Learning to Navigate in Cities Without a Map

Piotr Mirowski [1] **Matthew Koichi Grimes** [1] **Mateusz Malinowski** [1] **Karl Moritz Hermann** [1] **Keith Anderson** [1]
Denis Teplyashin [1] **Karen Simonyan** [1] **Koray Kavukcuoglu** [1] **Andrew Zisserman** [1] **Raia Hadsell** [1]

Abstract

Navigating through unstructured environments is a basic capability of intelligent creatures, and thus is of fundamental interest in the study and development of artificial intelligence. 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, that can simultaneously support continuous self-localisation ("I am *here*") and a representation of the goal ("I am going *there*"). Building upon recent research that applies deep reinforcement learning to maze navigation problems, we present an end-to-end deep reinforcement learning approach that can be applied on a city scale. Recognising that successful navigation relies on integration of general policies with locale-specific knowledge, we propose a dual pathway architecture that allows locale-specific features to be encapsulated, while still enabling transfer to multiple cities. We present an interactive navigation environment that uses Google StreetView for its photographic content and worldwide coverage, and demonstrate that our learning method allows agents to learn to navigate multiple cities and to traverse to target destinations that may be kilometres away. A video summarizing our research and showing the trained agent in diverse city environments as well as on the transfer task is available at: https://sites.google.com/view/streetlearn.

1. Introduction

The subject of navigation is attractive to various research disciplines and technology domains alike, being at once a subject of inquiry from the point of view of neuroscientists wishing to crack the code of grid and place cells, as well as a fundamental aspect of robotics research wishing to build mobile robots that can reach a given destination. The majority of algorithms involve building an explicit map during an exploration phase and then planning and acting via that representation. In this work, we are interested in pushing the limits of end-to-end deep reinforcement learning for navigation by proposing new methods and demonstrating their performance in large-scale, real-world environments. Just as humans can learn to navigate a city without relying on maps, GPS localisation, or other aids, it is our aim to show that a neural network agent can learn to traverse entire cities using only visual observations. In order to realise this aim, we designed an interactive environment that uses the images and underlying connectivity information from Google StreetView, and propose a dual pathway agent architecture that can navigate within the environment (see Fig. 1).

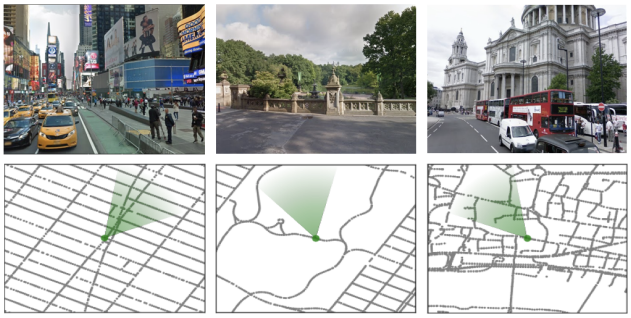

Figure 1. Our environment is built of real-world places from StreetView. The figure shows diverse views and corresponding local maps in New York City (Times Square, Central Park) and London (St. Paul's Cathedral). The green cone represents the agent's location and orientation.

Learning to navigate directly from visual inputs has been shown to be possible in some domains, by using deep reinforcement learning (RL) approaches that can learn from task rewards – for instance, navigating to a destination. Recent research has demonstrated that RL agents can learn to navigate house scenes (Zhu et al., 2017; Wu et al., 2018), mazes (e.g. Mirowski et al. 2016), and 3D games (e.g. Lample & Chaplot 2017). 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

[1]DeepMind, London, United Kingdom. Correspondence to: Piotr Mirowski <piotrmirowski@google.com>.

that they can be used for large-scale visual navigation based on real-world images, and hence it is the subject of our investigation.

Contributions. The primary contribution of this paper is to present a novel, dual-pathway agent architecture, trained using end-to-end reinforcement learning, that can handle a city-scale, real-world visual navigation task. Our proposed agent demonstrates goal-dependent learning, meaning that the policy and value function must learn to adapt to a sequence of goals that are given as inputs. The approach features a recurrent neural architecture that supports both locale-specific learning as well as general, transferable navigation behaviour. Balancing these two capabilities is achieved by separating a recurrent neural pathway from the general navigation policy of the agent. This pathway addresses two needs. First, it receives and interprets the current goal given by the environment, and second, it encapsulates and memorises the features and structure of a single city region. Thus, rather than using a map, or an external memory, we propose an architecture with two recurrent pathways that can effectively address a challenging navigation task in a single city as well as transfer to new cities or regions by training only a new locale-specific pathway.

The proposed agent architecture is demonstrated on a new interactive environment for reinforcement learning that features real-world images as agent observations, global scale and diversity, and real-world grounded content that is built on top of the publicly available Google StreetView. Within this environment we have developed a traversal task that requires that the agent navigates from goal to goal within London, Paris and New York City. The real-world analogy of our task would be a courier operating in a given city that starts at an arbitrary location called "A" and then is directed to go to a specific location "B", without having ever been shown the map featuring these locations or the path going from A to B, or been told its own position.

2. Related Work

Reward-driven navigation in a real-world environment is related to research in various areas of deep learning, reinforcement learning, navigation and planning.

Localisation from real-world imagery. Localising from only an image may seem impossible, but humans can integrate visual cues such as landmarks, vegetation, and architecture to geolocate a given image with surprising accuracy, motivating researchers to investigate machine learning approaches. For instance, convolutional neural networks (CNNs) achieve competitive scores on the geolocation task (Weyand et al., 2016), CNN+LSTM (Donahue et al., 2015; Malinowski et al., 2017) architectures improve on this by exploiting spatio-temporal continuity (Weyand et al., 2016), and other deep architectures can estimate camera pose or depth estimation from pixels (Walch et al., 2016; Kendall et al., 2015; Ummenhofer et al., 2017) or achieve robust location-based image retrieval (Arandjelovic et al., 2016). DeepNav (Brahmbhatt & Hays, 2017) and (Khosla et al., 2014) use a StreetView environment and propose that restricted "common-sense based" navigation-related tasks can be learned through supervised CNN-based architectures. RatSLAM demonstrates localisation and path planning over long distances using a biologically-inspired architecture (Milford et al., 2004). The aforementioned methods rely on supervised training with ground truth labels, something which is not provided in our environment. Moreover, we are not only interested in localisation, but also in planning a path to reach a given destination.

