Transfer Learning for Human Navigation and Triage Strategies Prediction in a Simulated Urban Search and Rescue Task

Yue Guo[1], Rohit Jena[1], Dana Hughes[1], Michael Lewis[2], Katia Sycara[1]

***Abstract*— To build an agent providing assistance to human rescuers in an urban search and rescue task, it is crucial to understand not only human actions but also human beliefs that may influence the decision to take these actions. Developing data-driven models to predict a rescuer's strategies for navigating the environment and triaging victims requires costly data collection and training for each new environment of interest. Transfer learning approaches can be used to mitigate this challenge, allowing a model trained on a *source environment/task* to generalize to a previously unseen *target environment/task* with few training examples. In this paper, we investigate transfer learning (a) from a source environment with smaller number of types of injured victims to one with larger number of victim injury classes and (b) from a smaller and simpler environment to a larger and more complex one for navigation strategy. Inspired by hierarchical organization of human spatial cognition, we used graph division to represent spatial knowledge, and Transfer Learning Diffusion Convolutional Recurrent Neural Network (TL-DCRNN), a spatial and temporal graph-based recurrent neural network suitable for transfer learning, to predict navigation. To abstract the rescue strategy from a rescuer's field-of-view stream, we used attention-based LSTM networks. We experimented on various transfer learning scenarios and evaluated the performance using mean average error. Results indicated our assistant agent can improve predictive accuracy and learn target tasks faster when equipped with transfer learning methods.**

I. INTRODUCTION

In an urban search and rescue (USAR) task, human rescuers may navigate better and rescue more victims with the help of an artificial agent that observes and predicts their navigation and rescue activities, and opportunistically intervenes to give them assistance. Simple assistance and guidance include reminding rescuers not to revisit an area already visited, how to efficiently go to desired places, and whether they are likely to find victims that they would consider high priority in saving. The utility of an agent's advice to rescuers is dependent on the accuracy of agent's predictions of the rescuer's intents; interventions based on incorrect predictions may be misleading.

Behavior prediction in an USAR task is challenging due to various factors, including but not limited to (a) incomplete information about where victims are, (b) changes in environmental and victim conditions, and (c) difficulty in obtaining data on rescue missions performed by humans. In a natural disaster scenario, such as after an earthquake, traversability of a building may change due to holes opening in walls or debris blocking passages, requiring rescuers to form ad-hoc navigation strategies during exploration.

Rescuers also may be faced with decisions regarding *triage* (providing essential medical care to victims) priority. For example, a rescuer may decide to temporarily disregard lightly injured victims in order to search for and triage critical victims first, triaging lightly injured victims later.

In USAR tasks, real data is expensive to obtain, and is usually associated with unique configurations of the environment. This raises the question: how can an agent learn navigation and victim triage prediction in USAR task that can efficiently generalize and transfer to more complex environments and tasks?

Representation and abstraction of spatial recognition in the humans has been studied widely. The knowledge human navigation relies on can be primarily characterized by a labeled graph [1]. Instead of perceiving environments into a global coordinate system, this cognitive map built from local information cannot guarantee geometric consistency [2]. There is a general agreement that people have hierarchical representations of space [3], [4], [5], [6], and this typically leads to the wrong answer to trick questions such as "What direction is Reno from San Diego?" (Answer: Northwest). Many people know San Diego is in California and Reno is in Nevada, and the spatial perception on state locations misleads that on cities.

Similar effects are found on a local scale where people cluster buildings and other landmarks together into regions. Distances between locations within a cluster are judged to be shorter than they actually are, while the distance between locations in different clusters are judged to be longer.

Simulations of learning and question answering based on this hypothesized hierarchical organization have been developed and analyzed in various works [7], [8], [9].

The goal of this paper is to develop transfer learning methods that provide zero- or few-shot transfer for human navigation and triage strategies from a *source* domain to a more complex *target* environment and task. In particular, we developed effective transfer learning techniques for navigation prediction from a smaller to a larger indoor environment and for a larger number of victim injury criticality classes. In this work, we used the 3D Minecraft platform [10], [11] as a testbed. Both source and target environments represented building interior spaces. The two environments had different room layouts, and we collected human rescuer navigation trajectories while they performed the simulated USAR tasks. We augmented the trajectories collected from

[1]The authors are with School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA. Email: {yueguo, rjena, danahugh, sycara}@andrew.cmu.edu

[2]The author is with the University of Pittsburgh, Pittsburgh, Pennsylvania, USA. Email: ml@sis.pitt.edu

human participants with trajectories generated by rule-based simulated participants, in order to introduce more diversity to the set of rescue strategies for training.

In this paper, we demonstrate several benefits of our approach: (1) the agent is able to make good predictions on the navigation and triage strategy of rescuers, (2) its prediction accuracy grows fast even with a small input number of trajectories, (3) the convergence time of training process is shortened significantly in the target domain while not affecting the accuracy, and (4) the experiment demonstrates the potential of transfer learning for USAR missions.

II. RELATED WORK

For the past decade, transfer learning has been studied extensively [12], [13], [14], [15], [16] . It has been recently used in reinforcement learning, where multiple tasks are learnt instead of a single one. Knowledge gained in some Markov Decision Processes can be leveraged to speed up the solutions of others [17], [18], [19], In computer vision, Joint Distribution Adaptation is proposed for robust transfer learning [20], which jointly adapts the marginal and conditional distribution. Domain invariant features of the source and target are extracted for visual object recognition [21].

