CSCI 5512 AI Project Final Report
Autonomous Wildfire Surveillance using Reinforcement Learning

Kerry Sun* and Peng Mun Siew†

May 13, 2018

Abstract

A fixed-wing aircraft was modeled to learn how to maximize forest coverage in a simulated environment. The aircraft (agent) takes one of the two possible actions at each time step to navigate in the pre-defined wildfire environment to maximize the reward, which is sum of the observed temperature gradient region. Simulation results show the aircraft was able to track wildfire expansion by maximizing its reward based solely on the immediate observations of the environment.

1 Introduction

Wildfires consumed 10.1 millions acres of land in 2015 and cost an estimated $6 billion of damage from 1995 to 2004 [1]. Obtaining accurate information about the state of a wildfire during the course of its evolution is crucial in deciding where to use fire suppressants and where to remove fuel. Using Unmanned aerial vehicles (UAVs) can increase safety and bring economic benefits for wild fire monitoring. For example, there is no need of a real pilot to fly in dangerous condition if the UAV can be used in real time. Furthermore, multiple UAVs can operate together without human control, reducing costs and increase efficiency. Therefore, having autonomous aircrafts monitor wildfire is an important topic to study.

In this project, we study how to navigate a fixed-wing aircraft autonomously to maximize forest fire coverage using deep reinforcement learning (DRL) approach. Specifically, given sensor information (aircraft states and partial wildfire image data), a deep neural network will be trained to maximize wildfire surveillance for a single aircraft. The idea of this project was originated from an recent AIAA conference paper [2]. The general structure of the methods are similar between [2] and our project, however, an improve wildfire model is used to simulate the environment for this project. As for the agent, a simple kinematic aircraft model (3 states) is used to capture the essence of dynamics of the agent. The Q learning was used to obtain approximately optimal strategy.

2 Methods and Algorithms

This section gives the details of the wildfire model (environment generation), the aircraft model (agent) and its possible actions, observation based reward, DRL setting and the Network Architecture. The propagated environment was generated in the Matlab setting prior to network

*Department of Aerospace Engineering and Mechanics, sunx0486@umn.edu
†Department of Aerospace Engineering and Mechanics, siewx007@umn.edu

training, and this environment data was then fed into the network. The network was built using Tensorflow [3]. The results of the DRL is visualized by TensorBoard.

2.1 Wildfire Model

A wildfire model is needed to create a simulation environment to train and evaluate the agent. We use the wildfire model in [4]. The model was derived from the conservation of energy, balance of fuel supply and the fuel reaction rate:

$$\frac{dT}{dt} = \nabla \cdot (k\nabla T) - \overrightarrow{v} \cdot \nabla T + A\big(Se^{-B/(T-T_a)} - C(T - Ta)\big) \tag{1}$$

$$\frac{dS}{dt} = -C_S S e^{-B/(T-T_a)}, \quad T > T_a \tag{2}$$

with the initial values

$$S(t_{init}) = 1 \quad and \quad T(t_{init}) = T_{init} \tag{3}$$

where $T(K)$ is the temperature of the fire layer. $S \in [0, 1]$ is the fuel supply mass fraction (the relative amount of fuel remaining), $k(m^2 s^{-1})$ is the thermal diffusivity, $A(Ks^{-1})$ is the temperature rise per second at the maximum burning rate with full initial fuel load and no cooling present, $B(K)$ is the proportionality coefficient in the modified Arrhenius law, $C(K^{-1})$ is the scaled coefficient of the heat transfer to the environment, $C_s(s^{-1})$ is the fuel relative disappearance rate, $T_a(K)$ is the ambient temperature, and finally $\overrightarrow{v}(ms^{-1})$ is wind speed.

The following parameters listed in Table 1 are used to for this model.

Table 1: A Summary of the Parameters Used

Parameter	**Symbol**	**A Typical Realistic Value**
Thermal diffusivity constant	k	0.28 $m^2 s^{-1}$
Wind direction	θ	0 deg
Wind magnitude	$\overrightarrow{v}$	0 m/s
Empirical burn rate	A	187.93 Ks^{-1}
Empirical proportionality coefficient	B	558.49 K
Empirical convection cooling coefficient	C	$4.8372e-5$ K^{-1}
Fuel relative disappearance rate	C_s	50 s^{-1}
Ignition temperature	T_a	300 K

