Pointer Networks

Learning to choose elements from an input

Oriol Vinyals, Meire Fortunato, Navdeep Jaitly

NeurIPS 2015

Learning a combinatorial solver

Training data

(problem instance P, solution C)

A neural model predicts a solution for a new instance.

The training distribution supplies patterns the model can learn.

The output must represent a discrete, structured solution.

A solution can be a sequence of indices

Input position	1	2	3	4
Input value	P4	P1	P3	P2

Target indices

$$C = (2, 4, 3, 1)$$

Selecting P_2, P_4, P_3, P_1

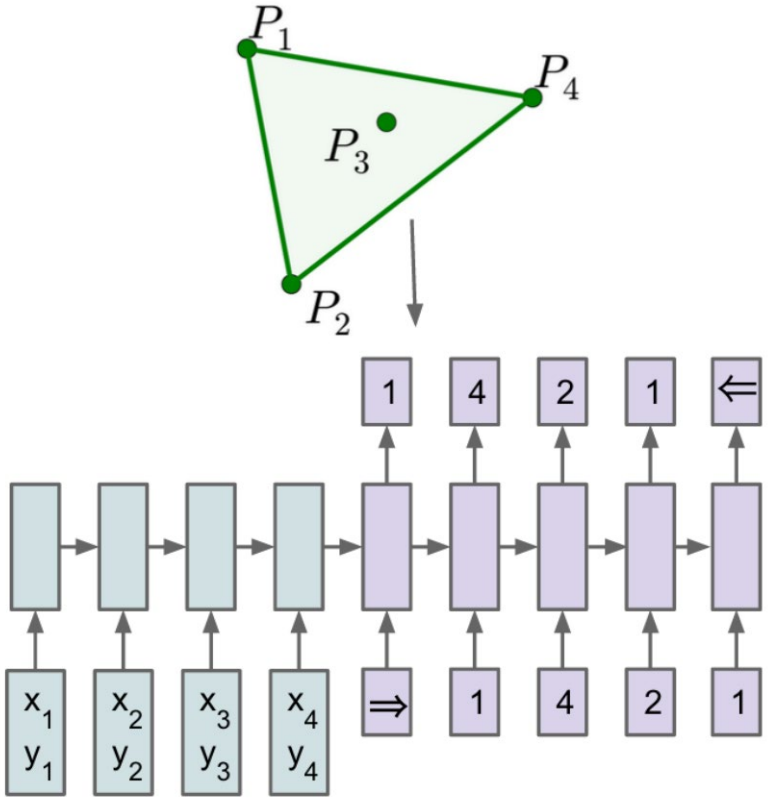
Variable output dictionaries

Input size	Candidate point indices at one output step
n = 4	{1, 2, 3, 4}
n = 7	{1, 2, 3, 4, 5, 6, 7}

Output length m(P) controls how many choices we make.

Dictionary size n controls which choices are available.

Sequence-to-sequence recap



Encoder: summarize the input in recurrent states.

Decoder: generate a sequence using previous choices.

A fixed output softmax assigns probability to a fixed vocabulary.

Paper, §2.1 and Figure 1(a)

The conditional sequence model

$$p(\mathcal{C}^{\mathcal{P}}|\mathcal{P};\theta) = \prod_{i=1}^{m(\mathcal{P})} p(C_i|C_1,\ldots,C_{i-1},\mathcal{P};\theta).$$

Symbol	Meaning
$P = (P_1, \ldots, P_n)$	Input sequence
$C = (C_1, \ldots, C_{m(p)})$	Output sequence of indices
$C_{(<i)}$	Choices made before output step i

Paper, Eq. (1) and Eq. (2)

Content-based input attention

e_j: encoder state for input j d_i: decoder state at output step i

$$
\begin{aligned}
u_j^i &= v^T \tanh(W_1 e_j + W_2 d_i) \quad j \in (1, \ldots, n) \\
a_j^i &= \text{softmax}(u_j^i) \quad j \in (1, \ldots, n) \\
d_i' &= \sum_{j=1}^{n} a_j^i e_j
\end{aligned}
$$

Each score compares one input state with the decoder state.