Deep RL methods for navigation. Many RL-based approaches for navigation rely on simulators which have the benefit of features like procedurally generated variations but tend to be visually simple and unrealistic (Beattie et al., 2016; Kempka et al., 2016; Tessler et al., 2017). To support sparse reward signals in these environments, recent navigational agents use auxiliary tasks in training. For instance, (Mirowski et al., 2016; Jaderberg et al., 2016) show visual navigation in various simulated mazes with self-supervised auxiliary tasks. Other methods learn to predict future measurements (Dosovitskiy & Koltun, 2016). Others have incorporated goal instructions for the agent, using simple text descriptions (Hill et al., 2017; Hermann et al., 2017; Chaplot et al., 2017); in our case, the goal is given to the agent in terms of distances to local landmarks. Deep RL is used for active localisation in (Chaplot et al., 2018).

To bridge the gap between simulation and reality, researchers have developed more realistic, higher-fidelity simulated environments (Dosovitskiy et al., 2017; Kolve et al., 2017; Shah et al., 2018; Wu et al., 2018). However, in spite of their increasing photo-realism, the inherent problems of simulated environments lie in the limited diversity of the environments and the antiseptic cleanliness of the observations. Our real-world dataset is diverse and visually realistic, comprising scenes with pedestrians, cars, buses or trucks, diverse weather conditions and vegetation and covering large geographic areas. However, we note that there are obvious limitations of our environment: It does not contain dynamic elements, the action space is necessarily discrete as it must jump between panoramas, and the street topology cannot be arbitrarily altered or regenerated.

Deep RL for path planning and mapping. Several recent approaches have used memory or other explicit neural structures to support end-to-end learning of planning or mapping. These include Neural SLAM (Zhang et al., 2017) that proposes an RL agent with an external memory to represent a global map and a SLAM-inspired algorithm, Neural

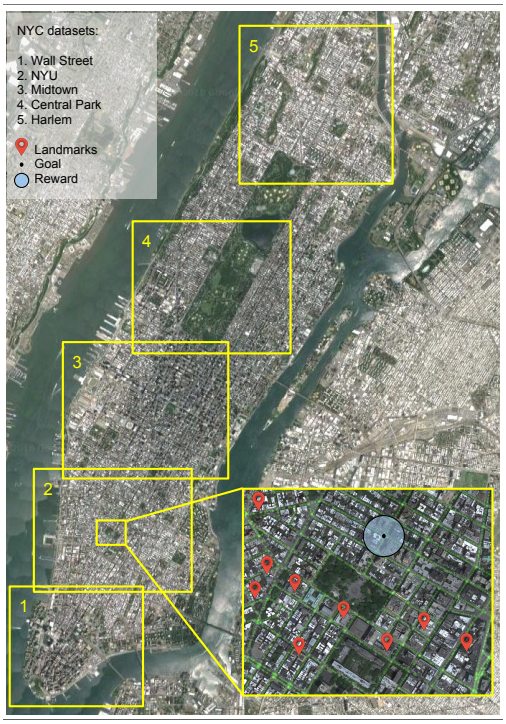

Figure 2. Map of the 5 environments in New York City; our experiments focus on the NYU area as well as on transfer learning from the other areas to Wall Street (see Section 5.3). In the zoomed in area, each green dot corresponds to a unique panorama, the goal is marked in blue, and landmark locations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

Map (Parisotto & Salakhutdinov, 2017) which proposes a structured 2D memory for navigation, and Memory Augmented Control Maps, which uses a hierarchical control strategy. Other work (Brunner et al., 2017) explicitly provide a global map that is input to the agent with the goal of learning to read and use the map. The proposed architecture from (Gupta et al., 2017) uses an explicit neural mapper and planner for navigation tasks, and argue that joint training makes the overall system more robust under errors introduced by single component. Similar to (Zhang et al., 2017), they also use extra memory that represents ego-centric position of the agent. Another recent work proposes a graph network solution (Nikolay Savinov, 2018). Finally, we mention the target-driven visual navigation approach of (Zhu et al., 2017), which learns a goal-conditional policy which is related to our approach.

3. Environment

This section presents an interactive environment constructed using Google StreetView, which provides a public API[1]. Since StreetView data has been collected worldwide, and includes both high-resolution imagery and graph connectivity,

[1] https://developers.google.com/maps/documentation/streetview/

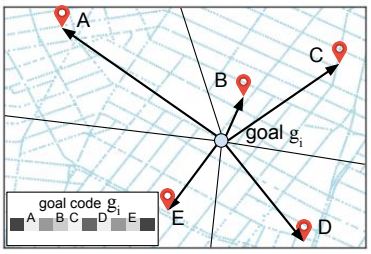

Figure 3. We illustrate the goal description by showing a goal and a set of 5 landmarks that are nearby, plus 4 that are more distant. The code g_i is a vector with a softmax-normalised distance to each landmark.

it is a valuable resource for studying navigation (Fig.1).

StreetView provides a set of geolocated 360°panoramic images which form the nodes of an undirected graph (we use the term node and panorama interchangeably). We selected a number of large regions in New York City, Paris and London that contain between 7,000 and 65,500 nodes (and between 7,200 and 128,600 edges, respectively), have a mean node spacing of 10m, and cover a range of up to 5km (see Fig.2).

We do not reduce or simplify the underlying connectivity but rather use the full graph, thus there are congested areas with many nodes, complex occluded intersections, tunnels and footpaths, and other ephemera. Although the graph is used to construct the environment, the agent never observes the underlying graph – it only sees the RGB images (overlay information, such as arrows, that are visible in the public StreetView product are also not seen by the agent). Examples of the RGB images and the graph are shown in Figure 1.

3.1. Agent Interface and the Courier Task

An RL environment needs to specify the observations and action space of the agent as well as define the task. The agent has two inputs: the image $\mathbf{x_t}$ and the goal description g_t. For the images, we simulate the challenging scenario of a first-person, partially observed 3D environment, thus $\mathbf{x_t}$ is a cropped, 60° square, RGB image that is scaled to 84×84 pixels (i.e. not the entire panorama). The action space is composed of five discrete actions: "slow" rotate left or right ($\pm 22.5^\circ$), "fast" rotate left or right ($\pm 67.5^\circ$), or move forward (this action becomes a *noop* if there is not an 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.

There are many options for how to specify the goal to the agent, from images to agent-relative directions, to text descriptions or addresses. We choose to represent the current goal in terms of its proximity to a set $\mathcal{L}$ of fixed landmarks: $\mathcal{L} = \{(Lat_k, Long_k)\}_k$, specified using the Lat/Long (lati-

tude and longitude) coordinate system. To represent a goal at $(Lat_t^g, Long_t^g)$ we take a softmax over the distances to the k landmarks, thus for distances $\{d_{t,k}^g\}_k$ the goal vector contains

$$g_{t,i} = \frac{\exp(-\alpha d_{t,i}^g)}{\sum_k \exp(-\alpha d_{t,k}^g)}$$

for the ith landmark with $\alpha = 0.002$. An example is shown in Fig. 3. This forms a goal description which 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. Note that 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). We manually define 644 landmarks covering New York, Paris and London (locations are given in the Supplementary material).

