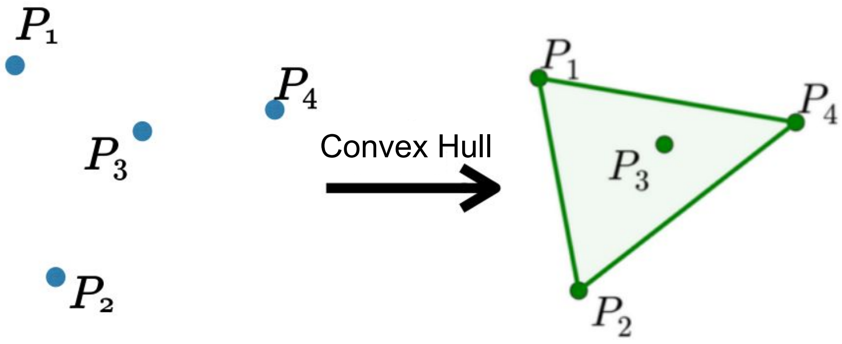
Pointer Networks

Pointer network slides by Keon Kim (36-43)

- https://www.slideshare.net/KeonKim/attention-mechanisms-with-tensorflow

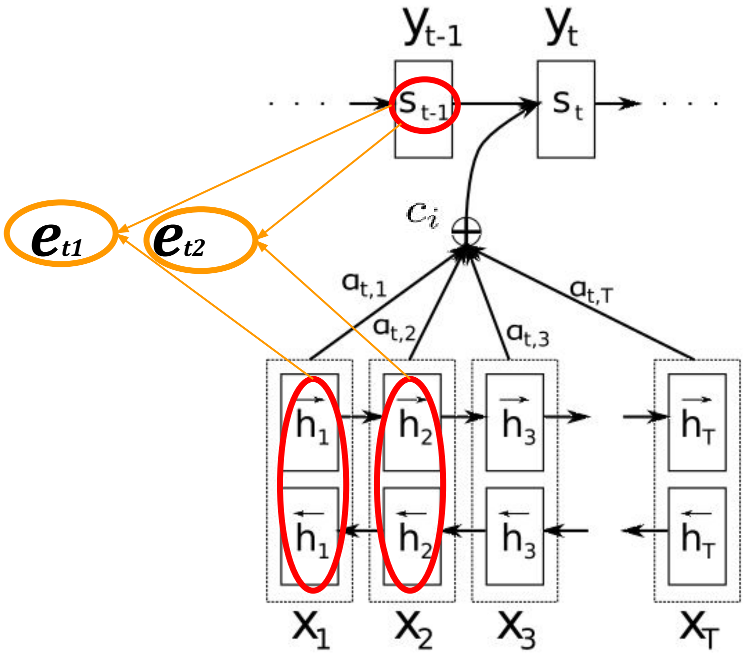


Pointer Networks Review

Pointer Networks ‘Point’ Input Elements!

In Ptr-Net, we do not blend the encoder state to propagate extra information to the decoder like standard attention mechanism.

But instead ...



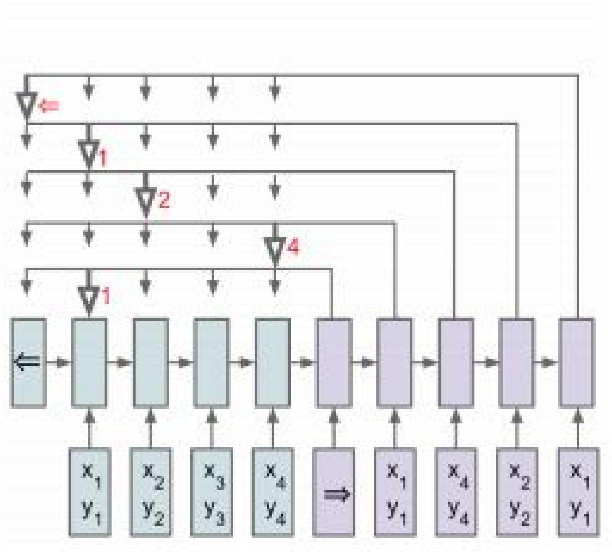
Standard Attention mechanism

Pointer Networks ‘Point’ Input Elements!

