Learning to Navigate Without a Map

Kannappan Sirchabesan
20-Nov-2018

Agenda

Introduction

Introduction

Links

Paper: https://arxiv.org/abs/1804.00168

DeepMind Blog: https://deepmind.com/blog/learning-to-navigate-cities-without-a-map/

StreetLearn website: https://sites.google.com/view/streetlearn

Introduction

Trajectory
Goal location
Agent position and field of view
Map of Paris Rive Gauche

Introduction

Long-range navigation is a complex cognitive task that relies on developing an internal representation of space, grounded by recognisable landmarks and robust visual processing.

Navigation should simultaneously support **continuous self-localisation** (“I am here”) and a representation of the **goal** (“I am going there”).

Successful navigation relies on integration of **general policies** with **locale-specific knowledge**. A **dual pathway architecture** is proposed, that allows locale-specific features to be encapsulated, while still enabling transfer to multiple cities.

An interactive navigation environment using **Google StreetView**, for its photographic content and worldwide coverage, is used.

The learning method demonstrates agents learning to navigate multiple cities and to traverse to target destinations that may be kilometres away.

Introduction

Learning to navigate directly from visual inputs has been shown to be possible in some domains, by using deep reinforcement learning approaches that can learn from task rewards – for instance, navigating to a destination.

Recent research has demonstrated that RL agents can learn to

- Navigate 3D games (e.g. Lample & Chaplot 2017).
- Navigate mazes (e.g. Mirowski et al. 2016), and
- Navigate house scenes (Zhu et al., 2017; Wu et al., 2018),

Successes notwithstanding, deep RL approaches are notoriously **data inefficient** and **sensitive to perturbations of the environment**, and are more well-known for their successes in games and simulated environments than in real-world applications. It is therefore not obvious that they can be used for large-scale visual navigation based on real-world images, and hence it is investigated in this paper.

Related Work

Related Work - Examples

Navigation in Games https://arxiv.org/abs/1609.05521

Related Work - Examples

Navigate Mazes: https://arxiv.org/abs/1611.03673

Related Work - Examples

Navigate House Scenes: https://arxiv.org/abs/1609.05143

Related Work

DeepNav: https://arxiv.org/abs/1701.09135 (CNN + A* on 10 city graphs and more than 1 million street-view images)

RatSLAM: https://youtu.be/t2w6kYzTbr8?t=430 (computational models on rodent hippocampus)

Neural SLAM: https://arxiv.org/abs/1706.09520 (RL agent with an external memory to represent a global map)

… and a lot more

Environment

Environment

The environment used is an interactive environment constructed using Google StreetView.

StreetView data includes both high-resolution imagery and graph connectivity

A set of geolocated 360 degree panoramic images which form the nodes of an undirected graph

In this paper, A number of large regions in New York City, Paris and London have been identified that contain between 7,000 and 65,500 nodes (and between 7,200 and 128,600 edges, respectively), having a mean node spacing of 10m, and covering a range of up to 5km.

The agent never observes the underlying graph. It only sees the RGB images.

Street View public API: https://developers.google.com/maps/documentation/streetview/intro

Environment

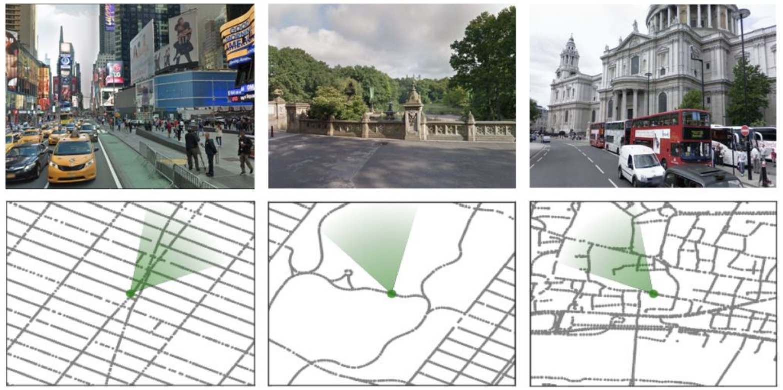


Figure 1. Our environment is built of real-world places from StreetView. The figure shows diverse views and corresponding local maps in New York City (Times Square, Central Park) and London (St. Paul's Cathedral). The green cone represents the agent's location and orientation.