In the *courier* task, which we define as the problem of 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; much like a delivery service, the agent receives a higher reward for longer journeys.

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[2].

4. Methods

This section describes our agents introduced to perform courier-like navigation tasks.

4.1. Goal-dependent Actor-Critic Reinforcement Learning

We formalise the learning problem as a Markov Decision Process, with state space $\mathcal{S}$, action space $\mathcal{A}$, environment $\mathcal{E}$, and a set of possible goals $\mathcal{G}$. The reward function depends on the current goal and state: $R : \mathcal{S} \times \mathcal{G} \times \mathcal{A} \rightarrow \mathbb{R}$. The usual reinforcement learning 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 $\{g_k\}_k$. The policy is a distribution over actions given the current state s_t and the goal g_t: $\pi(a|s, g) = Pr(a_t = a|s_t = s, g_t = g)$. We define the value function to be the expected return for the agent that is sampling actions from policy π from state s_t with goal g_t: $V^\pi(s, g) = E[R_t] = E[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t = s, g_t = g]$.

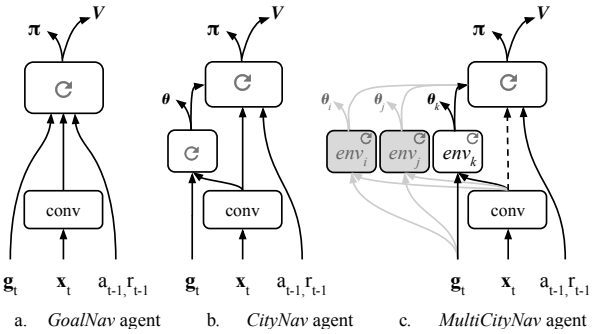

Figure 4. Comparison of architectures. *Left:* GoalNav is a convolutional encoder plus policy LSTM with goal description input. *Middle:* CityNav is a single-city navigation architecture with a separate goal LSTM and optional auxiliary heading (θ). *Right:* MultiCityNav is a multi-city architecture with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

We hypothesise the courier task should benefit from two types of learning: general, and locale-specific. A navigating agent not only needs an internal representation that is general, to support cognitive processes such as scene understanding, but also needs to organise and remember the features and structures that are unique to a place. Therefore, to support both types of learning, we focus on neural architectures with multiple pathways.

4.2. Architectures

The policy and the value function are both parameterised by a neural network which shares all layers except the final linear outputs. The agent operates on raw pixel images $\mathbf{x_t}$, which are passed through a convolutional network as in (Mnih et al., 2016). A Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997) receives the output of the convolutional encoder as well as the past reward r_{t-1} and previous action a_{t-1}. The three different architectures are described below. Additional architectural details are given in the Supplementary Material.

The **GoalNav** architecture (Fig. 4a) has a convolutional encoder and *policy LSTM*. The goal description g_t is input to this LSTM along with the previous action and reward.

The **CityNav** architecture (Fig. 4b) combines the previous architecture with an additional LSTM, called the *goal LSTM*, which receives visual features as well as the goal description. The CityNav agent also adds an auxiliary heading (θ)

[2]The Supplemental Material contains an analysis of the hidden representations of the agent by using a nonlinear decoder of both its current position and of the goal position.

prediction task on the outputs of the goal LSTM.

The **MultiCityNav** architecture (Fig. 4c) extends the CityNav agent to learn in different cities. The remit of the goal LSTM is to encode and encapsulate locale-specific features and topology such that multiple pathways may be added, one per city or region. Moreover, after training on a number of cities, we demonstrate that the convolutional encoder and the policy LSTM become general enough that only a new goal LSTM needs to be trained for new cities.

As shown in (Jaderberg et al., 2016; Mirowski et al., 2016; Dosovitskiy & Koltun, 2016), auxiliary tasks can speed up learning by providing extra gradients as well as relevant information. We employ a very natural auxiliary task: 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.

To train the agents, we use IMPALA (Espeholt et al., 2018), an actor-critic implementation that decouples acting and learning. In our experiments, IMPALA results in similar performance to A3C (Mnih et al., 2016). We use 256 actors for *CityNav* and 512 actors for *MultiCityNav*, with batch sizes of 256 or 512 respectively, and sequences are unrolled to length 50.

4.3. Curriculum Learning

Curriculum learning gradually increases the complexity of the learning task by choosing more and more difficult examples for the learning algorithm (Bengio et al., 2009; Graves et al., 2017; Zaremba & Sutskever, 2014). We propose to use a curriculum to help the agent to learn to navigate to increasingly more distant destinations. Similar to other RL problems such as Montezuma's Revenge, the courier task suffers from very sparse rewards; unlike that game, we are able to define a natural curriculum scheme. We start by sampling each new goal to be within 500m of the agent's position (phase 1). In phase 2, we progressively grow the maximum range of allowed destinations to cover the full graph (3.5km in the smaller New York areas, or 5km for central London or Downtown Manhattan).

5. Results

In this section, we demonstrate and analyse the performance of the proposed architectures on the courier task. We first show the performance of our agents in large city environments, next their generalisation capabilities on a held-out set of goals. Finally, we investigate whether the proposed approach allows transfer of an agent trained on a set of regions to a new and previously unseen region.

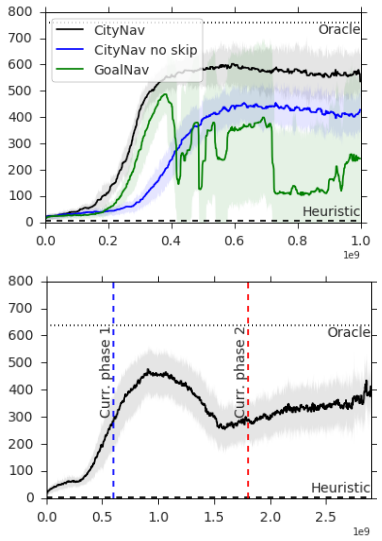

Figure 5. Average per-episode goal rewards (y axis) are plotted vs. learning steps (x axis) for the courier task in the NYU (New York City) environment (top), and in central London (bottom). We compare the *GoalNav* agent, the *CityNav* agent, and the *CityNav* agent without skip connection on the NYU environment, and the *CityNav* agent in London. We also compare the *Oracle* performance and a *Heuristic* agent, described below. The London agents were trained with a 2-phase curriculum– we indicate the end of phase 1 (500m only) and the end of phase 2 (500m to 5000m). Results on the Rive Gauche part of Paris (trained in the same way as in London) are comparable and the agent achieved mean goal reward 426.