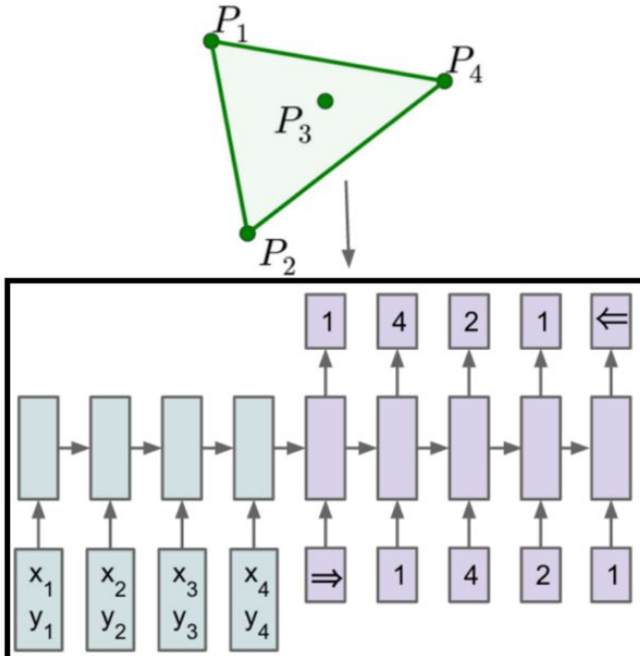
We use $e_{ij} = v_a^\top \tanh\left(W_a s_{i-1} + U_a h_j\right)$

as pointers to the input elements

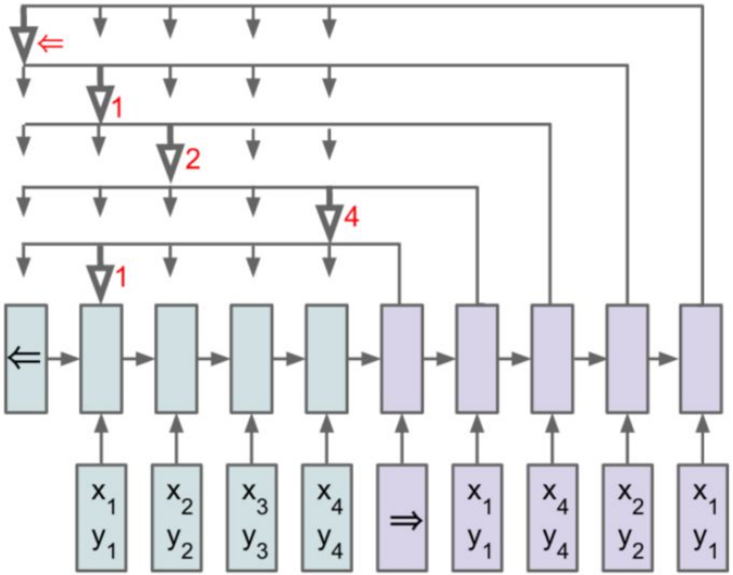
Ptr-Net approach specifically targets problems whose outputs are discrete and correspond to positions in the input

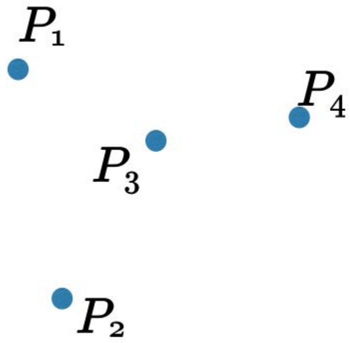


Pointer Network

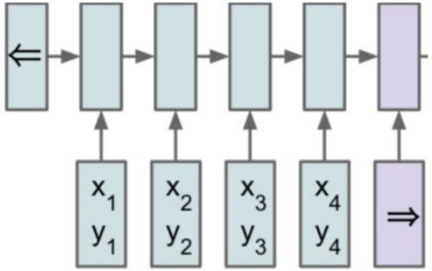


Another model for each
number of points?

