Deep Learning for Exploration and Recovery of Uncharted and Dynamic Targets from UAV-like Vision

William Andrew[1], Colin Greatwood[2] and Tilo Burghardt[1]

***Abstract*— This paper discusses deep learning for solving static and dynamic search and recovery tasks – such as the retrieval of all instances of actively moving targets – based on partial-view Unmanned Aerial Vehicle (UAV)-like sensing. In particular, we demonstrate that abstracted tactic and strategic explorational agency can be implemented effectively via a single deep network that optimises in unity: the mapping of sensory inputs *and* positional history towards navigational actions. We propose a dual-stream classification paradigm that integrates one Convolutional Neural Network (CNN) for sensory processing with a second one for interpreting an evolving long-term map memory. In order to learn effective search behaviours given agent location and agent-centric sensory inputs, we train this design against 400k+ optimal navigational decision samples from each set of static and dynamic evolutions for different multi-target behaviour classes. We quantify recovery performance across an extensive range of scenarios; including probabilistic placement and dynamics, as well as fully random target walks and herd-inspired behaviours. Detailed results comparisons show that our design can outperform naïve, independent stream and off-the-shelf DRQN solutions. We conclude that the proposed dual-stream architecture can provide a unified, rationally motivated and effective architecture for solving online search tasks in dynamic, multi-target environments. With this paper we publish[3] key source code and associated models.**

I. Introduction

In this work, we propose a map-based, unified deep learning framework (see Fig. 1) applicable to recovery tasks in structured environments where an agent with local sensing and spatial-temporal long-term memory is tasked with visiting within a confined space as quickly as possible *each* of a known-size set of static or dynamically moving targets.

In the special case where agents return and target locations are fully known over time and space, the task can be mapped to the static or dynamic Travelling Salesman Problem (TSP) [2], respectively. Practical solutions for this problem class have in the past been computed using Dynamic Programming [3] or Ant Colony System optimisation [4].

We consider a more realistic scenario, where target locations are initially unknown, consequently requiring exploratory agency. The search for targets with unknown locations typically arises in search and rescue (SAR) applications [5], [6], [7]. Here specifically, application of the proposed methodology is motivated by requirements to discover constituent positions of dynamic herds in confined-area

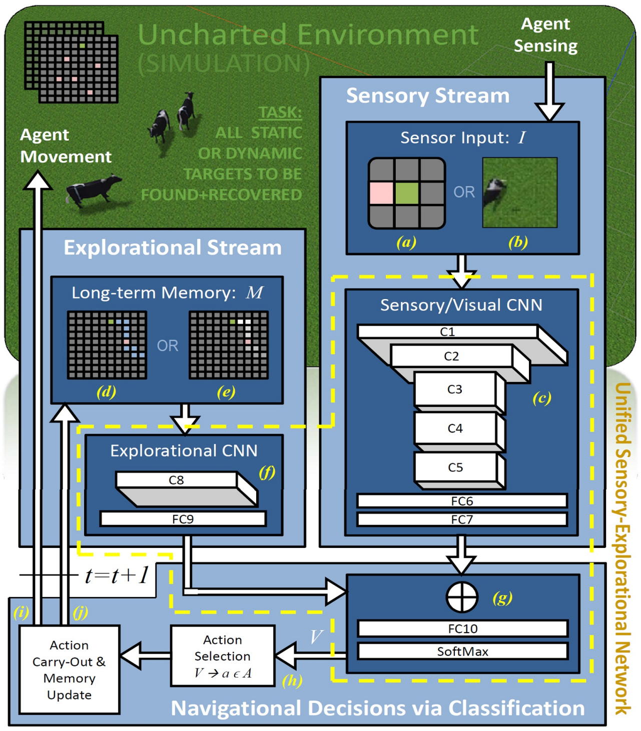


Fig. 1. **Unified Dual-Stream Deep Architecture for Search and Recovery Tasks.** The proposed design models the interpretation of sensory (tactical) and historic navigational (strategic) information within a single deep network (dotted yellow), which allows for unified back-propagation of navigation decision errors across both domains. Starting at the top right, the flow chart shows the agent's sensory input I, that is either ***(a)*** an abstracted, or ***(b)*** rendered, nearby environment sample. The input is processed via ***(c)*** a sensory/visual CNN utilising a basic AlexNet [1] design. In a second, parallel stream an evolving positional history memory M (holding either ***(d)*** spatial, or ***(e)*** spatial-temporal, long term information) is used as tensor input into ***(f)*** a network interpreting explorational histories. Both streams are concatenated into ***(g)*** a shallow integration network that culminates in a SoftMax map towards a score vector output V over the possible navigational actions $a \in A$. During training, the entire deep network (dotted yellow) is optimised based on triples (I, M, V) using one-hot encoding of V given a and cross-entropy loss. During inference at time step t, the network receives a sensory input I and ***(h)*** selects the top-ranking navigational action a based on V, which is ***(i)*** performed and, in-turn, ***(j)*** the positional history M is updated. In order to initiate a next iteration time-step $t+1$, a new sensory I is sampled from the environment E closing the operational loop.

outdoor farming, such that close-up UAV-based visual animal identification can take place [8]. The described recovery task may classically be interpreted as a partially-observable Markov decision process (POMDP), described in various surveys on visually-motivated robotic navigation [9], [10] prior to deep learning. However, in this paper, we propose to cast the task into a framework for optimising deep recurrent classification. That is, mapping positional history and current sensory inputs to new navigational actions via a single Deep Neural Network (DNN). Similar to Zhang et al. [11] in

*This work was supported by the EPSRC Centre for Doctoral Training in Future Autonomous and Robotic Systems (FARSCOPE) at the Bristol Robotics Laboratory (BRL).

[1]Department of Computer Science, [2]Department of Aerospace Engineering, Faculty of Engineering, University of Bristol, UK `{will.andrew, colin.greatwood}@bristol.ac.uk`, `tilo@cs.bris.ac.uk`

[3]Source code available at: `https://data.bris.ac.uk/data` and `https://github.com/CWOA/GTRF`

their recent work on reinforcement learning for exploration, we integrate explicit long-term memory into the design, experimenting with both storage of spatial as well as spatial-temporal information (see Figure 1 ***(d)*** and ***(e)***).