Transfer learning applied to graphs has recently gained attention. [22] proposes a network transfer learning framework using the adversarial domain and graph convolution. Although training and test data still require having the same feature space and distribution, new tasks that share similar representations can be resolved easier in [23], and this work transfers the geometric information from source to target. Graph-based domain mapping is used to identify previously encountered games, and this provides a good starting place for learning [24]. With graph based skill acquisition, [23] and [25] capture community detection from a connectivity graph, and speed up learning using the transferred knowledge. The theoretical grounded framework for the transfer learning of GNNs can be found in [26].

There are works on training an agent to navigate while adapting to new environments, including [27] where knowledge is transferred from previous navigation tasks using a successor-feature-based RL algorithm. In [28], the autonomous agent is trained to navigate in diverse city environments, while performing transferred tasks of navigating in target new locations. We build upon our previous work on the observing and predicting agent [29] by incorporating a graphical representation of the environment, amenable to inference using graph embedded Recurrent Neural Network models and transfer to new domains (i.e., environment layouts). We refer readers to [30] [29] for the data collection process where rescuers are equipped with Minecraft skills.

Our approach to navigation prediction builds upon the Transfer Learning Diffusion Convolutional Recurrent Neural Network (TL-DCRNN) architecture presented in [31] to infer participant navigation, partitioning each domain's map utilizing the concept of a "clique" in a graph (a set of connected nodes) to form our clique group assignment for graph division. The original work predicts on traffic flow based on the Diffusion Convolutional Recurrent Neural Network (DCRNN) model proposed in [32]. It was able to predict the traffic flow of one city using the data collected from another. In our work, instead of using the partition for cities [33], we partition the graph based on the spatial recognition of the rescuers when performing the search task.

III. METHODS

The transfer-learning agent we developed is able to predict the navigation and triage activities of rescuers in the source task, using training data from human Minecraft players. The agent divides the space using graph methods and utilizes the TL-DCRNN to predict navigation when the rescuers search for victims; it also uses LSTM to predict triage strategies when the rescuers found victims and made decisions. In this work, we transfer the prediction model which is trained with the old scenarios, to predict the navigation and rescue activities of human rescuers in an unobserved situation. The implementation code has been released.[1]

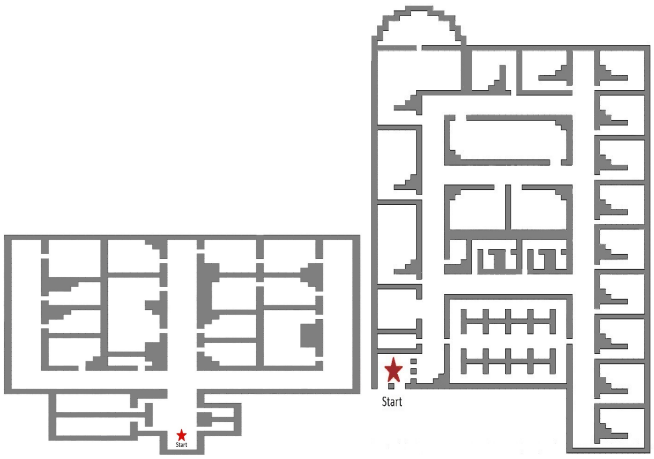
A. USAR Domains

To train agents and evaluate the ability of agents to transfer to new domains, multiple USAR task domains were developed using two distinct building layouts (shown in Figure 1): a smaller map with fewer rooms (referred to as *Sparky*), and a larger, more complex map (referred to as *Falcon*). For each domain, victims were placed in specific locations of the environment, and the environment was perturbed with blockage and holes on the walls, which are not initially known to rescuers, but are encountered as rescuers navigate the environment. Blockages in the hallways and rooms might make certain paths in the map impassable, while holes on the walls opened other possibilities for navigation. For prediction of rescuer navigation, a total of four task domains were created by varying the location of victims and perturbations: a single variant of the Sparky map, and three variants of the Falcon map of increasing difficulty (*Falcon-easy*, *Falcon-medium*, *Falcon-hard*), shown in Figure 2.

For prediction of rescuer triage strategy, we created two domains using the *Falcon* environment. The first domain, *Falcon-2victim*, contains victims of two severity levels, which are common to all domains: *regular* victims require 7 seconds to triage, and *critical* victims require 15 seconds to triage, and will expire 5 minutes after the start of the mission. The second domain, *Falcon-3victim*, introduces an additional victim severity level, *medium* victims, which require 12 seconds to triage and will expire 7 minutes into the mission.

The USAR mission required a human player/rescuer to work in an indoor environment in a limited time where victims were scattered. The rescuer is equipped with a medical kit which enabled them to triage victims in a few seconds. A device that beeps to report if one or more live victims were inside a room was utilized, however the human participant may or may not have understood the significance of the beeping. Knowledge of the beeping was a condition

[1] https://github.com/sophieyueguo/tl_navi



(a) Sparky (b) Falcon

Fig. 1: The Original Maps of Sparky and Falcon. Red stars indicate start locations.

Fig. 2: Map of the three versions of Falcon: easy, medium, and hard collected in [30]. Grey indicates walls, Magenta indicates blockages, and Cyan is for openings.

that was varied in different trials, i.e. in some conditions the participant was told the meaning of the beep at the beginning of the experiment, whereas in other conditions, the participant may not have been given this information.

B. Agent Observations

The agent observes a stream of information containing the following values from the rescuer's trajectory:

- **Rescuer State:** The position and orientation of the rescuer in the environment,
- **Field of View Contents:** Blocks of interest (e.g., victims) that are present in the rescuer's field of view,
- **Victim Interaction Events:** Including starting, completing, and abandoning triage attempts,
- **Environment Interaction Events:** Including opening or closing doors, and entering or exiting rooms,
- **Victim Location Device:** Emitted beep when the rescuer is near a room containing an untriaged victim.