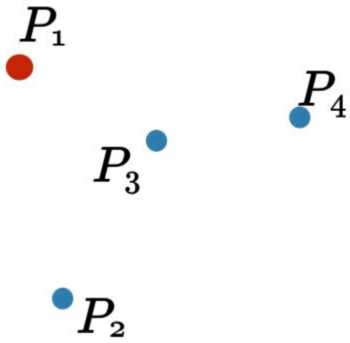




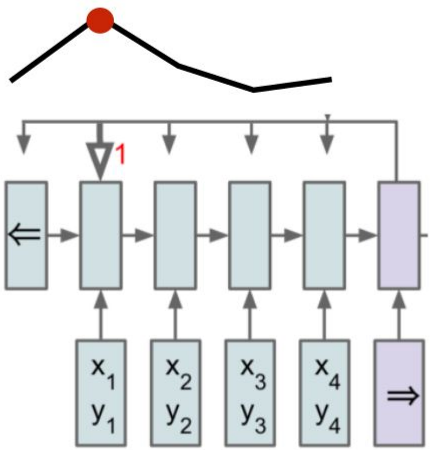
distribution of u



$$u_j^i = v^T \tanh(W_1 e_j + W_2 d_i)$$

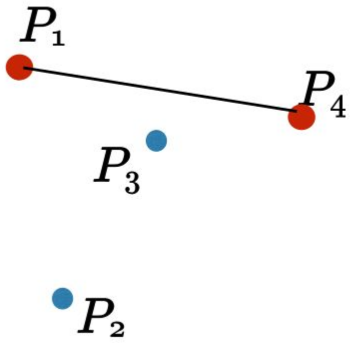


distribution of u

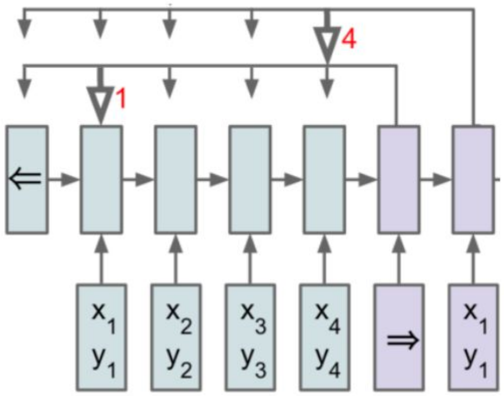


$$
\begin{aligned}
u_j^i &= v^T \tanh(W_1 e_j + W_2 d_i) \\
p(C_i | C_1, \ldots, C_{i-1}, \mathcal{P}) &= \mathrm{softmax}(u^i)
\end{aligned}
$$

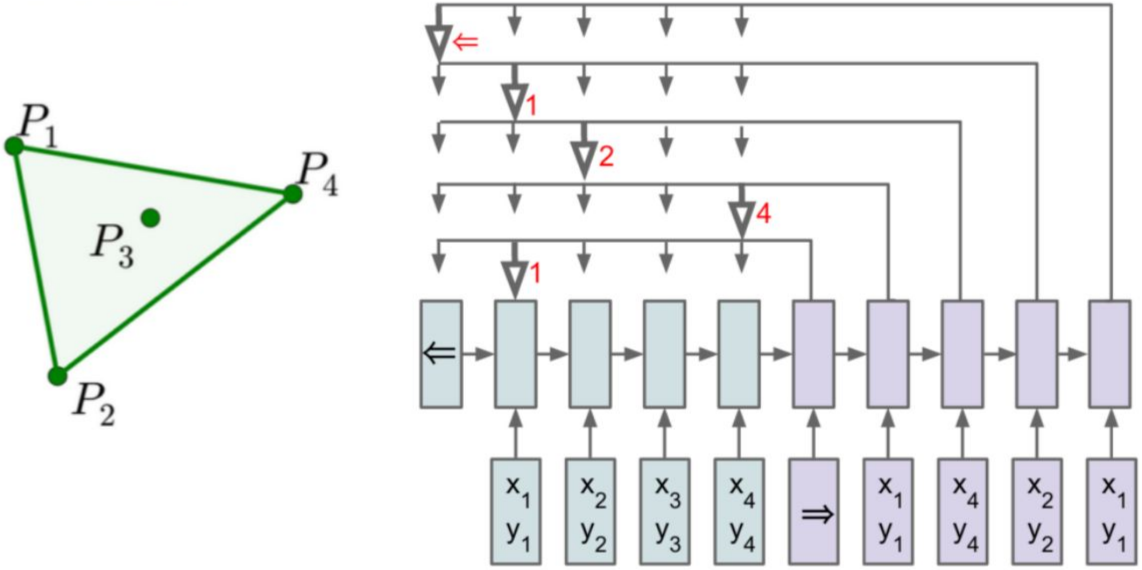
Distribution of the Attention is the Answer!



