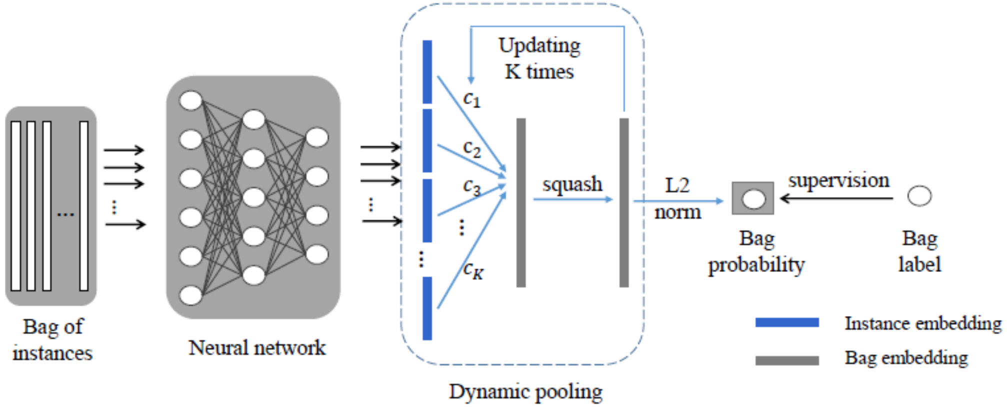


Figure 1: The architecture of Dynamic pooling for Multi-Instance Neural Network.

Functions on sets

Let $f: X^N \longrightarrow Y^M$ $\quad\pi$: any permutation

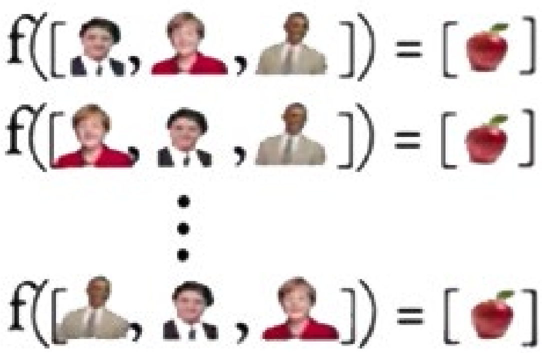
- **Permutation Invariant**

Permuting the input variables does not affect the output

$$f(x_1, x_2, x_3) = f(x_1, x_3, x_2) = f(x_3, x_2, x_1)$$

$M = 1$:

$$f(\pi X) = f(X)$$

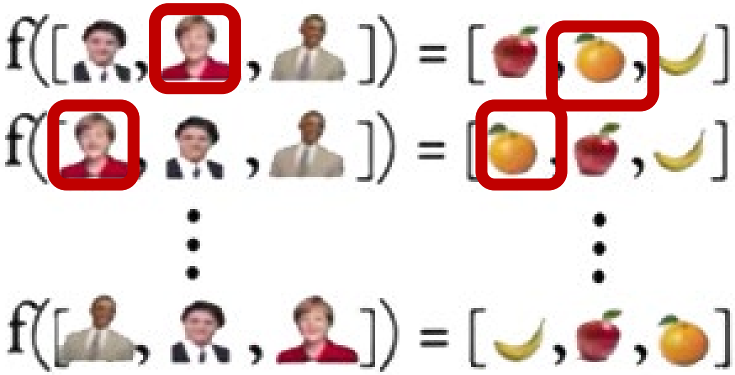


- **Permutation Equivariant**

Permuting the input variables permutes the output in same way

$M = N$

$$f(\pi X) = \pi f(X)$$

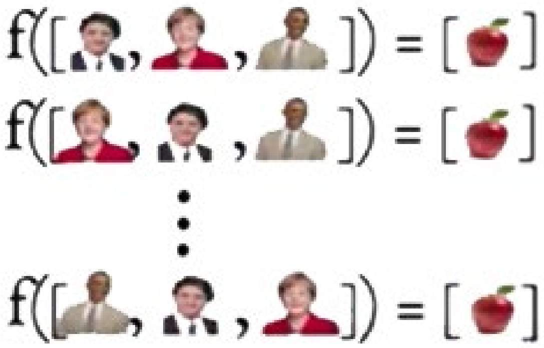


Functions on sets

- Permutation Invariant $M = 1$

Such functions must have the following structure:

$$f(X) = \rho\left(\sum_{x \in X} \emptyset(x)\right)$$



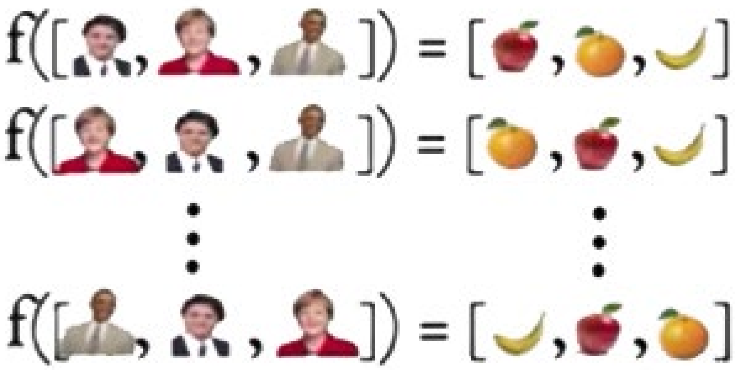
- Permutation Equivariant $M = N$

The neural network layer $\sigma(\theta X)$ must have following structure:

$$\Theta = \lambda \mathbf{I} + \gamma \left(\mathbf{1}\mathbf{1}^{\mathsf{T}}\right)$$

where $\lambda, \gamma \in \mathbb{R}$ $\quad \mathbf{1} = [1, \ldots, 1]^{\mathsf{T}} \in \mathbb{R}^M$

$$\theta = \begin{bmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & \lambda \end{bmatrix} + \begin{bmatrix} \gamma & \gamma & \gamma \\ \gamma & \gamma & \gamma \\ \gamma & \gamma & \gamma \end{bmatrix}$$



Functions on sets

- Permutation Invariant $M = 1$

Such functions must have the following structure:

$$f(X) = \rho(\sum_{x \in X} \emptyset(x))$$

$f(x_1, x_2) = x_1 x_2 (x_1 + x_2 + 3)$ can be represented with $\phi(x) = [x, x^2, x^3]$ and $\rho([u, v, w]) = uv - w + 3(u^2 - v)/2$.

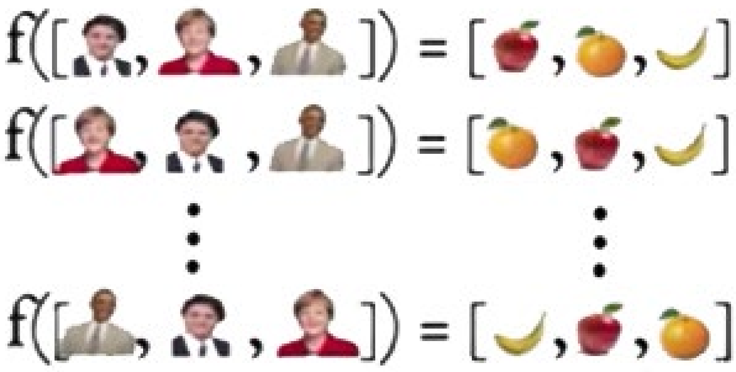
- Permutation Equivariant $M = N$

The neural network layer $\sigma(\theta X)$ must have following structure:

$$\Theta = \lambda \mathbf{I} + \gamma\ (\mathbf{1}\mathbf{1}^{\mathsf{T}})$$

where $\lambda, \gamma \in \mathbb{R}$ $\quad \mathbf{1} = [1, \ldots, 1]^{\mathsf{T}} \in \mathbb{R}^M$

$$\theta = \begin{bmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & \lambda \end{bmatrix} + \begin{bmatrix} \gamma & \gamma & \gamma \\ \gamma & \gamma & \gamma \\ \gamma & \gamma & \gamma \end{bmatrix}$$



Functions on sets

- Permutation Invariant $M = 1$

Such functions must have the following structure:

$$f(X) = \rho\left(\sum_{x \in X} \emptyset(x)\right)$$

$f(x_1, x_2) = x_1 x_2 (x_1 + x_2 + 3)$ can be represented with $\phi(x) = [x, x^2, x^3]$ and $\rho([u, v, w]) = uv - w + 3(u^2 - v)/2$.

- Permutation Equivariant $M = N$

The neural network layer $\sigma(\theta X)$ must have following structure:

$$\Theta = \lambda \mathbf{I} + \gamma\left(\mathbf{1}\mathbf{1}^{\mathsf{T}}\right)$$

where $\lambda, \gamma \in \mathbb{R}$ $\quad \mathbf{1} = [1, \ldots, 1]^{\mathsf{T}} \in \mathbb{R}^M$

$$\theta = \begin{bmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & \lambda \end{bmatrix} + \begin{bmatrix} \gamma & \gamma & \gamma \\ \gamma & \gamma & \gamma \\ \gamma & \gamma & \gamma \end{bmatrix}$$

$$\begin{bmatrix} \lambda+\gamma & \gamma & \gamma \\ \gamma & \lambda+\gamma & \gamma \\ \gamma & \gamma & \lambda+\gamma \end{bmatrix} \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix} = \begin{bmatrix} \lambda+6\gamma & 4\lambda+15\gamma \\ 2\lambda+6\gamma & 5\lambda+15\gamma \\ 3\lambda+6\gamma & 6\lambda+15\gamma \end{bmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} \lambda+\gamma & \gamma & \gamma \\ \gamma & \lambda+\gamma & \gamma \\ \gamma & \gamma & \lambda+\gamma \end{bmatrix} \begin{bmatrix} 1 & 4 \\ \underline{3} & 5 \\ \underline{2} & 6 \end{bmatrix} = \begin{bmatrix} \lambda+6\gamma & 4\lambda+15\gamma \\ \underline{3\lambda+6\gamma} & 5\lambda+15\gamma \\ \underline{2\lambda+6\gamma} & 6\lambda+15\gamma \end{bmatrix}$$