5.1. Courier Navigation in Large, Diverse City Environments

We first show that the *CityNav* agent, trained with curriculum learning, succeeds in learning the courier task in New York, London and Paris, with various types of street layout, parks, bridges and tunnels. We replicated experiments with 5 random seeds and plot the mean throughout the experimental results for a subset of New York around NYU and for London. Throughout the paper, and for ease of comparison with experiments that include reward shaping, we report only the rewards at the goal destination (*goal rewards*). Figure 5 compares different agents and shows that the CityNav architecture with the dual LSTM pathways and the heading prediction task attains a higher performance and is more stable than the simpler GoalNav agent. We also trained a CityNav agent without the skip connection from the vision layers to the policy LSTM. While this hurts the performance, we consider it because for the multi-city transfer scenario, we in order to regularise the interface between the goal LSTM and the policy LSTM. We also consider two baselines which give lower (*Heuristic*) and upper (*Oracle*) bounds on the performance. *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

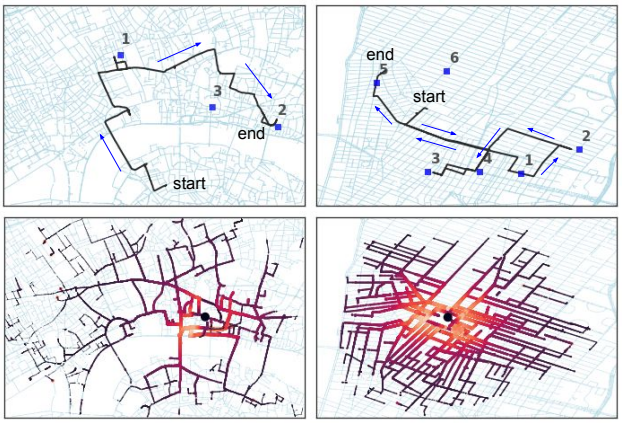

Figure 6. Trained *CityNav* agent's performance in two environments: Central London (left panes), and NYU (right panes). *Top*: examples of the agent's trajectory during one 1000-step episode, showing successful consecutive goal acquisitions. The arrows show the direction of travel of the agent. *Bottom*: 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.

GRID SIZE	TRAIN REWARDS	TEST REWARDS	TEST FAIL	TEST $T_{\frac{1}{2}}$
FINE	655	567	11%	229
MEDIUM	637	293	20%	184
COARSE	623	164	38%	243

Table 1. CityNav agent same-city generalization performance (goal acquisition reward) when separating a training and a held-out set of destination locations shows that the agent performs worse as the size of the held-out area increases. In addition to the reward metric and a fail metric, we also compute the *half-trip time* ($T_{\frac{1}{2}}$, or the number of steps necessary to reach halfway to the goal) to understand the lower performance.

with a probability $p = 0.95$. *Oracle* uses the full graph to compute the optimal path using breath-first search.

We visualise trajectories from the trained agent over two 1000 step episodes (Fig. 6 (top row)). In London, we see that the agent crosses a bridge to get to the first goal, then travels to goal 2, and the episode ends before it can reach the third goal. Figure 6 (bottom row) shows the value function of the agent as it repeatedly navigates to a chosen destination (respectively, St Paul's Cathedral in London and Washington Square in New York).

To understand whether the agent has learned a policy over the full extent of the environment, we plot the number of steps required by the agent to get to the goal. As the number grows linearly with the straight-line distance to that goal, this result suggests that the agent has successfully learnt the navigation policy on both cities (Fig. 7).

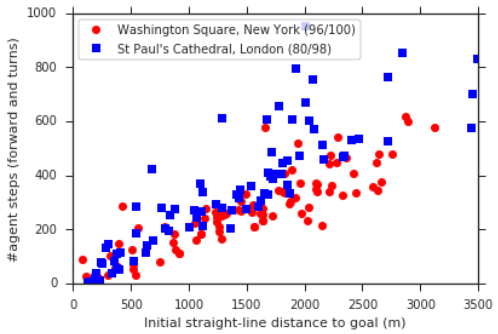

Figure 7. Number of steps required for the *CityNav* agent to reach a goal (Washington Square in New York or St Paul's Cathedral in London) from 100 start locations vs. the straight-line distance to the goal in metres. One agent step corresponds to a forward move of about 10m or a left/right turn by 22.5°or 67.5°.

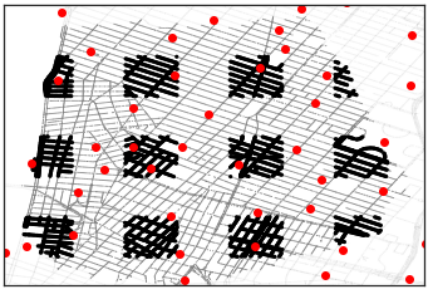

Figure 8. Illustration of medium-sized held-out grid with gray corresponding to training destinations, black corresponding to held-out test destinations. Landmark locations are marked in red.

5.2. Generalization on Held-out Goals

To investigate the generalisation capability of a trained agent, we mask 25% of the possible goals and train on the remaining ones (Fig. 8). At test time we evaluate the agent only on its ability to reach goals in the held-out areas. Note that the agent is still able to traverse *through* these areas, it just never samples a goal there. More precisely, the held-out areas are squares sized $0.01°$, $0.005°$ or $0.0025°$ of latitude and longitude (respectively roughly about 1km×1km, 0.5km×0.5km and 0.25km×0.25km). We call these grids respectively *coarse* (with few and large held-out areas), *medium* and *fine* (with many small held-out areas).

In the experiments, we train the *CityNav* agent for 1B steps, and next freeze the weights of the agent and evaluate its performance on held-out areas for 100M steps. Table 1 shows decreasing performance of the agents as the held-out area size increases. To gain further understanding, in addition to *Test Reward* metric, we also use missed goals (*Fail*) and half-trip time ($T_{\frac{1}{2}}$) metrics. The missed goals metric measures the percentage of times goals were not reached. The half-trip time measures the number of agent steps necessary to cover half the distance separating the

agent from the goal. While the agent misses more goal destinations on larger held-out grids, it still manages to travel half the distance to the goal within a similar time, which suggests that the agent has an approximate held-out goal representation that enables it to head towards it until it gets close to the goal and the representation is no longer useful for the final approach.

5.3. Transfer in Multi-city Experiments

A critical test for our proposed method is to demonstrate that it can provide a mechanism for transfer to new cities. As with humans, when our agent visits a new city we would expect it to have to learn a new set of landmarks, but not have to re-learn its visual representation, its behaviours, etc. Therefore, using the *MultiCityNav* agent, we train on a number of cities (actually regions in New York City), freeze both the policy LSTM and the convolutional encoder, and then train a new locale-specific pathway (the goal LSTM) on a new city. The gradient that is computed by optimising the RL loss is passed through the policy LSTM without affecting it and then applied only to the new pathway.

