Deep Set Prediction Networks

Yan Zhang, Jonathan Hare, Adam Prügel-Bennett

University of Southampton

Sets are unordered collections of things

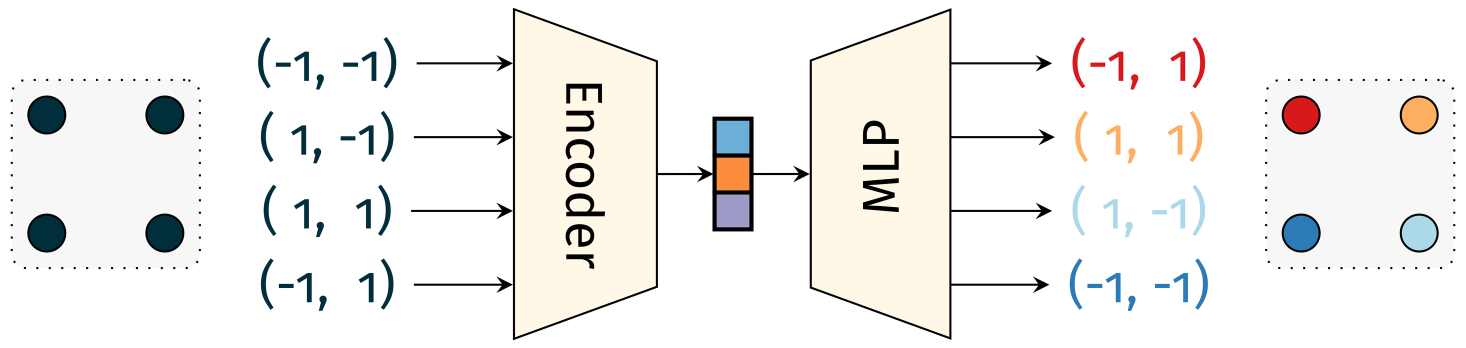
- Many things can be described as **sets of feature vectors**:
  - the set of objects in an image,
  - the set of points in a point cloud,
  - the set of nodes and edges in a graph,
  - the set of people reading this poster.
- Predicting sets means object detection, molecule generation, etc.
- This paper is about doing this **vector-to-set** mapping properly.
- Compared to normal object detection methods:
  - Anchor-free, fully end-to-end, no post-processing.

MLPs are not suited for sets

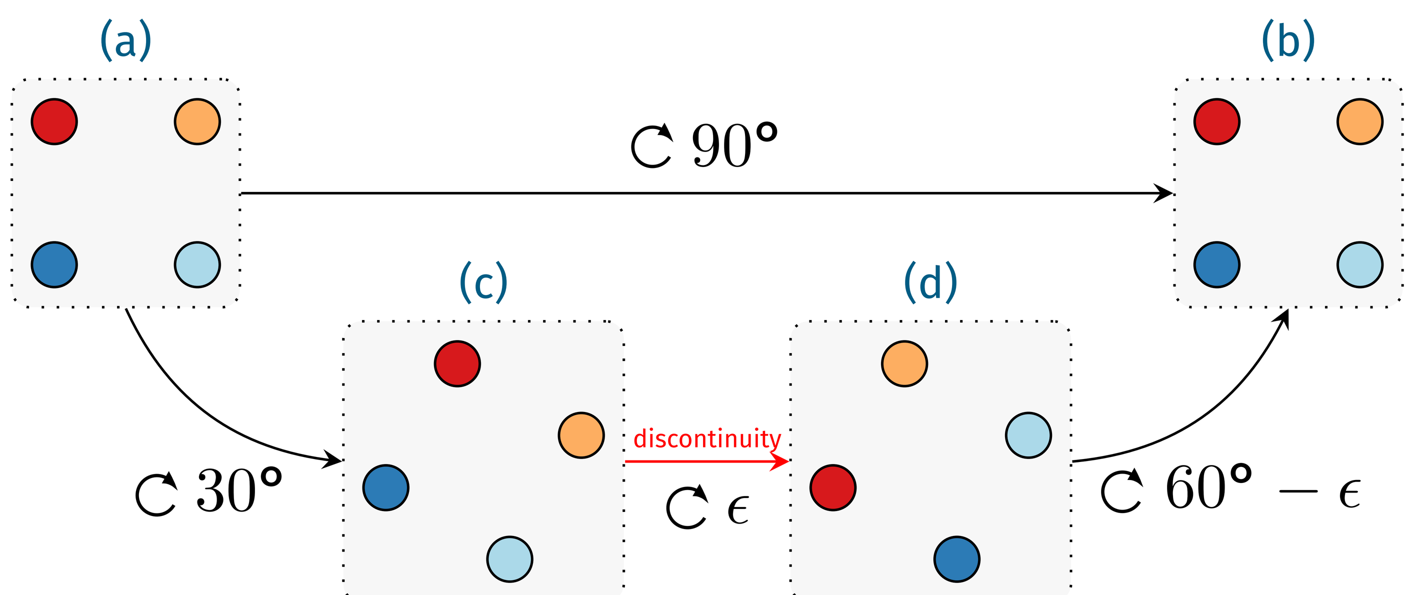
- Sets are **unordered**, but MLP and RNN outputs are **ordered**.

