Reinforcement Learning for Determining Spread Dynamics of Spatially Spreading Processes with Emphasis on Forest Fires

by

Sriram Ganapathi Subramanian

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2018

Abstract

Machine learning algorithms have increased tremendously in power in recent years but have yet to be fully utilized in many ecology and sustainable resource management domains such as wildlife reserve design, forest fire management and invasive species spread. One thing these domains have in common is that they contain dynamics that can be characterized as a Spatially Spreading Process (SSP) which requires many parameters to be set precisely to model the dynamics, spread rates and directional biases of the elements which are spreading.

We introduce a novel approach for learning in SSP domains such as wild fires using Reinforcement Learning (RL) where fire is the agent at any cell in the landscape and the set of actions the fire can take from a location at any point in time includes spreading into any point in the 3×3 grid around it (including not spreading).

This approach inverts the usual RL setup since the dynamics of the corresponding Markov Decision Process (MDP) is a known function for immediate wildfire spread. Meanwhile, we learn an agent policy for a predictive model of the dynamics of a complex spatially-spreading process. Rewards are provided for correctly classifying which cells are on fire or not compared to satellite and other related data.

We use 3 demonstrative domains to prove the ability of our approach. The first one is a popular online simulator of a wildfire, the second domain involves a pair of forest fires in Northern Alberta which are the Fort McMurray fire of 2016 that led to an unprecedented evacuation of almost 90,000 people and the Richardson fire of 2011, and the third domain deals with historical Saskatchewan fires previously compared by others to a physics-based simulator.

The standard RL algorithms considered on all the domains include Monte Carlo Tree Search, Asynchronous Advantage Actor-Critic (A3C), Deep Q Learning (DQN) and Deep Q Learning with prioritized experience replay. We also introduce a novel combination of Monte-Carlo Tree Search (MCTS) and A3C algorithms that shows the best performance across different test domains and testing environments. Additionally, some other algorithms like Value Iteration, Policy Iteration and Q-Learning are applied on the Alberta fires testing domain to show the performances of these simple model based and model free approaches. We also compare to a Gaussian process based supervised learning approach and discuss relation to state-of-the-art methods from forest wildfire modelling.

The results show that we can learn predictive, agent-based policies as models of spatial dynamics using RL on readily available datasets like satellite images which are at least as good as other methods and have many additional advantages in terms of generalizability and interpretability.

Acknowledgements

My sincere thanks to my supervisor Dr. Mark Crowley for all his support, ideas, help, feedback and kind encouragement during hard times. My research and this thesis would never be possible without his guidance.

I am grateful to all the members of the Machine Learning Lab. They have never failed to lend a helping hand during the times of crisis.

I would also like to thank Professor Sebastian Fischmeister and Professor Kate Larson for taking time off their busy schedules to read my thesis.

Above all I would like to express my gratitude to my parents. Their faith and confidence in me always inspires me to keep going.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

There's a clear and growing need for advanced analytical and decision making tools, as demands for sustainable management of scarce and important natural resources increase, and as the consequences of inaccurate decision making become more profound. Artificial Intelligence and Machine Learning methods provide ways to combine multiple modes of information such as spatial statistical ground data, weather data and satellite imagery into a unified model for classification or prediction.

One high impact example of this potential is Forest Wildfire Management. The risk, costs and impacts of forest wildfires are a perennial and unavoidable concern in many parts of the world. A number of factors contribute to the increasing importance and difficulty of this domain in future years including climate change, growing urban sprawl into areas of high wildfire risk and past fire management practices which focused on immediate suppression at the cost of increased future fire risk [46].

There are a wide range of challenging decision and optimization problems in the area of Forest Fire Management [40], many of which would benefit from more responsive fire behaviour models which could be run quickly and updated easily based on new data. For example, one simple decision problem is whether to allow a fire to burn or not, since burning is a natural form of fuel reduction treatment. Answering this question requires a great deal of expensive simulations to evaluate the policy options fully [28].

These simulations are built by an active research community for forest and wildland fire behaviour modelling. Data is collected using trials in real forest conditions, controlled lab burning experiments, physics-based fire modelling and more [19]. These hand crafted physics-based model simulations have high accuracy but are expensive to create and update and computationally expensive to use.

Recent advances in Deep **Reinforcement Learning (RL)** such as AlphaGo [60] and the more recent Alpha Zero [61] algorithms show that it is possible to combine **Monte Carlo Tree Search (MCTS)** intelligently with Neural Networks to get a sophisticated system that can beat the best Go players in the world. This serves as a motivation to try similar approaches on the forest fires. Forest fires present a more difficult challenge than these board games as the dynamics of the environment in the real world are more prone to uncertainties and randomness. Making learning agents that can give highly accurate solutions in such domains has been attempted before on simpler problems [22] but stopped at the simulation level as a demonstration. This thesis uses data from real fire events to show a proper proof of concept using a novel algorithmic framework.

Unfavourable weather conditions caused by climate change and increasing expansion of the urban-wildland interface make forest wildfire an increasingly important and complex problem to manage. Effective forest wildfire management and long term planning to improve the use fire in positive and negative ways requires accurate, adaptable and available models for fire dynamics. Machine learning offers the possibility to make such models adaptable to changing conditions, more accessible to poorer regions of the world that lack fire research programs and faster to use in real situations.

Forest wild fires are a phenomenon that is causing significant damage to the Earth's ecosystem in recent years. From 1980s to 2015 the number of acres burned has grown from 2 million a year to 8 million a year [1]. Specifically, the United States and Canada are seeing at least one large fire event every year. In 2017-2018 California has set a new record for area burned with the Thomas fire near Los Angeles becoming the largest fire in California history [70] having burned for an entire month in the depth of winter.

In this work, we model a learning agent to predict wildfire spread and carry out experiments. We report results based on the accuracy of interpolation and forward prediction using several different RL algorithms in a new formulation of agent-based learning. We have also implemented a Gaussian Process (GP) classifier [54] as a supervised learning model on different test domains for comparisons with the RL approach. The test domains comprise of a couple of well-known fires in Northern Alberta, historical fires in Saskatchewan and a widely used wild fire simulator. We also introduce a novel RL algorithm that beats all the preexisting RL algorithms in the different test domains.

Our approach promises to provide another source of validation for existing wildfire models as well as gives us the opportunity to perform faster and more accurate prediction by learning patterns in the raw data.

Another application of our approach would be learning transferable models from one data-rich region or time period and applying it to another region or time period where less data is available if there is evidence that fires in both regions behave similarly.

1.1 Problem Description

The question we ask is: **Can we learn a dynamics model from readily available satellite image and other related data and treating wildfire as an agent spreading across a landscape in response to neighbourhood environmental and landscape parameters?**

In this work we provide evidence for an affirmative answer to this question by introducing a new approach for using Reinforcement Learning (RL) [67] to automatically learn wildfire spread dynamics models by treating fire as an agent on the landscape taking spatial actions in reaction to its environment.

We see Forest wildfire spread as a specific case of a more general problem which we call a Spatially Spreading Processes (SSPs) which occurs when naturally occurring features change over time and space due to being strongly influenced by proximity in space, but without being caused by the movement of a single or multiple objects. Some good examples of SSPs include forest wild fires, floods, invasive species etc. They do not include scenarios of robot movements or car movements in space. The important difference is that in SSP domains we have an agent like the fire, which does not have an overall goal, but spread based on natural laws while in the other the agent does have an overall goal to achieve (e.g. robot moving in space).

In many of these domains the dynamics are often modelled by hand using agent-based models and geostatistics methods. In other areas, they are increasingly learned from data streams by being treated as videos. However, each of these approaches have their drawbacks. One of the goals of our research is to find novel representations of local dynamics for SSPs that can provide transparent and interpretative solutions to learning of dynamics models and decision making across many domains. The approach proposed here offers the tantalizing possibility to represent and learn a *causal* agent-based policy representation using RL which would be a much easier model to interpret and analyze for human decision makers.

Reinforcement Learning is being used in our research, for this domain due to several reasons. The first reason being that learning agents have never been used to understand the spread dynamics of complex natural processes using real world events. The second reason is that RL has recently shown superior performances in the spatial games like Go and chess; and robot movement problems which involve variations in space and time. The structure of the forest fire spreading problem can be seen to have some similarities to such domains. For the forest fire spread problem RL agent can be a possible solution as the fire is influenced by the environment (direction of spread) and it also influences the environment (phenomenon of burning) which is a requirement in any RL set up [68]. A RL agent has been attempted in spatial process before [22] and the author states that MDPs and RL being commonly used for representing and solving sequential decision making problems under uncertainty is a good fit for tackling problems like forest fires.

1.2 Thesis Contributions

The major contribution of this thesis is to make a Reinforcement Learning (RL) agent learn the dynamics model of a spatially spreading process such as forest wildfires and show the ability of Reinforcement Learning in decision making problems.

The following summarizes the important contributions of this thesis:

- Reinforcement learning is demonstrated to have advantages in learning dynamics models and is not restricted to games and robotic movements alone.
- We show that visualizing the forest fire as an agent learning in a complex environment is useful to the forest fire research.
- We demonstrate that the forest wildfire spreading problem can be modelled into a Markov Decision Process (MDP), which can be subsequently solved to get the fire spread policy.
- We present a solution that can use cheap and publicly available medium resolution satellite images as the primary source of data.
- We introduce a novel Reinforcement Learning algorithm and show that it performs better in spatially spreading domains than pre-existing algorithms.
- We have verified that the physics based simulators used for forest fire spread predictions have significant disadvantages and the use of Reinforcement Learning in such situations solves some of the problems.
- RL is shown to perform better than supervised learning approaches in the wild fire environment.
- We have provided evidence that the RL approach can learn the dynamics directly from noisy data without expert knowledge about fire spread and dynamics commonly used by fire practitioners.
- A novel approach of using Reinforcement Learning for decision making is presented.