We compare the performance using three different training regimes, illustrated in Fig. 9: Training on only the target city (*single training*); training on multiple cities, including the target city, together (*joint training*); and joint training on all but the target city, followed by training on the target city with the rest of the architecture frozen (*pre-train and transfer*). In these experiments, we use the whole Manhattan environment as shown in Figure 2, and consisting of the following regions "Wall Street", "NYU", "Midtown", "Central Park" and "Harlem". The target city is always the Wall Street environment, and we evaluate the effects of pre-training on 2, 3 or 4 of the other environments. We also compare performance if the skip connection between the convolutional encoder and the policy LSTM is removed.

We can see from the results in Figure 10 that not only is transfer possible, but that its effectiveness increases with the number of the regions the network is trained on. Remarkably, the agent that is pre-trained on 4 regions and then transferred to Wall Street achieves comparable performance to an agent trained jointly on all the regions, and only slightly worse than single-city training on Wall Street alone. This result supports our intuition that training on a larger set of environments results in successful transfer. We also note that in the single-city scenario it is better to train an agent with a skip-connection, but this trend is reversed in the multi-city transfer scenario. We hypothesise that isolating the locale-specific LSTM as a bottleneck is more challenging but reduces overfitting of the convolutional features and enforces a more general interface to the policy LSTM. While the transfer learning performance of the agent is lower than the stronger agent trained jointly on all the areas, the agent significantly outperforms the baselines and demonstrates goal-dependent navigation.

6. Analysis and Discussion

To better understand the environment, we present further experiments on reward, curriculum, and goal representation. Additional analysis, including architecture ablations and position and goal decoding, are presented in the Supplementary Material.

In our experiments, we focus on the NYU region in New York and analyse the contributions of reward shaping and curriculum learning (Section 6.1) and of the goal representation (Section 6.2). We quantify the performance in terms of average reward per episode obtained at goal acquisition. Each experiment was repeated 5 times with different seeds. We report average final reward and plot the mean and standard deviation of the reward statistic.

6.1. Reward Shaping and Curriculum Learning

Our navigation task assigns a goal to the agent; once the agent successfully navigates to the goal, a new goal is given to the agent. The long distance separating the agent from the goal makes this a difficult RL problem with sparse rewards. To simplify this challenging task, we investigate giving early rewards (*reward shaping*) to the agent before it reaches the goal (we consider an agent has reached a goal whenever it is within 100m thereof), or to add random rewards (*coins*) to encourage exploration (Beattie et al., 2016; Mirowski et al., 2016). Figure 11a suggests that *coins* by themselves are ineffective as our task does not benefit from wide explorations. At the same time large radii of reward shaping help as they greatly simplify the problem.

As a trade-off between task realism and feasibility, and guided by the results in Fig. 11a, we decide to keep a small amount of reward shaping (200m away from the goal) combined with curriculum learning. We choose a curriculum that starts by sampling the goal within a radius of 500m from the agent's location, and progressively grows that disc until it reaches the maximum distance an agent could travel within the environment (e.g., 3.5km, and 5km in the NYU and London environments respectively) by the end of the training. Note that this does not preclude the agent from going astray in the opposite direction several kilometres away from the goal, and that the goal may occasionally be sampled close to the agent. Hence, our curriculum scheme naturally combines easy with difficult cases (Zaremba & Sutskever, 2014), with the latter becoming more common over the period of time.

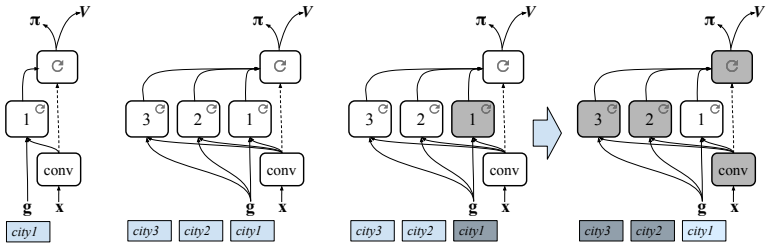

Figure 9. Illustration of training regimes: (a) training on a single city (equivalent to CityNav); (b) joint training over multiple cities with a dedicated per-city pathway and shared convolutional net and policy LSTM; (c) joint pre-training on a number of cities followed by training on a target city with convolutional net and policy LSTM frozen (only the target city pathway is optimised).

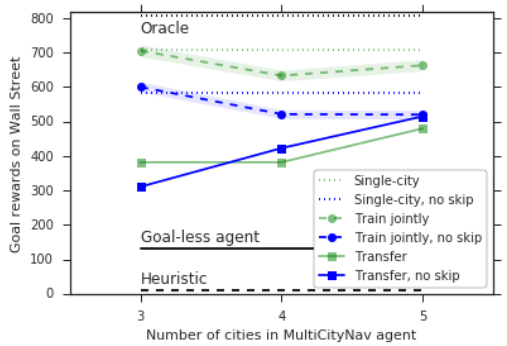

Figure 10. Joint multi-city training and transfer learning performance of variants of the *MultiCityNav* agent, evaluated only on the target city (Wall Street). We compare single-city training on the target environment alone vs. joint training on multiple cities (3, 4, or 5-way joint training including 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).

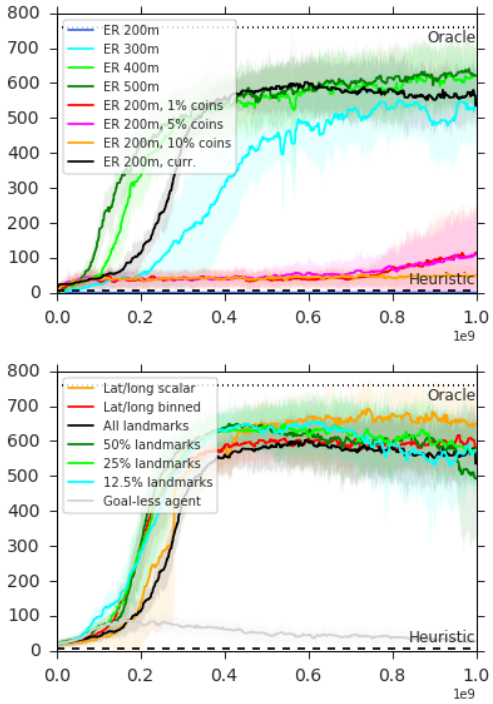

Figure 11. Top: Learning curves of the *CityNav* agent on NYU, comparing reward shaping with different radii of early rewards (ER) vs. ER with random coins vs. curriculum learning with ER 200m and no coins (ER 200m, curr.). Bottom: Learning curves for *CityNav* agents with different goal representations: landmark-based, as well as latitude and longitude classification-based and regression-based.