Architecture of Deepsets Invariant Model

We can use the structure of such functions to design neural networks that are universal for such tasks

$$f(X) = \rho(\sum_{x \in X} \emptyset(x))$$

Each input x in the set is transformed into $\emptyset(x)$

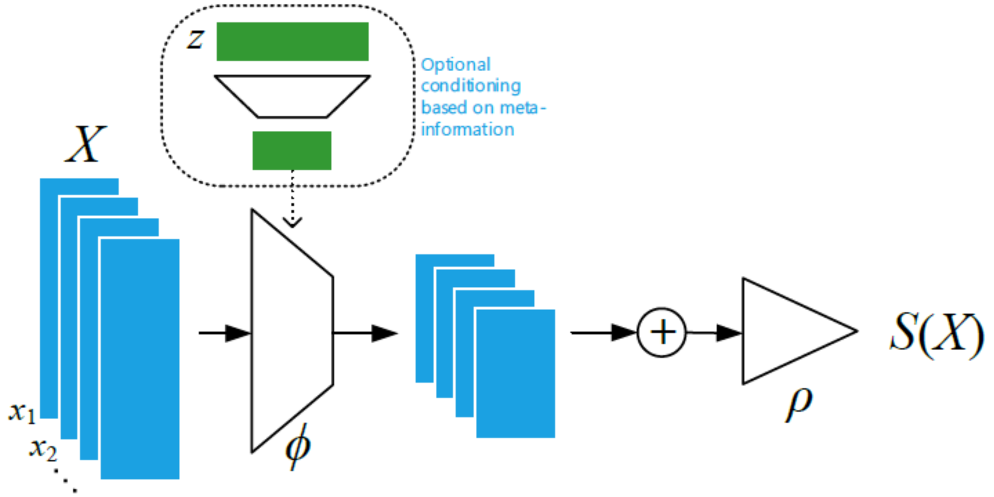


Figure 5: Architecture of DeepSets: Invariant

Simply add the representations $\emptyset(x)$

Sum is processed by ρ network

Architecture of Deepsets Equivariant Model

We can use the structure of such functions to design neural networks that are universal for such tasks

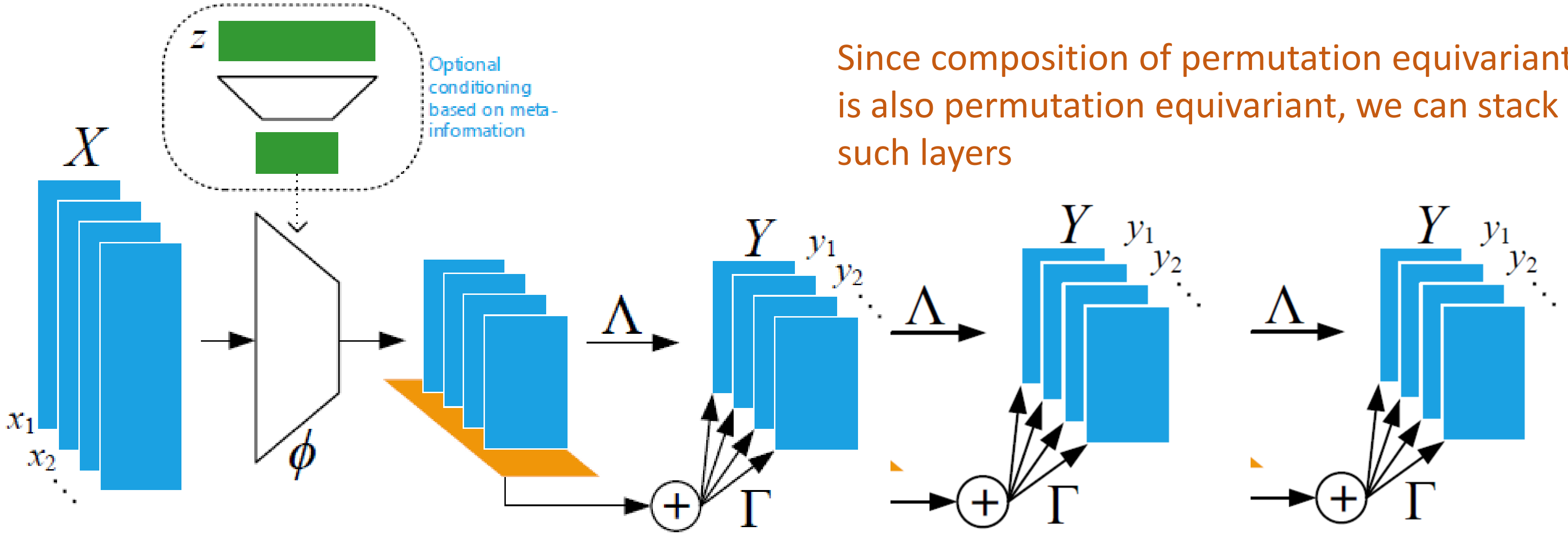
The neural network layer $f(X) = \sigma(\theta X)$ must have following structure:

Matrix Multiplication

$$\Theta = \lambda \mathbf{I} + \gamma\,(\mathbf{1}\mathbf{1}^{\mathsf{T}}) \longrightarrow \mathbf{f}(\mathbf{x}) \doteq \boldsymbol{\sigma}\,(\lambda \mathbf{I}\mathbf{x} + \gamma\ \mathrm{maxpool}(\mathbf{x})\mathbf{1}) \longrightarrow f(\mathbf{x}) = \sigma\big(\mathbf{x}\Lambda - \mathbf{1}\mathrm{maxpool}(\mathbf{x})\Gamma\big) \quad \Lambda, \Gamma \in \mathbb{R}^{D \times D'}$$

$$\mathbf{x} \in \mathbb{R}^{M \times D}$$
$$\downarrow$$
$$\mathbf{y} \in \mathbb{R}^{M \times D'}$$

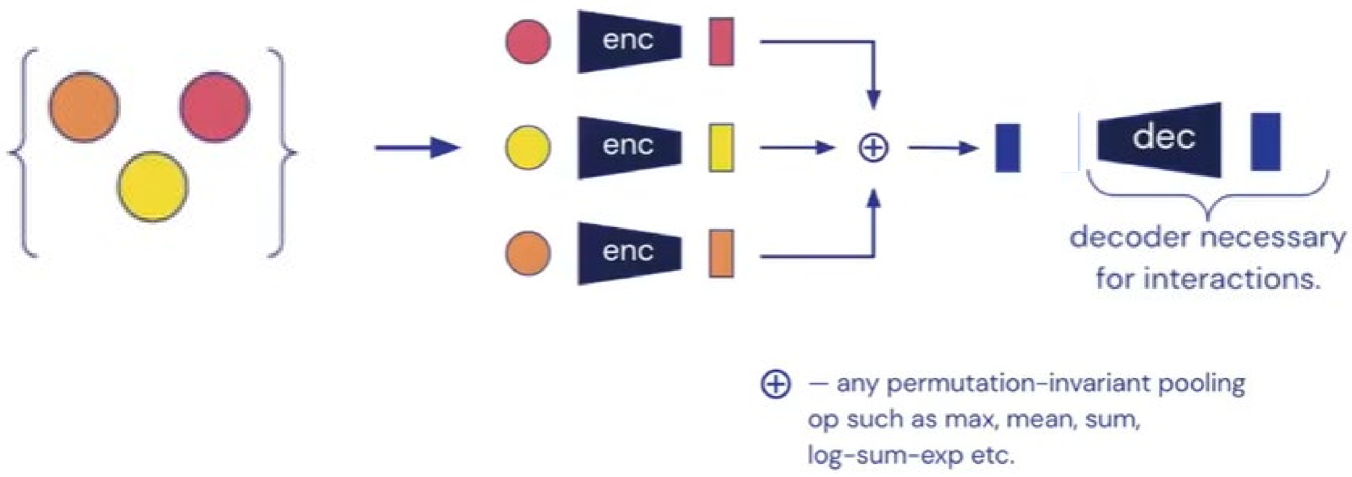
Each input x in the set is transformed into $\emptyset(x)$



Since composition of permutation equivariant functions is also permutation equivariant, we can stack multiple such layers