The weighted sum d'_i supplies context for output prediction.

Paper, §2.2, Eq. (3)

The pointer distribution

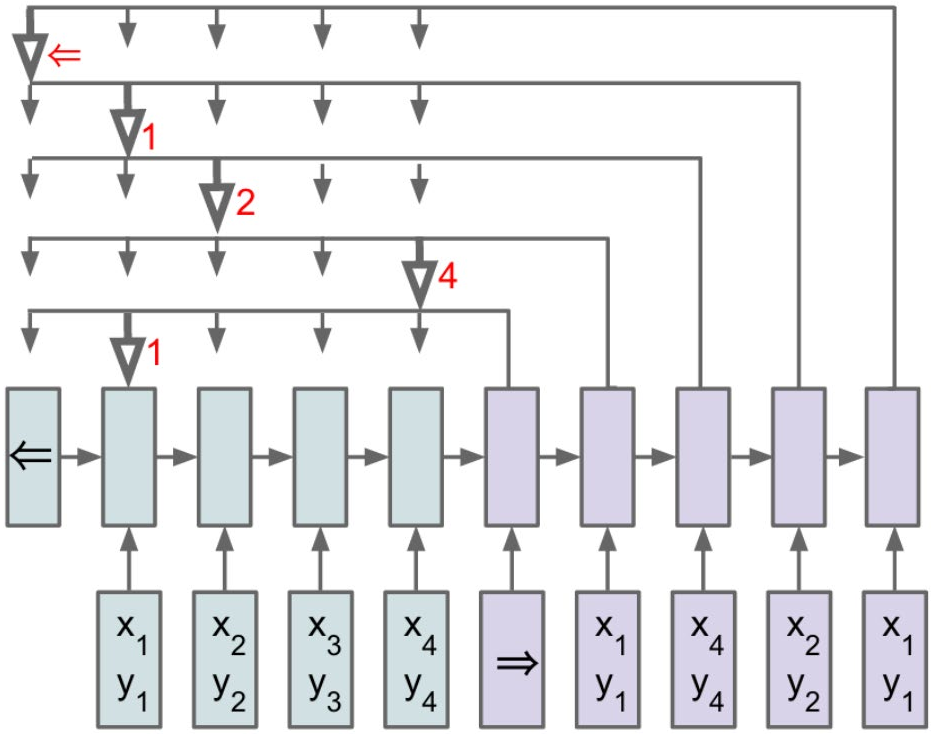
$$
\begin{aligned}
u_j^i &= v^T \tanh(W_1 e_j + W_2 d_i) \quad j \in (1, \ldots, n) \\
p(C_i | C_1, \ldots, C_{i-1}, \mathcal{P}) &= \text{softmax}(u^i)
\end{aligned}
$$

Mechanism	Role of the attention probabilities
Input attention	Weights in a summary of encoder states
Pointer Network	Probabilities of selecting input indices

The same scoring parameters can score 4, 7, or 100 input positions.

Paper, §2.3

One decoding step in a Pointer Network



1. Score every encoded input.

2. Select an input index C_i.

3. Feed $P[C_i]$ into the next decoder step.

Paper, §2.3 and Figure 1(b)

Worked example: the first pointer

Position j	1	2	3	4
Input P_j	4	1	3	2
$p(C_1 = j \mid P)$	0.03	**0.80**	0.07	0.10

$C_1 = 2$ selected input $P_2 = 1$

The output is an index. The next decoder input is the selected element.

Teaching example. Probabilities are illustrative.

Class exercise: the next choice

After choosing index 2, suppose the next distribution is:

Position j	1	2	3	4
$p(C_2 = j \mid C_1, P)$	0.05	0.00	0.15	0.80

Which index comes next, and which value does it select?

How many point scores are needed when n = 7?

Does pointing alone guarantee a permutation?

Teaching example with an optional no-repeat mask for sorting.

Supervised training

$$\theta^* = \arg\max_{\theta} \sum_{\mathcal{P},\mathcal{C}^{\mathcal{P}}} \log p(\mathcal{C}^{\mathcal{P}} | \mathcal{P}; \theta),$$

Teacher forcing

Use the correct previous choice to construct the next decoder input.