1.3 Thesis Organization

The rest of the thesis is organized as follows:

Chapter 2 presents a literature review of the related research to the work that we are doing. Additionally, the state of the art in forest fire spread prediction and Deep Reinforcement Learning research are summarized. The common Machine learning approaches to Spatially spreading problems are also elaborated.

Chapter 3 provides a brief overview of the different Reinforcement Leaning methods used in this study. It goes into the details of their implementations. The mathematical formulations of these algorithms are presented. The functional importance of the different parameters and variables in these algorithms are explained. A pseudo code or steps of implementation of some of these algorithms are also shown. Reasons and intuitions behind the consideration of these specific algorithms are explained in this chapter.

Chapter 4 explains the formulation of forest wildfire problem as a Markov Decision Process (MDP). The possible states and actions are enumerated with reasons. This chapter also goes into the details of the initial experimental setup and the first couple of algorithms implemented in this problem. The assumptions and approximations used in the experimental design are elaborated. The description also explains how the training of Reinforcement Learning agent happens and how the testing is done. The interaction between testing domain (environment) and the Reinforcement Learning agent is highlighted. This involves the details of the reward function and neural net parameters (like activation functions and batch sizes) used in the Deep Reinforcement Learning techniques. All the experimentation and results pertain to the Alberta wild fire scenario. *Note: The experiments and results presented in this chapter have been published as a paper in the proceedings of the Conference on Reinforcement Learning and Decision Making [65].*

Chapter 5 considers additional experiments on the same Alberta fire situation, which are an intuitive extension of the results from the Chapter 4. Additional reinforcement learning algorithms are also considered in this chapter, to test a comprehensive list of algorithms on the same domain and record the advantages and disadvantages each of them have in different test cases. The results gives a clear picture of the performances of different classes of algorithms in this problem domain. *Note: This chapter is To appear as a paper in the Frontiers Journal of ICT - Environmental Informatics, 2018.*

Chapter 6 introduces a new algorithm based on the results of the previous chapters and considers more testing domains such as simulated fires and Saskatchewan wild fires in addition to the Alberta fire scenario. The simulated fires have cleaner data and this domain is less affected by inherent noise and inaccuracies of satellite imagery. The Saskatchewan wild fires have been considered to compare the performances of reinforcement learning and a physics based simulator, Burn Probability 3 (BP3). This dataset has been previously used by wild fire researchers with the physics based simulator. A new algorithm that combines the advantages of previous algorithms is proposed and is shown to perform better than the other considered algorithms. Different other Deep Reinforcement learning methods are also considered for testing. *Note: This chapter is To appear as a paper in the Proceedings of the 30th Canadian Conference on Artificial Intelligence, Toronto, May 2018*

Chapter 7 draws the conclusions from the paper and the presented results. Possible future work that can step out of this research are also discussed.

Chapter 2

Related Work

We provide an overview of the relevant literature on the Forest Wildfire Prediction and Management problem in general, use of Machine learning on satellite imagery and the previous use of Machine Learning algorithms for the forest wildfire domain. We also mention how our work is different and advantageous compared to these approaches.

2.1 Forest Wildfire Prediction and Management

The author in [46] discusses the increased future fire risk at the consequence of fire management practices which focused on immediate suppression. The three core themes of externalities, incentives and risk based decision analysis in the case of wild fire suppression is described. The goal is to determine how the core themes contribute to the evolution of an effective future fire policy. This problem is also enumerated in [28]. We have approached this idea of learning a trade-off for immediate suppression and future fire risk using a policy by representing the fire as a learning agent in a spatially spreading environment.

The standard for wildland fire behaviour and forest fire modelling is described in [19] which carried out exhaustive lab experiments, real forest condition simulations and trials to build sound and coherent fire spread theory for model reliability. The resulting models are used by the US Forest Service but are very computationally expensive to run. The model accuracy also various widely across wild fires in different regions.

Recent works [30][21] use approximations such as ellipse-shaped spread and hexagonal grids to balance the accuracy and computational cost of a physics-based model. Cellular Automaton

Models are also widely used to predict wildfire spread [76]. Our approach is easier to apply than these methods and is shown to perform better than the same.

The work in [40] describes the need for a computationally simple and accurate model for wild fires. They highlight a number of challenging decision and optimization problems in the area of forest fire management and recent efforts to develop decision support tools to overcome them. The focus is on using methods of Operations Research to aid fire managers making complex decisions about fire suppression and resource allocation.

2.2 Machine Learning in Forest Fire Management

In [12] the authors discuss the application of an intelligent system based on genetic programming for the prediction of burned areas in a wild fire situation. They also compare the genetic programming methodology to state of the art machine learning techniques in fire modelling and conclude that genetic programming techniques are better. The major machine learning techniques used are SVM with a polynomial kernel, random forests, radial basis function network, linear regression, isotonic regression, neural networks. The data used is meteorological data and specific data sets from the forest under analysis. While this paper summarizes the current progress made in forest fire modelling and burned area prediction using the most widely used Machine learning methods, it does not touch upon the usage of deep networks with multiple levels of auto encoders or the use of reinforcement learning techniques, both of which are concentrated upon in our work.

Significant problems have arisen while dealing with large data bases or long periods of observation, (e.g. pattern recognition, geophysical monitoring, monitoring of rare events (natural hazards), etc). The authors in [22] stipulate that the major problems in such cases are how to explore, analyze and visualize the oceans of available information. Several important applications of Machine learning algorithms for geospatial data like regional classification of environmental data, mapping of continuous environmental data including automatic algorithms and optimization (design/redesign) of monitoring networks are presented. The result is that learning machines are usually universal tools, i.e. in principle they can model any mapping (either categorical or continuous data) with any desired precision. Several important software tools that include GeoMISC, GeoKNN, GeoMLP, GeoSVM and their relative pros and cons in several ecological applications are discussed in this paper which was useful for our work.

Machine learning algorithms use an automatic inductive approach to recognize patterns in data. Once learned, pattern relationships are applied to other similar data in order to generate predictions for data-driven classification and regression problems. The work in [16] takes a task of supervised lithology classification (geological mapping) using airborne images and multispectral satellite data and compares the application of popular machine learning techniques to the same. A ten fold cross validation was used to select the optimal parameters to be used in all the methods. These selected parameters were used to train the Machine Learning classification models on the entire set of samples. A prediction evaluation was carried out as a final step. Overall accuracy and Cohen's kappa statistic were used to evaluate classifier performance. A comparison of performances of all the Machine Learning algorithms considered is presented. The authors find random forests (RF) are a good first choice algorithm for this supervised classification problem, the reasons being that the RF is extremely stable, computationally efficient, easy to train and more accurate than the other common Machine learning algorithms. They explored the impacts of highly dispersed data and the inclusion of explicit spatial information on accuracy and sensitivity. The conclusion demonstrated that is critical to have high resolution spatial data with good distribution for the simple Machine learning models to work well.The airborne geophysical digital elevation model used in this study is expensive to obtain and is not readily available for most fires. Our work addresses this problem by simply using widely available easy to obtain medium resolution satellite imagery.

In [59] the authors use a machine learning approach for Geospatial Entity resolution which is the problem of consolidating data from diverse sources into a single data source referenced by location (in the form of coordinates). Several feature based matching techniques like location name matching, coordinate matching and location type matching are introduced and evaluated. These feature-based matching techniques use each location feature independently. A new method integrating spatial and non-spatial features and learning a combined spatial similarity measure is introduced.

Modelling forest areas is concentrated upon in [23]. The environmental variables consisting of both topographic and climatic factors are considered in this work. A modelling framework for habitat modelling is established to train,test and validate the popular predictive machine learning methods. Neural Networks, Random Forests and Tree-Based Classification are used as predictive models. A Receiver Operating Characteristic (ROC) (Specificity vs Sensitivity graph analysis) is used for parameter selection. Species distribution and habitat modelling are described to be complex problems much like the wild fire problem with many responsible factors and the authors admit that modelling all the factors are impossible with the current state of the art. Hence, having an agent based approach which learns the relevant factors on its own seems to be the most suitable idea.

Machine Learning is used for the spatial interpolation of environmental variables in [38]. In this study around 23 methods are considered including popular machine learning methods and their combinations. Along with machine learning the considered methods were drawn from a large pool of categories including geostatistical methods, non geostatistical methods, statistical methods and combined methods. The data set consists of about 177 samples of sea bed mud content in the southwest of Australian Exclusive Economic Zone and the problem is to determine the mud content in the other points by interpolation. Several primary and secondary variables are considered to influence this decision.Accuracy tests were done using 10 fold cross validation. Spatial predictions were additionally assessed using visual examination tests. Random forest seems to perform the best among the methods considered. RF performance was attributed to its relative robustness to outliers and noise, its nature of not over fitting with respect to the source data and its ability to model complex interactions.

The author in [55] discusses a summary of all the different ecological modelling applications of machine learning. Artificial Neural nets, Genetic Algorithm methods and Adaptive Agents seems to perform the best but they seem to have advantages for different problems. The conclusions are that artificial neural nets perform well in problems of non linear ordination, visualization, multiple regression, time series modelling and image recognition and classification. Genetic algorithms are suitable to evolve causal rules, process equations and optimize process parameters. Adaptive agents are suggested for providing a novel framework in the aid of the discovery and forecasting of emergent ecosystem structures and behaviours in response to environmental changes. This review demonstrates that while various classical AI and ML techniques have been applied to this domain in preliminary forms, there is a gap in the literature on attempting to use modern deep learning and RL techniques.

The existing state of the art methods for building fire prediction models directly satellite images include the Forest Fire risk prediction model used in China [78] and the Canadian systems like Forest Fire Weather Index (FWI) System and the Forest Fire Behavior Prediction (FBP) System ([64], [78]). These systems are still quite difficult to implement due to the requirement of large amount of heterogeneous sources of data such as fuel property and fire characteristics [78] which requires data from high resolution close-range sensors. The work by [2] explains the advantages of the satellite based fire prediction systems but specifies that the temporal resolution of available data serves as a serious impediment. Our work specifies a method to learn from available data and reliably interpolate to fill the missing spaces to get an acceptable overall performance.