→ **Discontinuities** from *responsibility problem*.

- Let's look at a normal set auto-encoder:



- The responsibility problem:



- (a) and (b) are the same set.

→ (a) and (b) encode to the same vector.

→ (a) and (b) have the same MLP output.

- (a) is turned into (b) by rotating 90°.

→ Rotation starts and ends with the same set.

→ MLP outputs can't just follow the 90° rotation!

→ There must be a discontinuity between (c) and (d)! All the outputs have to jump 90° anti-clockwise.

Conclusion:

- Smooth change of set requires discontinuous change of MLP outputs.
- To predict **unordered sets**, we should use an **unordered model**.

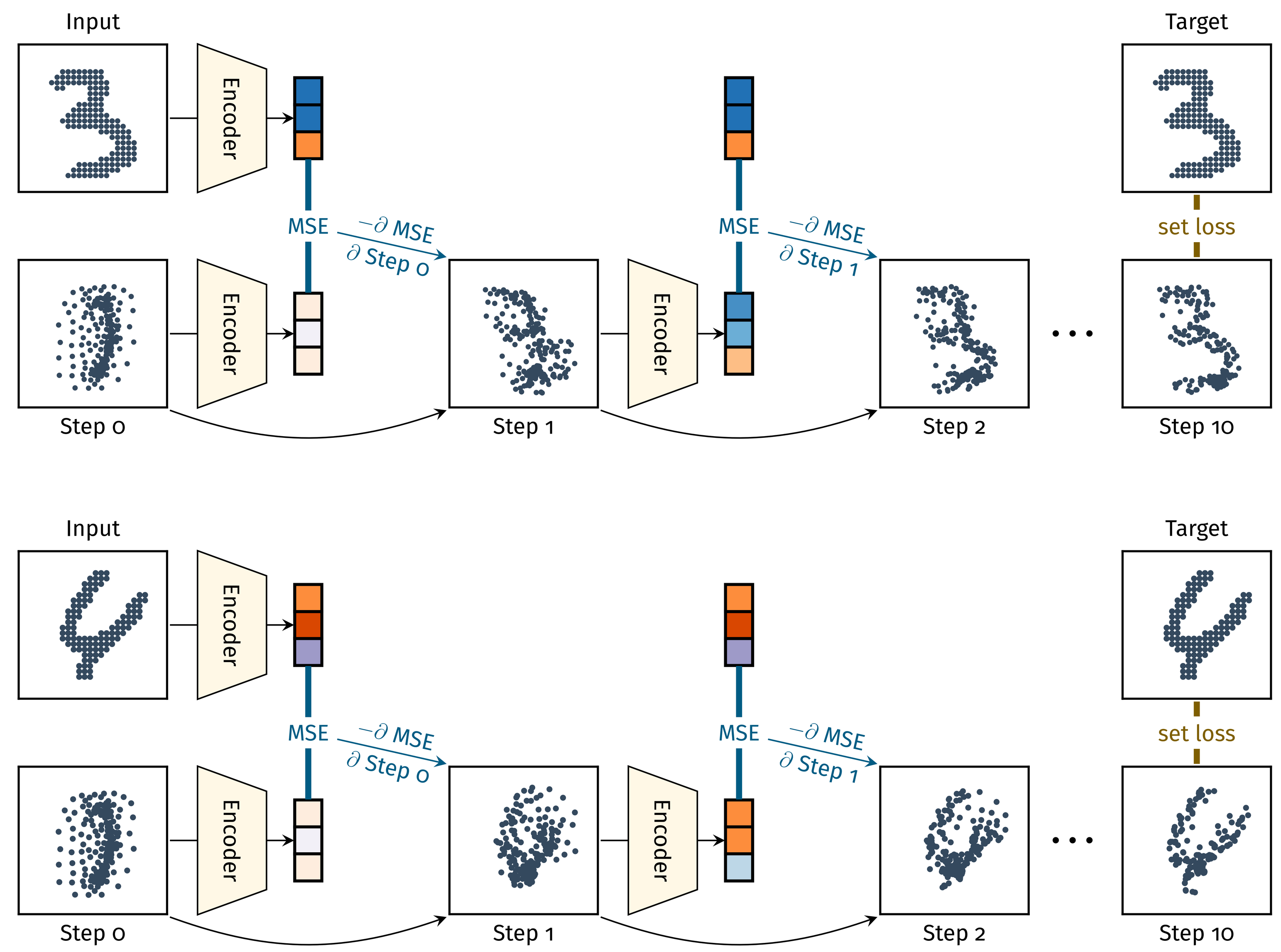
To **predict a set** from a vector, use gradient descent to find a set that **encodes** to that vector.