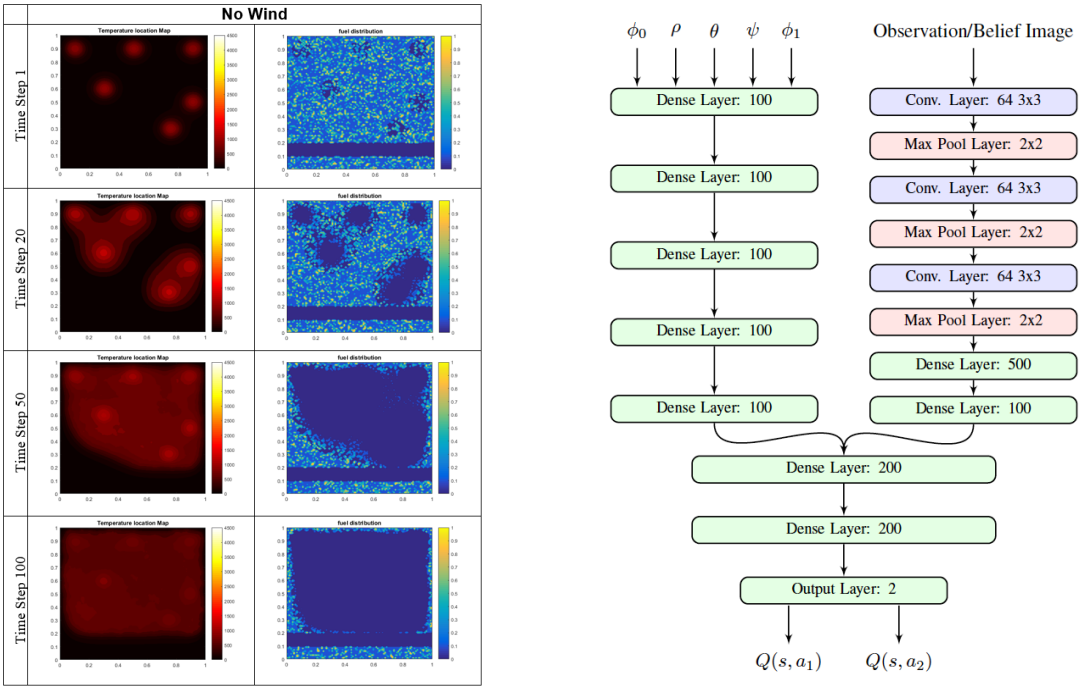
For simulation, we construct a spatial discretization using Laplacian matrix in Matlab. The grid size generated for the simulation is **300 by 300**. The backward Euler and Godunov splitting were used for the time stepping. Also, the wildfire model is initialized with randomized fuel distribution to make it more realistic. Fuel map and fire temperature maps are recorded at each time step. Those maps will be used as observations for the reinforcement learning network. Fig. 1(a) is an example of wildfire propagation over 100 iterations for no wind conditions from our simulation. In the later Simulation section, a different initial fire location was used to make the problem trivial to understand and analyze.

2.2 Partially Observable Markov Decision Processes

The problem is fundamentally a Markov decision process. Given an agent in state $s \in S$, taking action $a \in A$, the agent will transit to new state $s' \in S$ and receive reward $r \in R$. The agent is essentially making decisions sequentially. Also, even though we generated the full state of the wildfire, the agent (aircraft) is only able to observe partially to make decision. This is

realistic since the aircraft cannot see the full scope of the wildfire in real life. Hence, it makes the problem a partially observable Markov decision process (POMDP).

Fig. 2 shows an example of an aircraft's wildfire partial observation on the right while the full environment is shown on the left [2].



(a) Wildfire propagation over time

(b) Network Architecture [2]

Figure 1: Wildfire Simulation and Network Architecture Example.

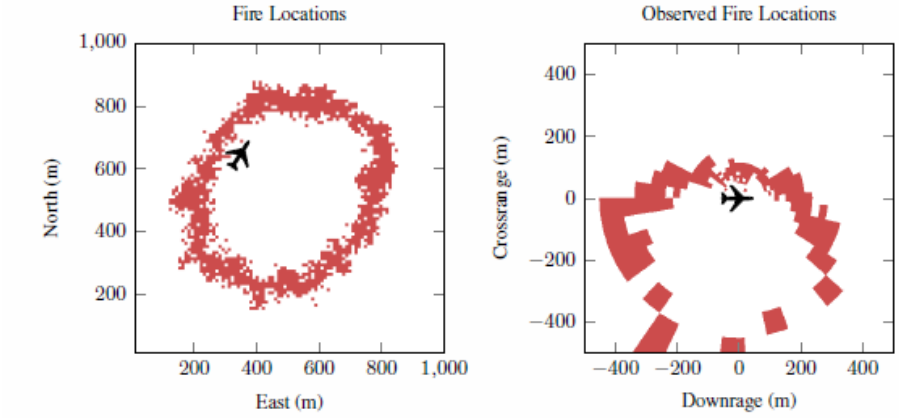


Figure 2: Example of an aircraft's wildfire observation [2]

In our simulation setting, the partial observed space is **40 by 40** at any given moment. The aircraft was assumed to have a 360° view of its surrounding.