P_1
P_4
P_3
P_2



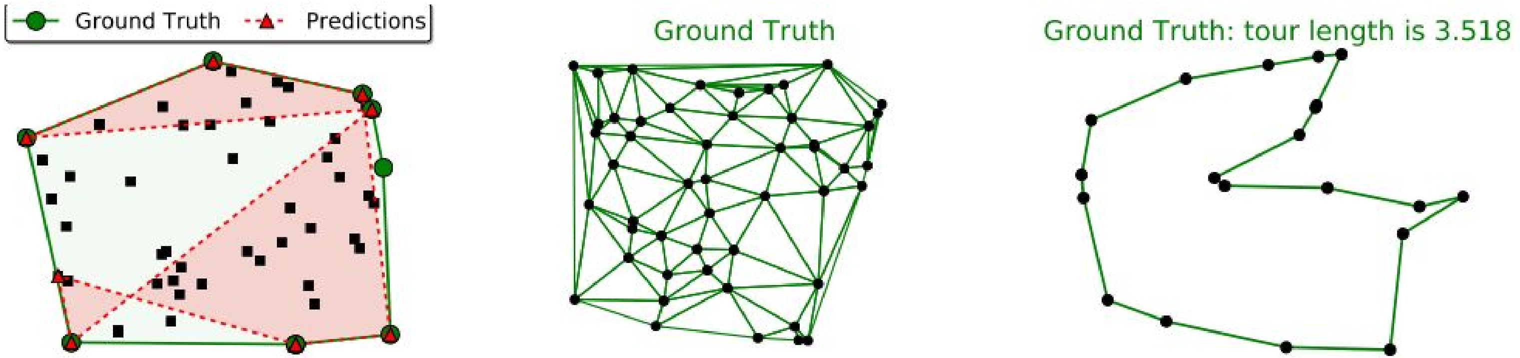
4
1
⇐
x_1 y_1
x_2 y_2
x_3 y_3
x_4 y_4
⇒
x_1 y_1



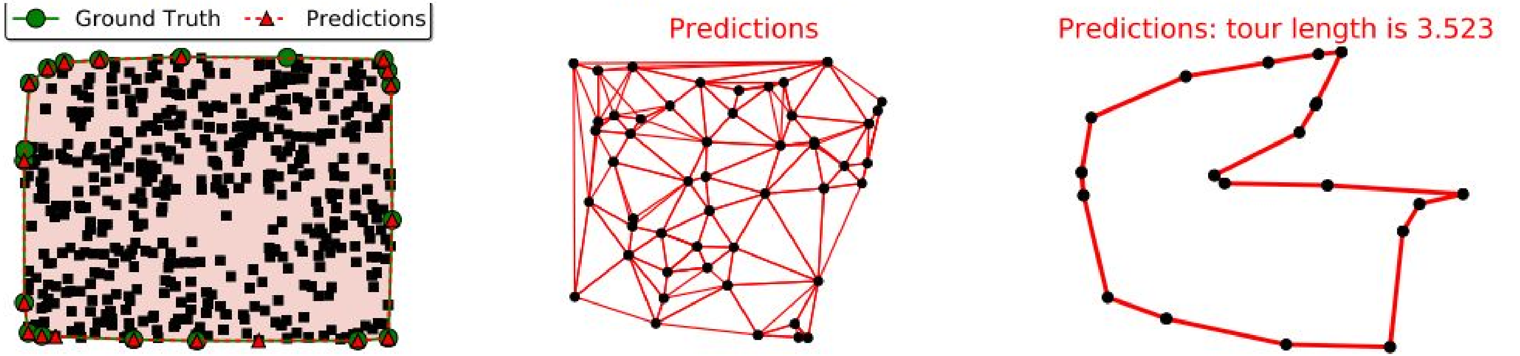
P_1
P_2
P_3
P_4
⇐
1
2
4
1
⇐
x_1 y_1
x_2 y_2
x_3 y_3
x_4 y_4
⇒
x_1 y_1
x_4 y_4
x_2 y_2
x_1 y_1

Method	trained n	n	Accuracy	Area
LSTM [1]	50	50	1.9%	FAIL
+attention [5]	50	50	38.9%	99.7%
Ptr-Net	50	50	72.6%	99.9%
LSTM [1]	5	5	87.7%	99.6%
Ptr-Net	5-50	5	92.0%	99.6%
LSTM [1]	10	10	29.9%	FAIL
Ptr-Net	5-50	10	87.0%	99.8%
Ptr-Net	5-50	50	69.6%	99.9%
Ptr-Net	5-50	100	50.3%	99.9%
Ptr-Net	5-50	200	22.1%	99.9%
Ptr-Net	5-50	500	1.3%	99.2%

Vinyals, O., Fortunato, M., & Jaitly, N. (2015). Pointer networks. In *Advances in Neural Information Processing Systems* (pp. 2692-2700).