Code and pre-trained models available at `https://github.com/Cyanogenoid/dspn`

The idea

- *Similar* set inputs encode to *similar* feature vectors.
- *Different* set inputs encode to *different* feature vectors.

→ Minimise the difference between predicted and target set by minimising the difference between their feature vectors.



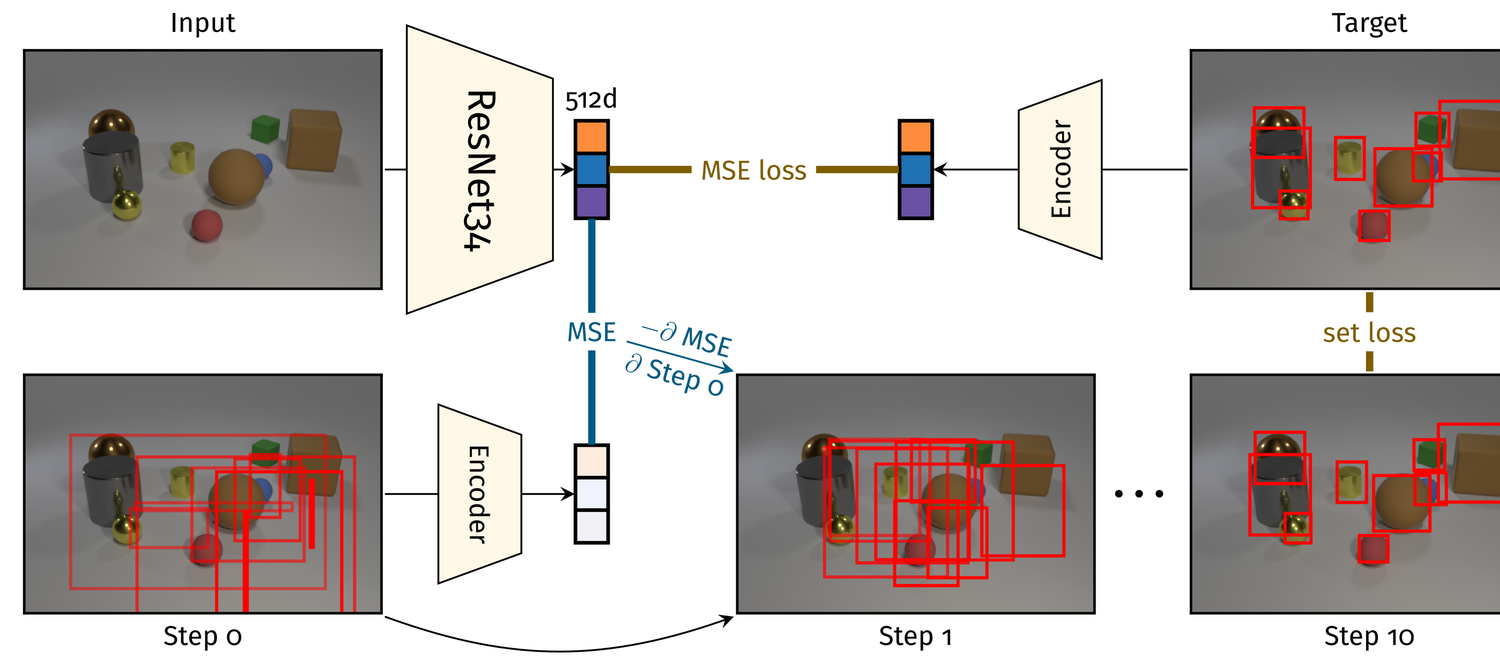
- Train (shared) encoder weights by minimising the set loss.
- Gradients of permutation-*invariant* functions are *equivariant*.