2.3 Aircraft Dynamic Modeling

There are only 3 states of the aircraft being modeled for the simulation: position x, y, and bank angle ϕ. Euler method is used for time propagation, and the aircraft was only able to maneuver in the restricted 300 by 300 grid setting. Once the aircraft can no longer partially observed the 40 by 40 surrounding, the episode of the simulation ends.

2.3.1 Dynamics

The dynamics are modeled using Dubin's kinematic model shown in Eq. 4, assuming steady level flight at constant speed [5]. The velocity is set at 15 m/s. Here a is the possible actions assigned to the aircraft.

$$\begin{aligned} \dot{x} &= vcos\phi \\ \dot{y} &= vsin\phi \\ \dot{\phi} &= a \end{aligned} \tag{4}$$

2.3.2 Action

At each time step, the aircraft has two possible actions: $a \in (+\dot{\phi}, -\dot{\phi}) = (+5°, -5°)$, that is, increase or decrease bank angle rate by 5°. The sampling frequency for the action is 10 Hz (0.1 sec), which ensures smooth turn for the aircraft.

2.4 Reward

As mentioned earlier, the aircraft chooses actions to maximize the accumulation of reward. In our case, the reward is based on the temperature difference ΔT between the current state s and previous state $s-1$. We sum the observed temperature difference as the total reward at any given moment in the environment. This reward physically represents the heat gradient of the wildfire. The stronger gradient is, the faster the fire is going to spread along that direction. Mathematically, the reward is calculated as the following:

$$r(s) = \sum_{s \in B_t} \Delta T(s(t), s(t-1)) = \sum_{s \in B_t} [T(s(t)) - T(s(t-1))] \tag{5}$$

where B_t is the the observed region by the aircraft at any given moment. Fig. 3 shows an example of the reward map, aircraft and its current observed reward region (the green square). The agent receives the sum of temperature difference, and moves toward the greater temperature gradient of the reward map.

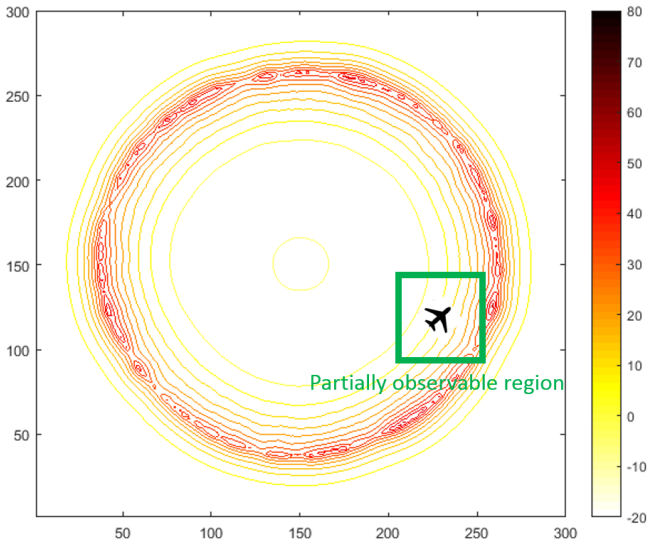


Figure 3: Example of an aircraft's partially observed environment(in green)

2.5 Solution Method

Since the problem we have is a POMDP, the solution is to have a policy that maps observation to actions. Because the space of possible observations is very large, an approximate method must be used. The deep reinforcement learning is used to obtain approximately optimal control strategy. DRL involves training a neural network to represent a policy mapping states to actions that maximizes the expected accumulation of reward. The network is trained offline through simulation, and the trained network can be used on-board to guide the aircraft. DRL generally maps states to actions, but here it makes decision using partial observations instead of full states. Therefore, the states used by DRL is consists of the continuous aircraft state variables and the observation.

