Deep Set Prediction Networks

Yan Zhang, Jonathon Hare, Adam Prügel-Bennett

University of Southampton

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

Set prediction

- Predicting sets means:
  - object detection (image to set of objects)
  - 3d shape inference (image to set of 3d points)
  - molecule generation (vector to set of nodes and edges)
  - clustering (set to set-of-sets)
- This paper is about the **vector to set** mapping, useful for all these applications.
- Existing approaches suffer from **responsibility problem**. *Explained at **FSPool** poster in this workshop!*
- Compared to normal object detection methods:
  - Anchor-free, fully end-to-end, no post-processing.

The idea

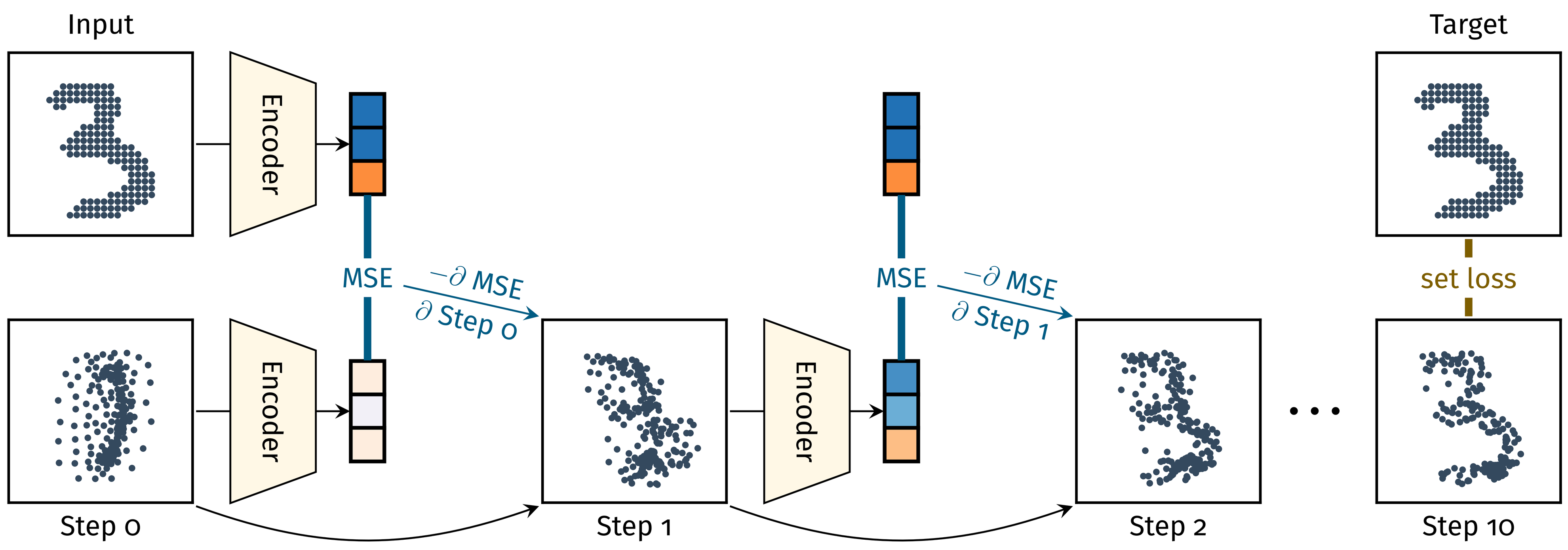
- To avoid responsibility problem, we want a model that has **unordered outputs**.
- *Similar* set inputs encode to *similar* feature vectors.
- *Different* set inputs encode to *different* feature vectors.