The Rescuer State and Field of View Contents update at a rate of 10Hz, all other observations are asynchronous events.

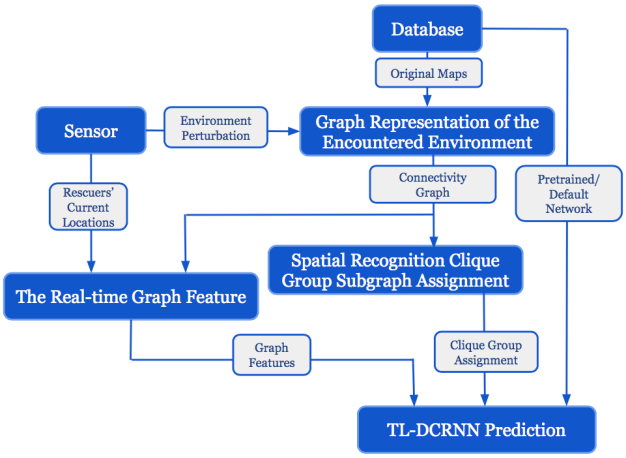


Fig. 3: The Architecture of the Navigation Prediction Process

C. Navigation Prediction

The navigation prediction task involves the agent predicting the next room a test (unknown) human will enter next, predicted at particular time intervals. Figure 3 illustrates the main components / processes of our architecture (blue blocks) and data flow between the components (grey blocks). In addition to the agent's observation stream listed above, the agent is provided with a map of the environment and victim and environmental perturbations (e.g., debris blockages and openings in walls), from which it generates a graph-based representation of the environment. The agent updates the representation based on victims and environment perturbations observed by the rescuer in the agent's data stream. Features are extracted from the graph-based representation, and a TL-DCRNN model is used to forecast future features, from which room visitation predictions can be made.

1) Graph Representation: The original map of the environment is converted into an abstract connectivity graph, $G = \{V, E\}$, where each $v \in V$ represents a predefined, semantically meaningful region (room, a segment of hallway, or intersections), and $e(v_1, v_2) \in E$ indicates connectivity between the regions v_1 and v_2 in the following way. Each (x, y, z) location in the original map is manually assigned to a region; environment perturbations result in edges either being added (in the case of a wall opening) or removed (in the case of rubble blockage).

2) Spatial Recognition Clique Group Subgraph Assignment: Given the connectivity graph $G = \{V, E\}$, the subgraph list $\{G_c\}$ is referred to as "clique group". A clique is defined as a set of rooms with a door adjacent to a common hallway segment. Formally, for a hallway segment, $v_h \in V$, the clique associated with the segment is defined as $G_c^h \triangleq \{v_i | (v_i, v_h) \in E; v_i \text{ not a hallway segment}\}$. Hallway segment boundaries are defined by walls or perturbations blocking passage, as well as intersections with other perpendicular hallway segments (i.e., T-intersections and corners). Clique

groups are the hallway clique unions that share elements, defined as $G_c \triangleq \{G_c^{h_1} \cup G_c^{h_2} | G_c^{h_1} \cap G_c^{h_2} \neq \emptyset\}$. We hypothesize that rescuers are likely to explore all rooms within a clique, and navigate to an adjacent, unexplored clique once the current clique has been fully explored.

3) Real-time Graph Feature Generation: Graph features are computed whenever the rescuer enters a new vertex in the graph representation of the map (i.e., enters a new room or hallway segment). Graph features are calculated using a history of the previous T vertices the rescuer occupied. A single graph feature at time t is a vector denoted as $f_t \in \mathbb{R}^k$, where k is the number of vertices in the connectivity graph. Each element in the feature vector is defined as $f_t^{(i)} = \gamma^{(t-t_i)}$, where t_i is the most recent time step when the rescuer was located in v_i. In essence, the feature vector captures the region the rescuer currently occupies (indicated with a value of 1 at the index of the region), and the recent history of the rescuer's trajectory (indicated by values less than 1 in the feature vector, where elements with larger values correspond to more recently visited regions).

4) TL-DCRNN Navigation Prediction: The future sequence of rooms visited by the rescuer is predicted using a TL-DCRNN model. The model takes as input as sequence of graph features, $F_t = \{f_{t-T+1}, f_{t-T+2} \ldots f_t\}$, and the subgraph list $\{G_c\}$, and predicts the future sequence of graph features, $F_t^{output} = \{f_{t+1}, f_{t+2} \ldots f_{t+T}\}$. The TL-DCRNN model learns as set of filter weights, $\mathbf{W}_O$ and $\mathbf{W}_I$, which intuitively represent the likelihood of the next room visited by the rescuer (outflow), and the likelihood of the room previously visited (inflow). After partitioning the graph, the sub-graph features are passed to a Recurrent Neural Network (RNN), which outputs the series of predicted graph features. Graph features are calculated using a diffusion process defined by the equation $\mathbf{W}_G F = \sum_{d=0}^{K-1} (\mathbf{W}_O (\mathbf{D}_O^{-1}\mathbf{A})^d + \mathbf{W}_I (\mathbf{D}_I^{-1}\mathbf{A})^d) F$ [31]. Here, K refers to the maximum number of steps taken for diffusion, $\mathbf{A}$ is the adjacency matrix of the graph, $\mathbf{D}_O$ and $\mathbf{D}_I$ are in-degree and out-degree diagonal matrices of graph nodes, thus $\mathbf{D}_O^{-1}\mathbf{A}$ and $\mathbf{D}_I^{-1}\mathbf{A}$ are transition matrices which indicate the flow, or transition frequency between regions. The predicted sequence of rooms visited by the rescuer can be extracted from the index of the maximum value of the predicted feature vectors at each time step.