2.5.1 Deep Reinforcement Learning

In deep reinforcement learning, each state-action pair has some value, $Q(s, a)$, which represents the expected value of taking action a from state s. The optimal Q follows the Bellman equation:

$$Q(s_t, a_t) = r(s_t) + \gamma \sum_{s_{t+1} \in S} p(s_{t+1}|s_t, a_t) \max_{a_{t+1} \in A} Q(s_{t+1}, a_{t+1}) \tag{6}$$

where γ is the discount factor, $r(s_t)$ is the reward received for being in state s at time t, and probably $p(s_{t+1}|s_t, a_t)$ is the probability of transitioning to state s_{t+1} by taking action a_t from state s_t. $\gamma = 0.99$ is used here to help the algorithm converge. The policy can be computed as:

$$\pi(s_t) = \underset{a_t \in A}{\operatorname{argmax}}\, Q(s_t, a_t) \tag{7}$$

However, since we do not have the explicit knowledge of the transition probabilities $p(s_{t+1}|s_t, a_t)$ and state space is large and intractable, Q-learning algorithm is used for this study. Q-learning learns the Q values iteratively through simulation, and a function approximation (deep neural networks) is used to represent the complex domains. Therefore, we use the following algorithm to update Q:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma \max_a Q(s_{t+1}, a_t) - Q(s_t, a_t)] \tag{8}$$

where $r_t + \gamma \max_a Q(s_{t+1}, a_t) - Q(s_t, a_t)$ is the Bellman error E. The Bellman error can be minimized through mean squared error loss and gradient descent methods. The Q Learning network uses the Bellman equation to estimate the Q-value, where it is assumed that the optimal action will be taken in the next state. Note that when using a function approximation, updating the parameters to change the Q values of one state will also update the Q values of very similar states, which can lead to faster convergence. However, since a given state s and next state s' will be similar, changing $Q(s, a)$ will also change $Q(s', a')$, which will increase the error, and this process can be unstable.

One technique implemented here to mitigate such instabilities is experience replay [6]. First, the state, action, reward, next state tuples can be stored in a large repository and random samples can be drawn to train the network, which helps to break up trajectories where states are very similar. Before network training even begins, this repository is supplied with a large number of samples to ensure that the network does not over-fit to the generated trajectories.

3 Experimental Setup

3.0.1 Network Architecture

We follow the deep Q-network architecture proposed by Julian and Mykel as shown in Fig. 1(b), use it as a benchmark architecture to train our network. Since the input include the observation (images) and the aircraft states (continuous inputs), the network begins with two separate networks: a convolutional network for the image, and fully connected layers for the continuous state variables. Convolutional neural networks exploit similarities between local regions of pixels, allowing them to be efficient and effective with large numer of inputs. In addition, the convolutional neural network contains max pooling layers, and these layers take the maximum of each non-overlapping 3×3 region in the previous layer, which significantly reduces the amount of weights needed for future layers. The convolutional and max pooling layers are then flattened and connected to two more fully connected layers, ending with 100 units in the last layer.