6.2. Goal Representation

As described in Section 3.1, our task uses a goal description which is a vector of normalised distances to a set of fixed landmarks. We manually define 644 landmarks covering New York, Paris and London, some which are visible on Figure 2. As we can see in Figure 11b, reducing the density of the landmarks to half, a quarter or an eighth (50%, 25%, 12.5%) does not seem to reduce the performance.

In spite of the advantages of the landmark-based goal representation (it is scalable, extends easily to new cities, and does not rely on arbitrary scaling or normalisation of map coordinates), we investigate some alternative representations: a) latitude and longitude scalar coordinates normalised to be between 0 and 1 in the region where the agent navigates (*Lat/long scalar* in Figure 11b), or b) binned representation *Lat/long binned* of the aforementioned world-coordinates that is expressed using 35 bins for X and 35 bins for Y, with each bin covering 100m. The latitude and longitude scalar goal representations performs best. However, since the *all landmarks* representation performs well while remaining independent of the coordinate system, we use this representation as the canonical one.

Finally, to asses the importance of the goal-conditioned agents, we also train a *Goal-less CityNav* agent by removing inputs g_t. The poor performance of this agent (Fig. 11b) confirms that the performance of our method cannot be attributed to clever street graph exploration alone. *Goal-less CityNav* learns to run in circles of increasing radii. A plausible explanation of such a behaviour is that the agent

"blindly" explores the environment by visiting each place at most once in the hope of finding the goal.

7. Conclusion

Navigation is an important cognitive task that enables humans and animals to traverse a complex world without maps. We have presented a deep RL approach to navigation that solves tasks in city-scale real-world environments, introduced and analysed a new courier task, and presented a multi-city neural network agent architecture that demonstrates transfer to new environments.

Acknowledgements

The authors wish to acknowledge Lasse Espeholt and Hubert Soyer for technical help with the IMPALA algorithm, Razvan Pascanu, Ross Goroshin, Pushmeet Kohli and Nando de Freitas for their feedback, Chloe Hillier and Vishal Maini for help with the project, and the Google Streetview team for their support in accessing the data.

References

Arandjelovic, Relja, Gronat, Petr, Torii, Akihiko, Pajdla, Tomas, and Sivic, Josef. Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5297–5307, 2016.

Beattie, Charles, Leibo, Joel Z, Teplyashin, Denis, Ward, Tom, Wainwright, Marcus, Küttler, Heinrich, Lefrancq, Andrew, Green, Simon, Valdés, Víctor, Sadik, Amir, et al. Deepmind lab. *arXiv preprint arXiv:1612.03801*, 2016.

Bengio, Yoshua, Louradour, Jérôme, Collobert, Ronan, and Weston, Jason. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48. ACM, 2009.

Brahmbhatt, Samarth and Hays, James. Deepnav: Learning to navigate large cities. *arXiv preprint arXiv:1701.09135*, 2017.

Brunner, Gino, Richter, Oliver, Wang, Yuyi, and Wattenhofer, Roger. Teaching a machine to read maps with deep reinforcement learning. *arXiv preprint arXiv:1711.07479*, 2017.

Chaplot, Devendra Singh, Sathyendra, Kanthashree Mysore, Pasumarthi, Rama Kumar, Rajagopal, Dheeraj, and Salakhutdinov, Ruslan. Gated-attention architectures for task-oriented language grounding. *arXiv preprint arXiv:1706.07230*, 2017.

Chaplot, Devendra Singh, Parisotto, Emilio, and Salakhutdinov, Ruslan. Active neural localization. *International Conference on Learning Representations*, 2018.

Donahue, Jeffrey, Anne Hendricks, Lisa, Guadarrama, Sergio, Rohrbach, Marcus, Venugopalan, Subhashini, Saenko, Kate, and Darrell, Trevor. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2625–2634, 2015.

Dosovitskiy, Alexey and Koltun, Vladlen. Learning to act by predicting the future. *arXiv preprint arXiv:1611.01779*, 2016.

Dosovitskiy, Alexey, Ros, German, Codevilla, Felipe, López, Antonio, and Koltun, Vladlen. Carla: An open urban driving simulator. *arXiv preprint arXiv:1711.03938*, 2017.

Espeholt, Lasse, Soyer, Hubert, Munos, Remi, Simonyan, Karen, Mnih, Volodymir, Ward, Tom, Doron, Yotam, Firoiu, Vlad, Harley, Tim, Dunning, Iain, Legg, Shane, and Kavukcuoglu, Koray. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. *arXiv preprint arXiv:1802.01561*, 2018.

Graves, Alex, Bellemare, Marc G, Menick, Jacob, Munos, Remi, and Kavukcuoglu, Koray. Automated curriculum learning for neural networks. *arXiv preprint arXiv:1704.03003*, 2017.

Gupta, Saurabh, Davidson, James, Levine, Sergey, Sukthankar, Rahul, and Malik, Jitendra. Cognitive mapping and planning for visual navigation. *arXiv preprint arXiv:1702.03920*, 2017.

Hermann, Karl Moritz, Hill, Felix, Green, Simon, Wang, Fumin, Faulkner, Ryan, Soyer, Hubert, Szepesvari, David, Czarnecki, Wojtek, Jaderberg, Max, Teplyashin, Denis, et al. Grounded language learning in a simulated 3d world. *arXiv preprint arXiv:1706.06551*, 2017.

Hill, Felix, Hermann, Karl Moritz, Blunsom, Phil, and Clark, Stephen. Understanding grounded language learning agents. *arXiv preprint arXiv:1710.09867*, 2017.

Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Jaderberg, Max, Mnih, Volodymyr, Czarnecki, Wojciech Marian, Schaul, Tom, Leibo, Joel Z, Silver, David, and Kavukcuoglu, Koray. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

Kempka, Michał, Wydmuch, Marek, Runc, Grzegorz, Toczek, Jakub, and Jaśkowski, Wojciech. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *Computational Intelligence and Games (CIG), 2016 IEEE Conference on*, pp. 1–8. IEEE, 2016.

Kendall, Alex, Grimes, Matthew, and Cipolla, Roberto. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Computer Vision (ICCV), 2015 IEEE International Conference on*, pp. 2938–2946. IEEE, 2015.

Khosla, Aditya, An An, Byoungkwon, Lim, Joseph J, and Torralba, Antonio. Looking beyond the visible scene. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3710–3717, 2014.

Kolve, Eric, Mottaghi, Roozbeh, Gordon, Daniel, Zhu, Yuke, Gupta, Abhinav, and Farhadi, Ali. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017.