Final representations are weighted combination of input and its sum

The Summary of Deep Sets

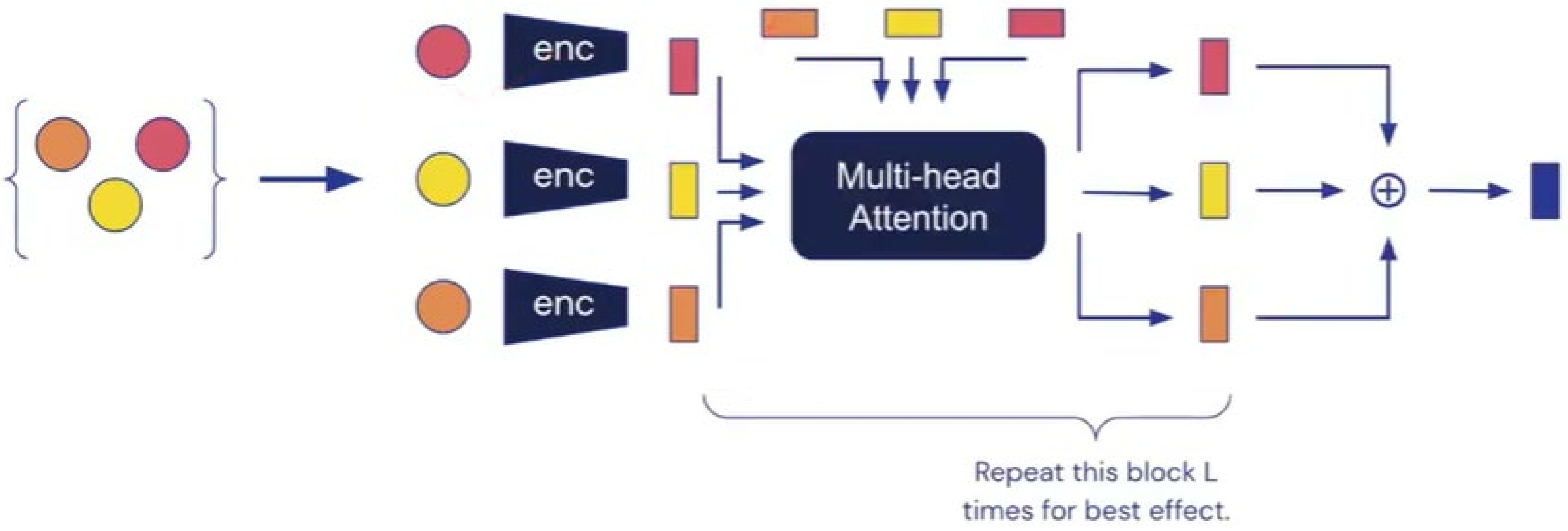


The encoder is shared between set elements.

It naturally supports a variable number of set elements.

Provably a universal approximator of set-input functions.

The Framework of Set Transformer



Self-attention can account for first-order interaction between points

Multiple layers of self-attention account for higer- order interactions

Computation is O(n^2) for n elements

Set Transformer Permutation Invariant

Such functions must have the following structure:

$$f(X) = \rho\Big(\sum_{x \in X} \emptyset\,(x)\Big)$$

$$\mathrm{net}(\{x_1, \ldots, x_n\}) = \underbrace{\rho(\mathrm{pool}}_{\text{Decoder}}(\{\underbrace{\phi(x_1), \ldots, \phi(x_n)}_{\text{Encoder}}\}))$$

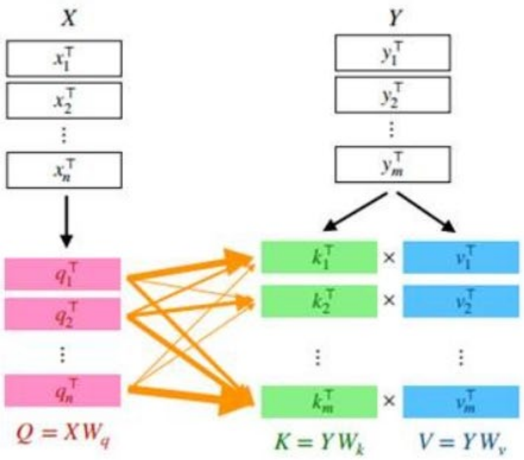
Set Transformer Permutation Equivariant

$$\mathbf{f}(\mathbf{x}) \doteq \boldsymbol{\sigma}\left(\lambda \mathbf{I}\mathbf{x} + \gamma\ \mathrm{maxpool}(\mathbf{x})\mathbf{1}\right)$$

$$\downarrow$$

$$\boxed{f_i}(x; \{x_1, \ldots, x_n\}) = \boxed{\sigma_i}(\lambda x + \gamma \mathrm{pool}(\{x_1, \ldots, x_n\}))$$

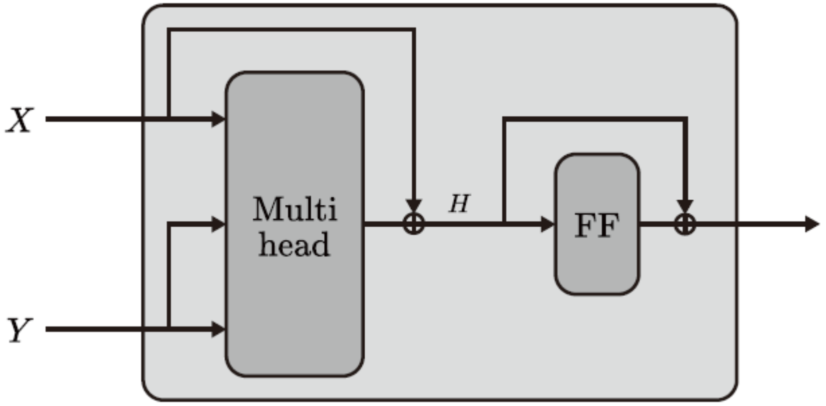
Set Transformer   Multi-head Attention Block (MAB)



$$\mathrm{Att}(X, Y) = \mathrm{softmax}\left(\frac{XW_q W_k^\top Y^\top}{\sqrt{d}}\right) Y W_v$$

$$\mathrm{Multihead}(Q, K, V; \lambda, \omega) = \mathrm{concat}(O_1, \cdots, O_h) W^O,$$