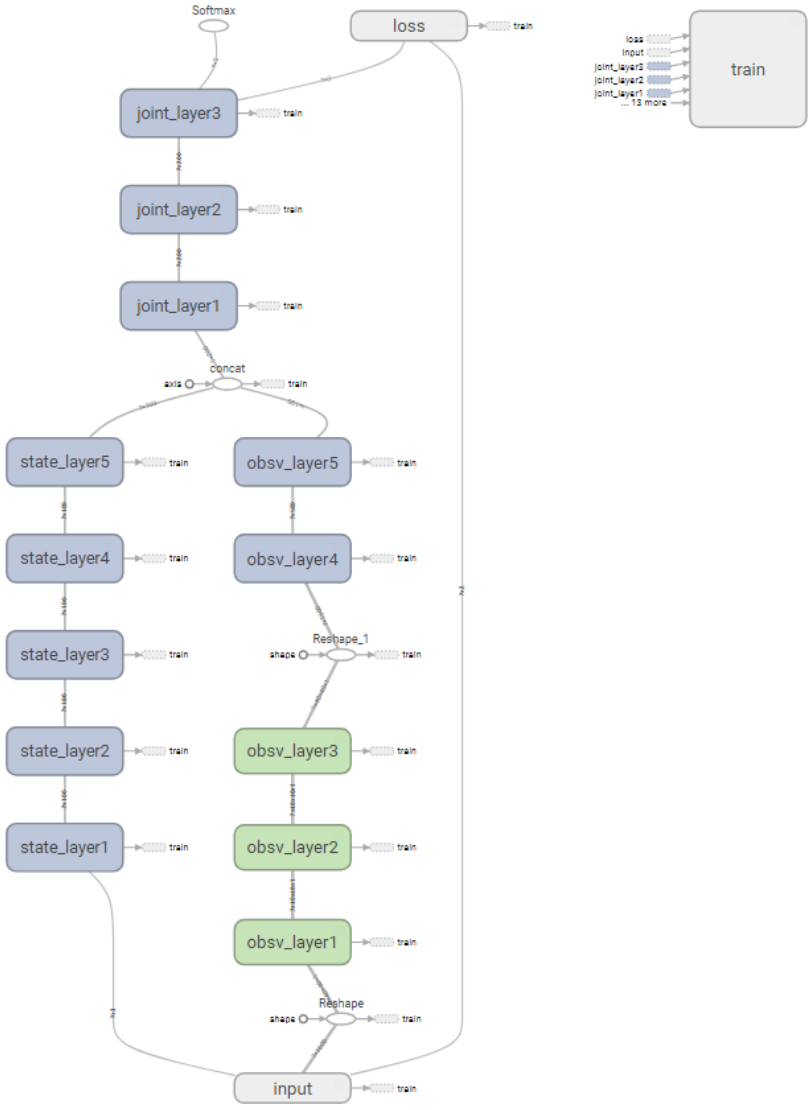


Figure 4: Network graph output from TensorBoard.

The fully connected lyaers for the continuous inputs consist of 5 hidden layers of 100 units each. At this point, both sides of the network have 100 units each. These two layers are then

concatenated into a layer of 200 hidden units. The merged layers are followed by two more fully connected hidden layers of 200 units each, which then end with an output layer with two inputs. The trained network graph from TensorBoard is shown in Fig. 4.

Each hidden layer uses rectified linear unit activations, which allow positive inputs to pass through unchanged while negative inputs are output as zero. The network parameters are updated using AdamMax optimization [7], which is an adaptive stochastic gradient descent based on the infinity norm. It estimates low order moments to change learning parameters and minimize loss quickly. The network is trained on mini-batches (20 batches) of 100 samples drawn from the experience replay, allowing the parameters to be updated to minimize a more global loss rather than the loss of a single sample. The network is updated every 1000 training iterations, which allows the loss to converge before moving the target. Only 300 episode total is allowed and is determined to be sufficient since the training environment is relatively small.

Note the input such as the environment (observation) and reward map are generated in Matlab with the wildfire model described in the earlier section. The reinforcement learning network is constructed using TensorFlow [3], which is an open-source software library for dataflow programming across a range of tasks.

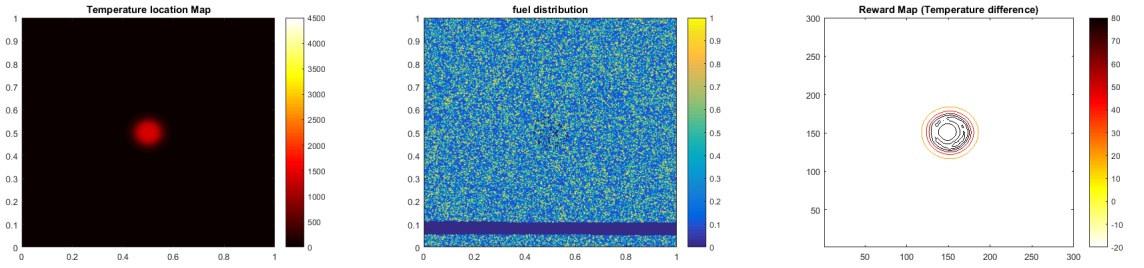
4 Results

We summarize a variety of simulation results. We begin by showing the environment and reward map outputs that are generated by Matlab. Then we show the outputs of the deep-neural network such as the optimal trajectory the aircraft took, the accumulative reward over episode some observations space diagram, and some weights outputs trained by Tensorflow.

4.1 Environment and Reward Map

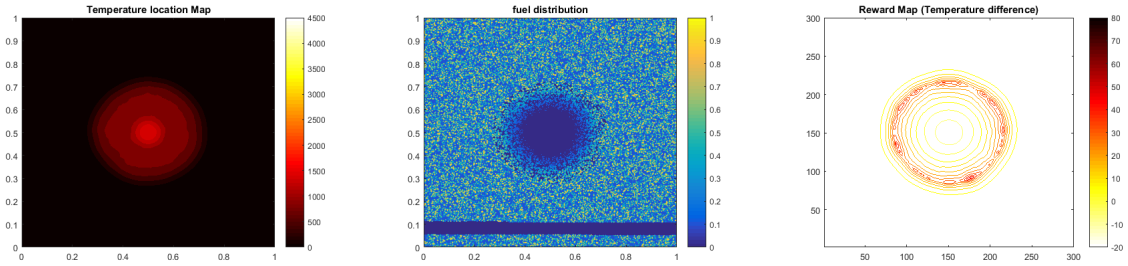
Fig. 5, 6, and 7 show the snapshots of the environment (observation) and reward map over 100 secs. Specifically, the fire, fuel distribution and reward map based on temperature difference are plotted here at t = 0, 50, 100 secs. The fire was chosen at the center of grid for simplicity. Notice there is an empty strip in the initial fuel distribution. This means there was no fuel in that region; it can represent a wall or a barren hill in real life. Those simulated observations and reward are then fed into the deep neural network as the observation and reward maps.