D. Triage Strategy Prediction

In the triage task, the agent predicts over time the next victim type the human rescuer will triage. We hypothesize that a rescuer's triage strategy is agnostic to the location of the victim, and only depends on the reward that a rescuer gets in triaging a victim of a particular severity class of injuries.

1) Triage Strategy: We formulate the triage strategy prediction as a classification problem, using a sequence of Field of View observations (specifically, the location and severity of victims within the rescuer's Field of View) and observation timestamp as input.

Based on the set of victim severity levels, we identify four categories of strategies a rescuer can employ for triaging victims:

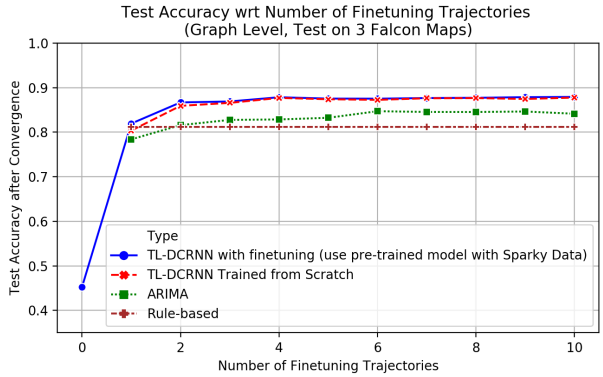
1) **Strict:** Rescuer exclusively triages *critical* victims during the first 5 minutes, *medium* victims from 5–7 minutes and *regular* victims from 7–10 minutes. In the *Falcon-2victim* domain, the rescuer following this strategy will triage *regular* victims from 5–10 minutes.
2) **Slack:** Rescuer triages *critical* and *medium* victims as they are discovered during the first 7 minutes, then triage *regular* from 7–10 minutes.
3) **Preemptive:** Rescuer triages victims as discovered regardless of victim severity.
4) **Probabilistic:** Rescuer triages a discovered victim only if the expected number of victims that can be triaged in the remaining time is less than the number of victims remaining in the environment. The rescuer is made aware of the total number of victims from each severity class, and can therefore calculate both the expected and remaining number of victims.

Note that in the *Falcon-2victim* domain, the **Strict** and **Slack** strategies are equivalent, and is therefore considered a single category for this domain.

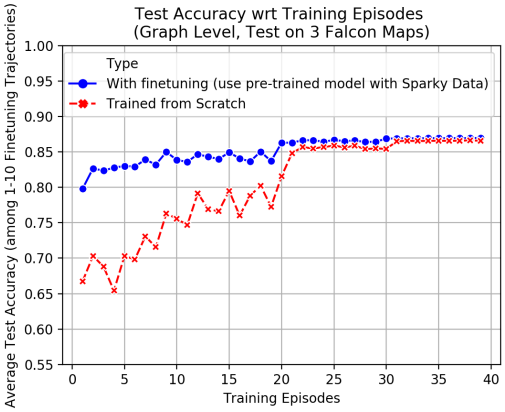
2) Attention based LSTM: We use a sinusoidal embedding for the position of the victims and the timestamp, following the works of [34]. At every timestep, we collect the list of victims, and use a common feedforward network E_ψ to convert the information into embeddings. To consider the case of no victims, we also include a "dummy" victim embedding at each timestep. Let there be N_t victims at time t. Each victim is depicted by a tuple of severity level, r, and (x,y) locations i.e. $v_i^t = (r_i^t, x_i^t, y_i^t), \quad i \in \{1 \ldots N_t\}$. This representation is fed into the feedforward network to get the victim embedding $e_i = E_\psi(v_i)$. The embeddings are given by $\{e_i\}\Big|_{i=1}^{N_t} \cup \{e_0\}$ where e_0 is the learned dummy victim embedding. This set of embeddings is passed into a self-attention network A_θ where θ are the learnable parameters, giving us modified embeddings $\{f_i\}_{i=0}^{N_t} = A_\theta\Big(\{e_i\}_{i=0}^{N_t}\Big)$. Next, we take the average of these embeddings as the "summary" vector that goes into the LSTM, $s_t = \frac{\sum_{i=0}^{N_t} f_i}{N_t+1}$. This architecture allows us to account for variable number of victims at each timestep without making the architecture task specific. This vector is used as input to the LSTM. The final triage strategy prediction y_t is given by the LSTM equation $y_t = \text{softmax}(g_\phi(h_t)), \quad h_t, c_t = \text{LSTM}(s_t, h_{t-1}, c_{t-1})$, where h_t, c_t are the hidden and context state of the LSTM, and g_ϕ is a feedforward network. The recurrent architecture allows the network to predict the triage strategy by taking into account the historical behavior of the player in terms of which victims they triaged (can be inferred from a change of victim state to "saved") and which victims are neglected.