Environment

Manhattan, New York

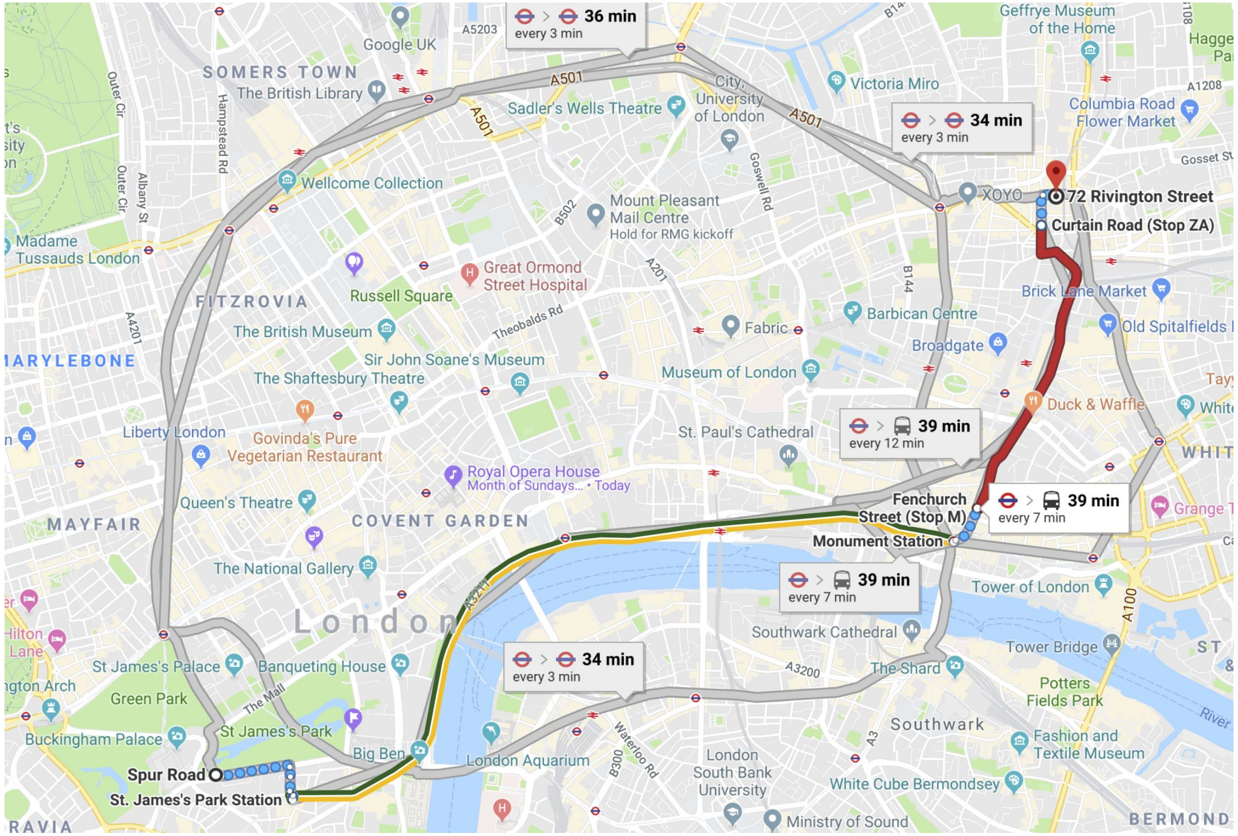
Manhattan StreetView graph has 256961 nodes and 266040 edges. Five areas have been selected using a starting point at a given coordinate and collecting panoramas in a panorama adjacency graph using breadth-first-search, until a given depth of the search tree.