(a) LSTM, m=50, n=50 (b) Truth, n=50 (c) Truth, n=20



(d) Ptr-Net, m=5-50, n=500 (e) Ptr-Net , m=50, n=50 (f) Ptr-Net , m=5-20, n=20

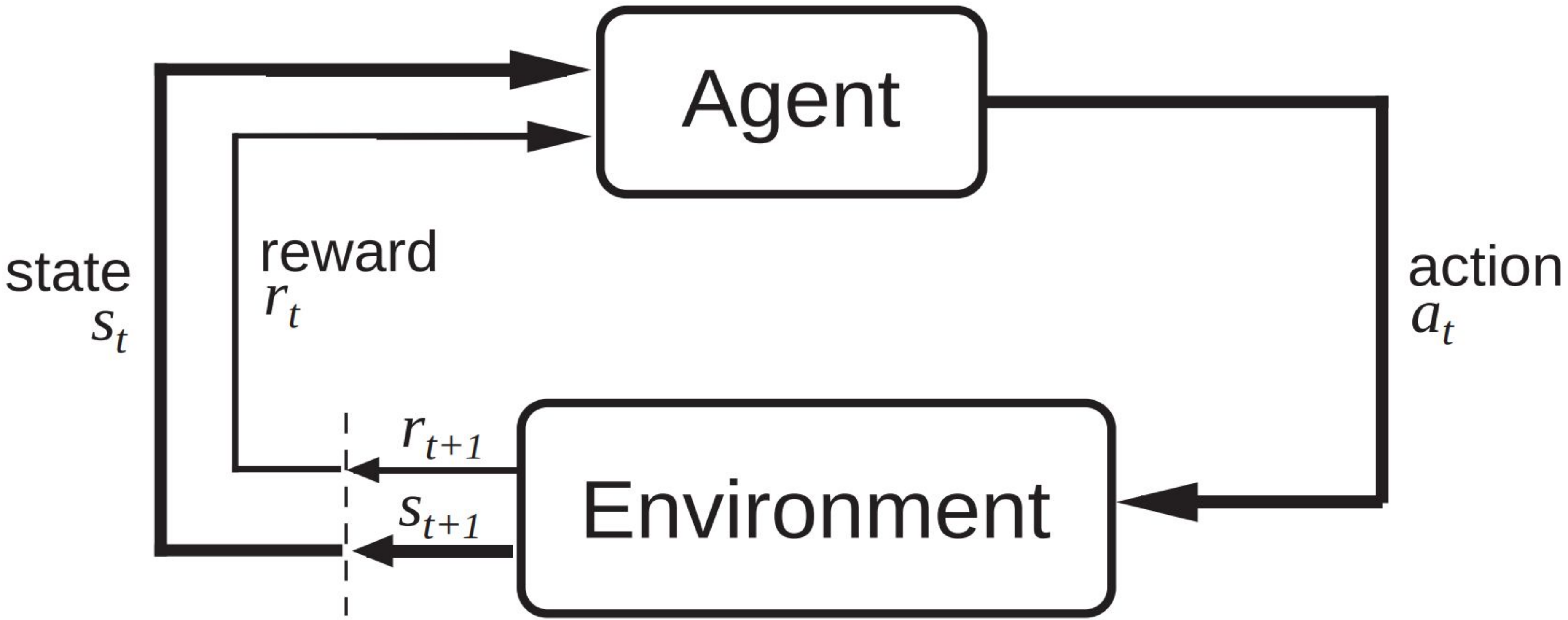
Vinyals, O., Fortunato, M., & Jaitly, N. (2015). Pointer networks. In *Advances in Neural Information Processing Systems* (pp. 2692-2700).

n	Optimal	A1	A2	A3	Ptr-Net
5	2.12	2.18	2.12	2.12	2.12
10	2.87	3.07	2.87	2.87	2.88
50 (A1 trained)	N/A	6.46	5.84	5.79	6.42
50 (A3 trained)	N/A	6.46	5.84	5.79	6.09
5 (5-20 trained)	2.12	2.18	2.12	2.12	2.12
10 (5-20 trained)	2.87	3.07	2.87	2.87	2.87
20 (5-20 trained)	3.83	4.24	3.86	3.85	3.88
25 (5-20 trained)	N/A	4.71	4.27	4.24	4.30
30 (5-20 trained)	N/A	5.11	4.63	4.60	4.72
40 (5-20 trained)	N/A	5.82	5.27	5.23	5.91
50 (5-20 trained)	N/A	6.46	5.84	5.79	7.66

Vinyals, O., Fortunato, M., & Jaitly, N. (2015). Pointer networks. In *Advances in Neural Information Processing Systems* (pp. 2692-2700).

Neural Combinatorial Optimization

with Reinforcement Learning



Sutton, Richard S., and Andrew G. Barto. Reinforcement learning: An introduction. Vol. 1. No. 1. Cambridge: MIT press, 1998.