In order to focus on the core of the proposed methodology and to be able to operate on tractable computational grounds, we abstract from real-world layouts to *(1)*: model time discretely, *(2)*: represent space as a simple two-dimensional grid world and *(3)*: assume grid-cell agent localisation to be resolved perfectly. Proposed as early as 1990 [12], grid worlds remain popular for exploring the viability of AI solutions today [13], [11]. In our work, agent-centric visual sampling – approximating a low-flying UAV with a downward-facing camera – is generated either from *(1)*: occupancy of local sub-grids of the world as content-independent abstractions, or *(2)*: rendering from 3D Gazebo simulations [14], [11] for domain-specific, high-resolution images of particular target and environment content (e.g. grazing cattle herds).

Principal contributions of this work can be summarised as:

- A novel dual-steam CNN-based navigation paradigm and architecture for discovering the positions of static or dynamic targets in uncharted environments.
- Demonstrable out performance of implemented baselines when training on optimal navigation decisions.
- Proof of concept across conducted experiments in meaningful feature learning within a realistic simulation environment applicable to real-world UAV scenarios.

II. Further Related Work

A. Categorisation for Navigation

Robotic navigational research naturally relies heavily on the categorisation of sensory input. In visually-motivated navigation, approaches can largely be categorised into map-based and mapless navigation [9], [10]. Mapless vision approaches – with no global environment representation – can be found to: be based in optical-flow [15] and template appearance matching [16], [17], track landmark features [18], [19] and more recently relevant, directly classify visual input via CNNs [20], [21]. In this work, we formulate a 2D global approximation of the environment (storing visited positions) inspired by occupancy grid maps [22], [23], [24], as opposed to post-exploration map-building approaches [25] and topological map representations [26].

B. Deep Reinforcement Learning for Exploration

Seminal work in deep reinforcement learning (DRL) occurred in 2013 when researchers from DeepMind Technologies used DNNs as an approximator for learning control policies from high-dimensional sensory input such as images. Trained with a variant of Q-learning towards the goal of replicating human-level performance in playing Atari 2600 games, the term Deep Q-Network (DQN) was coined [27], [28]. Deep Recurrent Q-Networks (DRQN) [29] extend this work in a recurrent design for partial agent-environment observability with the utilisation of a Long-Short Term Memory (LSTM) layer [30] in place of the first post-convolutional fully connected layer. These works gave rise to a new genre of reinforcement learning research, and aligned with this work: learning agent navigation in complex [31] and similar [32] environments, map-less navigation [33] achieved visually [34] or via other sensor measurements. One particularly noteworthy paper employs DRL towards target-driven visual agent navigation in simulated indoor environments [35] – bearing resemblance with the problem at hand here.

One could even argue that our work falls well under the domain of classical reinforcement learning, yet we propose its application here to be unnecessarily complex and intensive; within generated episodes, target locations (containing reward) can be known (even though computationally expensive due to TSP's complexity bounds) and a globally optimal path that visits all targets whilst minimising path length can be inferred (see Section IV). Put differently, an understanding of what constitutes a good solution is known and we can train against this. We will demonstrate this knowledge to be a strong basis for DNN training, accelerating the learning process and making it applicable to sample sets considerably smaller than those required to employ DRQN successfully.

III. Problem Formalisation

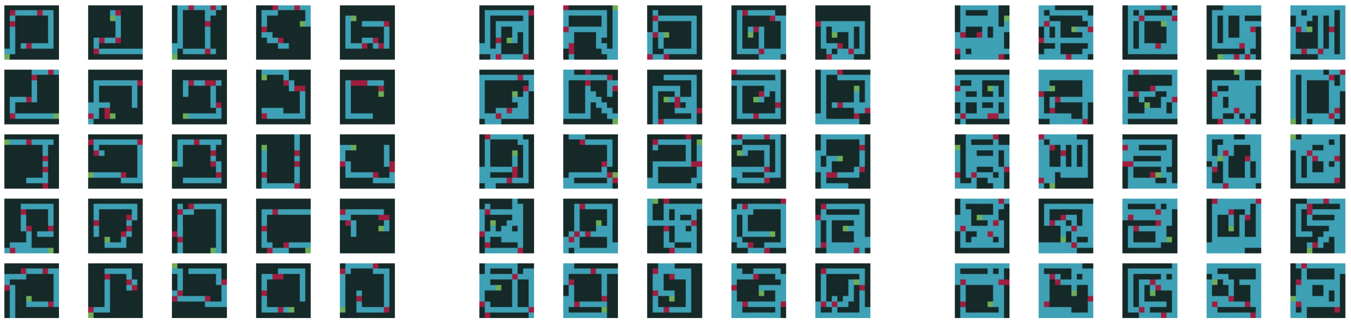
A. Base Case – Static Target Recovery

For target recovery in a grid world E under discrete time t, let E be defined as a rectangular 2D matrix with dimensions $w \times h$. The agent G and static targets $r^i \in R$ both have discrete coordinates (x, y) within the map boundaries $0 \leq x < w,\ 0 \leq y < h$ where $x, y \in \mathbb{Z}$. Exactly $|R|$ targets are placed according to some unknown (possibly random) distribution P in this world. Note that G may take position anywhere, whilst multiple targets cannot occupy the same map position:

$$\forall i, j \in |R| : r^i_x \neq r^j_x \wedge r^i_y \neq r^j_y \quad \text{where } i \neq j \text{ and } i, j \in \mathbb{I}. \tag{1}$$

Possible agent actions $a \in A$ are the four possible navigation directions for non-boundary coordinates: forward $(-y)$, backward $(+y)$, left $(-x)$, right $(+x)$ or $A = \{f, b, l, r\}$, respectively, for a top-left origin. Particular actions are inapplicable in case they would move the agent outside the map (e.g. if $G_x = G_y = 0$, corresponding possible actions are $A|G = \{b, r\},\ \forall w, h > 0$). Performing one particular action (e.g. 'move right one unit', or $x := x + 1$) is defined to take one agent step in discrete time t. The agent is deemed to have recovered a target r^i iff $G_x = r^i_x \wedge G_y = r^i_y$ at some time-step t_j. A recovery solution S to an episode (i.e. visiting all $r^i \in R$) is defined as an ordered action sequence given initial agent coordinates (e.g. $S = [f, f, l, b, r, r, b]$, $G^{init}_x = G^{init}_y = 1$). An episode terminates in S once full target visitation is achieved, or in failure due to time expiry: $t > t_e$. The quality of a solution S is defined by its cardinality $|S|$, the quality of explorational agency is defined as the average solution quality (e.g. measured as target recovery rate) computed by scenario sampling given P.