1. Wall Street / Lower Manhattan: 6917 nodes and 7191 edges, 200-deep search tree starting at (40.705510, - 74.013589).
2. NYU / Greenwich Village: 17227 nodes and 17987 edges, 200-deep search tree starting at (40.731342, - 73.996903).
3. Midtown: 16185 nodes and 16723 edges, 200-deep search tree starting at (40.756889, -73.986147).
4. Central Park: 10557 nodes and 10896 edges, 200-deep search tree starting at (40.773863, -73.971984).
5. Harlem: 14589 nodes and 15099 edges, 220-deep search tree starting at (40.806379, -73.950124).

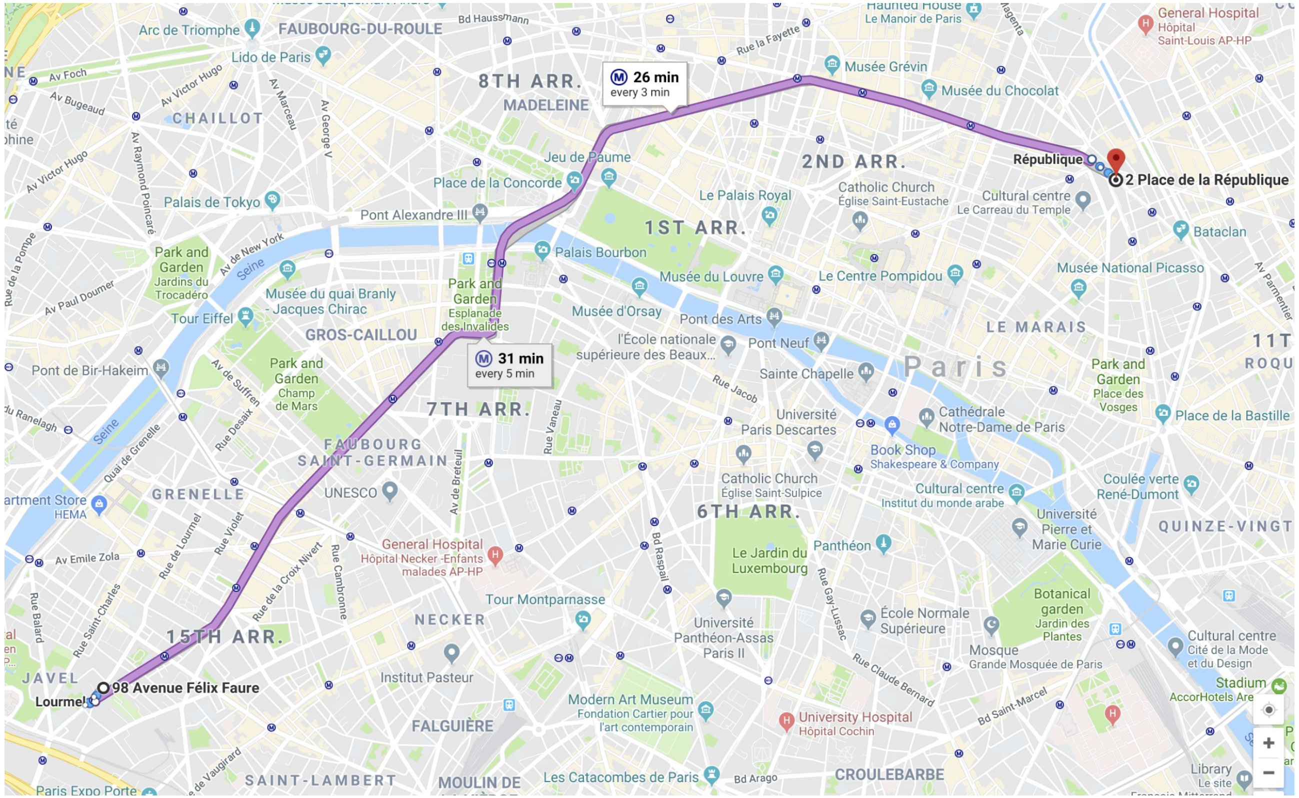
Central London: The Central London StreetView environment contains 24428 nodes and 25352 edges, and is defined by a bounding box between the following Lat/Long coordinates: (51.500567, -0.139157) and (51.526175, -0.080043).