REINFORCE with Baseline (episodic)

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta}), \forall a \in \mathcal{A}, s \in \mathcal{S}, \boldsymbol{\theta} \in \mathbb{R}^n$
Input: a differentiable state-value parameterization $\hat{v}(s,\mathbf{w}), \forall s \in \mathcal{S}, \mathbf{w} \in \mathbb{R}^m$
Parameters: step sizes $\alpha > 0$, $\beta > 0$

Initialize policy weights $\boldsymbol{\theta}$ and state-value weights $\mathbf{w}$
Repeat forever:
  Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \boldsymbol{\theta})$
  For each step of the episode $t = 0, \ldots, T-1$:
    $G_t \leftarrow$ return from step t
    $\delta \leftarrow G_t - \hat{v}(S_t,\mathbf{w})$
    $\mathbf{w} \leftarrow \mathbf{w} + \beta \delta \nabla_{\mathbf{w}} \hat{v}(S_t,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \cancel{\gamma^t} \delta \nabla_{\boldsymbol{\theta}} \log \pi(A_t|S_t, \boldsymbol{\theta})$

Sutton, Richard S., and Andrew G. Barto. Reinforcement learning: An introduction. Vol. 1. No. 1. Cambridge: MIT press, 1998.

Algorithm 1 Actor-critic training

procedure $\textsc{Train}$(training set S, number of training steps T, batch size B)
  Initialize pointer network params θ
  Initialize critic network params θ_v
  for $t = 1$ to T **do**
    $s_i \sim \textsc{SampleInput}(S)$ for $i \in \{1, \ldots, B\}$
    $\pi_i \sim \textsc{SampleSolution}(p_\theta(.|s_i))$ for $i \in \{1, \ldots, B\}$
    $b_i \leftarrow b_{\theta_v}(s_i)$ for $i \in \{1, \ldots, B\}$
    $g_\theta \leftarrow \frac{1}{B}\sum_{i=1}^{B}(L(\pi_i|s_i) - b_i)\nabla_\theta \log p_\theta(\pi_i|s_i)$
    $\mathcal{L}_v \leftarrow \frac{1}{B}\sum_{i=1}^{B}\|b_i - L(\pi_i)\|_2^2$
    $\theta \leftarrow \textsc{Adam}(\theta, g_\theta)$
    $\theta_v \leftarrow \textsc{Adam}(\theta_v, \nabla_{\theta_v}\mathcal{L}_v)$
  end for
  return θ
end procedure

Bello, Irwan, et al. "Neural Combinatorial Optimization with Reinforcement Learning." *arXiv preprint arXiv:1611.09940* (2016).

Algorithm 2 Active Search

procedure ACTIVESEARCH(input s, θ, number of candidates K, B, α)
  $\pi \leftarrow$ RANDOMSOLUTION()
  $L_\pi \leftarrow L(\pi \mid s)$
  $n \leftarrow \lceil \frac{K}{B} \rceil$
  for $t = 1 \dots n$ **do**
    $\pi_i \sim$ SAMPLESOLUTION$(p_\theta(. \mid s))$ for $i \in \{1, \dots, B\}$
    $j \leftarrow$ ARGMIN$(L(\pi_1 \mid s) \dots L(\pi_B \mid s))$
    $L_j \leftarrow L(\pi_j \mid s)$
    if $L_j < L_\pi$ **then**
      $\pi \leftarrow \pi_j$
      $L_\pi \leftarrow L_j$
    end if
    $g_\theta \leftarrow \frac{1}{B} \sum_{i=1}^{B} (L(\pi_i \mid s) - b) \nabla_\theta \log p_\theta(\pi_i \mid s)$
    $\theta \leftarrow$ ADAM(θ, g_θ)
    $b \leftarrow \alpha \times b + (1 - \alpha) \times (\frac{1}{B} \sum_{i=1}^{B} b_i)$
  end for
  return π
end procedure

Bello, Irwan, et al. "Neural Combinatorial Optimization with Reinforcement Learning." *arXiv preprint arXiv:1611.09940* (2016).