→ All gradient updates $\partial\mathrm{MSE}/\partial\mathrm{set}$ don't rely on the order of the set.

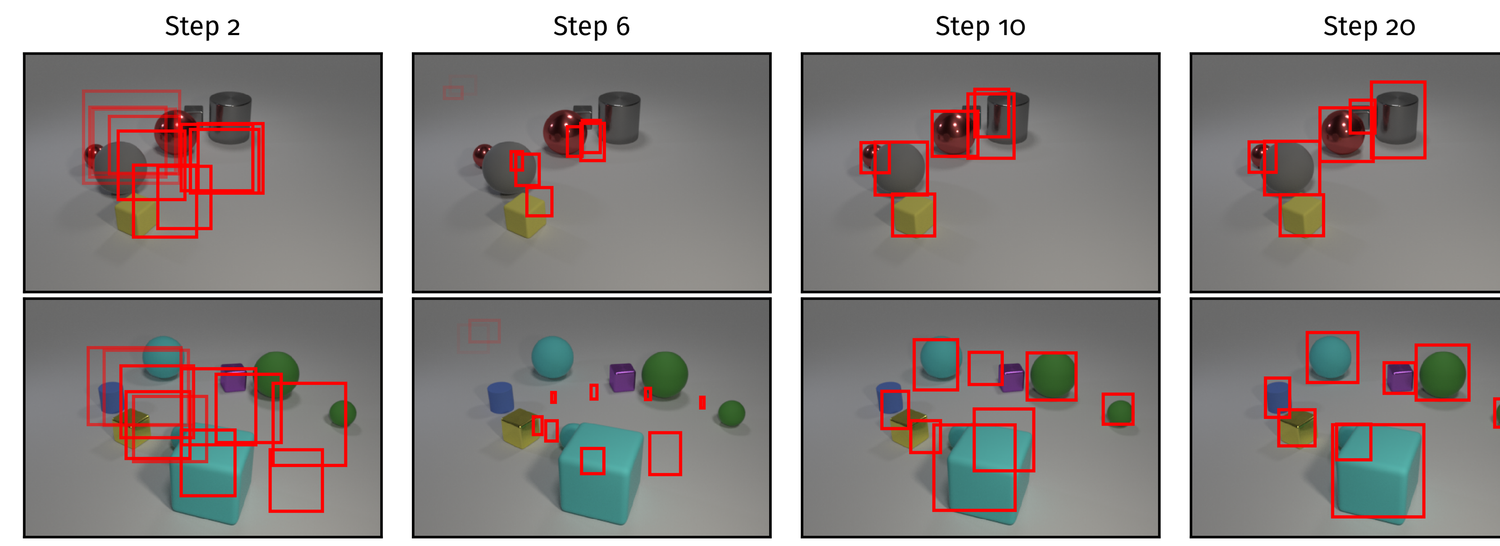
→ Our model is completely **unordered**, exactly what we wanted!