We model the forest fire domain as a Markov Decision Process (MDP). This approach is becoming more common [41], but earlier works usually focused on finding treatment actions to be taken to reduce or alter fire spread. We take the novel approach of modelling the fire itself as a decision agent attempting to minimize prediction error. This is related to work in [22] which investigated using the state variable to represent land vegetation cover and environmental characteristics but to use the action variable to represent interaction between characteristics of nearby locations. This was not on a domain as dynamic as forest fire however and our approach differs entirely in implementation.

Machine learning has also been used to detect and classify burn areas in Indian forests [57].

The authors of [57] use Spatial data mining to obtain useful information from a series of data sets and subsequently apply supervised algorithms like Artificial Neural Nets (ANN) and Sequential minimal optimization (SMO) to quantify ignition risk of different regions and hence predict the occurrence of fires. Similarly [62] uses Decision Trees and a series of IF-THEN rules to develop a classification model for forest fires in Indonesia. We prove the superiority of RL techniques to such supervised methods in our work. For instance, the results in [62] show an accuracy of about 63%, whereas our best RL models perform much better than that in similar test cases. Additionally, the authors in [3] use fuzzy logic and fuzzy membership rules to make a forest fire detection system from satellite images. The methods outlined, is only capable of predicting fires at the time of satellite image availability and it is not possible to predict forward or in-between as demonstrated in our work.

Our work also has similarities to the use of intelligent systems for predicting burned areas as suggested in [12]. However, that work focused on burned area alone whereas we look at the more specific problem of prediction of actual fire spread location over the short term. That work also relies on high resolution field data which may not be possible to obtain for all fires.

2.3 Machine learning on Satellite Imagery

Researchers have attempted to use Machine Learning in combination with satellite imagery in a series of ecological applications in the past. In [36] the authors use satellite radar images and machine learning for the detection of oil spills. Algorithms such as C4.5 and 1-nearest neighbour is used along with expert rules to train the classifier which the authors specify was hard and not completely accurate. Our work completely removes the need for such expert rules.

In [29] satellite imagery and machine learning has been used to tackle the case of poverty. Convolutional Neural Nets(CNNs) and transfer learning have been used to derive models having good performances in terms of accuracy. This research demonstrated how machine learning tools, that are typically suited for data rich domains, can be used for data scarce settings too.

Coral reef research is another aspect that has been studied using satellite images and Machine learning [32]. In [32] a series of statistical and Machine learning models have been used on the IKONOS satellite imagery to produce spatially explicit predictions of species richness, biomass and diversity of fish community. This research motivates the exploration of importance of different variables in the predictive models by using permutations techniques.

2.4 Summary

This section outlines a series of research efforts that are related to the topic of this thesis. We have seen that use of any form of Machine learning have not been applied on real fire events to demonstrate the effectiveness of learning agents. We have also seen that making learning agents that can give highly accurate solutions in such domains have been attempted before on simpler clones of real world problems but have stopped at the simulation level as a demonstration. This paper uses data from real fire events to show a proper proof of concept using a novel algorithmic framework. Reinforcement Learning have not been applied to any form of actual spatially spreading process using real world data sets. This is the main motivation for us to try the effectiveness of Reinforcement Learning agents on real world spatially spreading processes. Another observation is that the physics based simulators used in forest fire research need a copious amount of expert knowledge that explain the role of different factors/variables that affect wild fires. This is not needed, when learning agents are used to present a solution and that is a single greatest push for the use of learning agents in such scenarios.

Chapter 3

Algorithms Considered

Reinforcement learning is an active area of Machine Learning in which the agents must act in an environment in such a way that the overall cumulative rewards are maximized. The RL approach aims to solve the Markov Decision Process (MDP) where there are a set of states determining the environment S, set of actions A, that the agent can legally take, a transition function $P_a(s, s')$ which gives the probability of transition from one state to another and a Reward function $R_a(s, s')$ that gives the agent a reward (or punishment) for its action choice in a particular state. The aim of the RL is to solve the MDP and obtain a policy $\pi(s, a)$ which gives a mapping of state to actions. This policy stabilized over time represents what the agent should do at every state to maximize the rewards. There are many different types of RL algorithms that can solve this MDP. We use several very different, widely used RL algorithms on the testing domains in order to see the range and application of the idea. We begin with three classic approaches for iteratively solving MDPs, Value Iteration, Policy Iteration and, Q-learning. We then consider the recent and very different approaches of Monte Carlo Tree Search and A3C, a recent policy gradient RL algorithm which utilizes Deep Learning for value function representation. We have also considered the Deep Q learning (DQN) deep learning approach and also one of its variants with the use of prioritized experience replay. These algorithms will be reviewed briefly with modifications needed for our problem highlighted.

3.1 Asynchronous Value Iteration (VI)

Value Iteration computes the optimal state value function by iteratively improving the estimate of the value function. The algorithm starts with initial arbitrary values for the value function and repeatedly updates the same until convergence or termination is reached. The optimal value of the state $V^*(s)$ is given by the following Bellman Equation:

$$V^*(s) = R(s) + max_a \gamma \sum_{s'} P(s'|s, a)V^*(s') \tag{3.1}$$

where s' is the successor state and γ denotes the discount factor. The policy is the action that maximizes the value function. This is obtained by taking the $argmax_a$ of the same. States are randomly sampled to update the value function according to (3.1) and the updates continue till the termination is reached. The asynchronous value iteration algorithm is a refined version of the value iteration algorithm with the improvements that the states are updated one at a time, in any order, and the values are stored in a single array.

The steps used in implementing the Asynchronous Value Iteration are given in Algorithm 1. The algorithm is terminated after it runs for about (4-10) hours. The exact time is determined based on monitoring runtime plots to figure out the time at which the relative improvements in scores gained becomes very small or zero. This approach is taken instead of explicitly monitoring convergence values as it is computationally simpler (in both memory and time) and eliminates the need for complete saturation in a large state space. Many researchers have used different methods for terminating the algorithm early as the convergence takes prohibitively large amounts of time and the improvements become relatively minor in the later stages [14],[79], [8]. This is the case for the termination conditions of all the subsequent algorithms as well. The policy is guaranteed to progressively get better and converge[6].

3.2 Policy Iteration (PI)

While value iteration keeps improving the value function at each iteration, policy iteration targets the the policy directly. The optimal policy could converge before the value function. We begin with an initial random policy for acting in each state and iteratively improve it through alternating policy evaluation (Equation 3.2) and improvement (Equation 3.3) steps defined as follows:

$$V^\pi(s) = \sum_{s'} P(s, a, s')[R(s, a, s') + \gamma V^\pi(s')] \tag{3.2}$$

$$\pi(s) = argmax_a \sum_{s'} P(s, a, s')[R(s, a, s') + \gamma V^\pi(s')] \tag{3.3}$$

where s belongs to set of all states S and γ is the discount factor. For this algorithm, the value function $V_\pi(s)$, for the optimal policy π, is an array of **Net Policy Worth** values for all states,

Algorithm 1 Asynchronous Value Iteration

```
1:  procedure ASYNCHRONOUS_VALUE_ITERATION(S,A,P,R,T)
2:      Inputs:
3:      S: Set of all States
4:      A: Set of all Actions
5:      P: State Transition function
6:      R: Reward Function
7:      T: Time for completion
8:      Outputs:
9:      $\pi[S]$: Approximate Optimal Policy
10:     $V[S]$: Value Function
11:     Local Variables:
12:     $V[S]$: real array
13:     $\pi[S]$: action array
14:     assign $V[S]$ arbitrarily
15:     assign Time = 0, Time is tied to the system clock and tracks the total time the program is
    being executed.
16:     while Time $<=$ T do
17:         select a state s
18:         select an action a
19:         Update Value V according to $V(s) = R(s) + max_a \gamma \sum_{s'} P(s'|s,a)V(s')$
20:     end while
21:     while state s = true do
22:         $\pi[s]$ = $argmax_a\, V[S]$
23:     end while
24:     return $\pi$, V
25: end procedure
```

which is the highest cumulative reward that could be obtained for an optimal policy passing through that state at the time of consideration. The steps used in implementation are given in Algorithm 2. The loop termination and times are similar to value iteration for all experiments. The most recent policy upon termination is returned as the optimal policy and the value function under this optimal policy is also returned. Convergence guarantee is given in [53]

3.3 Q-learning (QL)

Both value iteration and policy iteration are model based methods that attempt to model the environment and then choose the best policy based on this model. Q Learning algorithm on the other hand is a model free approach where the attempt is to learn the optimal policy without learning this model. This algorithm [74] performs off-policy exploration and uses temporal differences to estimate the optimal policy. In Q-learning, the agent maintains a state-action value function $Q(S, A)$ instead of a state value function. This is updated as follows:

$$\hat{Q}(s,a) = (r + \gamma(\max Q(s', a'))) \tag{3.4}$$

and

$$Q(s,a) = Q(s,a) + \alpha(\hat{Q}(s,a) - Q(s,a)) \tag{3.5}$$

where α denotes the learning rate and γ denotes the relative value of delayed vs immediate rewards also known as the discount factor. s' is the new state after action a. a, a' are actions taken in states s, s' respectively. $\max_a Q(s', a')$ denotes the estimate of maximum discounted future reward expected. Both the learning rate and discount factors are in the range [0-1].

Higher learning rate for Q-learning makes the recent information to have a higher impact, which is needed in a continuous spatial environment like forest fires. There is a high positive spatial auto-correlation in spatial datasets corresponding to fire as the similar pixels(on fire or not fire) tend to cluster together. The Q-learning is made to exploit this property using the appropriate learning rate. The discount factor is also kept high as the long term rewards are more important than the short term rewards in our model. Using a lower discount rate (closer to 0) decreases the level of exploration and the risk of falling into a local optimum becomes high [73]. Each state-action pair is considered as a step taken in the real world. For every valid state the highest Q value (Utility value) for the state is recorded. Algorithm 3 gives implementation details. The termination condition is as discussed in the Value Iteration algorithm.