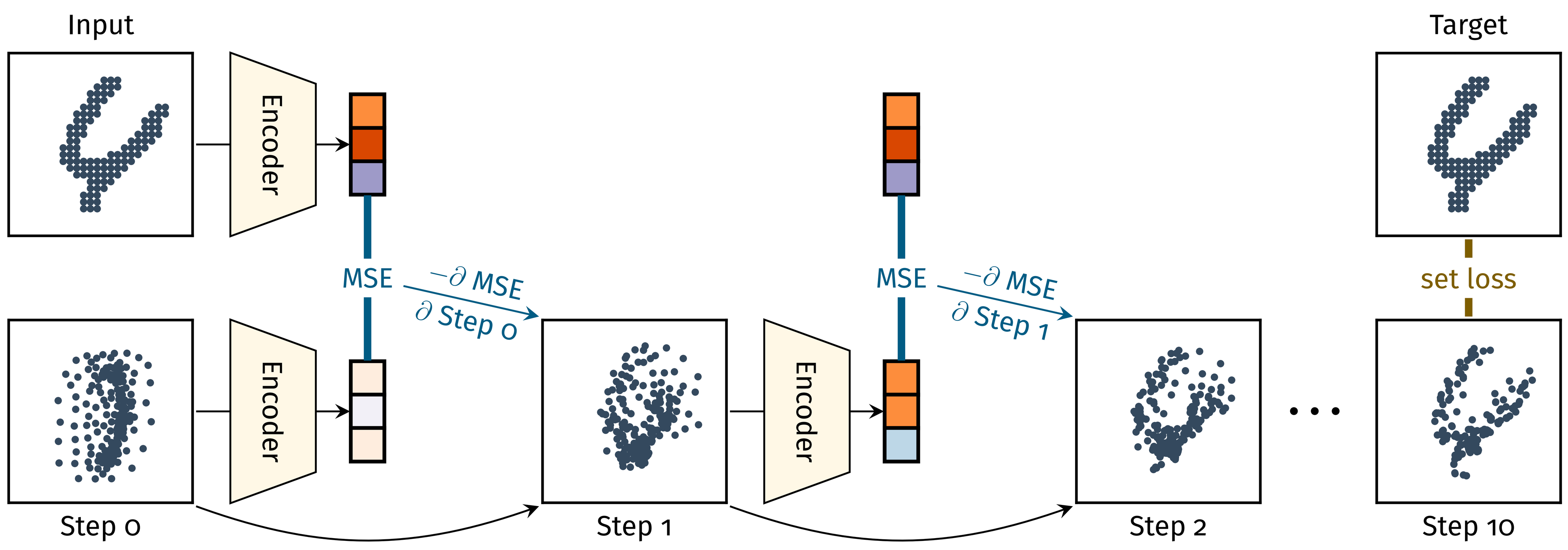
> Minimise the difference between predicted and target set by minimising the difference between their feature vectors.

Algorithm for auto-encoding

1. Start with "random" guess for our prediction.
2. For a fixed number of steps:
   1. Encode current set and input set into feature vectors.
   2. Compute MSE between the two feature vectors.
   3. Gradient descent on MSE by changing current set.

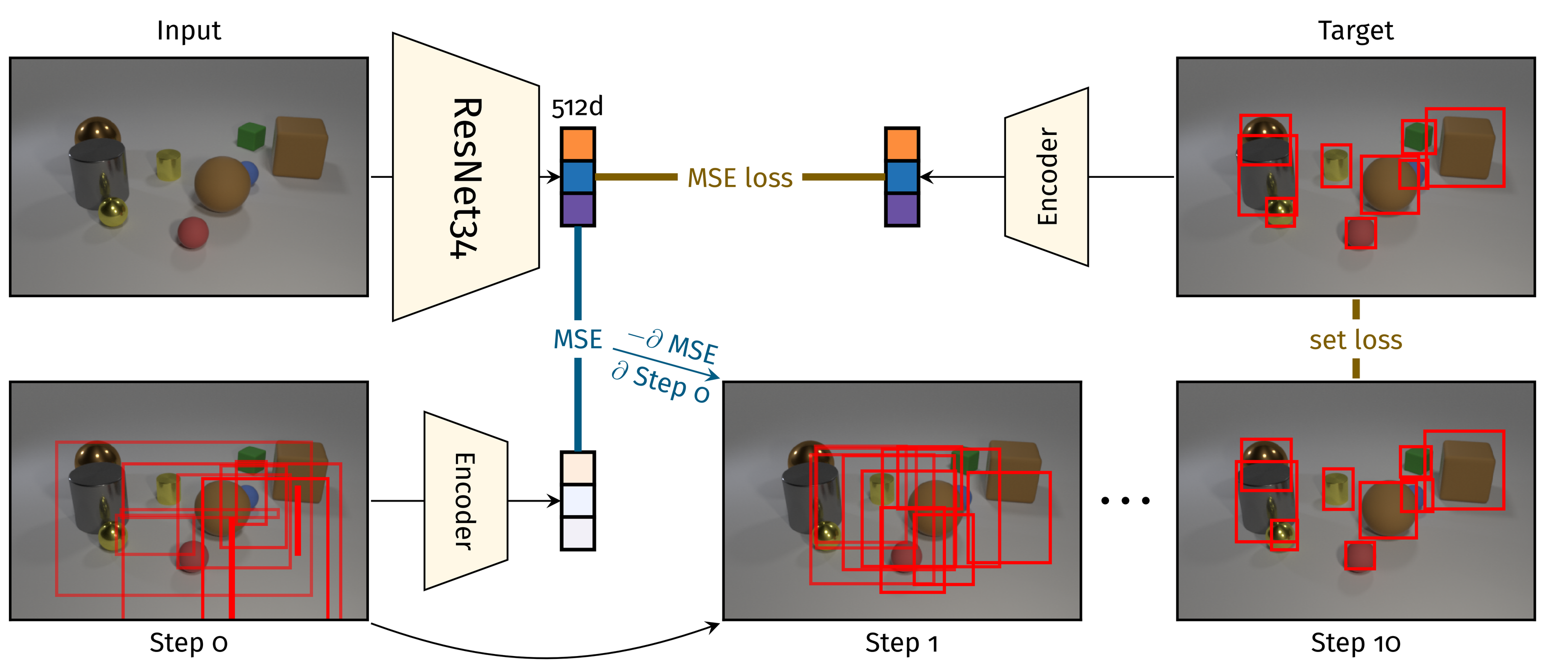
- Train (shared) encoder weights by minimising the set loss, differentiating through the algorithm.
- Gradients of permutation-*invariant* functions are always **permutation-equivariant**.
  - > All gradient updates $\partial \text{MSE} / \partial \text{set}$ don't rely on the order of the set.
  - > Our model is completely **unordered**!