Paris: The Paris Rive Gauche environment contains 34026 nodes and 35475 edges, and is defined by a bounding box between Lat/Long coordinates: (48.839413, 2.2829247) and (48.866578, 2.3653221).

Environment - Central London



Environment - Paris



Environment

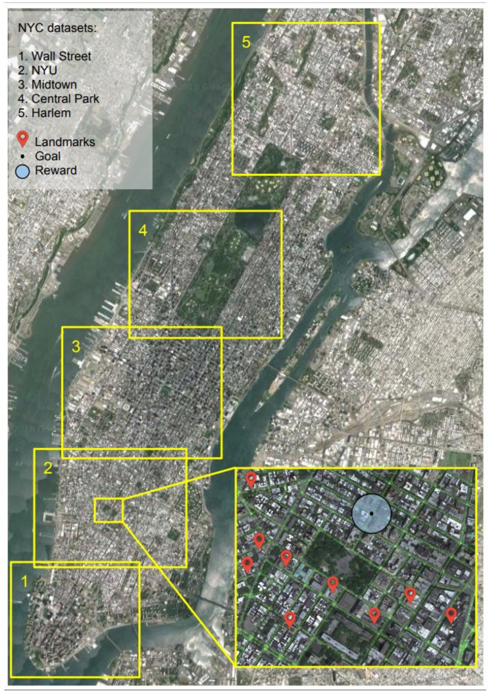


Figure 2. Map of the 5 environments in New York City; our experiments focus on the NYU area as well as on transfer learning from the other areas to Wall Street (see Section 5.3). In the zoomed in area, each green dot corresponds to a unique panorama, the goal is marked in blue, and landmark locations are marked with red pins.

Environment

Image Input x_t is a cropped RGB image scaled to 84 x 84 pixels

Action space is composed of five discrete actions:

slow rotate left or right (22.5 degrees),

“fast” rotate left or right (67.5 degrees),

move forward (*noop* if no edge in view from the current agent pose). If there are multiple edges in the viewing cone of the agent, then the most central one is chosen.

Environment

Goal

There are many options for how to specify the goal to the agent, from images to agent-relative directions, to text descriptions or addresses.

In this paper the goal is represented in terms of its proximity to a set L of fixed landmarks specified using Lat/Long

A goal is represented as a softmax over the distances to the k landmarks

$$g_{t,i} = \frac{\exp(-\alpha d^g_{t,i})}{\sum_k \exp(-\alpha d^g_{t,k})} \qquad (\alpha = 0.002)$$

This has a number of desirable qualities: it is a scalable representation that extends easily to new regions, it does not rely on any arbitrary scaling or normalising of map coordinates, and it has intuitive meaning – humans and animals also navigate with respect to fixed landmarks.

The goal description is *absolute*; it is not relative to the agent's position and only changes when a new goal is drawn (either upon successful goal acquisition or at the beginning of an episode).

Environment

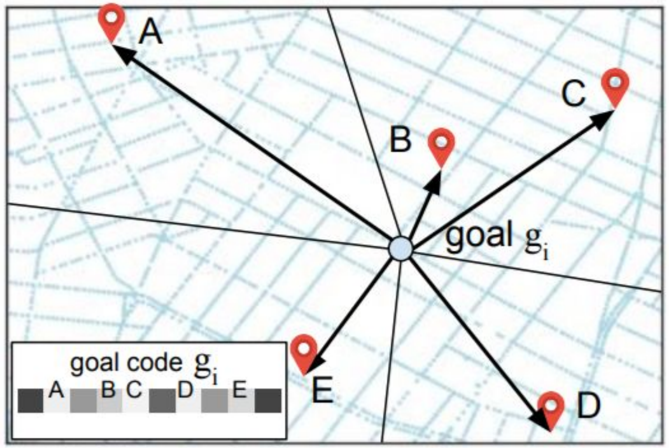


Figure 3. We illustrate the goal description by showing a goal and a set of 5 landmarks that are nearby, plus 4 that are more distant. The code g_i is a vector with a softmax-normalised distance to each landmark.