Algorithm 2 Policy Iteration

```
1: procedure POLICY_ITERATION(S,A,P,R,T)
2:     Inputs:
3:     S: Set of all States
4:     A: Set of all Actions
5:     P: State Transition function
6:     R: Reward Function
7:     T: Time for completion
8:     Outputs:
9:     $\pi[S]$: Approximate Optimal Policy
10:    $V_\pi[S]$: Value Function (Net policy worth)
11:    Local Variables:
12:    $V_\pi[S]$ : real array
13:    $\pi[S]$: action array
14:    choose an arbitrary policy $\pi$
15:    assign Time = 0, Time is tied to the system clock and tracks the total time the program is
       being executed.
16:    while Time <= T do
17:        Compute the value function of the policy according to
               $$V_\pi(s) = \sum_{s'} P(s,a,s')[R(s,a,s') + \gamma V_\pi(s')]$$
18:        Improve the policy according to
               $$\pi(s) = argmax_a \sum_{s'} P(s,a,s')[R(s,a,s') + \gamma V_\pi(s')]$$
19:    end while
20:    return $\pi$, $V_\pi$
21: end procedure
```

Algorithm 3 Q Learning

```
1: procedure Q_LEARNING(S,A,P,R,T)
2:     Inputs:
3:     S: Set of all States
4:     A: Set of all Actions
5:     P: State Transition function
6:     R: Reward Function
7:     T: Time for completion
8:     Outputs:
9:     $Q[S, A]$: Q function
10:    Local Variables:
11:    $Q[S, A]$ : Q array
12:    Initialize $Q(S, A)$ arbitrarily
13:    assign Time = 0, Time is tied to the system clock and tracks the total time the program is
   being executed.
14:    while Time <= T do
15:        Initialize s
16:        while state s = true do
17:            Choose a from s using policy derived from Q ($\epsilon$ greedy)
18:            Take action a, observe r,s'
19:            Update $Q(s, a)$ according to Equation
                   $Q(s, a) = Q(s, a) + \alpha(r + \gamma(maxQ(s', a') - Q(s, a)))$
20:            s = s'
21:        end while
22:    end while
23:    return $Q$
24: end procedure
```

3.4 Monte Carlo Tree Search (MCTS)

MCTS [33] are a class of algorithms that perform approximate, but confidence-bounded, short roll-outs of the current policy in order to obtain enough statistical information about a state to keep it or discard it as belonging to the optimal path. A good survey is provided in [9]. MCTS [33] uses rollouts to determine the value of every state, which are represented as nodes in a search tree connected by actions along the branches. In every Monte carlo tree search run we have four steps: Selection, Expansion, simulation and back propagation. In the first step, selection, the UCT (Upper Confidence Bound applied to trees) [24] calculates an upper bound B.

$$B = X + \sqrt{2log(p)/n}$$

where X is the average of the rewards obtained from the node for which B is calculated, n is the number of times the node has been visited and p is the number of times the node's parent node has been visited. This is used to produce the value estimate in MCTS. Selection is continued till we reach the end (leaf nodes) of the tree with visited nodes. The next step is the expansion where an unvisited child is added to the tree, followed by simulation where there is a random play out till the goal state and the final step is back propagation at which the counter variables (like visit count and reward count) are incremented. The pseudo code is given in Algorithm 4.

3.5 DQN

The first Deep Learning technique that we use is the Deep Q learning [45]. Neural Networks are used to represent the Q values in this algorithm as opposed to simpler tabular representation of the Q values seen in the Q learning algorithm (Section 3.3).

Deep Learning is commonly used in many classification problems [77],[35]. The Neural networks are typically organized in layers in all deep learning representations. Layers are made up of a number of interconnected nodes which contain an activation function. Patterns are presented to the network via the input layer, which communicates to one or more hidden layers where the actual processing is done via a system of weighted connections. The hidden layers then link to an output layer where the output are the classes in a classification problem. Deep RL techniques like Deep Q Learning, use the deep learning representations in Reinforcement Learning. The Deep Q Learning is a model free Deep Reinforcement Learning method used in this work. This algorithm is off-policy in the sense that an exploratory step is undertaken with epsilon probability at every action step.

Figure 3.1 shows the representation of the Q values in DQN. The inputs are only the state values and outputs are the Q-value for each possible action. This approach has the advantage,

Algorithm 4 MCTS

```
1:  procedure VOID MCTS (NODE ROOTNODE, TIME T)
2:      assign Time = 0, Time is tied to the system clock and tracks the total time the program is
    being executed.
3:      while Time <= T do
4:          currentnode ← rootnode
5:          while currentnode!=leafnode do
6:              lastnode ← currentnode
7:              currentnode ← SELECTION(currentnode)
8:          end while
9:          lastnode ← EXPANSION(lastnode)
10:         SIMULATE
11:         while currentnode do
12:             Backpropogate(currentnode)
13:             currentnode ← parent(currentnode)
14:         end while
15:     end while
16: end procedure
```

that if we want to perform a Q-value update or pick the action with highest Q-value, we only have to do one forward pass through the network and have all Q-values for all actions immediately available.

The DQN has two networks having the same structure but with different weights. The networks are called as a Target Net and Eval Net respectively. The Target Net is considered to be a stable net with network updating every 200 steps. The Eval Net is updated every step. A separate target net is used as a stable network is needed for calculating loss. If the network keeps changing then the whole system would fall into local feedback loops and hence, such a set up is necessary. Both the Target Net and the Eval Net are used to calculate the associated Q values (q_target and q_eval) respectively. The network architecture is shown in Figure 3.3. The same architecture is used for both the eval and the target nets. There are three fully connected layers in addition to the input layer. The first twp are the hidden layers and the final one is the output layer. The Rectified Linear Unit (ReLU) activation function is used as the activation function for all the layers. Input to the network are all the different state variables as followed previous implementations [44]. Outputs of the network are Q-values for each possible action. Q-values can be any real values, which makes it a regression task, that can be optimized with a simple squared error loss.

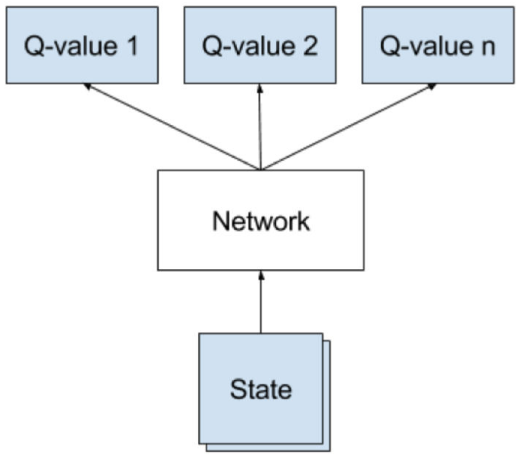


Figure 3.1: Representation of the Q function using Neural Networks.

$$L = \frac{1}{2}[r + \gamma max_{a'} Q(s', a') - Q(s, a)]^2$$

where the variables have the usual meaning as mentioned in (Section 3.3).

The principle of experience replay is used to train the network. This involves random batches of memory to be used for training, than the whole memory, for improving the learning efficiency of the Neural Net. For all action steps, the experiences $< s, a, r, s >$ are stored in a replay memory. When training the network, random samples from the replay memory are used instead of the most recent transition. This breaks the similarity of subsequent training samples, which otherwise might drive the network into a local minimum. Additionally, experiences in successive steps may not be too different from each other. The loss as defined in Equation 3.5 is used to update the Eval Net based on a stochastic gradient back propagation step. In this equation the $Q(s, a)$ is the predicted value obtained from the Eval Net and the $r + \gamma max_{a'} Q(s', a')$ is the target value obtained from the Target Net. After every fixed number of iterations (200) the Eval Net parameters are simply copied onto the Target Net. The policy keeps getting refined over several thousands of iterations, before an acceptable policy can be learned. A parameter γ called the discount factor is used to indicate that the relative importance of the most recent rewards as compared to the future rewards obtainable. This variable normally takes a value in the range from 0 - 1, where 1 means that the future rewards dominate whereas 0 means that only the most recent rewards mater. The Q values of the target network (q_target) after termination, is considered as the utility values for all states. The high level architecture of the entire network is shown in

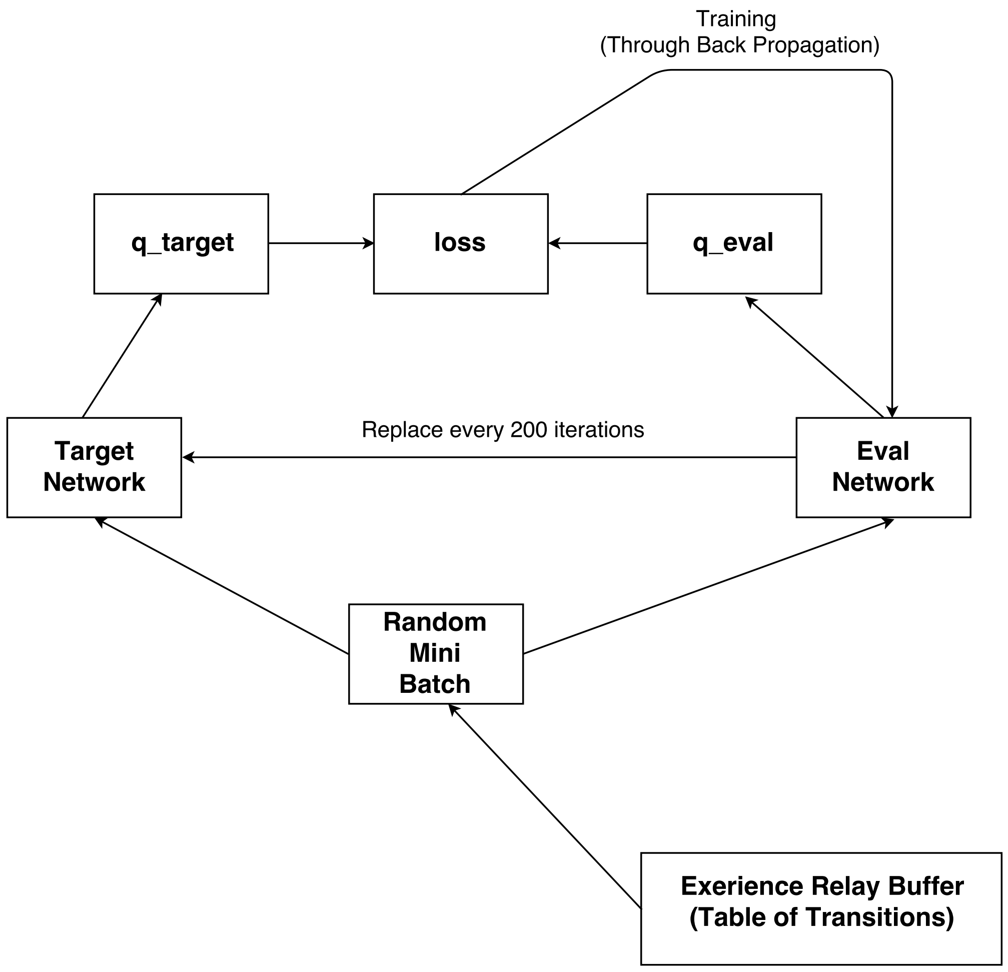


Figure 3.2: High level architecture of the DQN algorithm.