Per time step, let the agent G be provided with two observable inputs: its own current position (G_x, G_y) in coordinates of E, and an agent-centric 3×3 grid image I resolving target occupancy in E topologically adjacent to G's



(a) $0 \leq \tau \leq 10$ (b) $20 \leq \tau \leq 30$ (c) $40 \leq \tau \leq 50$

Fig. 2. **Examples of Solved Testing Episodes.** We depict 75 example episodes where the agent (trained on our base case: Random (PT)) has recovered all targets at varying degrees of efficiency (avg. 19 steps for 5 targets over 20k samples – compared to lawn-mower pattern requiring ≥ 41 steps). *(Red)*: target positions uniformly randomly generated (specifying P) at the beginning of an episode, *(blue)*: past agent positions, *(green)*: current/finishing agent position, and *(black)*: unvisited grid positions. Instances are sorted (a) - (c) according to the difference in generated and optimal solution lengths $\tau = |S| - |S_{GO}|$.

location – in essence, neighbouring targets and unoccupied cells can be sensed (see Figure 1 ***(a)***). For the base case, this abstracts from realistic visual inputs (such as Figure 1 ***(b)***) and provides a content-independent problem isolation of tactical sensing from more fine-grained visual recognition tasks. The current position is continuously recorded to keep a $w \times h$ spatial occupancy map M up-to-date, encoding for each position of world E: 1) if exploration has taken place, and 2) if the agent is currently located there (see Figure 1 ***(d)***). M in this form provides long-term positional history. Given this setup, the problem can be stated as: how well can we optimise the quality of explorational agency (see examples depicted in Figure 2) by learning information of P using only samples of navigation decisions (I, M, a)?

B. Dynamic Case – Actively Moving Target Recovery

Here the base case is extended for target motion over time. Targets now apply individual actions $a_i \in A_{ext}$ at velocity $\frac{1}{s}u/t$ (spatial grid units per time-step) – whereas the agent displaces at velocity $1u/t$. We select $s = 3$, given $s{>}1$ makes agent-target visitation eventually always possible. The action set is extended here to include action 'do nothing' denoted by n yielding $A_{ext} = \{f, b, l, r, n\}$ to permit optimised agent-target path intersection. Furthermore, single cell target occupation conditions given in Equation 1 are mandated $\forall t_i \in \{t_0, t_1, ..., t_e\}$. Architectural and temporal modifications are enacted on the agent's memory such that both agent and recovery locations are encoded spatiotemporally in the map (see Figure 1 ***(e)***), i.e. time annotations are memorised. These modifications are made to allow the system to learn about the uncertainty of target locations by relating agent paths over time and spatiotemporal recovery points along these paths – note that this information reveals properties of target *motion* beyond placement statistics according to P.

IV. Ground Truth Synthesis

To yield ground-truth solutions to generated episodes for the purpose of training data synthesis of appropriate navigation decisions (I, M, a), we employ the use of two strategies detailed as follows.

A. Fast Approximation: Closest Unvisited Target (CU)

Approximative solutions can be generated *fast* by determining the position of the closest unvisited target $r^i \in R$, selecting an appropriate action $a \in A$ towards visiting r^i and repeating until all targets have been visited. This forms an approximation in a nearest-neighbour fashion to any environment configuration irrespective of w, h, or $|R|$.

Action selection is determined by first finding angle θ between the agent G and the closest unvisited target r^i using:

$$\theta = \underline{atan2}(r^i_y - G_y, r^i_x - G_x). \tag{2}$$

Second, the action $a \in A$ is selected via $\frac{\pi}{2}$-wide intervals:

$$a = \begin{cases} rand(\{f,r\}), & \text{if } \theta = \frac{\pi}{4} \\ rand(\{b,r\}), & \text{if } \theta = -\frac{\pi}{4} \\ rand(\{b,l\}), & \text{if } \theta = -\frac{3\pi}{4} \\ rand(\{f,l\}), & \text{if } \theta = \frac{3\pi}{4} \\ f, & \text{if } \frac{\pi}{4} < \theta < \frac{3\pi}{4} \\ b, & \text{if } -\frac{3\pi}{4} < \theta < -\frac{\pi}{4} \\ l, & \text{if } \frac{3\pi}{4} < \theta < -\frac{3\pi}{4} \\ r, & \text{otherwise.} \end{cases} \tag{3}$$

where $rand(X)$ randomly selects an element $x \in X$.

B. Optimal Solution: Permutation of Targets (PT)

Globally-optimal solution(s) are some ordering on R that are associated with minimal $|S|$ for a given G^{init}_x, G^{init}_y. A target ordering can be transformed into navigation decisions by using Equations 2 and 3. The number of possible orderings is factorially dependent on the cardinality of the target set, that is, within $O(|R|!)$. Episode-specific, optimal target sequence orderings are generated via exhaustive search, whose runtime intrinsically degrades exponentially with growing $|R|$, but is independent of dimensions $w \times h$. Experimental implications are discussed in Section VI-A and related performance data is summarised in Figure 3.