We can see as fire spreads radially outward, the inner fire starts to die down as expected. This is because there was no longer any fuel (i.e. trees) left to be burned. The reward map represents where the fire gradient is going. The fire gradient is also spread radially as expected. In an ideal setting, the aircraft should be able to track the greatest gradient of temperature and guide itself towards that region.



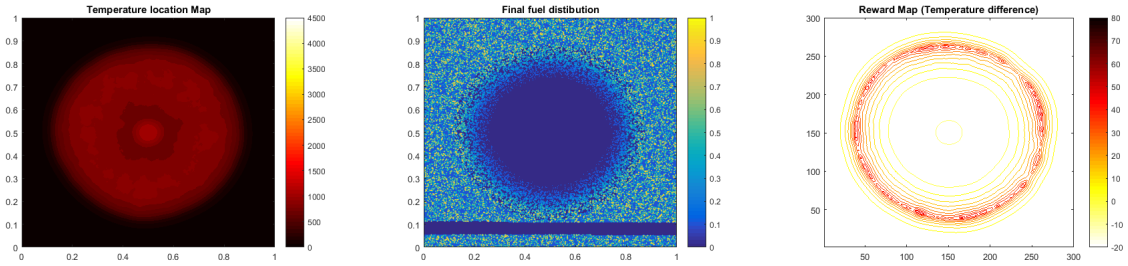
(a) Initial fire distribution (t = 0 s) (b) Initial fuel distribution (t = 0 s) (c) Initial Reward Map (t = 0 s)

Figure 5: Simulated Wildfire Environment and Reward Map at t = 0 sec.



(a) Fire distribution at half time (b) Fuel distribution at half time (c) Reward Map at half time

Figure 6: Simulated Wildfire Environment and Reward Map at half time.



(a) Final fire location (b) Final fuel distribution (c) Final Reward Map at half time

Figure 7: Simulated Wildfire Environment and Reward Map at final time.

4.2 DRL Output

As mentioned earlier, we use the reward function of the observation based network to guide the aircraft. The average accumulated reward over the training is shown in Fig. 8. We can see the reward only improved a little over 300 episodes, and there is some variance in the reward trend. The exact reason is still to be determined, but we suspect more tunning is needed to make improve accumulative reward. Note the reward was actually normalized against the size of grid (normalization factor 300^2) to make it reasonable.

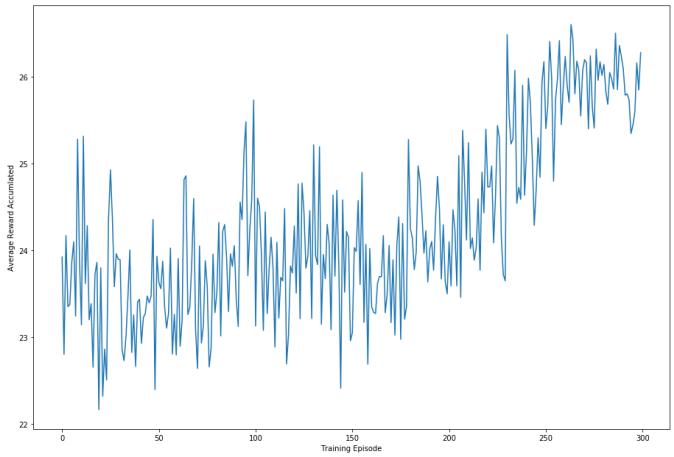
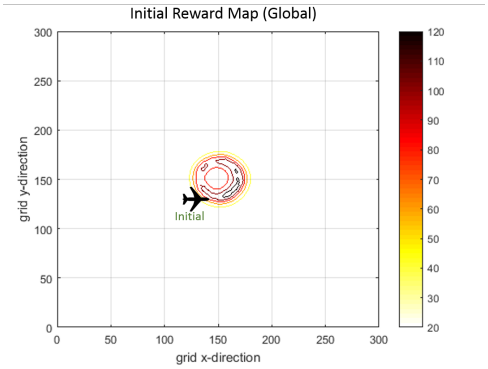