Paper, Eq. (2), §2.3, and §4.1

The experimental setting

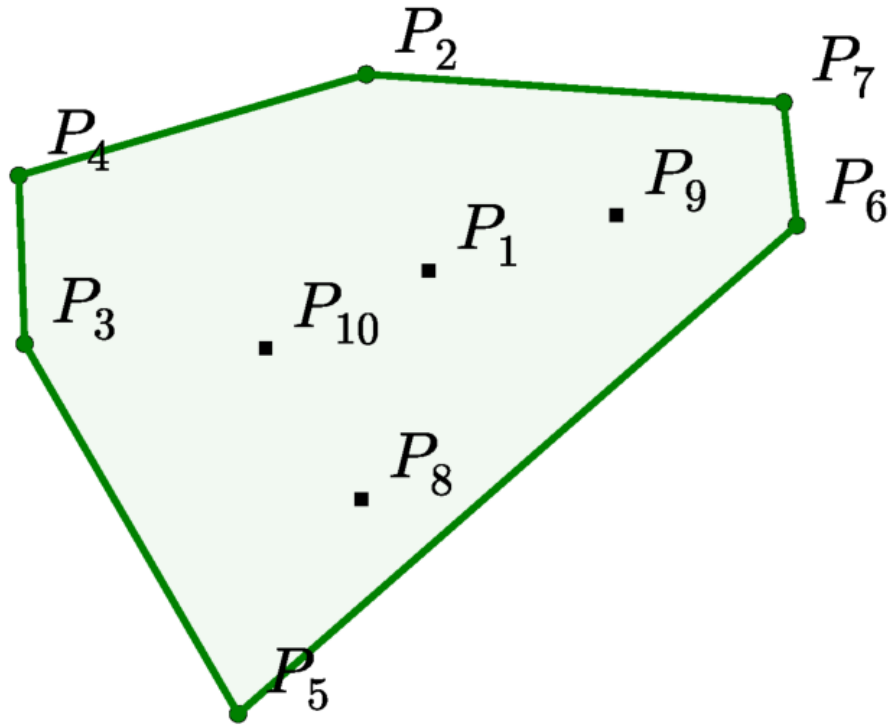
Inputs: 2D point sets sampled uniformly from $[0, 1]^2$

Task	Output representation	Label source
Convex hull	Boundary vertex indices	Exact geometric algorithm
Delaunay triangulation	Vertex triples for triangles	Exact geometric algorithm
Euclidean TSP	Permutation of city indices	Exact or heuristic tours

Single-layer LSTMs with 256 or 512 hidden units. 1M training pairs.

Paper, §3 and §4.1

Convex hull representation



Boundary indices

(2, 4, 3, 5, 6, 7, 2)

Start with the lowest-index hull vertex.

Traverse counter-clockwise.

Add start and end symbols for sequence boundaries.

Paper, §3.1 and Figure 2(a)

Convex hull results at n = 50

Model	Exact polygon	Area covered
LSTM	1.9%	FAIL
LSTM with attention	38.9%	99.7%
Pointer Network	**72.6%**	**99.9%**

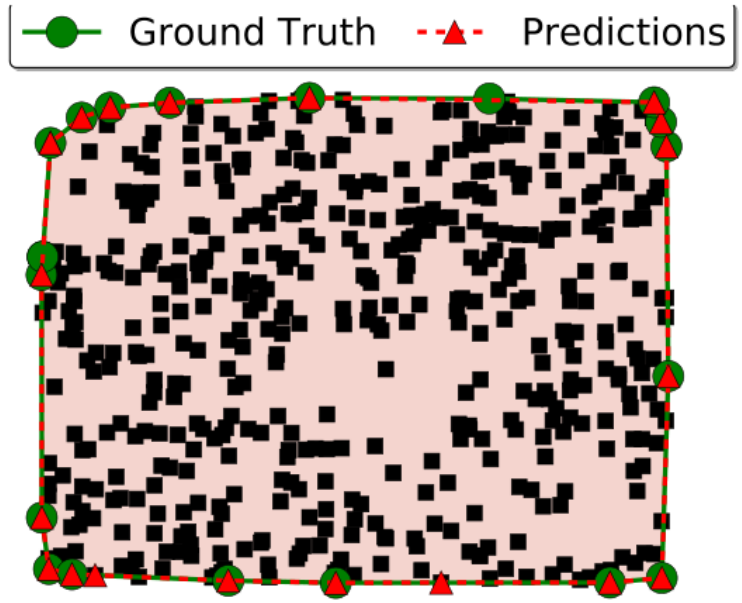
All three models train and test on 50 input points.