IV. EXPERIMENTS

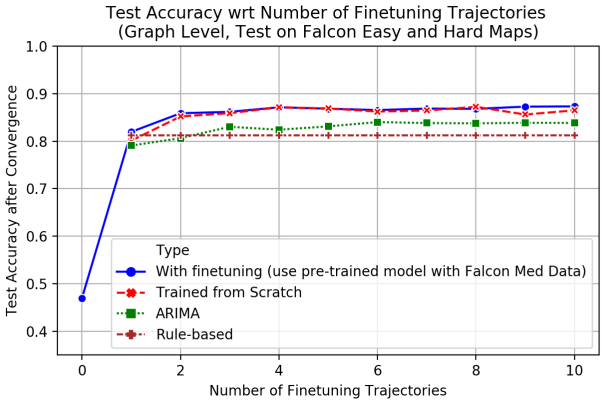
We performed a series of experiments to evaluate the agent's ability transfer prediction models trained on a source domain to a target domain in the USAR domains described in Section III-A. To train and evaluate the networks, we used a previously collected set of trajectories generated by human participants on the Falcon map variants [30],



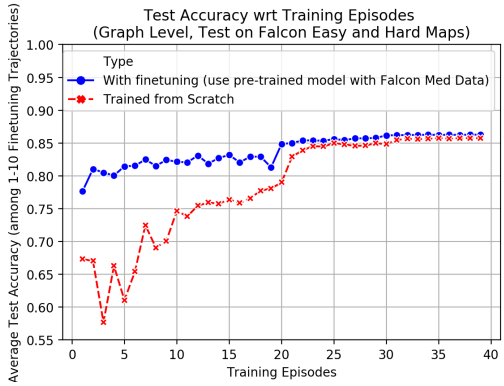
(a) Transfer from Map Sparky to Map Falcon



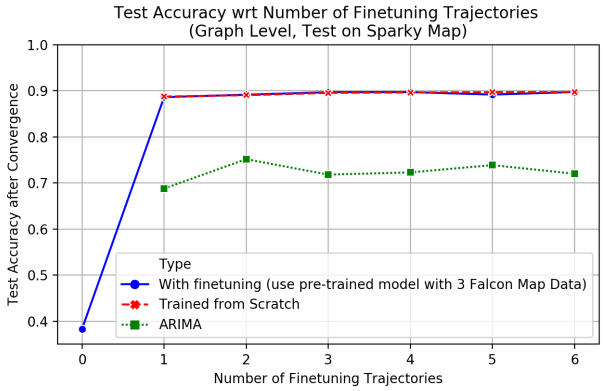
(b) Transfer from Map Sparky to Map Falcon



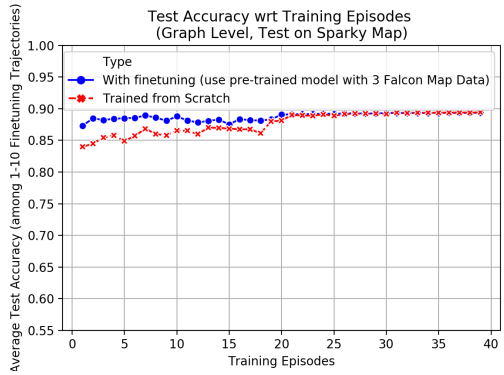
(c) Transfer from Map Falcon Med to Map Falcon Hard and Easy



(d) Transfer from Map Falcon Med to Map Falcon Hard and Easy



(e) Transfer from Map Falcon to Map Sparky



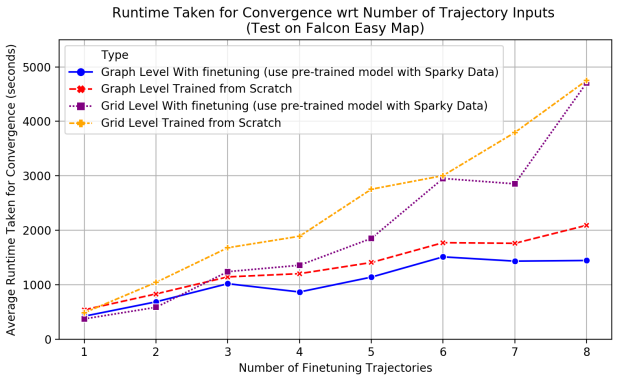
(f) Transfer from Map Falcon to Map Sparky

Fig. 4: Graph Level Transfer Learning on Navigation Prediction among Different Maps and Perturbations

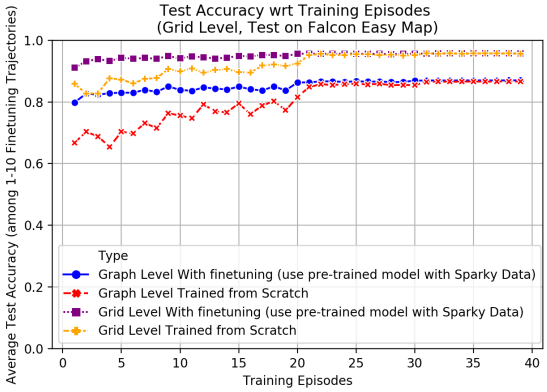
and collected trajectories generated by human participants on the Sparky map. In each experimental run, a single human participant navigated a given map and triaged victims, accumulating score points for each triaged victim. Score point were allocated in ascending order on seriousness of injury. Each participant was assigned to only one map or map variant, in the experiment. In other words no participant repeated the experiment in two or more maps.

For navigation strategy prediction, there were 8 trajectories of map Sparky for training, 138 trajectories of map Falcon for training, and 33 trajectories of map Falcon for test. Trajectories of different map Falcon perturbations were considered as different trials. For triage strategy prediction, we used a rule-based agent conditioned on rescue strategy to generate trajectories in the Falcon map. For domains for triage prediction (*Falcon-2victim*, *Falcon-3vicim*), we generated 100 trajectories for each triage strategy, resulting in a total of 300 trajectories for the *Falcon-2victim* domain and 400 trajectories for the *Falcon-3victim* domain. To introduce variation in the trajectories, we introduced perturbations in victim locations by making the victims do a "random walk" around their starting locations. Victim severity levels were also randomized in each training run.