V. Implementation

A. Architectural Details

As introduced, Figure 1 illustrates and explains the end-to-end deep architecture used here. The reader can observe there that network output consists of a class membership score vector V with $|V| = |A|$ and $\sum_{v \in V} = 1$ as enforced by a final SoftMax layer. In general, the model-selected action $a \in A$ is taken to be the maximal-likelihood value of V. Note that, if enacting a results in the agent moving outside of the environment boundaries, a random valid action is performed instead and equally, loop-detection may also alter action

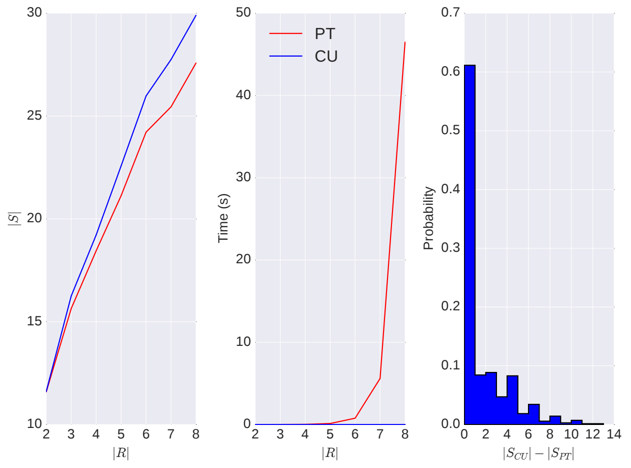


Fig. 3. **Ground-truth Synthesis – Comparison of Strategies CU vs PT.** Comparison of ground-truth synthesis methods for newly randomly generated episodes. At each value of $|R|$, 100 additional episodes were generated and solved. *(Left)*: average ground-truth solution quality $|S|$ for generated solutions against the number of targets $|R|$. *(Middle)*: average time required for solution generation against the number of targets. *(Right)*: normalised histogram for the difference in generated solution length between CU and PT for 100 instances for each $|R| \in [2, 8]$.

selection (see Figure 1 ***(h)***). Furthermore, note that the grid occupancy/visitation map M (see Figure 1 ***(e)***) intrinsically forms spatiotemporal memory for the agent with explicitly-defined architecture. This is as opposed to popular recurrent units such as gated recurrent units (GRU) [36] and long-short term memory (LSTM) [30] where the encoding and selection of temporal knowledge is implicit and hidden. Since all information of the path can contribute to learning information about P and target movements, full explicit long-term memory should intuitively perform well without such designs. Furthermore, since all training instances are modelled temporally independent/non-correlated, standard batch-based training can be employed here as opposed to resource-intensive back-propagation methods for recurrent architectures such as back-propagation through time (BPTT) [37] and real-time recurrent learning (RTRL) [38].

B. *Infinite Loop Detection*

An inherent property of model output (i.e. next-step action selection) is the potential for performing infinite agent loops. In their simplest forms, infinite loops are easily detectable via direct substring search into past actions (e.g. $\{f, b, f, b, ...\}$, $\{r, b, l, f, ...\}$). However, specific substring rules for more complex loop formations – consisting of many actions – are difficult to formulate and do not generalise well. Instead, loop detection here is primitively indicated in the event that the agent visits a particular grid coordinate α times (we empirically set $\alpha = 3$ to minimise solution length).

Upon indication of a loop, action selection control (see Figure 1 ***(h)***) is given to an alternate algorithm, which in turn navigates the agent towards the closest preferred unvisited location in the occupancy map M for current G_x, G_y. This is achieved by examining occupancy map values surrounding the agent in a $\beta_{init} = 1$ radius. Unvisited locations in that radius vote to determine the next action. If no unvisited cell exists within that radius value β, it is incremented $\beta := \beta + 1$.

C. *Training Setup*

Synthetic training data produced by the selected ground-truth synthesis method is utilised to train the dual-stream DNN model defined previously. For training end-to-end, a single instance (I, M, a) consists of inputs: agent-centric visual input I and occupancy grid map M, whilst the output is an one-hot action-class vector encoding ground-truth action a used for back-propagation. To verify pure ground-truth classification performance, $k = 10$-fold cross validation is performed over the respective experimental dataset. At each fold k_i, training is performed for 50 epochs over the partitioned training set with weights initialised randomly from a truncated normal distribution and using a batch size of 64. Costs are optimised via categorical cross-entropy loss using stochastic gradient descent (SGD) with momentum [39] and a fixed learning rate $e = 0.001$. Mean and standard deviation of cross-validated classification accuracies for each experiment (if applicable) are given in Table I: ***(h)***.

D. *Baseline Algorithms*

We implement three baseline methodologies to compare our dual-stream, unified deep learning approach against:

1) Naïve Solution (NS): As a simple, naïve strategy to provide a primitive baseline, we employ an algorithm whereby if *(1)*: an unvisited target is currently present within the visual field (verified by examining M), select an angle-appropriate action towards it using Equation 3. Otherwise, *(2)*: navigate towards a next unvisited location using the voting strategy given in Section V-B (otherwise used to break loops), and repeat *(1)* until the episode terminates.

2) Deep Recurrent Q-Network (DRQN): We also implement off-the-shelf DRQN [29] for comparison employing a 1000-time-step strong experience replay buffer. We establish this baseline to validate the hypothesis that explorational agency benefits from known solution samples and the inclusion of structured, map-based long-term memory. Visualisation of the environment was modified with a white border surrounding the map used to create the partial visual input I given to an agent. This serves to provide agent knowledge of the environment boundaries – which is an implicit property of the occupancy grid map M's architecture for our method. Additionally, target visitation is visually signified to the agent who can then no longer receive reward from that target.

3) Split Stream Network (SSN): To validate the motivation of our proposed dual-stream approach – that explorational agency benefits from information exchange between sensor and positional history inputs trained under a single architecture – we split the proposed dual-stream architecture during training. That is, resulting split stream networks 1 and 2 optimise on (I, a) and (M, a) using network architectures ***(c)*** and ***(f)*** as given in Figure 1, respectively. Network outputs are element-wise summated and normalised to yield the final action-class score vector used for action selection.

VI. Experiments

All experiments were conducted using a 3.6 GHz AMD 8-core Bulldozer CPU with a Nvidia GTX 1080Ti GPU and 32 GB of DDR3 RAM. Throughout experiments, 'random'