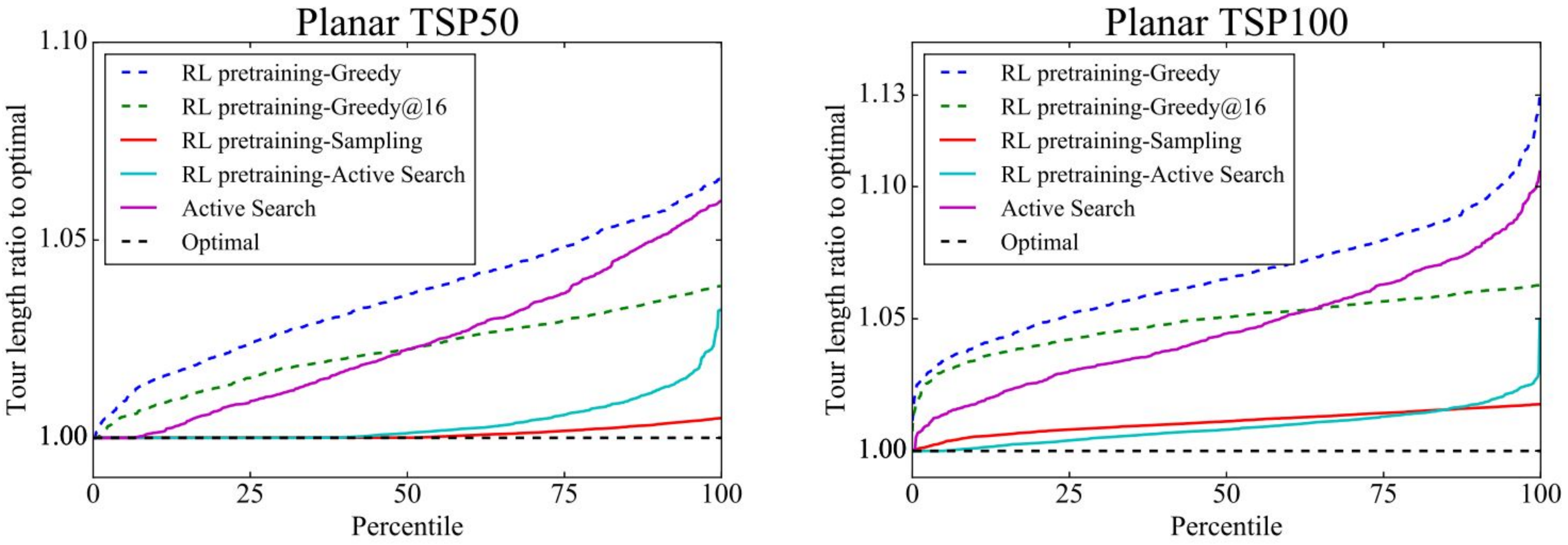
Table 2: Average tour lengths (lower is better). Results marked $^{(\dagger)}$ are from (Vinyals et al., 2015b).

Task	Supervised Learning	RL pretraining				AS	Christo -fides	OR Tools' local search	Optimal
		greedy	greedy@16	sampling	AS				
TSP20	$3.88^{(\dagger)}$	3.89	–	3.82	3.82	3.96	4.30	3.85	3.82
TSP50	$6.09^{(\dagger)}$	5.95	5.80	5.70	5.70	5.87	6.62	5.80	5.68
TSP100	10.81	8.30	7.97	7.88	7.83	8.19	9.18	7.99	7.77

Bello, Irwan, et al. "Neural Combinatorial Optimization with Reinforcement Learning." *arXiv preprint arXiv:1611.09940* (2016).

Task	# Solutions	RL pretraining		
		Sampling $T = 1$	Sampling $T = T^*$	Active Search
TSP50	128	5.80 (3.4s)	5.80 (3.4s)	5.80 (0.5s)
	1,280	5.77 (3.4s)	5.75 (3.4s)	5.76 (5s)
	12,800	5.75 (13.8s)	5.73 (13.8s)	5.74 (50s)
	128,000	5.73 (110s)	5.71 (110s)	5.72 (500s)
	1,280,000	5.72 (1080s)	5.70 (1080s)	5.70 (5000s)
TSP100	128	8.05 (10.3s)	8.09 (10.3s)	8.04 (1.2s)
	1,280	8.00 (10.3s)	8.00 (10.3s)	7.98 (12s)
	12,800	7.95 (31s)	7.95 (31s)	7.92 (120s)
	128,000	7.92 (265s)	7.91 (265s)	7.87 (1200s)
	1,280,000	7.89 (2640s)	7.88 (2640s)	7.83 (12000s)

Bello, Irwan, et al. "Neural Combinatorial Optimization with Reinforcement Learning." *arXiv preprint arXiv:1611.09940* (2016).