From each rescue strategy, we used up to 70 trajectories



(a) Compare with Lower Level abstraction



(b) Compare with Lower Level abstraction

Fig. 5: Grid Level Transfer Learning on Navigation Prediction from Map Sparky to Map Falcon Easy

for training, 10 for validation, and 20 for testing.

Evaluating the transferability of the prediction models to new domains involves pre-training a model on trajectories from a *source* domain (e.g., *Falcon-easy*), and evaluating the performance of the model on predicting trajectories from a *target* domain (e.g., *Falcon-hard*) after performing additional training on trajectories from the target domain.

A. Evaluation Metrics

For evaluating navigation prediction, we used Mean Average Error (MAE), as used in [32] and [31],

$$MAE(\mathbf{f},\hat{\mathbf{f}}) = \frac{1}{|\Omega|}\sum_{i\in\Omega}|f_i - \hat{f}_i|$$

where the ground truth is represented by $\mathbf{f} = f_1, f_2 \ldots f_T$, the predictions are represented by $\hat{\mathbf{f}} = \hat{f}_1, \hat{f}_2 \ldots \hat{f}_T$, and Ω referred to the observed samples.

For triage strategy prediction, we used the mean accuracy over time (MAT) as our final evaluation metric. In the initial stages, the prediction can be random due to lack of rescuer's behavior data around the victims. Therefore, the average is taken only after one minute. Given a sequence of observations $\mathbf{O}$ and the ground truth rescue strategy $\mathbf{c}$, the metric is given by:

$$\mathbf{MAT}(\mathbf{O},\mathbf{c}) = \frac{1}{|\mathbf{O}|}\sum_{t}\mathbb{I}(\mathbf{y_t} = \mathbf{c})$$

where $\mathbf{y_t}$ is the predicted strategy at timestep t. $\mathbf{c}$ is one of the four classes mentioned in Section III-D.

B. Navigation Prediction with Transfer Learning

With the transfer learning experiments, we aim to answer the following questions regarding the navigation prediction model: (1) How effective/accurate is a model in the test domain that has learned to predict with a certain accuracy in the source domain? Does it achieve zero-shot learning? (2) How does the performance of the model in the source domain compared with a model that has been trained from scratch in the target domain? (3) How does the time required to train a source model and refine in the target domain with few-shot learning compare with the time to train a model in the target domain from scratch? (4) How does the number of input trajectories affect performance? (5) How do maps with different room configurations, vs the same map with different perturbations affect the transfer learning process? (6) Does the abstraction level of graph affect performance?

1) Baselines: We compare our mechanism with a Rule-based Navigation Prediction and a traditional time-series prediction method Autoregressive Integrated Moving Average (ARIMA) Model.

Rule-Based Model. The rule-based agent could make two levels of prediction on the rescuer's navigation behavior: (1) the next clique group, and (2) the next room(s). The rule-based prediction was dependent on the knowledge condition of rescuers, i.e. if they knew about what beeping meant, how long triaging victims would take, and how much reward they would get after finishing triaging the victims. Generally the prediction accuracy was higher for rescuers who had the knowledge of the beep. The rule-based method considers the following factors to predict next room(s):

- The current location of the rescuer
- The rooms not visited in the current clique group
- Whether the rescuer knows about beep
- Whether seriously injured victims are dead
- The rooms connected to the current location with a hole/internal door

The factors to predict next clique group:

- The current clique group:
- Whether seriously injured victims are dead
- Whether rooms in the current clique group have victims
- Whether there is a new clique group nearby

ARIMA Model. Autoregressive integrated moving average (ARIMA) was widely used [35] to perform time series prediction. One of its important variation models is Seasonal Autoregressive Integrated Moving Average with eXogenous variables (SARIMAX) [36], which includes a seasonal pat-

tern in the prediction for short-term forecasting. By indexing the active regions and concatenating trajectories of different rescuers, we are able to observe a "seasonal" pattern, where different rescuers might visit same locations in a similar order. This enabled us to implement SARIMAX as a baseline. However, this extended ARIMA model does not take a graph structure of the map into consideration, and cannot be trained with data from other maps or perturbations. It only took in the training trajectories of the target map and perturbations.

2) From Sparky to Falcon: We compared the accuracy of the pre-trained model with fine-tuning *Sparky* trajectories (in blue), i.e. adding *Falcon* trajectories to the *Sparky* pre-trained model and the accuracy of the model trained on *Falcon* from scratch (in red) in Fig. (4a) and (4b). We gradually added a few training and validation trajectories of the *Falcon* domains (*Falcon-easy*, *Falcon-medium*, *Falcon-hard*) to the Sparky model (ranging from 1 to 10 incrementally, distinguished by different map perturbations respectively and took the average). We had the following observations on the converged accuracy and the training process: (a) TL-DCRNN outperformed ARIMA (in green) and the rule-based (in brown) baselines. (b) With increasing number of train/val trajectories, the accuracy after convergence of all models slightly improved with respect to varying the number of input *Falcon* trajectories. (c) TL-DCRNN with fine-tuning and TL-DCRNN trained from scratch have similar converged accuracy, but the one with fine-tuning converged faster.

3) From Falcon-med to Falcon-hard and Falcon-easy: When using the trajectories of *Falcon* domains with different perturbation and victim densities instead of using the ones of *Sparky* domain, we obtained the results in Fig. (4c) and (4d). All the conclusions reported previously still held, and there was no significant difference in terms of accuracy after convergence or training process. This indicated that the data from the same map but with different perturbations was not necessarily more helpful than data from a different map. With perturbations, the navigation behavior can be as different as from another map.