TABLE I

EXPERIMENTAL PERFORMANCE

			(a)	*(b)*	*(c)*	*(d)*	*(e)*	*(f)*	*(g)*	*(h)*
			Loop Detected (%)	>1 Loops Detected (%)	>100 Moves (%)	Optimal Solution (%)	≤ 10 Difference (%)	$t > t_e$ Moves (%)	Target Recovery Rate ($^d/_t$)	10-fold CV mAP (%)
	Baselines for	NS	-	-	8.13	0.34	5.96	0	0.260±0.113	-
VI-B	**Static Recovery**	DRQN [29]	-	-	58.09	0.02	2.18	45.56	0.237±0.092	-
	(random targets)	SSN (PT)	56.31	63.16	23.52	0.66	3.89	0.01	0.217±0.112	68.95±0.256
	Learning	Random (CU)	37.49	53.01	9.64	2.21	13.76	0	0.262±0.113	67.84±0.237
VI-A	**Static**	Random (PT)	32.45	46.78	7.57	2.85	15.72	0	**0.265±0.114**[1]	71.15±0.233
+	**Recovery**	Fixed Grid	0.94	100	0.76	99.06	99.06	0.63	0.337±0.054	91.47±0.696
VI-D	(agent location	Equidistant	54	82.01	19.63	21.78	45.75	0.91	0.333±0.109	83.66±0.182
	memorised)	Gaussian	24.46	30.17	0.58	39.99	78.61	0.07	0.616±0.143	75.81±0.515
VI-C	(simulation)	Random (+S)	36.73	44.94	6.88	6.53	35.72	0.164	0.289±0.109	73.97±0.175
	(agent+recovery	Random (+M)	70.16	78.9	25.55	2.08	12.67	0.59	0.261±0.115	70.17±0.276
VI-E	locations	Equidistant (+M)	47.59	74.79	7.41	30.25	60.87	0.16	**0.380±0.085**[2]	84.99±0.224
	memorised)	Gaussian (+M)	23.52	63.6	3.28	45.46	78.28	0.15	**0.622±0.141**[3]	77.30±0.429
	Learning	Random Walk	-	-	79.17	0.08	0.93	12.23	0.155±0.176	60.91±0.418
VI-F	**Dynamic**	Herd Walk	-	-	77.18	0.88	3.18	21.39	0.225±0.128	62.33±0.273
	Recovery	Herd Walk (+S)	-	-	61.35	0.12	2.90	5.67	**0.203±0.098**[4]	63.26±0.181

Experimental Performance Overview. Row sections in the table group results of our approach across three task categories, and against three baselines, respectively: VI-B: ***Baselines*** for recovery of static, randomly placed targets all outperformed by [1]our dual-stream approach; VI-A+VI-D: ***Static Recovery*** attempting to learn recovery under various spatial target distributions P; VI-C: Using ***Gazebo Simulations*** (+S) instead of abstracted sensing; VI-E: Additionally ***Memorising Recovery Locations*** instead of only agent locations, shows superior results for all [2,3]non-random target placements; VI-F: ***Dynamic Recovery*** attempting to learn recovery under target motion incl. [4]Gazebo simulated herd dynamics. Columns hold the following values: ***(a)***: % of episodes where a loop was detected; and ***(b)***: of those cases, % of episodes where another, subsequent loop was detected; ***(c)***: % of episodes where the model required more steps than simply exhaustively exploring every environment position (for fixed $w = h = 10$); ***(d)***: % of episodes where the generated solution length was equal to the optimum; and ***(e)***: % of instances where the model generated a solution less than 10 moves longer than the optimum; ***(f)***: % of instances where time expiry (model failure) occurred with $t_e = 300$; ***(g)***: $\mu \pm \sigma$ target discovery rates (no. of target recoveries per time-step); ***(h)***: 10-fold cross validated ground-truth classification mean average precision mAP$\pm\sigma$.

numbers are generated using the P-RNG Mersenne Twister algorithm [40]. We fix the number of targets to $|R| = 5$, set the environment E dimensions $w = h = 10$ and unless specified otherwise, synthesise $20,000$-episode strong training datasets per experiment containing approximately $400,000$ instances of (I, M, a) data tuples. Note also that the agent's initial position is randomly generated at the beginning of every episode $G_x^{init} \in [0, w-1], G_y^{init} \in [0, h-1]$. Targets remain static throughout episodes (solving the base case given in Section III-A, where distribution P is two-dimensional) in all experiments up until those given in Section VI-F. For testing, $20,000$ episodes are presented to the trained models to solve; each model recovery performance is then measured and reported. Results for all experiments are given in Table I and evaluated over the following subsections.

A. Episode and Ground Truth Generation

In this experiment, we compare and contrast the aforementioned ground-truth synthesis strategies (see Section IV) – that is, Closest Unvisited Target (CU) and Permutation of Targets (PT) – for training data generation utilised for subsequent model training. Figure 3 illustrates comparative results on newly-generated episodes with fixed environment dimensions $w = h = 10$ and random target distribution. As theorised for PT, the time required to solve episodes increases factorially with the number of targets, i.e. $O(|R|!)$, whilst CU requires negligible linear time (see Figure 3: *(middle)*). The tradeoff being that as $|R| \longrightarrow \infty$, the average difference in generated solution lengths diverge (see Figure 3: *(left)*) – since PT guarantees optimality. Yet, as illustrated in Figure 3: *(right)*, employing the CU strategy yields optimality in the majority of instances ($\sim 60\%$). Comparison of both training synthesis strategies against validation episodes demonstrably proves that PT prevails (see both Table I and Figure 4) in learning the conditions for optimal decisions given random target distribution. Thus, we employ the PT approach for episode solution synthesis for all other experiments and establish this result (i.e. using PT) as the base case benchmark for static, uniformly randomly distributed targets.

B. Baseline Comparison

These experiments evaluate the three implemented baseline algorithms (see Section V-D) against our proposed dual-stream architecture. For each algorithm, targets are uniformly randomly spatially distributed within E. Resulting performance statistics illustrate under-performance in all aspects in comparison to our benchmark. The employed naïve approach (NS) yields highest baseline performance and is accordingly shown for comparison in Figure 4 against benchmark results achieved by our solution. Off-the-shelf DRQN [29] demonstrably performs poorly despite access to experiences containing decisions leading to reward. This occurs arguably as a result of the agent having no global notion of current environment localisation necessary to locate possible future reward in an overall context. Splitting inputs into separately-trained neural network streams (SSN) is also shown to yield poor performance since model exposure to any single input (I or M) alone is insufficient in producing optimal reasoning (the problem becomes increasingly partially-observable). This being said, knowledge of occupancy grid M (encoding G_x, G_y, visited cells, etc.) inherently encodes more information on E than I, which is reflected in our findings of significantly higher mAP classification accuracy in the network optimising on (M, a). Additionally, performing end-to-end training on two separate networks to address the imbalance attributes significant additional computation over our proposed network architecture.