Lample, Guillaume and Chaplot, Devendra Singh. Playing FPS games with deep reinforcement learning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Malinowski, Mateusz, Rohrbach, Marcus, and Fritz, Mario. Ask your neurons: A deep learning approach to visual question answering. *International Journal of Computer Vision*, 125(1-3):110–135, 2017.

Milford, Michael J, Wyeth, Gordon F, and Prasser, David. Ratslam: a hippocampal model for simultaneous localization and mapping. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 1, pp. 403–408. IEEE, 2004.

Mirowski, Piotr, Pascanu, Razvan, Viola, Fabio, Soyer, Hubert, Ballard, Andrew J, Banino, Andrea, Denil, Misha, Goroshin, Ross, Sifre, Laurent, Kavukcuoglu, Koray, et al. Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673*, 2016.

Mnih, Volodymyr, Badia, Adria Puigdomenech, Mirza, Mehdi, Graves, Alex, Lillicrap, Timothy, Harley, Tim, Silver, David, and Kavukcuoglu, Koray. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pp. 1928–1937, 2016.

Nikolay Savinov, Alexey Dosovitskiy, Vladlen Koltun. Semi-parametric topological memory for navigation. *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SygwwGbRW.

Parisotto, Emilio and Salakhutdinov, Ruslan. Neural map: Structured memory for deep reinforcement learning. *arXiv preprint arXiv:1702.08360*, 2017.

Shah, Shital, Dey, Debadeepta, Lovett, Chris, and Kapoor, Ashish. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*, pp. 621–635. Springer, 2018.

Tessler, Chen, Givony, Shahar, Zahavy, Tom, Mankowitz, Daniel J., and Mannor, Shie. A deep hierarchical approach to lifelong learning in minecraft. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Tieleman, Tijmen and Hinton, Geoffrey. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

Ummenhofer, Benjamin, Zhou, Huizhong, Uhrig, Jonas, Mayer, Nikolaus, Ilg, Eddy, Dosovitskiy, Alexey, and Brox, Thomas. Demon: Depth and motion network for learning monocular stereo. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 5, 2017.

Walch, Florian, Hazirbas, Caner, Leal-Taixé, Laura, Sattler, Torsten, Hilsenbeck, Sebastian, and Cremers, Daniel. Image-based localization with spatial lstms. *arXiv preprint arXiv:1611.07890*, 2016.

Weyand, Tobias, Kostrikov, Ilya, and Philbin, James. Planet-photo geolocation with convolutional neural networks. In *European Conference on Computer Vision*, pp. 37–55. Springer, 2016.

Wu, Yi, Wu, Yuxin, Gkioxari, Georgia, and Tian, Yuandong. Building generalizable agents with a realistic and rich 3d environment. *arXiv preprint arXiv:1801.02209*, 2018.

Zaremba, Wojciech and Sutskever, Ilya. Learning to execute. *arXiv preprint arXiv:1410.4615*, 2014.

Zhang, Jingwei, Tai, Lei, Boedecker, Joschka, Burgard, Wolfram, and Liu, Ming. Neural slam: Learning to explore with external memory. *arXiv preprint arXiv:1706.09520*, 2017.

Zhu, Yuke, Mottaghi, Roozbeh, Kolve, Eric, Lim, Joseph J., Gupta, Abhinav, Fei-Fei, Li, and Farhadi, Ali. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation, ICRA*, pp. 3357–3364, 2017.

A. Video of the Agent Trajectories and Observations

The video available at https://sites.google.com/view/streetlearn shows the performance of trained *CityNav* agents in the Paris Rive Gauche and Central London environments, as well as of the *MultiCityNav* agents trained jointly on 4 environments (Greenwich Village, Midtown, Central Park and Harlem) and then transferred to a fifth environment (Lower Manhattan). The video shows the high-resolution StreetView images (actual inputs to the network are 84×84 RGB observations), overlaid with the map of the environment indicating its position and the location of the goal.

B. Further Analysis

B.1. Architecture Ablation Analysis

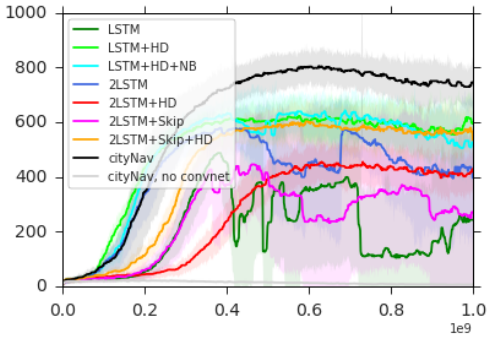

Figure 12. Learning curves of the *CityNav* agent (2LSTM+Skip+HD) on NYU, comparing different ablations, all they way down to *GoalNav* (LSTM). 2LSTM architectures have a global pathway LSTM and a policy LSTM with optional Skip connection between the convnet and the policy LSTM. HD is the heading prediction auxiliary task.

In this analysis, we focus on agents trained with a two-day learning curriculum and early rewards at 200m, in the NYU environment. Here, we study how the learning benefits from various auxiliary tasks as well as we provide additional ablation studies where we investigate various design choices. We quantify the performance in terms of average reward per episode obtained at goal acquisition. Each experiment was repeated 5 times with different seeds. We report average final reward and plot the mean and standard deviation of the reward statistic. As Fig. 12 shows, the auxiliary task of heading (*HD*) prediction helps in achieving better performance in the navigation task for the *GoalNav* architecture, and, in conjunction with a skip connection from the convnet to the policy LSTM, for the 2-LSTM architecture. The *CityNav* agent significantly outperforms our main baseline *GoalNav* (LSTM in Fig. 12), which is a goal-conditioned variant of the standard RL baseline (Mnih et al., 2016). *CityNav* perfoms on par with *GoalNav* with heading prediction, but the latter cannot adapt to new cities without re-training or adding city-specific components, whereas the *MultiCityNav* agent with locale-specific LSTM pathways can, as demonstrated in the paper's section on transfer learning. Our weakest baseline (*CityNav* no vision) performs poorly as the agent cannot exploit visual cues while performing navigation tasks. In our investigation, we do not consider other auxiliary tasks introduced in prior works (Jaderberg et al., 2016; Mirowski et al., 2016) as they are either unsuitable for our task, do not perform well, or require extra information that we consider too strong. Specifically, we did not implement the reward prediction auxiliary task on the convnet from (Jaderberg et al., 2016), as the goal is not visible from pixels, and the motion model of the agent with widely changing visual input is not appropriate for the pixel control tasks in that same paper. From (Mirowski et al., 2016), we kept the 2-LSTM architecture and substituted depth prediction (which we cannot perform on this dataset) by heading and neighbor traversability prediction. We did not implement loop-closure prediction as it performed poorly in the original paper and uses privileged map information.