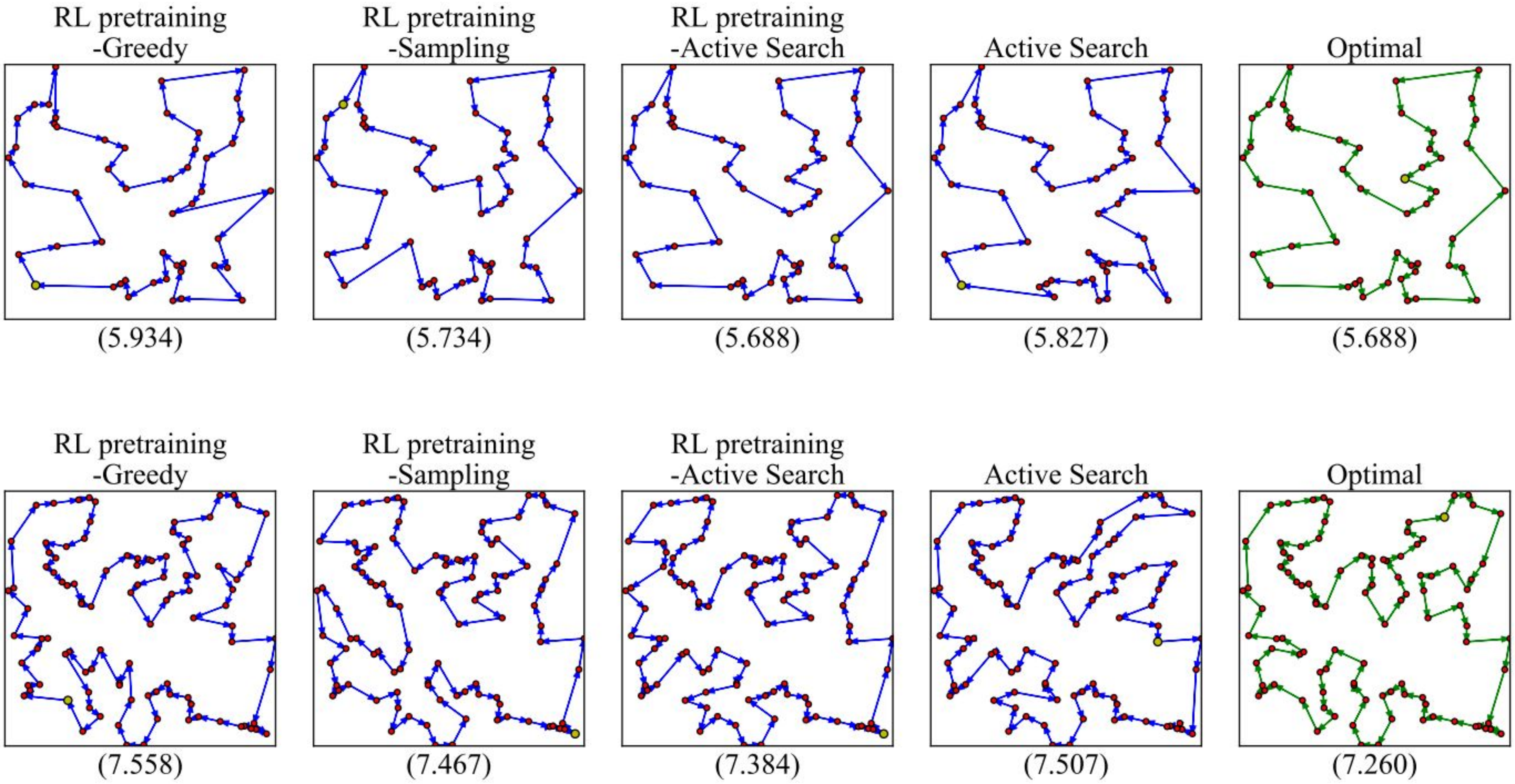


Bello, Irwan, et al. "Neural Combinatorial Optimization with Reinforcement Learning." *arXiv preprint arXiv:1611.09940* (2016).

Table 5: Results of RL pretraining-Greedy and Active Search on KnapSack (higher is better).

Task	RL pretraining greedy	Active Search	Random Search	Greedy	Optimal
KNAP50	19.86	**20.07**	17.91	19.24	**20.07**
KNAP100	40.27	**40.50**	33.23	38.53	**40.50**
KNAP200	57.10	**57.45**	35.95	55.42	**57.45**

Bello, Irwan, et al. "Neural Combinatorial Optimization with Reinforcement Learning." *arXiv preprint arXiv:1611.09940* (2016).



Bello, Irwan, et al. "Neural Combinatorial Optimization with Reinforcement Learning." *arXiv preprint arXiv:1611.09940* (2016).

Unofficial implementation by Taehoon Kim

- https://github.com/devsisters/neural-combinatorial-rl-tensorflow