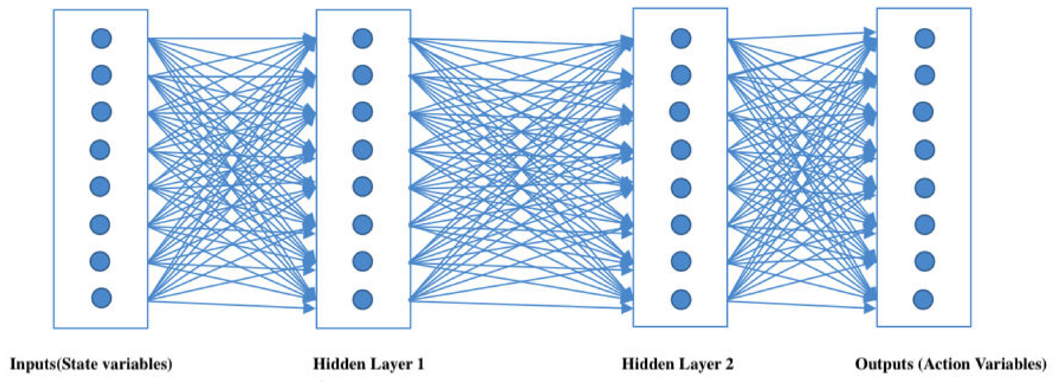


Figure 3.3: Deep layers in DQN. The architecture is same for both the Target and the Eval Nets.

Figure 3.2. Pseudo code is given in Algorithm 5.

The DQN set up is capable of handling delayed rewards as previous implementations have shown [37], [63]. There are several ways this has been achieved, the most common being using an exponentially reducing discount rate so that the earlier rewards accumulated matter much less compared to the later rewards. In some other cases the final moves are made to give large rewards, which are weighted highly for the experience replay using methods such as prioritized experience replay as mentioned in the next subsection.

3.6 DQN with Prioritized Experience Replay

The above explained DQN algorithm is used with a prioritized experience replay [58] for this algorithmic implementation. The idea of this implementation is that the transitions that do not fit well to our current estimate of the Q function are preferred as they are the transitions from which the maximum knowledge can be gained. The prioritized experience replay weights all the different experiences with the probability of the absolute value of error between the Q target and calculated Q values. The prioritized values are stored in a binary heap tree as a priority queue and an Experience class is built to store and retrieve the sample. This replaces the simple experience replay seen in the previous section.

Algorithm 5 Deep Q Learning

```
 1: procedure DQN(time T)
 2:     Initialize replay memory $D$
 3:     Initialize action-value function $Q_{eval}$ with random weights $D$
 4:     Consider the initial state $s$
 5:     assign Time = 0, Time is tied to the system clock and tracks the total time the program is
    being executed.
 6:     while Time $<=$ T do
 7:         select an action $a$ = $argmax'_a Q_{eval}(s, a')$
 8:         select $a$ random action with probability $\epsilon$
 9:         perform $a$
10:         observe reward $r$ and new state $s'$
11:         store experience $< s, a, r, s' >$ in replay memory $D$
12:         sample random transitions $< ss, aa, rr, ss' >$ from replay memory $D$
13:         calculate target for each mini batch transition
14:         if $ss'$ is terminal, $tt = rr$
15:         Otherwise, $tt = rr + \gamma max'_a Q(ss', aa')$
16:         train the Eval Q network using $\frac{1}{2}(tt - Q(ss, aa))^2$as loss
17:         For every 200 moves, $Q_{target}$ = $Q_{eval}$
18:         s = s'
19:     end while
20: end procedure
```

3.7 A3C

The final preexisting RL algorithm used for comparison is the **Asynchronous Advantage Actor-Critic (A3C)** algorithm [42] which represents the state-action value function using a Deep Q-Network (DQN)[45]. In the DQN algorithm we had a single agent interacting with a single environment. A3C on the other hand, has multiple incarnations of the same to learn more effectively. The high level architecture of A3C is given in Figure 3.4. The A3C algorithm has become the most popular algorithm for most RL algorithms with complex state and action spaces. It is developed in such a way that there is a global network interacting with a group of worker agents who have their own environment and network parameters. Each worker agent is independent of the others and the overall experiences are accumulated by the global network estimating the value and policy functions. This algorithm estimates a value function and a policy simultaneously. These are maintained as a separate fully-connected layers at the top of the global network, thus combining the benefits of different kinds of RL algorithms that optimize either the value or policy. The difference between the Q estimate and Value estimate is called as the advantage estimate and it tells us how much better the actions performed than expected as against just determining how good the actions were using discounted rewards. The advantage estimate segregates the good and bad actions which will help the agent converge towards a good solution. The advantage estimate is given by

$$A = R(s, a) - V(s)$$

where R is the discounted reward given by $R = \gamma(r)$ where γ is the discount factor, r is the immediate reward obtained and V is the value estimate of state s.

The deep network, was the same as that defined in the DQN implementation. For our problem A3C has the advantage of defining multiple worker agents. The pseudo code is given in Algorithm 6.

3.8 Supervised Classification - Gaussian Process

In this supervised classification method of performing the experiments the Gaussian process algorithm [54] is used to estimate if a particular cell is burning or not based on the burning conditions of other cells in the scene. For each particular cell, the attributes of temperature, rain, relative humidity, land cover type, wind speed, and wind direction are considered to influence the decision.

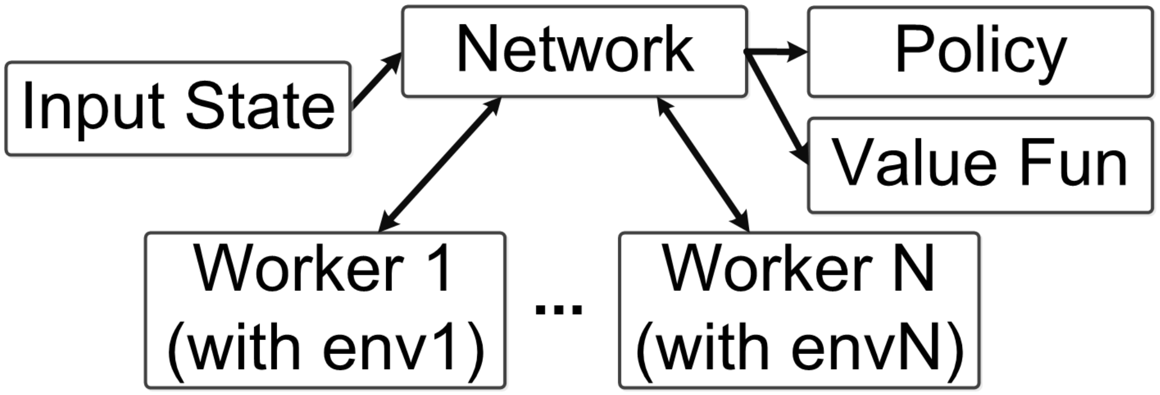


Figure 3.4: A3C high-level architecture.

Algorithm 6 A3C (for each worker thread)

```
 1: procedure A3C(time T)
 2:     Initialize thread step counter t $\leftarrow 1$
 3:     assign Time = 0, Time is tied to the system clock and tracks the total time the program is
    being executed.
 4:     while Time $<=$ T do
 5:         Reset Gradients: $d\theta \leftarrow 0$ and $d\theta_v \leftarrow 0$
 6:         Synchronize thread-specific parameters: $d\theta'$ = $\theta$ and $d\theta'_v$ = $\theta_v$
 7:         $t_{start} = t$
 8:         Get state $s_t$
 9:         while terminal $s_t$ or $t - t_{start} == t_{max}$ do
10:             Perform $a_t$ according to policy $\pi(a_t|s_t;\theta')$
11:             Receive reward $r_t$ and new state $s_t + 1$.
12:             $t \leftarrow t + 1$
13:         end while
14:         $R = 0$ for terminal $s_t$
15:         $R = V(s_t, \theta'_v)$ for non-terminal $s_t$
16:         for $i \in t - 1, ...t_{start}$ do
17:             $R \leftarrow r_i + \gamma R$
18:             Accumulate Gardients wrt $\theta$ and $\theta_v$
19:         end for
20:         Perform asynchronous update of $\theta$ and of $\theta_v$
21:     end while
22: end procedure
```

A distribution over function f can be specified by the Gaussian process as given by:

$$p(f) = GP(f|0, k(x, x'))$$

where the mean function is zero, and the covariance is defined by some kernel function. We use the RBF kernel in our experiments because we have an assumption of 2D proximity of features being relevant in any direction.