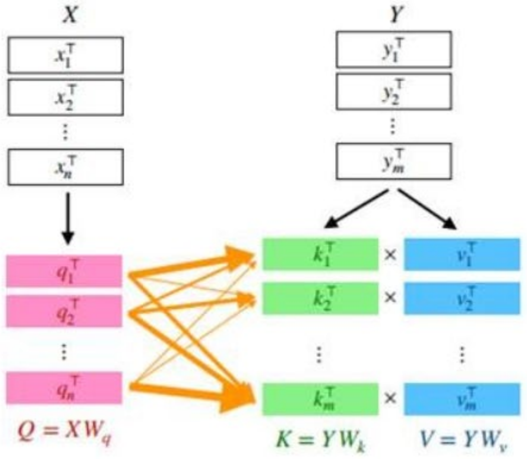
$$\text{where } \; O_j = \mathrm{Att}(QW_j^Q, KW_j^K, VW_j^V; \omega_j)$$



(b) MAB

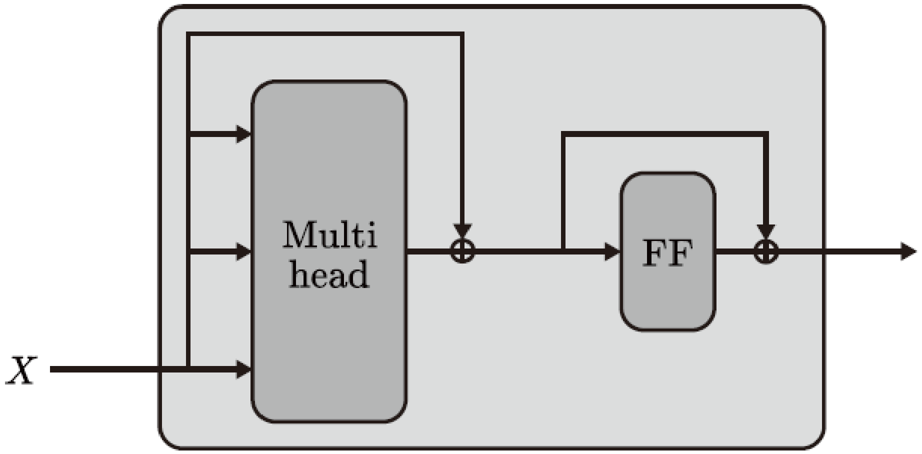
$$\mathrm{MAB}(X, Y) = \mathrm{LayerNorm}(H + \mathrm{rFF}(H)),$$
$$\text{where } \; H = \mathrm{LayerNorm}(X + \mathrm{Multihead}(X, Y, Y; \omega)),$$

Set Transformer   Set Attention Block (SAB)



$$\mathrm{Att}(X, Y) = \mathrm{softmax}\left(\frac{XW_q W_k^\top Y^\top}{\sqrt{d}}\right) Y W_v$$

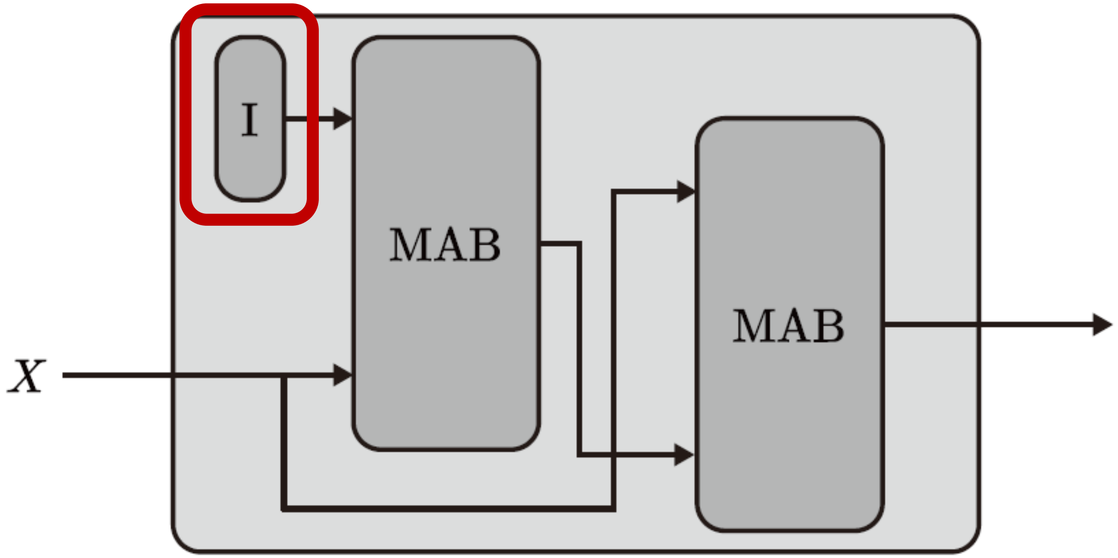
$$\boxed{\mathrm{SelfAtt}(X) = \mathrm{Att}(X, X)\,.}$$