Environment

Task

Courier Task

Navigating to a series of random locations in a city, the agent starts each episode from a randomly sampled position and orientation.

If the agent gets within 100m of the goal (approximately one city block), the next goal is randomly chosen and input to the agent.

Each episode ends after 1000 agent steps.

The reward that the agent gets upon reaching a goal is proportional to the shortest path between the goal and the agent's position when the goal is first assigned

In order to solve the *courier* task, the agent needs to learn the association between the goal vector and the images observed at the goal location, as well as the association between the images observed at its current location and the policy to reach the goal destination.

Architecture

Architecture

Goal-dependent Actor-Critic Reinforcement Learning

Goal Dependent RL: https://bair.berkeley.edu/blog/2018/09/06/rig/

The usual RL objective is to find the policy that maximises the expected return defined as the sum of discounted rewards starting from state s_0 with discount γ. In this navigation task, the expected return from a state s_t also depends on the series of sampled goals.

The learning problem is formulated as a MDP with state space S, action space A, environment E, and a set of possible goals G.

Policy is a distribution over actions given the current state and goal

$$\pi(a|s,g) = Pr(a_t = a|s_t = s, g_t = g)$$

Value function is Expected return for the agent that is sampling actions from policy π from state s_t with goal g_t

$$V^\pi(s,g) = E[R_t] = E[\sum_{k=0}^{\infty} \gamma^k r_{t+k}|s_t = s, g_t = g]$$

General Learning: To support cognitive processes such as scene understanding

Locale specific Learning: To organise and remember the features and structures that are unique to a place.

Architecture

Neural Network Architecture

- GoalNav - CNN + *Policy LSTM*
- CityNav - CNN + *Policy LSTM* + ***Goal LSTM + auxiliary heading (θ) prediction***
- MultiCityNav - CNN + *Policy LSTM* + *Goal LSTM(**one per city**) + auxiliary heading (θ) prediction*
- **Auxiliary tasks** can speed up learning by providing extra gradients. In this case, the prediction of the agent's heading θ_t, defined as an angle between the north direction and the agent's pose, using a multinomial classification loss on binned angles.

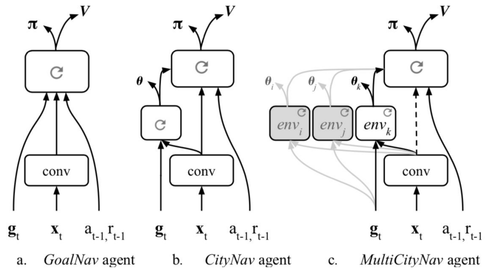


Figure 4. Comparison of architectures. *Left:* GoalNav is a convolutional encoder plus policy LSTM with goal description input. *Middle:* CityNav is a single-city navigation architecture with a separate goal LSTM and optional auxiliary heading (θ). *Right:* MultiCityNav is a multi-city architecture with individual goal LSTM pathways for each city.

Architecture

Neural Network Architecture - Vision Model

Standard vision model for Deep RL with 2 convolutional layers followed by a fully connected layer.

Conv1 layer = *8x8 kernel, 4x4 stride, and 16 feature maps*

Conv2 layer = *4x4 kernel, 2x2 stride, and 32 feature maps*

Fully connected layer = 256 units. outputs 256-dimensional visual features.

(ReLU) separate each of the layers.

Architecture

Neural Network Architecture - LSTMs

The *GoalNav* architecture has a single recurrent layer (LSTM) to predict the policy and value function. The convnet is connected to the policy LSTM. The policy LSTM has additional inputs: past reward and previous action expressed as a one-hot vector of dimension 5 (one for each action: forward, turn left or right by 22.5 degrees, turn left or right by 67.5 degrees).

The goal information g_t is provided as an extra input, either to the policy LSTM (*GoalNav* agent) or to each *goal* LSTM in the *CityNav* and *MultiCityNav* agents.

In case of landmark-based goals, g_t is a vector of 460 elements (460 landmarks).