Area measures geometric coverage. Exact accuracy checks the full polygon.

Paper, Table 1 and §4.2

Generalization to larger convex hull inputs

Test n	Exact polygon	Area
50	69.6%	99.9%
100	50.3%	99.9%
200	22.1%	99.9%
500	1.3%	99.2%

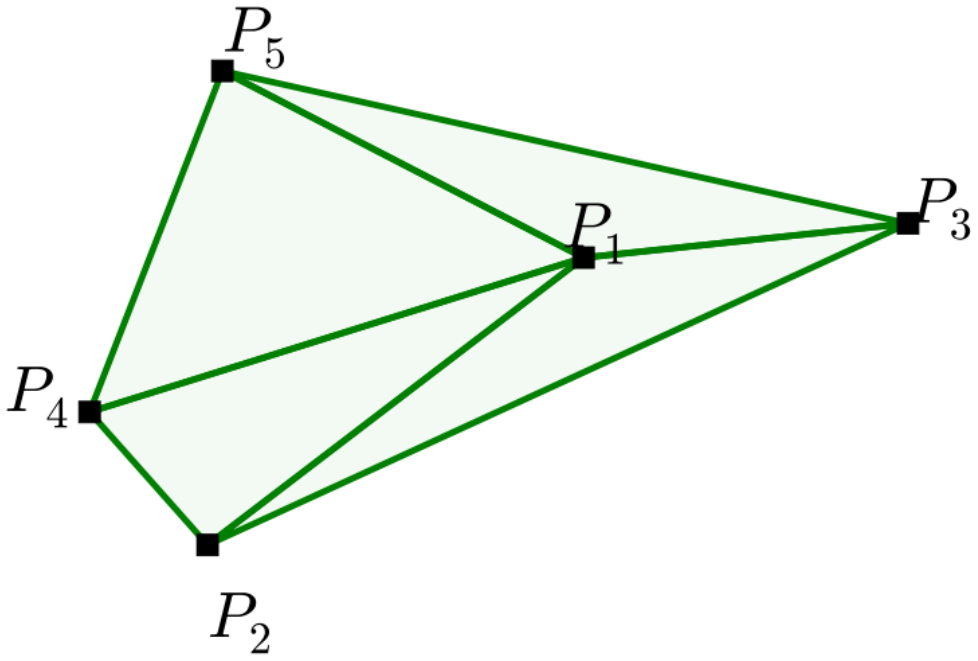


Training sizes: 5-50

Large-input area coverage can hide incorrect boundary vertices.

Paper, Table 1 and Figure 3(d)

Delaunay triangulation representation



Each triangle has an empty circumcircle.

(1, 2, 4), (1, 4, 5),

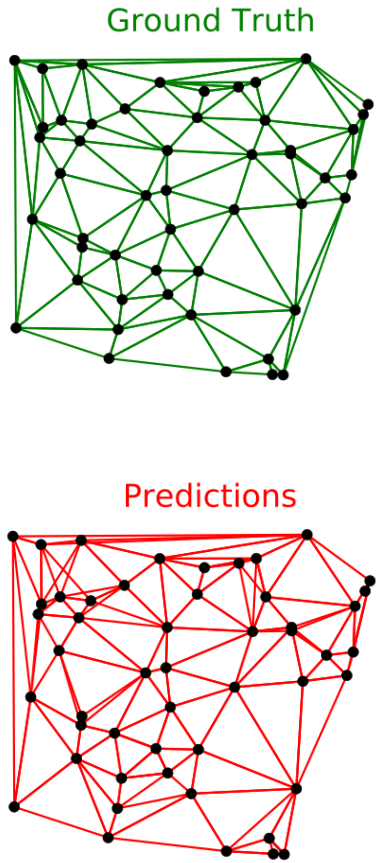
(1, 3, 5), (1, 2, 3)

Training fixes an order for vertices and triangles to reduce ambiguity.