The GP prior is a multivariate normal given by

$$p(y) = Normal(y|0, K)$$

where K is a covariance matrix given by evaluating $k(x_n, x_m)$for each pair of inputs. The mean of the GP prior is assumed to be 0 in accordance with the above equations.

To obtain a probability the output response is condensed into the range $[0, 1]$ which is an appropriate choice for classification. The probability is given by the condensed value and a Bernoulli distribution is used to determine the label.

The likelihood of an observation (x_n, y_n) is given by

$$p(y_n|z, x_n) = Bernoulli(y_n|logit^{-1}(z_n))$$

Gaussian processes were chosen for this problem as they are known to work well for modelling non-linear relationships between variables especially in spatial domains. For the GP classifier we use the logistic (sigmoid) function, whose integral is approximated for the binary case to map output to a probability. For the covariance function we use an RBF kernel (stationary kernel) because we have an assumption of 2D proximity of features being relevant in any direction. The hyperparameters of the kernel are optimized by maximizing the log-marginal-likelihood of the optimizer. A Cholesky decomposition was used to decompose the kernel matrix. The result is a binary output response which can be used for classification.

3.9 Summary

This chapter outlines all the different reinforcement learning algorithms and supervised learning algorithm considered in this study. The algorithms have been derived from a wide range of model free and model based approaches in RL. The mathematical formulations of the algorithms are given in this chapter. Notice that for all the utility equations in this chapter, (For example, Equations 3.1, 3.2, 3.4) the immediate reward value for the state action transitions is clearly the most major influence on the utility function. The other term/terms are normally heavily discounted using a discount factor (like γ) so that they don't matter as much. Small increase/decrease in the reward function will show a corresponding small increase/decrease in the local utility values uniformly for all cells as they are almost linearly related. Thus, the local maxima, which is almost always (except in exploration with very low probability) chosen as the corresponding action for a state, is not affected by the small reward changes. Since the overall goal for the agent is to maximize the global utility in a given number of moves to determine the policy, these small changes to the reward function will not alter the global policy in any way. It may only increase/decrease the time of convergence/saturation. Thus, the system is resilient to small changes in the reward function. This was observed while experimenting with the reward function too.

Chapter 4

Reinforcement Learning for Spread Dynamics

This chapter introduces the idea of using Reinforcement Learning for determining spread dynamics of spatially spreading processes like forest fires. It explains the initial formulation of the forest fire spreading environment as a Markov Decision Processes (MDP) which can be solved using Reinforcement Learning. The results are shown on a set of forest fires in Alberta using a couple of RL algorithms.

Note: This chapter is published as a paper in the proceedings of Conference on Reinforcement Learning and Decision Making [65].

4.1 Problem Formulation

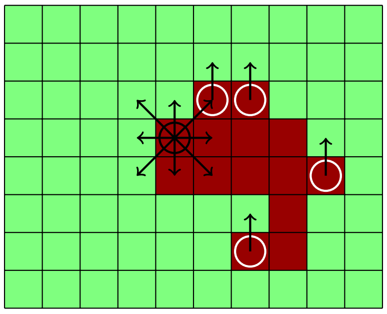
We define a Markov Decision Process (MDP) $< S, A, P, R >$ where the set of states S describes any location on the landscape and where the 'agent' taking actions is a fire spreading across the landscape. A state $s \in S$ corresponds to the state of a cell in the landscape $(x, y, te, l, w, d, rh, r, i)$ where x, y are the grid coordinates of the cell, te is the temperature at the particular time and location and l is the land cover type of the cell derived from satellite images (values include: water, vegetation, built up, bare land, other), w is wind speed, d is wind direction, rh is relative humidity, i is the fireline intensity of fire at a cell as defined in [31], and r is the average amount of the rainfall at the spatial coordinates during the time of study. These variables are considered to be the most contributing factors to fire spread as they are the primary variables in the Canadian Forest fire weather index as specified in [15]. The action $a \in A$ indicates the direction the fire at a particular cell 'chooses' to move: North, North-West, North-East, South, South-West, South-East, East or West or to stay put.

The dynamics function for any particular cell $P(s'|s, a)$ is a mapping from one state s to the next s' given an action a. Note, unlike most RL domains where the unknown dynamics of the system can be very complex, in this formulation the dynamics are actually straightforward. Once the direction of fire spread is determined, simulating the impact of burning and the resulting state is almost deterministic based on local conditions. Most properties of the cell state do not change quickly, or at all, in direct response to the agent's actions.

The reward function R maps a cell state to a continuous value in the range [-1,1]. Rewards are based on the land-cover, for example, a cell predominantly filled with water has less chance catching fire compared to one filled with vegetation. The reward function scores spread choices in proportion to flammability of a cell (Veg: 0.5, Water = -1, urban = -0.1, empty land = 0.05, Other = 0). These are defined as locally based rewards.

Other reward function components, which are globally based are derived from the ground truth data for the training sets. As this directly depict the ground scenario, they are more important than the locally based rewards. Thus, in the training phase the fire(agent) is given an additional positive reward (+1) for taking action choices similar to the actual scenario and negative rewards(-1) otherwise. These are applied differently for different experiments as explained in the experimental setup. This additional reward helps the agent understand the influence of each variable on the dynamics of spread and weights them accordingly.

Figure 4.1 shows a schematic representation of the domain where some cells are currently affected by fire and have a potential to spread fire to other cells nearby which is treated as the 'decision' of an agent at that location. The policy for this agent tries to minimize the total error at different future steps. Here a square grid is used as opposed to a different grid like a hexagonal grid due to several reasons. The first one being, almost all wildfire simulators in existence [48] use cellular models with square grids. Also, increasing the action space will exponentially increase the computational complexity. A researcher [56] argues that for a sparse setting like an idealized forest, having more action spaces is detrimental to the model as when the flammable objects (trees, etc.) are sparse, the fire will be unable to spread over any considerable area but will have too many choices. He recommends only four cardinal directions as choices for a sparse fuel setting. We consider the testing domains in this work to be not too sparse or not too dense, hence having nine action choices seems to be ideal. Also, square grids help us to have direct comparison to cellular automata and physics based models having the same set up. Additionally, as the grids in the source data (satellite images) are square, the square grids are the most straight forward implementation for the problem.



(a) Schematic of the state and actions

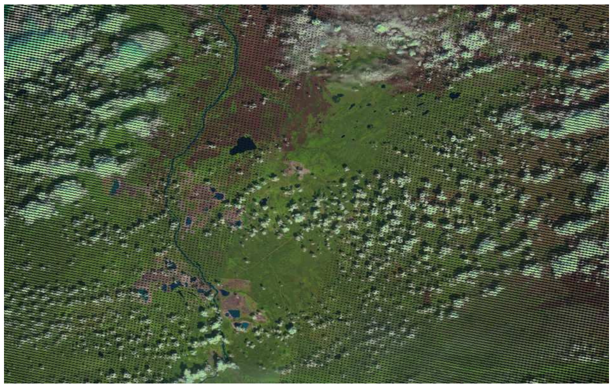
Figure 4.1: Forest Wildfire Satellite Data Domain: A schematic of the wildfire motion domain at a particular state and timestep. The red (dark) cells are on fire, the green (light) are not on fire, the dark circle indicates the current cell or agent being spread by the policy. The arrows around the dark circle indicates the action choices possible. The white circle indicates that other cells will be considered for spread. The arrows from the white circle indicates that there is a strong wind blowing towards North and the north action is the most likely action choice for these cells (effect of wind).

The initial state will come from satellite images that correspond to the beginning of a fire. We are focusing on fire growth rather than ignition detection, so we set certain cells to have just ignited fire and assign these to the initial state. As it is impossible to predict precisely from where a "fire starts", these ignited cells are decided based on thermal image data, media reports and approximations based on the burning areas in the first day on which the satellite image is available for Alberta fires.

The goal is to learn a policy for this agent(fire) that recreates the spread of the fire observed in later satellite images by maximizing discounted rewards designed to reward high accuracy simulation. Actions are constrained to not cross the boundary of the domain of study.

4.2 Experimental Domain - Alberta wild fires

Two well known fires in Northern Alberta were considered for this study. They are the Richardson fire and the Fort McMurray Fire. The Richardson fire occurred in a region called Richardson Backcountry north of Fort McMurray in Northern Alberta in 2011. Fort McMurray was also affected by forest fires in 2016 during the massive Fort McMurray fires[75]. The coordinates of Fort McMurray is ($56°43'36''$ N, $111°22'51''$ W) and the coordinates of Richardson Backcountry


(a) Satellite Image of Northern Alberta

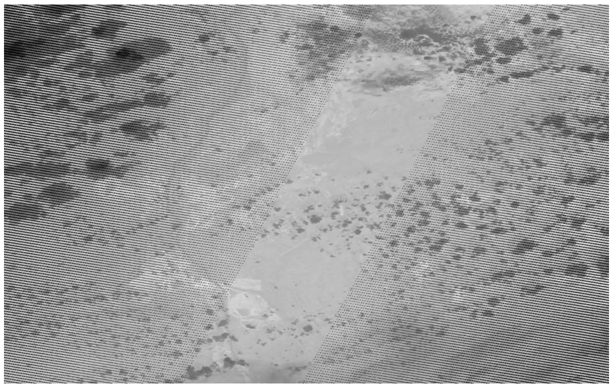

(b) Thermal Image of the same area

Figure 4.2: Satellite images of a target area from Landsat.

is ($57°22'2.3''$ N, $111°19'27.1''$ W). The satellite images are downloaded from the USGS Earth Explorer [66].