Goal LSTM takes 256-dimensional inputs from the convnet. The goal LSTM contains 256 hidden units and is followed by a *tanh* nonlinearity, a dropout layer with probability p=0.5, then a 64-dimensional linear layer and finally a *tanh* layer. It is this (*CityNav*) or these (*MultiCityNav*) 64-dimensional outputs that are connected to the policy LSTM.

Policy LSTM contains 256 hidden units, followed by two parallel layers: one linear layer going from 256 to 1 and outputting the value function, and one linear layer going from 256 to 5 (the number of actions), and a softmax nonlinearity, outputting the policy.

Architecture

Curriculum Learning

Link: https://ronan.collobert.com/pub/matos/2009_curriculum_icml.pdf

- The courier task suffers from very sparse rewards similar to RL problems like Montezuma’s Revenge.
- A curriculum is used to overcome this problem by helping the agent to learn to navigate to increasingly more distant destinations.
- The curriculum scheme used here is by sampling each new goal to be within 500m of the agent’s position (phase 1). In phase 2, the maximum range of allowed destinations is grown progressively to cover the full graph (3.5km in the smaller New York areas, or 5km for central London or Downtown Manhattan).

Architecture

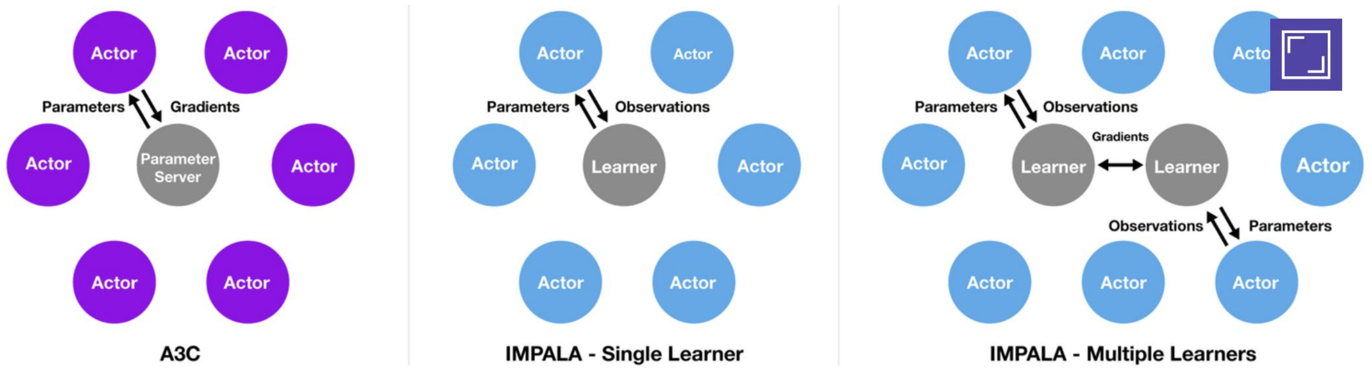
Learning Hyperparameters

- The costs for all auxiliary heading prediction tasks, of the value prediction, of the entropy regularization and of the policy loss are added before being sent to the RMSProp gradient learning algorithm. The weight of heading prediction is 1, the entropy cost is 0.004 and the value baseline weight is 0.5.
- The agent is trained using IMPALA, an actor-critic implementation of deep reinforcement learning that decouples acting and learning. IMPALA results in similar performance to A3C on a single city task, but as it has been demonstrated to handle better multi-task learning than A3C.
- 256 actors are used for *CityNav* and 512 actors for *MultiCityNav*, with batch sizes of 256 or 512 respectively, and sequences are unrolled to length 50.
- Learning rate of 0.001, linearly annealed to 0.
- The discounting coefficient in the Bellman equation is 0.99.