C. *Simulated Full Visual Input (+S)*

To this point, sensory environment observability has been encoded by small 3×3 sensory abstractions. To simulate a more realistic target location recovery assignment – where visual target detection is no longer trivial – we employ the task of visual outdoor farming census related to [8]; discovering cattle positions within a field using a quadrotor UAV.

The task is simulated utilising randomly-oriented 3D cattle models within the Gazebo robot simulation environment [14] (see Figure 5: *(top)*). The simulation environment is used to directly replace the agent's sensory field I with a 50×50 pixel image (see Figure 5: *(bottom)*) for a $100°$ FoV camera, whilst M remains identical in architecture and operation. The UAV is flown in discrete 2m increments within an $x-y$-plane at fixed height $z = 3.5$m.

The experimental setup remains identical to the benchmark case with agent and cattle targets uniformly randomly spatially distributed. Whilst obtained results (see Table I and Figure 4) indicate an advantage for the simulator case, these results are not directly comparable within the setup used, since environment observability improves in a three dimensional projection beyond 2D gridding given the camera FoV (see Figure 5: *(bottom)*), visibility of object shadows, etc.

Whilst efforts could have been made to strictly enforce agent visibility to a 1 grid ground cell radius (as for the grid world), this would have failed to model real scenarios well since visual artefacts (e.g. shadows) are present in real-world sensing. Note that the employed visual processing CNN architecture (AlexNet [1]) clearly demonstrates that it can recover and utilise additional visual information found in such realistic robotic scenarios (see Table I, Section VI-C).

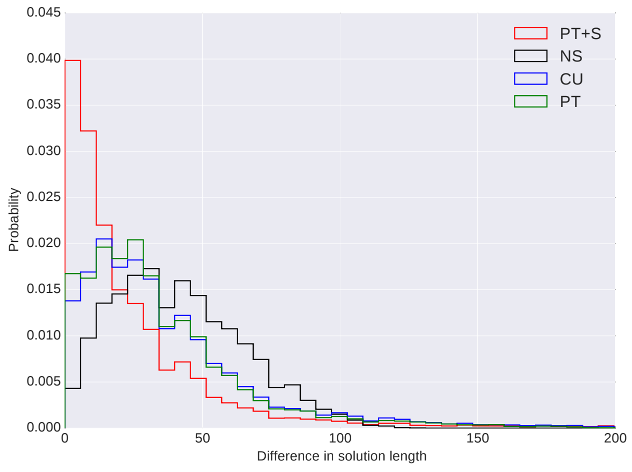


Fig. 4. **Model Solution Length Difference.** Normalised histogram of the difference in generated solution length using models trained with Closest Unvisited (CU) and Permutation of Targets (PT) ground-truth synthesis strategies against the global optimum over $20,000$ test instances. Distributions when using the naïve solution (NS) and Gazebo simulator [14] (PT+S) are also included. See Table I for further detailed performance statistics.

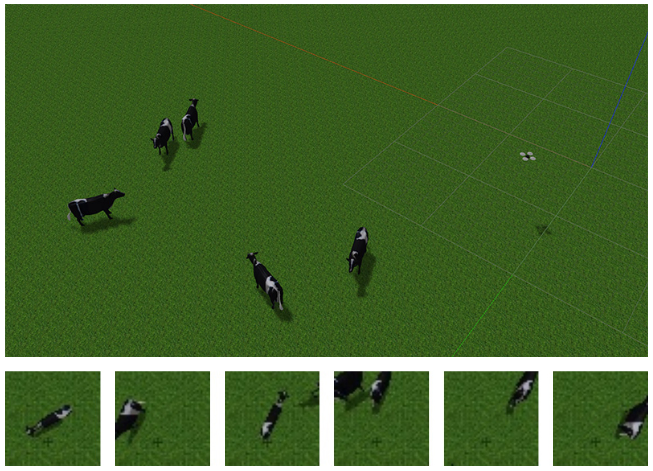


Fig. 5. **Visual Simulation Environment.** *(Top):* overview of the Gazebo-powered [14] simulation (+S) environment for a randomly generated static episode. *(Bottom):* example 50×50 pixel images from the simulated downward facing UAV camera provided as input to the agent.

D. *Learning Static Recovery under Spatial Distributions*

Experiments to this point have dealt with fully random uniform target distributions. Here we show that simpler and arguably more realistic spatial distributions can also be learnt (naturally, far more effectively) under the solution proposed here. We employ three additional distributions as follows:

1) Fixed Grid: As a baseline distribution case, targets are distributed in a fixed, static grid across episodes (spatial distribution P is known completely), whilst agent position initialisation varies as before. Since there are many orders of magnitude less possible environment initialisation configurations, $wh - |R| = (10 \times 10) - 5 = 95$ – since only the agent position is varied per-episode – one would expect highly improved model performance, which indeed turns out to be the case both in pure ground-truth classification accuracy and online model performance (see Table I, row 'Fixed Grid').

2) Equidistant Grid: Targets are distributed spatially in a grid that satisfies target-target nearest neighbour (NN) equidistance (Voronoi-like distribution) – a property that is observed in real cattle-herd distributions [41]. The position of the grid in E is varied randomly per-episode and inter-target NN spacing satisfies 2 grid-unit spacing. Since the grid is fixed in shape across episodes, positional knowledge of just two targets provides full information of remaining target positions. Ergo, the problem is then vastly less complex than the fully random case, which is reflected in model performance on this distribution (see Table I, rows 'Equidistant').

3) Gaussian: Target positions are sampled from a two-dimensional Gaussian with constant parameters $\mu_x = 3$, $\mu_y = 5$ and $\sigma_x = \sigma_y = 1$ at the beginning of each episode. Values are sampled from the distribution as required. Results demonstrably show strong model performance in efficiently discovering target positions – relatable to an ability to effectively learn the distributional parameters of the underlying 2D Gaussian P (see Table I, rows 'Gaussian').

E. *Memorising Recovery Locations (+M)*

In these experiments, the visitation map M is modified such that target locations are marked upon agent visitation.