A series of visual and thermal images (Figures 4.2(a) and 4.2(b)) corresponding to the duration of occurrence of the Fort McMurray fire (approx. April 2016 to September 2016) and Richardson fire (approx. May 2011 to October 2011) were collected for the corresponding regions of Alberta. Pre-processing steps were carried out as in [16]. The image classification and matrix construction are done using steps outlined in [80]. These images gave a clear description of the ground scenario in case of fire and are a reliable source of burn areas. In the Figure 4.2(b) the areas in darker shades of black are on fire. The spatial resolution of the satellite images was 30m. This means that a pixel in the satellite image corresponds to 30 meter x 30 meter on the ground. This also means that two objects, at least thirty meters long or wide, sitting side by side, can be separated (resolved) on a Landsat image. The temporal resolution of this satellite is 16 days. Thus a 16 day periodic information of a particular area during the course of the fires are available. The Landsat Enhanced Thematic Mapper Plus (ETM+) sensor is capable of sensing in 8 different spectral bands which are made available in its data sets. Only the bands in the visible spectrum (red, blue and green) and the thermal band is used in this work. Landsat 8 imagery has a radiometric resolution of 12-bits (16-bits when processed into Level-1 data products). Temperature is obtained from processing thermal images. Wind speed, rain and wind direction are obtained from the historical data sets present in the Canada Information Portal [11] and World Clim [27] data sets for the region of study. Relative Humidity is obtained from processing satellite data sets from the USGS as explained in [52]. Land cover is obtained from the satellite images. The Canadian wildfire information system gives us information about the fireline intensity (mentioned as head fire intensity) of the fires [69].

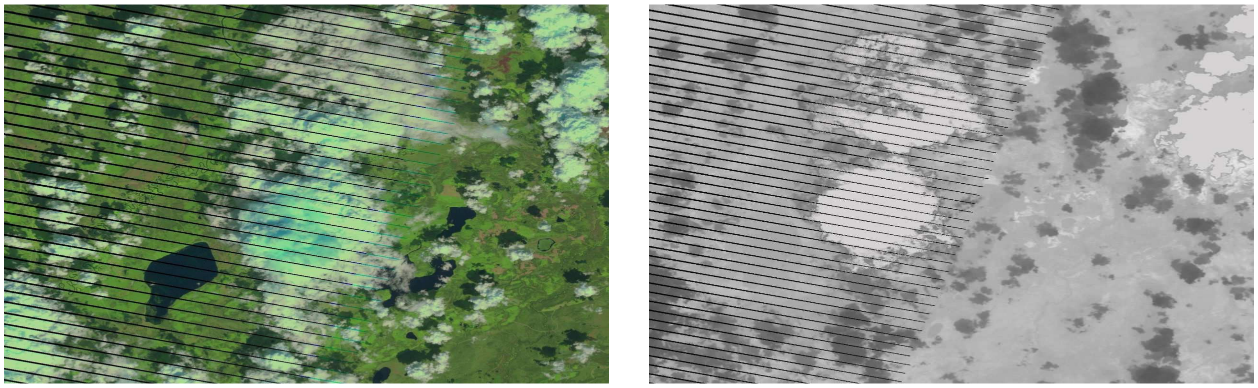

(a) Satellite Image of Northern Alberta (b) Thermal Image of the same area

Figure 4.3: Satellite images of a target area from Landsat - 2.

The data used in this study is to be trusted as the thermal data sets gives a very clear scenario for which cells are burning as the burning ones are clearly at a very high temperature as compared to its surrounding cells. The visual data sets gives a clear description of the land cover scenario. The spatial coordinates of fire spread given in more accurate data portals like the Canadian wildfire information systems, corroborates strongly with the burned locations in the Landsat images. The Landsat data continues to be used across the world by a large number of researchers for spatial applications [10], [26]. Figure 4.3 clearly depicts a scenario in which there are fire, water and clouds in the same scene. The clouds are at a fairly lower temperature as compared to smoke in these scenarios. Hence they can be separated out from the thermal image even if the visual images are a source of confusion. Though Landsat tries to give cloud free images using multiple spectral cameras and varying angles to image a scene [71], in some rare cases cloud cover is too strong and it just blocks the scene. Hence, no information can be derived. In such cases we rely on the secondary sources to get the missing information. Figure 4.3 also has some water bodies present in the scene which correctly appear as low temperature objects in the corresponding thermal image.

4.3 Experimental Setup

To evaluate this idea of agent-based RL for SSP prediction we performed a cross-comparison of the RL algorithms using a two-stage training process.

In the first stage we learn the cell-based fire spread policy directly using the satellite images from the start of the fire and the image at its next successive time step. This is the policy from the MDP which describes spread of fire from one cell to another based on local conditions. The training data images (each time step) are spaced 16 days apart, so many successive calls to this policy are required to reach the fire spread for 16 days. Thus, the second stage of training is to choose a cap C for the number of calls to this policy to obtain the correct spread area. The cap is used to set the upper limit of the total number of moves in all experiments. Previous researchers working on non single step transition RL scenarios maintain a cap for the number of moves or time [17]. During test time, once this cap on the number of moves has been reached, the conditions are reset to the initial state and a new iteration starts. This is determined based on a validation set not used for training or testing.

In the first two experiments, (A) and (B), we consider three consecutive, evenly spaced, time steps and we want to test the ability of the algorithms to predict the middle time step. The ignition points for training come from the time step 1 and the training happens using the information in time step 3. This is used to determine the fire spread at the intermediate time step 2. Thus, we learn the policy and optimize C based on the ground truth data for time step 3.

The satellite images from May, June and July 2011 of the Richardson wildfire were chosen for these experiments. The images from the beginning of May (right from before the ignition of the fire) were taken and used as the start state for RL. Experiment (A) and (B) differ only in the dates of training and test. The satellite images in June (A) and July (B) were used to determine the target burned area and so inform the reward function. The test was done on the immediate previous step (16 days earlier) of the training image. The values obtained as a result for the tests were used to obtain a value function that predicts the growth and characteristics of fire after fixed time duration. The output from the value iteration was compared to an actual satellite image from halfway through the training period in the same manner.

Note that these experiments are important as in many fire fighting scenarios the available data is made up of multiple sources with different temporal resolution. For example, highly accurate spatial spread and fire status information is obtained by drones launched regularly by the fire fighting teams. Meanwhile, satellite based data provide information about large areas, vegetation cover and conditions but images are taken much less frequently. This experiment interpolates the intermediate state information needed for fire fighters.

In the third experiment (C), we applied the policy learned on the Richardson fire to satellite images of a different fire in Fort McMurray five years later. We tune the cap value C using the first transition in the Fort McMurray data. This is a similar region of Northern Alberta (but not exactly the same area) so the general properties of the policy should be applicable but these particular images were never part of the training set. The ignition location and initial spread are taken from the satellite images of the month of May 2016. As before, the burned areas predicted by the learned policy 16 days after ignition were compared with the true satellite data. The three experiments were carried out using value iteration and the $A3C$ method. This experiment answers the question, Can the agent which has learnt the dynamics of a fire in a region, predict the spreads of a different fire in the same region?

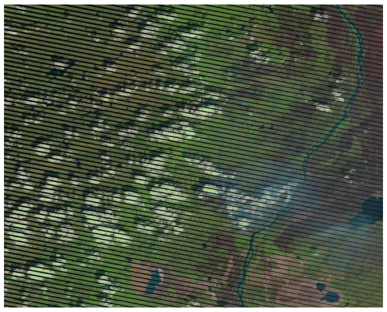
In all experiments we provide the start state (satellite image corresponding to the start date of the Richardson Fire, for experiments (A, B) and Fort McMurray fire for experiments(C-F)) to understand the spatial and landscape features of the region under consideration. All the experiments are repeated 20 times and the average accuracies are reported.

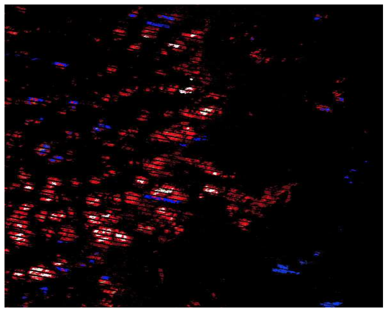
4.4 Algorithms considered

The asynchronous value iteration and the A3C algorithms were considered in this initial set of experiments. We wanted to try a classic reinforcement learning and also a deep reinforcement learning technique to note the performances. The A3C algorithm has become the most popular algorithm for most RL problems with complex state and action spaces. These two representative sets of algorithms will give us a good idea as to the advantages and disadvantages of particular classes of RL algorithms in this wild fire domain. For all of the methods each cell in the target area that is visited is marked as a burning cell and has the resulting value estimate stored in a hash data structure so it expands only as new states are encountered. The discount factor for the value iteration is set to be 0.9. For A3C, each separate instance of a fire (unconnected to other fires) in a neighbourhood is given its environment (84×84 cells as used in [43]) as an input and the fire is defined as an individual worker. Based on need in the experiments, the number of workers considered for training and testing ranged from about 40 - 96.

4.5 Initial Results

Figure 4.4(b) shows the results obtained from the experiment (A). The red in the output image corresponds to the pixels which were on fire and were classified correctly as fire by the algorithm (ie. true positives). The blue pixels represent false positives, those which were classified as burning but were not yet at that point. White pixels represent false negatives where the policy predicted no fire but fire was indeed present. Black pixels represents true negatives, the pixels which were not on fire and rightly classified. It can be seen visually that the value iteration algorithm was able to capture most of the large fire zones of the Richardson fire in the region of representation on June 8. Still some of the smaller ignitions were not captured by the value


(a) The satellite image of june 8, 2011 in the region of Alberta (Richardson Fire)


(b) Output obtained from value iteration

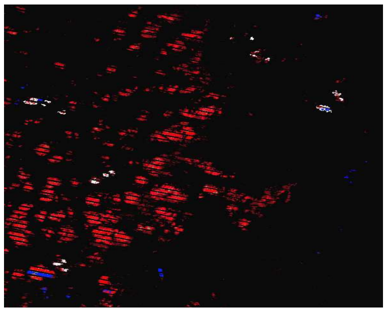

(c) Output obtained from A3C

Figure 4.4: The wildfire spread domain as a grid model representing input satellite data and output wildfire prediction. The prediction is for Richardson Fire (Experiment A).