4) From Falcon to Sparky: We reversed the process to transfer from the *Sparky* domain to the *Falcon* domain in Fig. (4e) and (4f). The *Sparky* map was relatively easier for TL-DCRNN to learn from and resulted in a higher accuracy than the previous comparison, while ARIMA was not able to improve. The TL-DCRNN also converged faster.

5) Comparison with Grid-Based Representation: We compared the graph-based representation prediction with predictions made using a grid representation of the map in Fig. (5a) and (5b). In the grid representation, each 3 cell by 3 cell region was considered as a graph node; edges connected adjacent node. The state space was 12.5 times larger. The denser graph led to a higher accuracy after convergence when transferring from *Sparky* domain to *Falcon-easy* domain. However, the run-time to convergence increased.

C. Triage Strategy Prediction with Transfer Learning

In this section, we compare the performance of the model trained on the harder *Falcon-3victim* domain from scratch, and transferring a model pre-trained on the easier *Falcon-2victim* source domain, and finetuning on few trajectories from the *Falcon-3victim* target domain, reducing the required training data and time. We therefore perform the following experiments:

1) Models trained on the *Falcon-3victim* domain from scratch, i.e. without finetuning, with a varying number of training trajectories.
2) Models pre-trained on the *Falcon-2victim* domain as a source domain with maximum number of training trajectories, and transferring the model to the *Falcon-3victim* as the target domain by finetuning with varying number of target-domain trajectories.

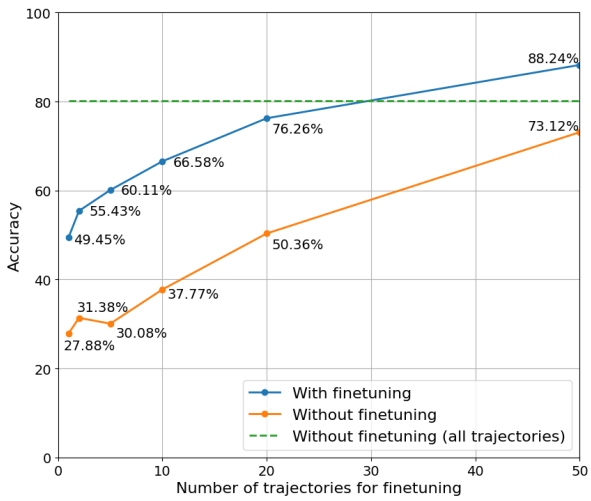


Fig. 6: Transfer learning on triage strategy prediction

The results are shown in Figure 6. The dotted plot shows the test accuracy with 70 training trajectories in the *Falcon-2victim* domain. We train with 1, 2, 5, 10, 20 and 50 training trajectories from each strategy. Finetuning improves generalization performance even given a small number, such as 2, of finetuning trajectories from target *Falcon-3victim*. In our case, the source task does not contain information about the target task at all (for example, no *Falcon-2victim* trajectory contains information about *medium* severity victims), so the finetuning indeed improves generalization. Moreover, using even as few as 30 trajectories for finetuning the *Falcon-2victim* source model reaches the performance of the *Falcon-3victim* model trained from scratch with *all* trajectories (green constant line in the figure) and for more than 30 finetuning trajectories, it outperforms the *Falcon-3victim* model.

V. CONCLUSIONS

We have built an agent that makes predictions on the navigation and rescue strategies of a human rescuers in a simulated urban search and rescue mission. We showed experimentally that: (a) using an abstract representation, i.e graphs enables efficient navigation strategy transfer from source to target domains from smaller to large maps with

different victim and perturbation configurations; (b) for triage strategy, training a source model with smaller number of victim classes and adding a few finetuning trajectories form the target domain with larger number of triage victim classes, is not only high performing and efficient, but surprisingly outperforms a model trained from scratch in the target domain, while also converging faster. This is an interesting finding and we will explore it further in additional domains in future work. We also plan to study transfer with a team of rescuers instead of a single one. This is a very challenging task, not only because of the increase in the number of humans but also because of the inter-dependency of rescuer policies since they coordinate as a team.

ACKNOWLEDGMENT

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001120C0036 and by the AFRL/AFOSR award FA9550-18-1-0251. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA).

REFERENCES

[1] E. R. Chrastil and W. H. Warren, "From cognitive maps to cognitive graphs," *PloS one*, vol. 9, no. 11, p. e112544, 2014.

[2] W. H. Warren, "Non-euclidean navigation," *Journal of Experimental Biology*, vol. 222, no. Suppl 1, 2019.

[3] A. Car, G. Taylor, and C. Brunsdon, "An analysis of the performance of a hierarchical wayfinding computational model using synthetic graphs," *Computers, environment and urban systems*, vol. 25, no. 1, pp. 69–88, 2001.

[4] S. C. Hirtle and J. Jonides, "Evidence of hierarchies in cognitive maps," *Memory & cognition*, vol. 13, no. 3, pp. 208–217, 1985.

[5] Y. Gong, Y. Liu, J. Yang, and G. Li, "Structural hierarchy of spatial knowledge based on landmarks and its application in locality descriptions," in *2010 18th International Conference on Geoinformatics*. IEEE, 2010, pp. 1–5.

[6] A. Tapus, S. Vasudevan, and R. Siegwart, "Towards a multilevel cognitive probabilistic representation of space," in *Human Vision and Electronic Imaging X*, vol. 5666. International Society for Optics and Photonics, 2005, pp. 39–48.