B.2. Allocentric and Egocentric Goal Representation

We do an analysis of the activations of the 256 hidden units of the region-specific LSTM, by training decoders (2-layer multi-layer perceptrons, MLP, with 128 units on the hidden layer and rectified nonlinearity transfer functions) for the allocentric position of the agent and of the goal as well as for the egocentric direction towards the goal. Allocentric decoders are multinomial classifiers over the joint Lat/Long position, and have 50×50 bins (London) or 35×35 bins (NYU), each bin covering an area of size 100m$\times$100m. The egocentric decoder has 16 orientation bins. Fig.13 illustrates the noisy decoding of the agent's position along 3 trajectories and the decoding of the goal (here, St Paul's), overlayed with the ground truth trajectories and goal location. The average error of the egocentric goal direction decoding is about 60°(as compared to 90°for a random predictor), suggesting some decoding but not a cartesian manifold representation in the hidden units of the LSTM.

B.3. Reward Shaping: Goal Rewards vs. Rewards

The agent is considered to have reached the goal if it is within 100m of the goal, and the reward shaping consists in giving the agent early rewards as it is within 200m of the goal. Early rewards are shaped as following:

$$r_t = \max\left(0, \min\left(1, \frac{200 - d_t^g}{100}\right)\right) \times r^g$$

where d_t^g is the distance from the current position of the agent to the goal and r^g is the reward that the agent will receive if it reaches the goal. Early rewards are given only

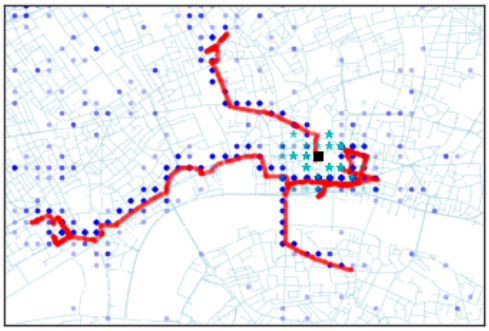

Figure 13. Decoding of the agent position (blue dots) and goal position (cyan stars) over 3 trajectories (in red) with a goal at St Paul's Cathedral, in London (in black).

once per panorama / node, and only if the distance d_t^g to the goal is decreasing (in order to avoid the agent developing a behavior of harvesting early rewards around the goal rather than going directly towards the goal). However, depending on the path taken by the agent towards the goal, it could earn slightly more rewards if it takes a longer path to the goal rather than a shorter path. Throughout the paper, and for ease of comparison with experiments that include reward shaping, we report only the rewards at the goal destination (*goal rewards*).

C. Implementation Details

C.1. Neural Network Architecture

For all the experiments in the paper we use the standard vision model for Deep RL (Mnih et al., 2016) with 2 convolutional layers followed by a fully connected layer. The baseline *GoalNav* architecture has a single recurrent layer (LSTM), from which we predict the policy and value function, similarly to (Mnih et al., 2016).

The convolutional layers are as follows. The first convolutional layer has a kernel of size 8x8 and a stride of 4x4, and 16 feature maps. The second layer has a kernel of size 4x4 and a stride of 2x2, and 32 feature maps. The fully connected layer has 256 units, and outputs 256-dimensional visual features f_t. Rectified nonlinearities (ReLU) separate the layers.

The convnet is connected to the policy LSTM (in case of two-LSTM architectures, we call it a *Skip* connection). The policy LSTM has additional inputs: past reward r_{t-1} and previous action $\mathbf{a}_t$ expressed as a one-hot vector of dimension 5 (one for each action: forward, turn left or right by 22.5°, turn left or right by 67.5°).

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 (see Section D for the complete list of landmark locations in the New York and London environments). In the case of Lat/Long scalars, g_t is a 2-dimensional vector of Lat and Long coordinates normalized to be between 0 and 1 in the environment of interest. In the case of binned Lat/Long coordinates, we bin the normalized scalar coordinates using 35 bins for Lat and 35 bins for Long in the NYU environment, each bin representing 100m, and the vector g_t contains 70 elements.

The goal LSTM also takes 256-dimensional inputs from to 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. We chose to use this bottleneck, consisting of a dropout, linear layer from 256 to 64, followed by a nonlinearity, in order to force the representations in the goal LSTM to be more robust to noise and to send only a small amount of information (possibly related to the egocentric position of the agent w.r.t. the goal) to the policy LSTM. Please note that the *CityNav* agent can still be trained to solve the navigation task without that layer.

Similarly to (Mnih et al., 2016), the policy LSTM contains 256 hidden units, followed by two parallel layers: one linear layer going from 256 to 1 and outputing the value function, and one linear layer going from 256 to 5 (the number of actions), and a softmax nonlinearity, outputting the policy.

The heading θ_t prediction auxiliary task is done using an MPL with a hidden layer of 128 units, connected to the hidden units of each goal LSTM in the *CityNav* and *MultiCityNav* agents, and outputs a softmax of 16-dimensional vectors, corresponding to 16 binned directions towards North. The auxiliary task is optimized using a multinomial loss.

C.2. 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 (Tieleman & Hinton, 2012). The weight of heading prediction is 1, the entropy cost is 0.004 and the value baseline weight is 0.5.

In all our experiments, we train our agent with IMPALA (Espeholt et al., 2018), an actor-critic implementation of deep reinforcement learning that decouples acting and learning. In our experiments, IMPALA results in similar performance to A3C (Mnih et al., 2016) on a single city task, but as it has been demonstrated to handle better multi-task learning than A3C, we prefer it to A3C for our multi-city and transfer learning experiments. We use 256 actors for *CityNav* and 512 actors for *MultiCityNav*, with batch sizes of 256 or 512

respectively, and sequences are unrolled to length 50. We used a learning rate of 0.001, linearly annealed to 0 after 2B steps (NYU), 4B steps (London) or 8B steps (multi-city and transfer learning experiments). The discounting coefficient in the Bellman equation is 0.99. Rewards are clipped at 1 for the purpose of gradient calculations.

C.3. Curriculum Learning

Because of the distributed nature of the learning algorithm, it was easier to implement the duration of phase 1 and phase 2 of curriculum learning using the Wall clock of the actors and learners rather than by sharing the total number of steps with the actors, which explains why phase durations are expressed in terms of days, rather than in a given number of steps. With our software implementation, hardware and batch size as well as number of actors, the distributed learning algorithm runs at about 6000 environment steps/sec, and a day of training corresponds to about 500M steps.

D. Environment

For the experiments on data from Manhattan, New York, we relied on sub-areas of a larger StreetView graph that contains 256961 nodes and 266040 edges. We defined 5 areas by selecting 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. We defined areas as following:

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

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). 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).

We provide, in a text file[3], the locations of the 644 landmarks used throughout the study.

[3] Available at https://sites.google.com/view/streetlearn