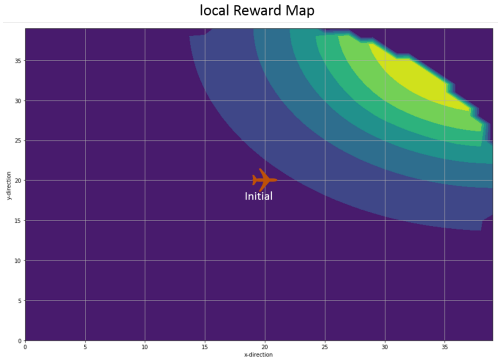
Figure 8: Average Reward Accumulated over iterations

After fully training the networks, the decisions of neural networks can be understood qualitatively through simulation and inspection of the flight trajectory. The path taken by the aircraft using observation neural network controller is plotted with the global reward map and

local reward map respectively, as shown in Fig. 9, 10 and 11. Fig. 9, 10 and 11 show the progression of the aircraft trajectory over 100 secs. The aircraft begins at a location that is close to the origin of the fire shown in Fig. 9(a). We can also see the observed area (40 by 40) from the local reward map in Fig. 9(b). Again, the aircraft was only able to observe partial of the whole reward to make decision.



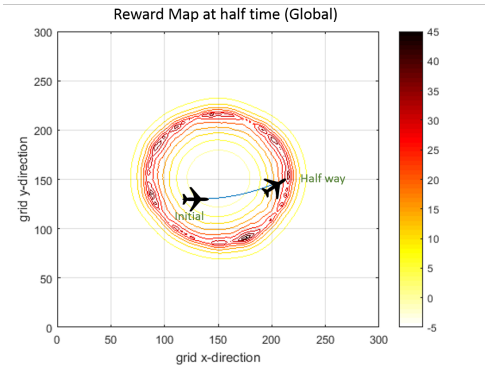
(a) Optimal Trajectory the Aircraft taken projected on the global inital reward map (t = 0 secs)



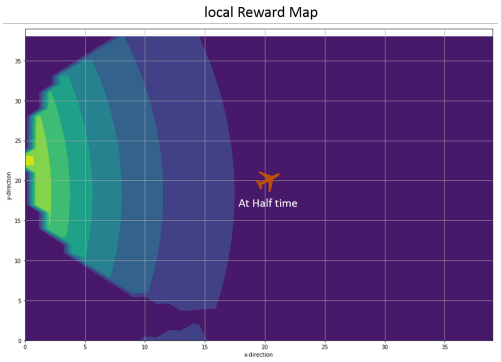
(b) contour of observed reward by the aircraft at t = 0 secs.

Figure 9: Partial Observed Reward by the aircraft from both global and local frame

In Fig. 10, we can see the aircraft was able to adjust its heading and move towards the greatest gradient of temperature, that is, the maximum reward region it can achieve. Since the aircraft is physically constraint by the given dynamic model, it only moves towards to right side of reward circle in Fig. 10(a), which is perfectly reasonable. However, Fig. 10(b) is a bit hard to explain; it seems the greatest gradient of temperature is actually on the left side of the aircraft. The aircraft perhaps cannot instantly change its direction, therefore moves away from it.



(a) Optimal Trajectory the Aircraft taken projected on the global reward map (t = 50 secs)

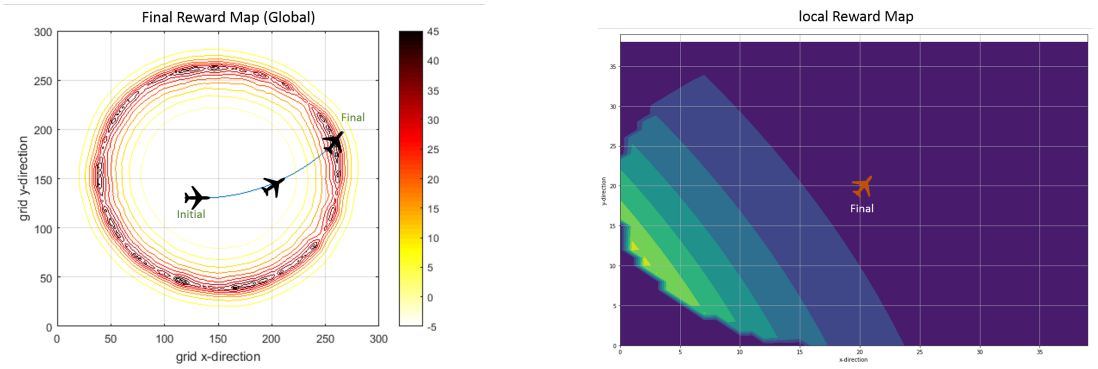


(b) contour of observed reward by the aircraft at t = 50 secs.