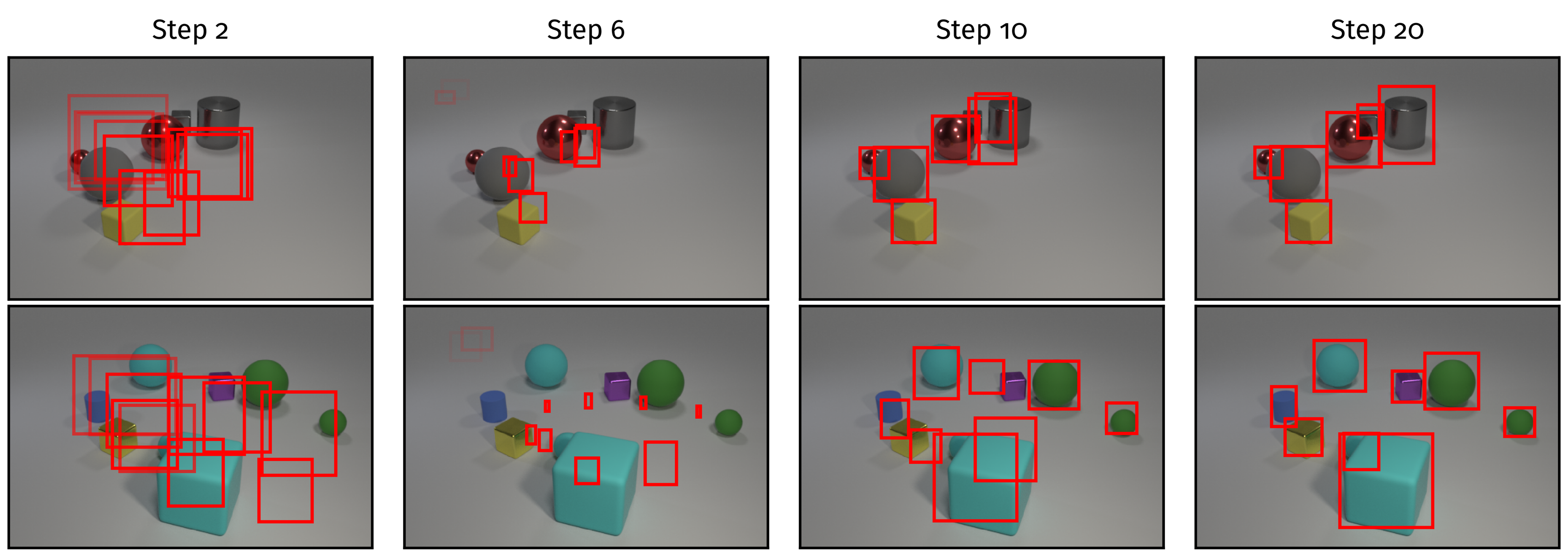


Bounding box prediction

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	$99.4_{\pm 0.2}$	$94.9_{\pm 2.0}$	$65.0_{\pm 10.3}$	$2.4_{\pm 0.0}$	$0.0_{\pm 0.0}$
Ours (train 10 steps, eval 10 steps)	$98.8_{\pm 0.3}$	$94.3_{\pm 1.5}$	$85.7_{\pm 3.0}$	$\mathbf{34.5}_{\pm 5.7}$	$\mathbf{2.9}_{\pm 1.2}$
Ours (train 10 steps, eval 20 steps)	$\mathbf{99.8}_{\pm 0.0}$	$\mathbf{98.7}_{\pm 1.1}$	$\mathbf{86.2}_{\pm 7.2}$	$24.3_{\pm 8.0}$	$1.4_{\pm 0.9}$
Ours (train 10 steps, eval 30 steps)	$\mathbf{99.8}_{\pm 0.1}$	$96.7_{\pm 2.4}$	$75.5_{\pm 12.3}$	$17.4_{\pm 7.7}$	$0.9_{\pm 0.7}$

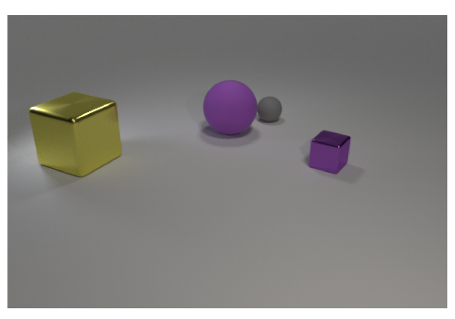


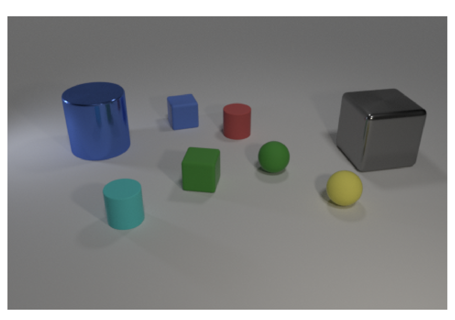
- Simply replace input encoder with ConvNet image encoder.
- Add MSE loss to set loss when training the encoder and ResNet weights.
  - Forces minimisation of MSE to converge to something sensible.



Object attribute prediction

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	$4.0_{\pm 1.9}$	$1.8_{\pm 1.2}$	$0.9_{\pm 0.5}$	$0.2_{\pm 0.1}$	$0.0_{\pm 0.0}$
Ours (train 10 steps, eval 10 steps)	$72.8_{\pm 2.3}$	$59.2_{\pm 2.8}$	$39.0_{\pm 4.4}$	$12.4_{\pm 2.5}$	$1.3_{\pm 0.4}$
Ours (train 10 steps, eval 20 steps)	$84.0_{\pm 4.5}$	$80.0_{\pm 4.9}$	$\mathbf{57.0}_{\pm 12.1}$	$\mathbf{16.6}_{\pm 9.0}$	$\mathbf{1.6}_{\pm 0.9}$