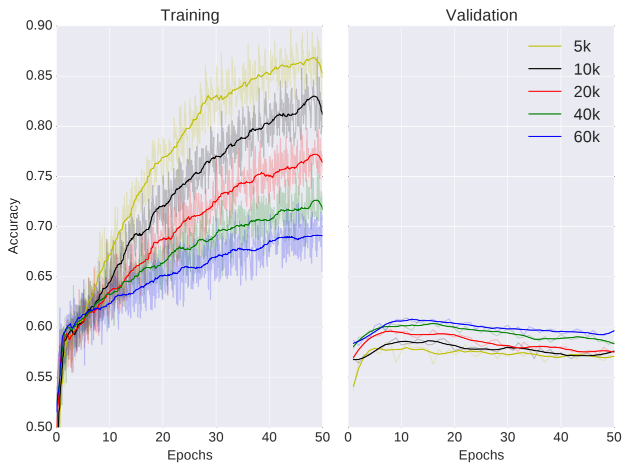


Fig. 6. **Training Evolution of Classification Accuracy for Dynamic Targets.** Classification accuracy for training and validation sets over 50 epochs over optimal generated target motion datasets (implementing random walk) with various numbers of thousands of episodes used for training.

This small implementation detail leads to improved performance upon all tested, non-random target distributions, that is 'Equidistant' and 'Gaussian'. These improvements occur intuitively; previously discovered target positions directly encode information about a non-random P. This is demonstrably not the case for fully random distributions – knowledge of randomly positioned target r_i reveals no information about the position of r_j, and yields no increase (i.e. slight decrease) in performance (see Table I, compare rows 4, 5 and 10).

F. Learning Dynamic Multi-Target Recovery

Thus far, targets have remained static throughout episodes. Here we extend our implementation to include individual target motion in two forms. Formalisation of this, now dynamic recovery problem, is given in Section III-B. Note also that infinite loop detection – applicable to the static recovery task – is disabled here.

1) Random Walk: Targets implement random walk via random action selection $rand(A)$ with a fixed velocity of $\frac{1}{3}u/t$ (one grid unit every 3 time-steps). To determine globally-optimal solutions towards training data generation, $\varepsilon = t_e = 300$ random actions are pre-determined for each target such that Equation 1 – mandating single grid cell target occupancy – remains satisfied for every time-step $t_i \in [0, \varepsilon]$. The yielded matrix containing target coordinates over time is then exhaustively searched for each possible combination of future target visitation orderings towards finding the shortest solution using a full 'lookahead' extension of the PT solver.

2) Herd-like Motion: Looking towards more realistic scenarios closer to a robotic UAV application in 'smart farming', cattle herd-inspired distribution initialisation and motion is applied. Targets are initially distributed in an equidistant grid (as for Section VI-D.2) with random position in E. Herd-inspired motion is implemented with an overall group direction $d_i \in A$ randomly selected at the beginning of each episode. Direction d_i is applied to each target at velocity $\frac{1}{3}u/t$ with a 10% per-individual likelihood that they instead perform a random action $rand(A_{ext})$ excluding in-axis directions for d_i (e.g. if $d_i = r$, $A_{ext} = \{f, b, n\}$).

This motion behaviour approximation is supported by literature observing collective dynamics for grazing cattle herds [42]. Finally, upon reaching the boundaries of E, a new group direction d_{i+1} is randomly selected where $d_i \neq d_{i+1}$. Identically to random walk, $\varepsilon = t_e = 300$ individual motion actions are pre-determined and solved for optimally via the full 'lookahead' extension of the PT solver.

Following network training on the random walk experiment, the so far employed $20,000$-episode dataset cardinality was found to be insufficient for optimal performance leading to significant model overfitting (see Figure 6). This observation illustrates the significant increase in problem complexity introduced by individual target motion. Increasingly larger datasets improving validation accuracy were thus synthesised at the cost of overall computation time. However, reasonable synthesis, training and evaluation times were quickly exceeded – we opted to empirically limit dynamic experiment datasets to $60,000$ episodes as a reasonable accuracy versus time tradeoff. Findings suggest that there is more validation accuracy to be gained by providing even larger sets of episode datasets (see Figure 6, right).

Quantitative ground-truth classification and online model performance statistics given in Table I demonstrably indicate the capability of the proposed unified network to generalise well to the case of individual target dynamics. In particular, herd-inspired motion generally improves search and recovery capabilities (see Table I, compare rows 13 vs 14 and 15).

Finally, experimental outcomes to this point culminate in our last, highlighted experiment: 'Herd Walk (+S)', combining target dynamics enacting herd-like motion *and* visual sensory input rendered via the cattle-census simulation environment (see Section VI-C and Figure 5). As for the static case, we observe increased environment observability and reduced partiality in view compared to the 2D case. Yet, the experiment clearly validates the employed dual-stream, single network architecture in yielding competitive explorational decisions whilst processing higher resolution, complex visual imagery in unity with spatial-temporal navigational memory.

VII. Conclusion

This work demonstrates that recovery tasks can be effectively modelled by combining visually-motivated sensing and map-based positional histories under a single deep classification architecture, comprising the combined optimisation of both information streams in unity. We have shown that this approach outperforms the various tested baselines including split stream optimisation. Our proposed architectural choices demonstrably generalise well to a wide range of scenarios, target distributions and dynamism given training data comprised of good or optimal decision strategies.

Future work will look towards deployment of models developed to onboard, real-world UAV visual navigation complementing individual target identification tasks. Within this, the visitation map M will be augmented probabilistically to account for outdoor UAV localisation variance introduced by using GPS. Additionally, environment-observability benefits introduced by varying UAV altitude will be investigated.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[2] C. S. Helvig, G. Robins, and A. Zelikovsky, “The moving-target traveling salesman problem,” *Journal of Algorithms*, vol. 49, no. 1, pp. 153–174, 2003.

[3] M. Held and R. M. Karp, “A dynamic programming approach to sequencing problems,” *Journal of the Society for Industrial and Applied Mathematics*, vol. 10, no. 1, pp. 196–210, 1962.

[4] M. Dorigo and L. M. Gambardella, “Ant colony system: a cooperative learning approach to the traveling salesman problem,” *IEEE Transactions on evolutionary computation*, vol. 1, no. 1, pp. 53–66, 1997.

[5] L. Lin and M. A. Goodrich, “Uav intelligent path planning for wilderness search and rescue,” in *Intelligent robots and systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. IEEE, 2009, pp. 709–714.