Figure 10: Partial Observed Reward by the aircraft from both global and local frame

In Fig. 11, the trend is similar; the aircraft continues to track the location of the biggest reward boundary, but it moves towards the opposite of greatest gradient. Unfortunately, the environment was not simulated longer enough to see what happens next. I suspect if the fire keep burning at final circular location for a while, the aircraft would turn around and make its way back and circle above where the fire is burning.

Finally, we show some examples of weights distribution from TensorBoard in Fig. 12 and 13. Since we are using the rectified linear unit activation functions, most of the weight should



(a) Optimal Trajectory the Aircraft taken projected on the global reward map (t = 100 secs)

(b) contour of observed reward by the aircraft at t = 100 secs.

Figure 11: Partial Observed Reward by the aircraft from both global and local frame

start from the positive side. This is indeed case shown in Fig. 12. In Fig. 13, we can see the weights stay bounded or at least did not blow up. One thing worth mentioning here is the Loss of the neural network was increase first, then decreasing. Again, there might be caused the poor tunning be used in the current setting.

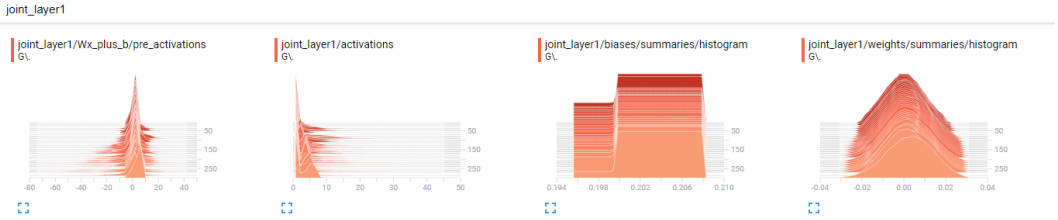


Figure 12: Example of TensorBoard: Histogram

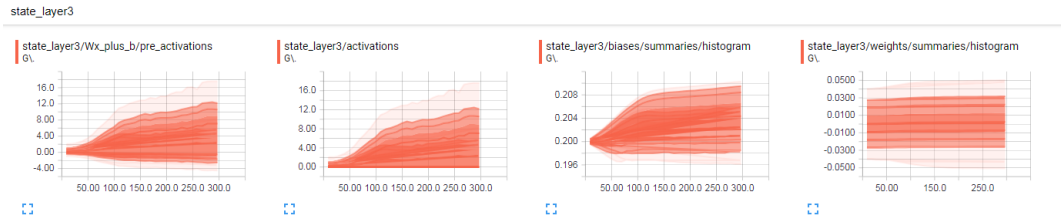


Figure 13: Example of TensorBoard: Distribution

5 Conclusion

In this project, we successfully demonstrate how to train an agent to act by maximizing its assign reward function. Specifically, we showed how an aircraft can maneuver autonomously with partial observed environment to track along the fire front. A deep neural network is developed and trained using Q-learning in a TensorFlow setting. The trained network could be incorporated into the on-board guidance system of real aircraft to generate intelligent flight trajectories for monitoring wildfires. Future work include impoving the training procedure and tuning different parameter to achieve a betteer performance. A more realistic aircraft (including aerodynamic model) should be considered for this study. Also, different wildfire situation should be tried to see how the aircraft responds. This work is inspired by [2], we would like to thank

them for showing us great work.

References

[1] "Fereral firefighting costs (supression only)," National Interagency Fire Center, 2017.

[2] K. D. Julian and M. J. Kochenderfer, "Autonomous distributed wildfire surveillance using deep reinforcement learning," in *AIAA SciTech Forum*. American Institute of Aeronautics and Astronautics, Jan. 2018. [Online]. Available: https://doi.org/10.2514/6.2018-1589

[3] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[4] J. Mande, L. S. .Bennethum, J. D. Beezley, J. L. Coen, C. C. Douglas, M. Kim, and A. Vodacek, "A wildland fire model with data assimilation," *Mathematics and Computers in Simulation*, vol. 79, no. 3, pp. 584–606, Dec. 2008.

[5] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, vol. 79, no. 3, pp. 497–516, 1957. [Online]. Available: http://www.jstor.org/stable/2372560

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015. [Online]. Available: http://dx.doi.org/10.1038/nature14236

[7] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980