Ours (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
	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 = (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 = (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 = (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

Input	Step 5	Step 10	Step 20
	(0.22, 0.12, 3.47) small brown rubber cube	(-2.76, -1.42, 0.68) large blue metal cylinder	(-2.68, -1.64, 0.77) large blue metal cylinder
	(0.41, 0.11, 3.77) large gray metal cube	(-1.56, -0.61, 0.35) small blue rubber cylinder	(-2.43, 0.03, 0.34) small blue rubber cube
	(0.50, 0.44, 3.61) small gray rubber cube	(-1.08, 0.23, 0.33) small green rubber cube	(-1.00, 1.18, 0.33) small red rubber cylinder
	(0.83, 0.53, 3.45) small cyan rubber sphere	(-0.07, 0.97, 0.36) small green rubber cylinder	(0.21, -2.88, 0.40) small cyan rubber cylinder
	(0.86, 0.85, 3.50) small gray rubber sphere	(0.28, -2.44, 0.49) small cyan rubber cylinder	(-0.01, -1.00, 0.46) small green rubber cube
	(1.86, 2.34, 3.80) large gray metal cube	(1.36, -0.63, 0.38) small green rubber sphere	(0.99, 0.17, 0.37) small green rubber sphere
	(1.97, 0.55, 3.61) small green rubber sphere	(2.01, 3.07, 0.65) large gray metal cube	(1.97, 2.89, 0.39) large gray metal cube
		(2.69, 0.63, 0.34) small yellow rubber sphere	(2.87, 0.51, 0.25) small yellow rubber sphere