[7] E. Remolina, J. A. Fernandez, B. Kuipers, and J. Gonzalez, "Formalizing regions in the spatial semantic hierarchy: An ah-graphs implementation approach," in *International Conference on Spatial Information Theory*. Springer, 1999, pp. 109–124.

[8] H. Voicu, "Hierarchical cognitive maps," *Neural Networks*, vol. 16, no. 5-6, pp. 569–576, 2003.

[9] T. Madl, S. Franklin, K. Chen, R. Trappl, and D. Montaldi, "Exploring the structure of spatial representations," *PloS one*, vol. 11, no. 6, p. e0157343, 2016.

[10] S. C. Duncan, "Minecraft, beyond construction and survival," 2011.

[11] M. Johnson, K. Hofmann, T. Hutton, and D. Bignell, "The malmo platform for artificial intelligence experimentation." in *IJCAI*. Citeseer, 2016, pp. 4246–4247.

[12] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[13] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2020.

[14] Z. Zhu, K. Lin, and J. Zhou, "Transfer learning in deep reinforcement learning: A survey," *arXiv preprint arXiv:2009.07888*, 2020.

[15] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey." *Journal of Machine Learning Research*, vol. 10, no. 7, 2009.

[16] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *International conference on artificial neural networks*. Springer, 2018, pp. 270–279.

[17] A. Barreto, D. Borsa, J. Quan, T. Schaul, D. Silver, M. Hessel, D. Mankowitz, A. Zidek, and R. Munos, "Transfer in deep reinforcement learning using successor features and generalised policy improvement," in *International Conference on Machine Learning*. PMLR, 2018, pp. 501–510.

[18] A. Barreto, S. Hou, D. Borsa, D. Silver, and D. Precup, "Fast reinforcement learning with generalized policy updates," *Proceedings of the National Academy of Sciences*, vol. 117, no. 48, pp. 30 079–30 087, 2020.

[19] A. Barreto, W. Dabney, R. Munos, J. J. Hunt, T. Schaul, H. Van Hasselt, and D. Silver, "Successor features for transfer in reinforcement learning," *arXiv preprint arXiv:1606.05312*, 2016.

[20] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu, "Transfer feature learning with joint distribution adaptation," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 2200–2207.

[21] M. Baktashmotlagh, M. T. Harandi, B. C. Lovell, and M. Salzmann, "Unsupervised domain adaptation by domain invariant projection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 769–776.

[22] Q. Dai, X. Shen, X.-M. Wu, and D. Wang, "Network transfer learning via adversarial domain adaptation with graph convolution," *arXiv preprint arXiv:1909.01541*, 2019.

[23] J. Lee, H. Kim, J. Lee, and S. Yoon, "Transfer learning for deep learning on graph-structured data," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.

[24] G. Kuhlmann and P. Stone, "Graph-based domain mapping for transfer learning in general games," in *European Conference on Machine Learning*. Springer, 2007, pp. 188–200.

[25] F. Shoeleh and M. Asadpour, "Skill based transfer learning with domain adaptation for continuous reinforcement learning domains," *Applied Intelligence*, vol. 50, no. 2, pp. 502–518, 2020.

[26] Q. Zhu, Y. Xu, H. Wang, C. Zhang, J. Han, and C. Yang, "Transfer learning of graph neural networks with ego-graph information maximization," *arXiv preprint arXiv:2009.05204*, 2020.

[27] J. Zhang, J. T. Springenberg, J. Boedecker, and W. Burgard, "Deep reinforcement learning with successor features for navigation across similar environments," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 2371–2378.

[28] P. Mirowski, M. K. Grimes, M. Malinowski, K. M. Hermann, K. Anderson, D. Teplyashin, K. Simonyan, K. Kavukcuoglu, A. Zisserman, and R. Hadsell, "Learning to navigate in cities without a map," *arXiv preprint arXiv:1804.00168*, 2018.

[29] V. Jain, R. Jena, H. Li, T. Gupta, D. Hughes, M. Lewis, and K. Sycara, "Predicting human strategies in simulated search and rescue task," in *Artificial Intelligence for Humanitarian Assistance and Disaster Response Workshop, NeurIPS*, 2020.

[30] L. Huang, J. Freeman, N. Cooke, M. Cohen, X. Yin, J. Clark, M. Wood, V. Buchanan, C. Carrol, F. Scholcover, A. Mudigonda, L. Thomas, A. Teo, M. Freiman, J. Colonna-Romano, L. Lapujade, and K. Tatapudi, "Using humans' theory of mind to study artificial social intelligence in minecraft search and rescue," in *(to be submitted to the) Journal of Cognitive Science*, 2021.

[31] T. Mallick, P. Balaprakash, E. Rask, and J. Macfarlane, "Transfer learning with graph neural networks for short-term highway traffic forecasting," *arXiv preprint arXiv:2004.08038*, 2020.

[32] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," *arXiv preprint arXiv:1707.01926*, 2017.

[33] T. Mallick, P. Balaprakash, E. Rask, and J. Macfarlane, "Graph-partitioning-based diffusion convolutional recurrent neural network for large-scale traffic forecasting," *Transportation Research Record*, vol. 2674, no. 9, pp. 473–488, 2020.

[34] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS*, 2017.

[35] N. D. Uri, "Forecasting peak system load using a combined time series and econometric model," *Applied Energy*, vol. 4, no. 3, pp. 219–227, 1978.

[36] N. Liu, V. Babushkin, and A. Afshari, "Short-term forecasting of temperature driven electricity load using time series and neural network model," *Journal of Clean Energy Technologies*, vol. 2, no. 4, pp. 327–331, 2014.