Paper, §3.2 and Figure 2(b)

Delaunay accuracy drops as the input grows

n	Exact result	Triangle coverage
5	80.7%	93.0%
10	22.6%	81.3%
50	0%	52.8%



Many correct triangles can still form an incorrect triangulation.

Paper, §4.3 and Figure 3(b,e)

Travelling Salesman Problem

Find a shortest tour that visits each city once and returns to its start.

$$L(C, P) = \sum_{i=1}^{n} \|P[C_i] - P[C_{i+1}]\|_2$$

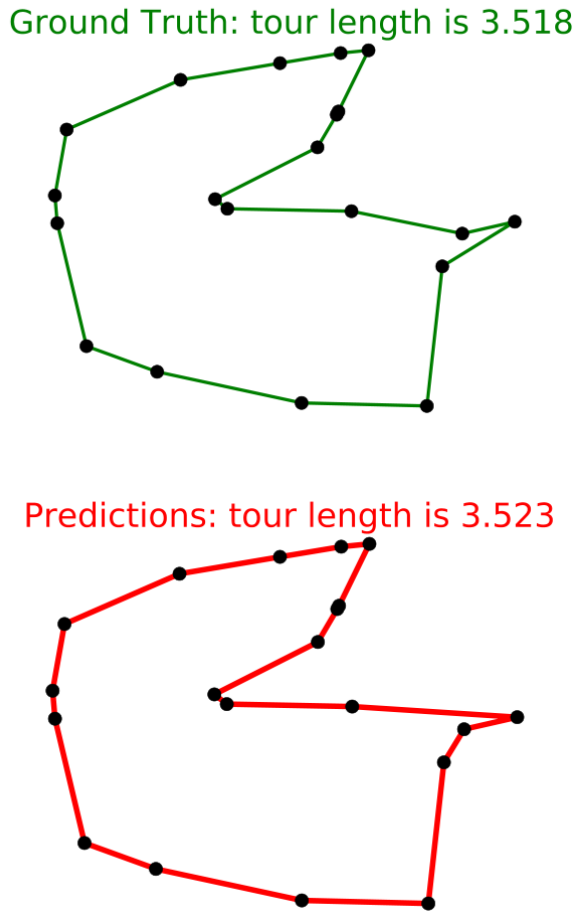
C is a permutation and $C_{n+1} = C_1$.

Up to 20 cities: exact Held-Karp solutions provide labels.

At 50 cities: heuristic algorithms provide labels.

TSP solution quality within the training range

n	Optimal mean	Ptr-Net mean
5	2.12	2.12
10	2.87	2.87
20	3.83	3.88



Training sizes: 5-20

At n = 20, the mean tour length is close to optimal.

Paper, Table 2 and Figure 3(c,f)

TSP extrapolation and invalid tours

n	Ptr-Net mean	A3 mean
25	4.30	4.24
30	4.72	4.60
40	5.91	5.23
50	**7.66**	5.79

The length columns use valid-tour beam search. Training sizes: 5-20.

Supporting a larger output dictionary does not ensure good extrapolation.

Paper, Table 2 and §4.4

Key ideas

A shared attention scorer produces a distribution over input indices.

Supervised sequence learning turns solved instances into training data.

The experiments separate representational flexibility from solver reliability.

References

Vinyals, O., Fortunato, M., and Jaitly, N. (2015).
Pointer Networks. Advances in Neural Information Processing Systems 28.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014).
Sequence to Sequence Learning with Neural Networks. NeurIPS.

Bahdanau, D., Cho, K., and Bengio, Y. (2015).
Neural Machine Translation by Jointly Learning to Align and Translate. ICLR.