[6] S. Waharte and N. Trigoni, “Supporting search and rescue operations with uavs,” in *Emerging Security Technologies (EST), 2010 International Conference on*. IEEE, 2010, pp. 142–147.

[7] K. Ryu, “Autonomous robotic strategies for urban search and rescue,” Ph.D. dissertation, Virginia Polytechnic Institute and State University, 2012.

[8] W. Andrew, C. Greatwood, and T. Burghardt, “Visual localisation and individual identification of holstein friesian cattle via deep learning,” in *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, Oct 2017, pp. 2850–2859.

[9] F. Bonin-Font, A. Ortiz, and G. Oliver, “Visual navigation for mobile robots: A survey,” *Journal of intelligent and robotic systems*, vol. 53, no. 3, p. 263, 2008.

[10] G. N. DeSouza and A. C. Kak, “Vision for mobile robot navigation: A survey,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 24, no. 2, pp. 237–267, 2002.

[11] J. Zhang, L. Tai, J. Boedecker, W. Burgard, and M. Liu, “Neural slam,” *arXiv preprint arXiv:1706.09520*, 2017.

[12] R. S. Sutton, “Integrated architectures for learning, planning, and reacting based on approximating dynamic programming,” in *Proceedings of the seventh international conference on machine learning*, 1990, pp. 216–224.

[13] A. A. Melnikov, A. Makmal, and H. J. Briegel, “Projective simulation applied to the grid-world and the mountain-car problem,” *arXiv preprint arXiv:1405.5459*, 2014.

[14] N. Koenig and A. Howard, “Design and use paradigms for gazebo, an open-source multi-robot simulator,” in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3. IEEE, pp. 2149–2154.

[15] J. Santos-Victor, G. Sandini, F. Curotto, and S. Garibaldi, “Divergent stereo for robot navigation: Learning from bees,” in *Computer Vision and Pattern Recognition, 1993. Proceedings CVPR’93., 1993 IEEE Computer Society Conference on*. IEEE, 1993, pp. 434–439.

[16] Y. Matsumoto, M. Inaba, and H. Inoue, “Visual navigation using view-sequenced route representation,” in *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, vol. 1. IEEE, 1996, pp. 83–88.

[17] S. D. Jones, C. Andresen, and J. L. Crowley, “Appearance based process for visual navigation,” in *Intelligent Robots and Systems, 1997. IROS’97., Proceedings of the 1997 IEEE/RSJ International Conference on*, vol. 2. IEEE, 1997, pp. 551–557.

[18] N. Pears and B. Liang, “Ground plane segmentation for mobile robot visual navigation,” in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 3. IEEE, 2001, pp. 1513–1518.

[19] P. Saeedi, P. D. Lawrence, and D. G. Lowe, “Vision-based 3-d trajectory tracking for unknown environments,” *IEEE transactions on robotics*, vol. 22, no. 1, pp. 119–136, 2006.

[20] A. Giusti, J. Guzzi, D. C. Cireşan, F.-L. He, J. P. Rodríguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. Di Caro, *et al.*, “A machine learning approach to visual perception of forest trails for mobile robots,” *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 661–667, 2016.

[21] L. Ran, Y. Zhang, Q. Zhang, and T. Yang, “Convolutional neural network-based robot navigation using uncalibrated spherical images,” *Sensors*, vol. 17, no. 6, p. 1341, 2017.

[22] J. Borenstein and Y. Koren, “Real-time obstacle avoidance for fast mobile robots in cluttered environments,” in *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*. IEEE, 1990, pp. 572–577.

[23] ——, “The vector field histogram-fast obstacle avoidance for mobile robots,” *IEEE transactions on robotics and automation*, vol. 7, no. 3, pp. 278–288, 1991.

[24] G. Oriolo, M. Vendittelli, and G. Ulivi, “On-line map building and navigation for autonomous mobile robots,” in *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, vol. 3. IEEE, 1995, pp. 2900–2906.

[25] H. P. Moravec, “The stanford cart and the cmu rover,” *Proceedings of the IEEE*, vol. 71, no. 7, pp. 872–884, 1983.

[26] A. Kosaka and A. C. Kak, “Fast vision-guided mobile robot navigation using model-based reasoning and prediction of uncertainties,” *CVGIP: Image understanding*, vol. 56, no. 3, pp. 271–329, 1992.

[27] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing atari with deep reinforcement learning,” *arXiv preprint arXiv:1312.5602*, 2013.

[28] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[29] M. Hausknecht and P. Stone, “Deep recurrent q-learning for partially observable mdps,” *CoRR, abs/1507.06527*, 2015.

[30] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[31] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, *et al.*, “Learning to navigate in complex environments,” *arXiv preprint arXiv:1611.03673*, 2016.

[32] J. Zhang, J. T. Springenberg, J. Boedecker, and W. Burgard, “Deep reinforcement learning with successor features for navigation across similar environments,” *arXiv preprint arXiv:1612.05533*, 2016.

[33] L. Tai, G. Paolo, and M. Liu, “Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation,” *arXiv preprint arXiv:1703.00420*, 2017.

[34] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik, “Cognitive mapping and planning for visual navigation,” *arXiv preprint arXiv:1702.03920*, 2017.

[35] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, “Target-driven visual navigation in indoor scenes using deep reinforcement learning,” in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 3357–3364.

[36] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” *arXiv preprint arXiv:1406.1078*, 2014.

[37] P. J. Werbos, “Generalization of backpropagation with application to a recurrent gas market model,” *Neural networks*, vol. 1, no. 4, pp. 339–356, 1988.

[38] A. Robinson and F. Fallside, *The utility driven dynamic error propagation network*. University of Cambridge Department of Engineering, 1987.

[39] N. Qian, “On the momentum term in gradient descent learning algorithms,” *Neural networks*, vol. 12, no. 1, pp. 145–151, 1999.

[40] M. Matsumoto and T. Nishimura, “Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator,” *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 8, no. 1, pp. 3–30, 1998.

[41] L. A. Syme, G. Syme, T. Waite, and A. Pearson, “Spatial distribution and social status in a small herd of dairy cows,” *Animal Behaviour*, vol. 23, pp. 609–614, 1975.

[42] K. Zhao and R. Jurdak, “Understanding the spatiotemporal pattern of grazing cattle movement,” *Scientific reports*, vol. 6, p. 31967, 2016.