Bounding box set prediction

Bounding box prediction	AP_{50}	AP_{90}	AP_{95}	AP_{98}	AP_{99}
MLP baseline	$99.3_{\pm 0.2}$	$94.0_{\pm 1.9}$	$57.9_{\pm 7.9}$	$0.7_{\pm 0.2}$	$0.0_{\pm 0.0}$
RNN baseline	$99.4_{\pm 0.2}$	$94.9_{\pm 2.0}$	$65.0_{\pm 10.3}$	$2.4_{\pm 0.0}$	$0.0_{\pm 0.0}$
DSPN (train 10 steps, eval 10 steps)	$98.8_{\pm 0.3}$	$94.3_{\pm 1.5}$	$85.7_{\pm 3.0}$	$\mathbf{34.5}_{\pm 5.7}$	$\mathbf{2.9}_{\pm 1.2}$
DSPN (train 10 steps, eval 20 steps)	$\mathbf{99.8}_{\pm 0.0}$	$\mathbf{98.7}_{\pm 1.1}$	$\mathbf{86.2}_{\pm 7.2}$	$24.3_{\pm 8.0}$	$1.4_{\pm 0.9}$
DSPN (train 10 steps, eval 30 steps)	$\mathbf{99.8}_{\pm 0.1}$	$96.7_{\pm 2.4}$	$75.5_{\pm 12.3}$	$17.4_{\pm 7.7}$	$0.9_{\pm 0.7}$

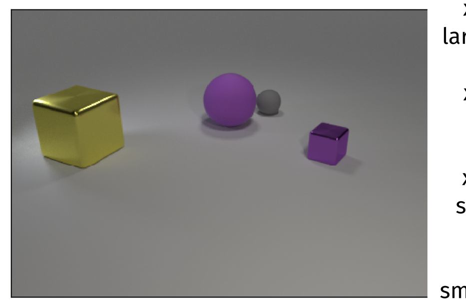


- Simply replace input encoder with ConvNet image encoder.
- Add MSE loss to set loss when training the encoder and ResNet weights.
  - Forces minimisation of MSE to converge to something sensible.



Object detection

Object attribute prediction	AP_{∞}	AP_{1}	$AP_{0.5}$	$AP_{0.25}$	$AP_{0.125}$
MLP baseline	$3.6_{\pm 0.5}$	$1.5_{\pm 0.4}$	$0.8_{\pm 0.3}$	$0.2_{\pm 0.1}$	$0.0_{\pm 0.0}$
RNN baseline	$4.0_{\pm 1.9}$	$1.8_{\pm 1.2}$	$0.9_{\pm 0.5}$	$0.2_{\pm 0.1}$	$0.0_{\pm 0.0}$
DSPN (train 10 steps, eval 10 steps)	$72.8_{\pm 2.3}$	$59.2_{\pm 2.8}$	$39.0_{\pm 4.4}$	$12.4_{\pm 2.5}$	$1.3_{\pm 0.4}$
DSPN (train 10 steps, eval 20 steps)	$84.0_{\pm 4.5}$	$80.0_{\pm 4.9}$	$\mathbf{57.0}_{\pm 12.1}$	$\mathbf{16.6}_{\pm 9.0}$	$\mathbf{1.6}_{\pm 0.9}$
DSPN (train 10 steps, eval 30 steps)	$\mathbf{85.2}_{\pm 4.8}$	$\mathbf{81.1}_{\pm 5.2}$	$47.4_{\pm 17.6}$	$10.8_{\pm 9.0}$	$0.6_{\pm 0.7}$

Input	Step 5	Step 10	Step 20	Target
	x, y, z = (-0.14, 1.16, 3.57) large purple rubber sphere	x, y, z = (-2.33, -2.41, 0.73) large yellow metal cube	x, y, z = (-2.33, -2.42, 0.78) large yellow metal cube	x, y, z = (-2.42, -2.40, 0.70) large yellow metal cube
	x, y, z = (0.01, 0.12, 3.42) large gray metal cube	x, y, z = (-1.20, 1.27, 0.67) large purple rubber sphere	x, y, z = (-1.21, 1.20, 0.65) large purple rubber sphere	x, y, z = (-1.18, 1.25, 0.70) large purple rubber sphere
	x, y, z = (0.67, 0.65, 3.38) small purple metal cube	x, y, z = (-0.96, 2.54, 0.36) small gray rubber sphere	x, y, z = (-0.96, 2.59, 0.36) small gray rubber sphere	x, y, z = (-1.02, 2.61, 0.35) small gray rubber sphere
	x, y, z = (0.67, 1.14, 2.96) small purple rubber sphere	x, y, z = (1.61, 1.57, 0.36) small yellow metal cube	x, y, z = (1.58, 1.62, 0.38) small purple metal cube	x, y, z = (1.74, 1.53, 0.35) small purple metal cube