(c) SAB

$$\mathrm{SAB}(X) := \mathrm{MAB}(X, X). \longrightarrow \mathcal{O}(n^2)$$

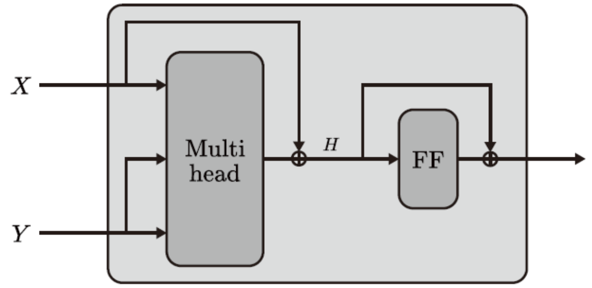
Set Transformer Induced Set Attention Block (ISAB)



$$\mathrm{ISAB}_m(X) = \mathrm{MAB}(X, H) \in \mathbb{R}^{n\times d}, \quad \text{where } H = \mathrm{MAB}(I, X) \in \mathbb{R}^{m\times d}. \longrightarrow \mathcal{O}(nm)$$

SAM(X) and ISAB(X) are permutation-equivariant

Set Transformer   Pooling by Multihead Attention

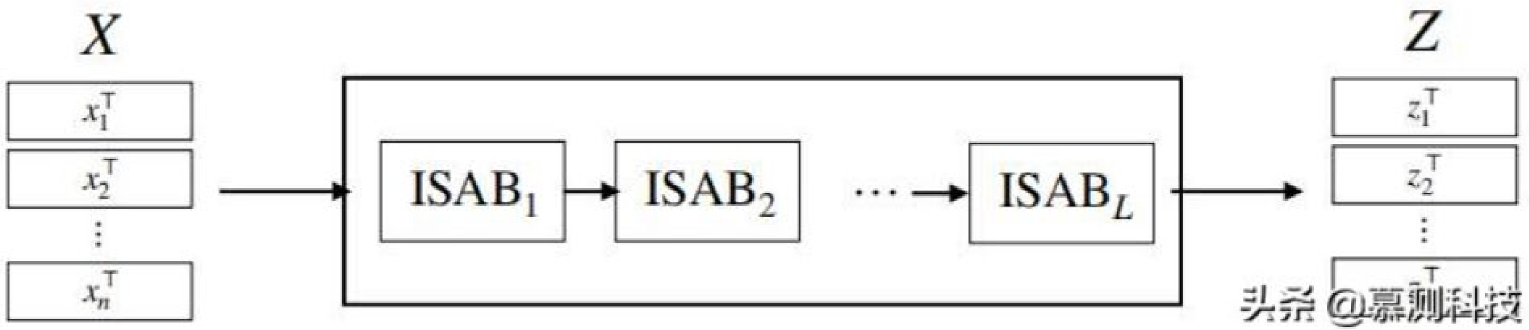


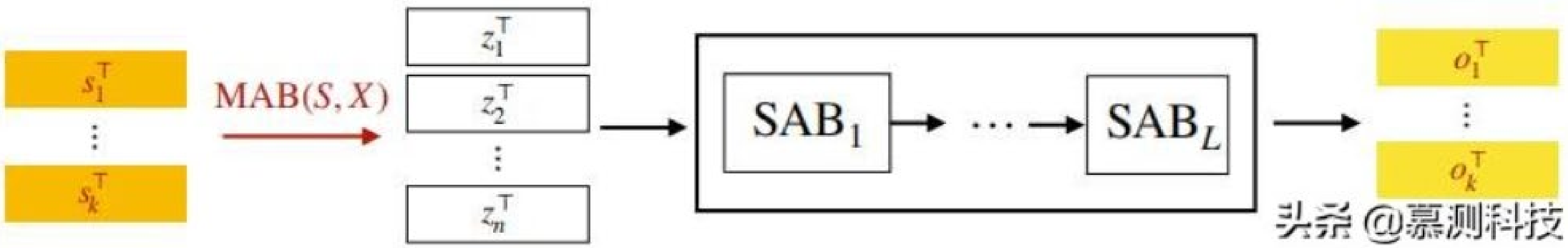
(b) MAB

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

$S \in \mathbb{R}^{k \times d}$: Learnable set of k seed vectors

Set Transformer





permutation-equivariant

Set Transformer

$$\mathrm{Encoder}(X) = \mathrm{SAB}(\mathrm{SAB}(X))$$
$$\mathrm{Encoder}(X) = \mathrm{ISAB}_m(\mathrm{ISAB}_m(X)).$$

$$\mathrm{Decoder}(Z;\lambda) = \mathrm{rFF}(\mathrm{SAB}(\mathrm{PMA}_k(Z))) \in \mathbb{R}^{k\times d}$$
$$\text{where } \mathrm{PMA}_k(Z) = \mathrm{MAB}(S, \mathrm{rFF}(Z)) \in \mathbb{R}^{k\times d},$$