Architecture

IMPALA vs A3C

- IMPALA = Importance Weighted Actor Learner Architecture
- https://deepmind.com/blog/impala-scalable-distributed-deeprl-dmlab-30/
- https://arxiv.org/abs/1802.01561



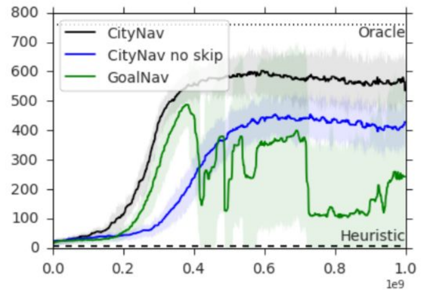
Results

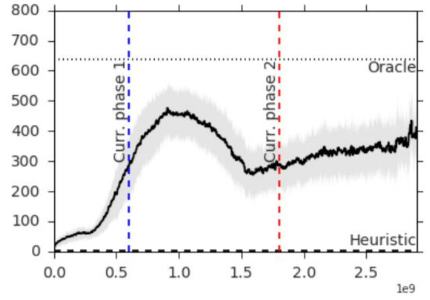
Results

Heuristic is a random walk on the street graph, where the agent turns in a random direction if it cannot move forward; if at an intersection it will turn with a probability p=0.95.

Oracle uses the full graph to compute the optimal path using breadth-first search.

The bottom diagram shows performance of **curriculum learning**.





Results

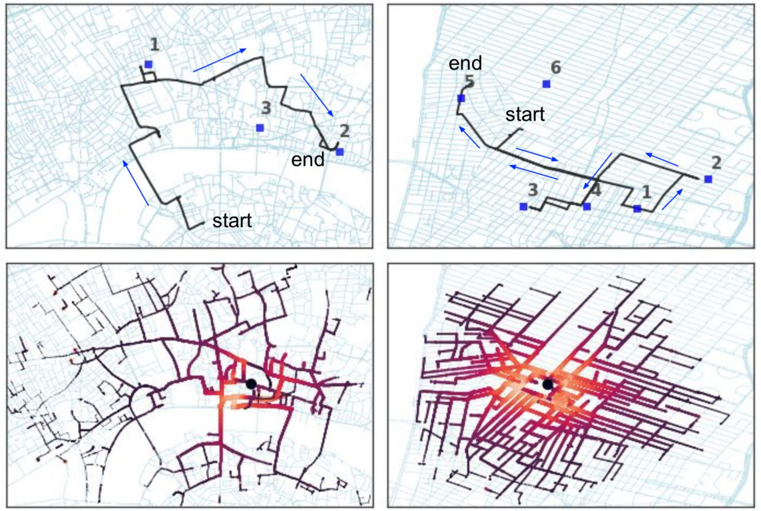


Figure 6. Trained *CityNav* agent's performance in two environments: Central London (left panes), and NYU (right panes). *Top*: examples of the agent's trajectory during one 1000-step episode, showing successful consecutive goal acquisitions. The arrows show the direction of travel of the agent. *Bottom*: We visualise the value function of the agent during 100 trajectories with random starting points and the same goal (respectively St Paul's cathedral and Washington Square). Thicker and warmer colour lines correspond to higher value functions.

Results

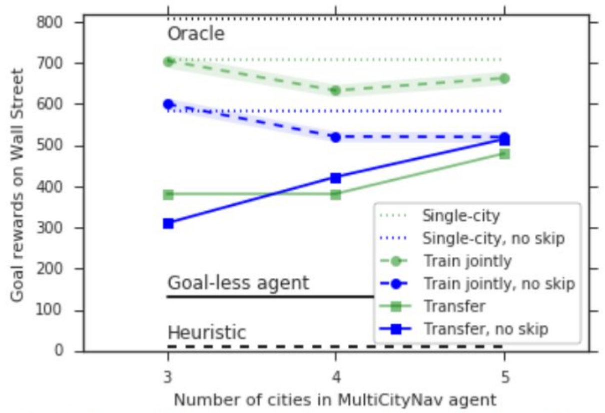


Figure 10. Joint multi-city training and transfer learning performance of variants of the *MultiCityNav* agent, evaluated only on the target city (Wall Street). We compare single-city training on the target environment alone vs. joint training on multiple cities (3, 4, or 5-way joint training including Wall Street), vs. pre-training on multiple cities and then transferring to Wall Street while freezing the entire agent except the new pathway (see Fig. 10). One variant has skip connections between the convolutional encoder and the policy LSTM, the other does not (no-skip).

Thanks

Q & A