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

Experiments

Set Anomaly Detection

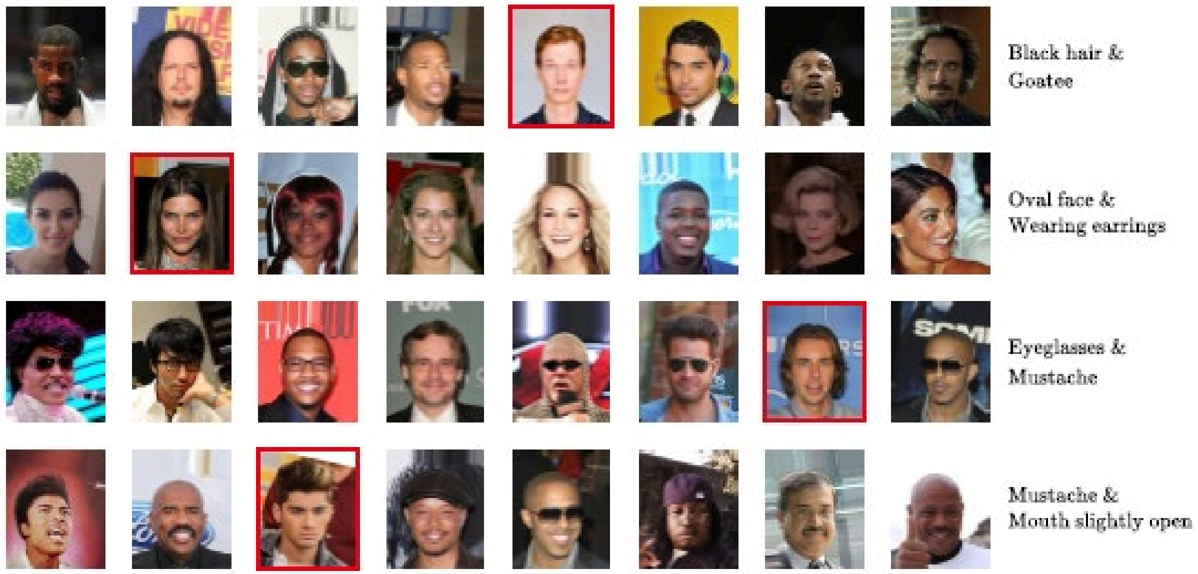


Table 5. Meta set anomaly results. Each architecture is evaluated using average of test AUROC and test AUPR.

Architecture	Test AUROC	Test AUPR
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
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)	**0.5941 ± 0.0170**	**0.4386 ± 0.0089**

Experiments Point-cloud Classification

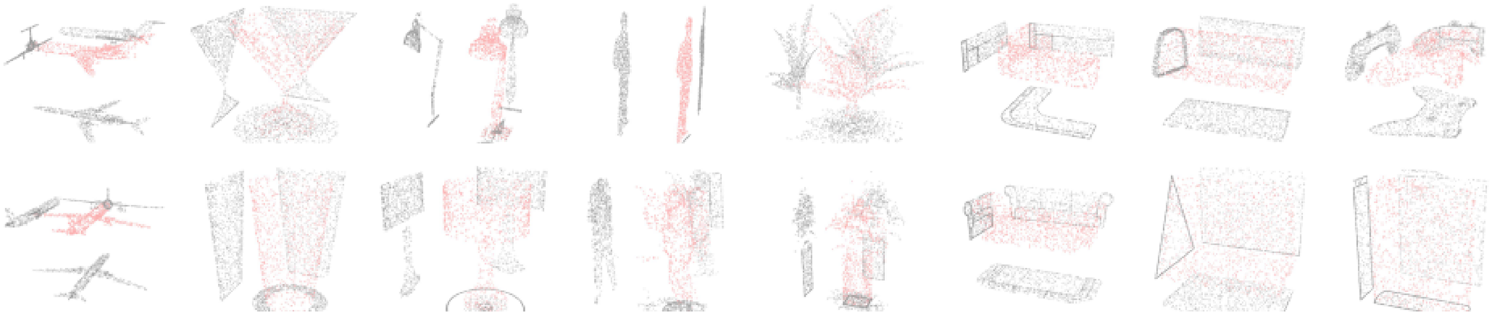


Figure 12: Examples for 8 out of 40 object classes (column) in the ModelNet40. Each point-cloud is produces by sampling 1000 particles from the mesh representation of the original MeodelNet40 instances. Two point-clouds in the same column are from the same class. The projection of particles into xy, zy and xz planes are added for better visualization.

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

Architecture	100 pts	1000 pts	5000 pts
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
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

Experiments

Counting Unique Characters

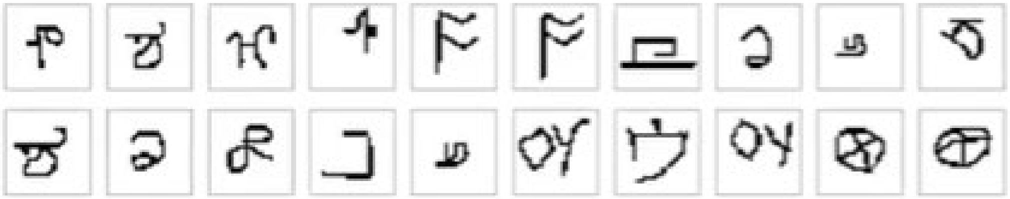


Figure 2. Counting unique characters: this is a randomly sampled set of 20 images from the Omniglot dataset. There are 14 different characters inside this set.

Method	Accuracy
Deep Set	43.82+-072%
Set Transformer	**56.59+-0.77%**

Experiments

Sum of Digits