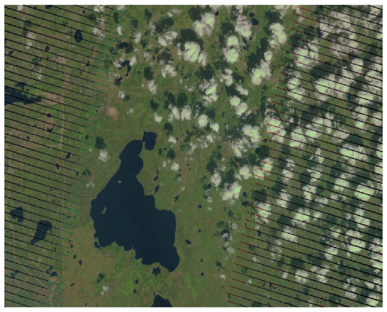
iteration. Also, the larger fires are not captured fully. Some spots in the middle and the edges of the major fires are missed by the VI algorithm.

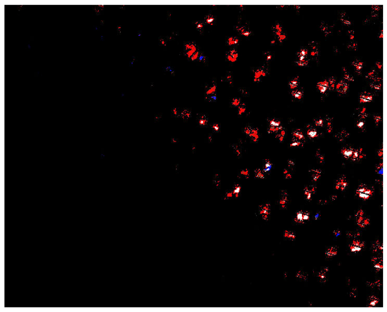
Figure 4.5 shows the results from the third experiment (C). The day of comparison was on June 14, 2016. Again we can see that the value iteration algorithm did fairly well capturing the major fire spots but could not capture them fully in some cases.

Figure 4.4(c) shows the results obtained from the A3C method. The A3C method performs slightly better than a simple value iteration as many small fire instances missed by the VI method are also being identified by $A3C$ in addition to the ones identified by value iteration technique. The comparison was done on the same day (June 8, 2011) for Richardson fire. This performance was also maintained in the case of Experiment (C). A3C method is also able to capture the major fires fully. In both cases, the A3C method also misses some small fires completely.

The Receiver Operating Characteristic (ROC) curve (plot between ratio of false positive and ratio of true positives) for experiment A in Figure 4.6 shows the effect of varying the threshold used for classifying if a pixel is affected by fire or not. Note that this is the standard technique for determining the proper design parameter for threshold based classifiers or predictors [23], [5]. The threshold was chosen to be 0.83 for the value iteration method and 0.7 for the Deep Learning $A3C$ method. This is because these thresholds correspond to high degree of true positives with only a small number of false positives.

Table 4.1 presents the accuracy (as a percentage) obtained in both the methods across different experiments. Columns (A) and (B) test the policy's prediction on the same fire event, where the test point is 1 month or 2 months after ignition, where the reward comes from a point a month


(a) The satellite image of june 14, 2016 in the region of Alberta (Fort McMurray fire)


(b) Output obtained from value iteration

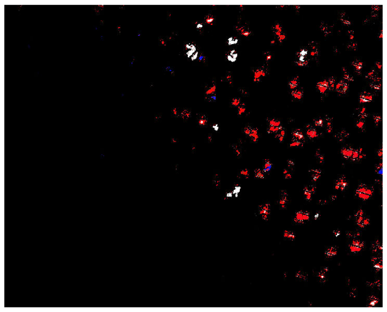

(c) Output obtained from A3C

Figure 4.5: Comparing the output from value iteration to the actual scenario in case of Fort McMurray fire (Experiment (C).

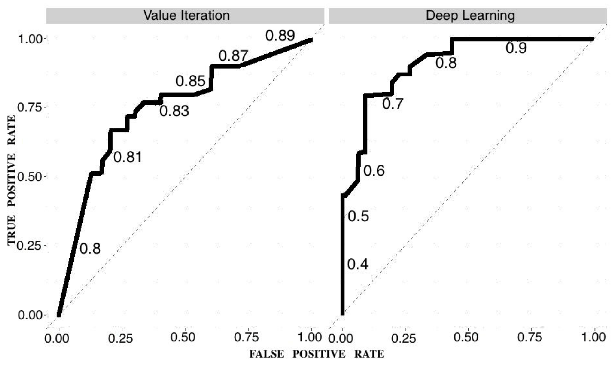


Figure 4.6: ROC curve for analyzing sensitivity of policy to classification threshold parameter.

Method	(A) Richardson(1st Month)	(B) Richardson(2nd Month)	(C) Fort McM
Value Iteration	82.1%	79.2%	88.5%
A3C	89.5%	85.7%	90.1%

Table 4.1: Average Accuracy of each algorithm on the three different test scenarios.

later. Column (C) is testing the policy's prediction on a completely different fire five years later, in the same region, predicting a fire one month after ignition. The maximum standard deviation from the average accuracies for all the experiments was 1.3% for Value Iteration and 3.4% for the A3C method upon repetition.

4.6 Summary

In this chapter we explained the formulation of the Markov Decision process for the forest fire spread problem and demonstrated that a simple agent based model of fire can obtain reasonable fire spread predictions using value iteration and that this can be significantly improved using a Deep RL representation. In the subsequent chapters we will perform a wider range of comparisons on a more comprehensive set of RL algorithms.

Chapter 5

Comparison to Tree search and Supervised Learning Algorithms

This chapter extends the range of algorithms considered on the forest fire problem pertaining to the Alberta fire domain discussed in the previous chapter. The baseline used for comparison is the Gaussian Processes supervised classification algorithm. The Monte Carlo Tree Search (MCTS) which is a tree based RL algorithm is also considered in this chapter. Experiments are modified and expanded from the previous chapter to include additional scenarios.

Note: This chapter is To appear as a paper in the Frontiers Journal of ICT - Environmental Informatics, 2018.

5.1 Algorithms considered

In this expanded set of experiments more algorithms are considered. The algorithms considered include Monte Carlo Tree Search (MCTS), Value Iteration, Policy Iteration, Q-Learning, Gaussian Processes and A3C. As Value Iteration was noted to have some advantages in predicting large fire occurrences, an expanded set of classic algorithms that include model based approaches like Policy Iteration and model free approaches like Q-Learning are considered. A tree based approach like MCTS is considered to try out a new class of algorithm and also as Monte Carlo simulations have proven to have advantages in physics based simulators [50]. The supervised technique Gaussian Processes (GP) was used to provide a baseline for comparison. The Deep Learning scheme A3C is retained from the previous chapter to see its performance on an expanded set of experiments. The discount factors for the classical algorithms was 0.9. This makes the globally based rewards matter more than the locally based rewards. The learning rate for Q-Learning was also 0.9. In our implementation of MCTS each node in the search tree is a valid cell state in the fire model. From each state any possible action could be taken, which is modelled as branches in the tree.The nodes in the tree are made to be selected by the **Upper Confidence Bound for trees (UCT)**[34] method to minimize the cumulative regret. Each "step" in the MCTS tree is defined as a possible action taken from any valid state. There is a stay action defined as a legal action in our scenario. Thus, in this implementation it is possible to go a few levels down the tree without that corresponding to any real action in the world. Rewards are given based at each step of the roll-out and when complete the combined reward is used to update the value for the initial state at the root of the roll-out tree. For computational simplicity, during the roll-out a simpler state representation is used which focuses only on number of cells burning. The algorithm is configured to stop after the desired time comes to an end for the experiment of consideration. Also, if the intensity of fire at a cell falls below a particular threshold (different for different experiments and empirically determined), it is set as ceased to burn and is stopped from spreading further until it is reignited. This is done to keep the branching factor down.

5.2 Experiments

In this chapter we consider different and extensive sets of experiments compared to the previous chapter.

The experiment (A) is similar to the experiments (A) and (B) in the previous chapter. The only difference being many such three consecutive pairs in the Richardson fire are considered and the average performance for the algorithms are reported.

For Experiment (B) of this chapter, we start with the initial state of the fire, providing rewards based on the time step 2 of the fire and the algorithm predicts fire locations at time step 3. This experiment is similar to asking the question: Where will the fire spread in the next 16 days, given its position currently? This experiment pertains to the Richardson fire. Similar to experiment (A) of this chapter, many such 3 consecutive pairs are considered and the average performances are reported. For both experiments (A) and (B), about five pairs are considered.

We also extend experiment (C) from the previous chapter to three more experiments (D), (E) and (F). Each one pertains to application of the learned policy from the Richardson fire to the Fort McMurray fire over four 16-day time steps in the Fort McMurray data after giving the ignition points as the input. The time of comparison is the only difference between each of these experiments. Again, all experiments are repeated 20 times and the average values are reported.

Experiment	VI	PI	QL	MCTS	A3C
A	0.78	0.93	0.72	0.80	**0.96**
B	0.70	0.72	0.67	**0.82**	0.80
C	0.82	0.81	0.80	0.71	**0.90**
D	0.75	0.74	0.72	0.69	**0.87**
E	0.64	0.66	0.62	0.67	**0.77**
F	0.54	0.53	0.55	0.60	**0.63**

Table 5.1: Average AUC for all the methods. The best values for all experiments are in bold.

5.3 Results

The task in these experiments is essentially to classify each cell at each time step correctly in terms of burning or not burning. The policy learned by the fire gives the direction of the fire spread. The fire as an agent in any RL domain has the goal of increasing its overall utility accumulated. In this regard, it may decide to take low utility moves, which could either be the best move in the given context or may lead to higher utility moves subsequently. As the reward function (and hence utility) is strongly determined by the cell flammability, low utility cells have low/no chances of burning. This is taken into account during the process of determining if a cell burns or does not burn on a landscape.

We use the validation set to determine true positive rate and false positive rate and compute a receiver operating curve (ROC) and a corresponding area under the curve (AUC) metric as done in the previous chapter. This helps us in choosing a threshold for the utility function to determine burn/not burn. The Average AUC for all experiments and algorithms are shown in Table 5.1. This shows that the A3C algorithm provides an excellent threshold for most experiments while classic algorithms are not so favourable. This seems to augur well with the overall accuracy for individual experiments seen in Table 5.2. The ROC curves for a particular case of experiment (A) is shown in Figure 5.1 which corroborates with the AUC values seen in Table 5.1 and gives an idea about the thresholds.

Figures 5.3 and 5.4 show the visualization of some of the results for the different experiments. For all the images the color scheme is same as that followed in Figure 4.4(b)

Analyzing the average accuracies in Tables 5.2 and 5.5 we can see that, from all the algorithms considered, A3C seems to be the best overall, which is not surprising as it has the most flexible state-action value function representation. Policy Iteration in general comes second best in most tests. Q learning a model free method of learning gives a lesser accuracy than model

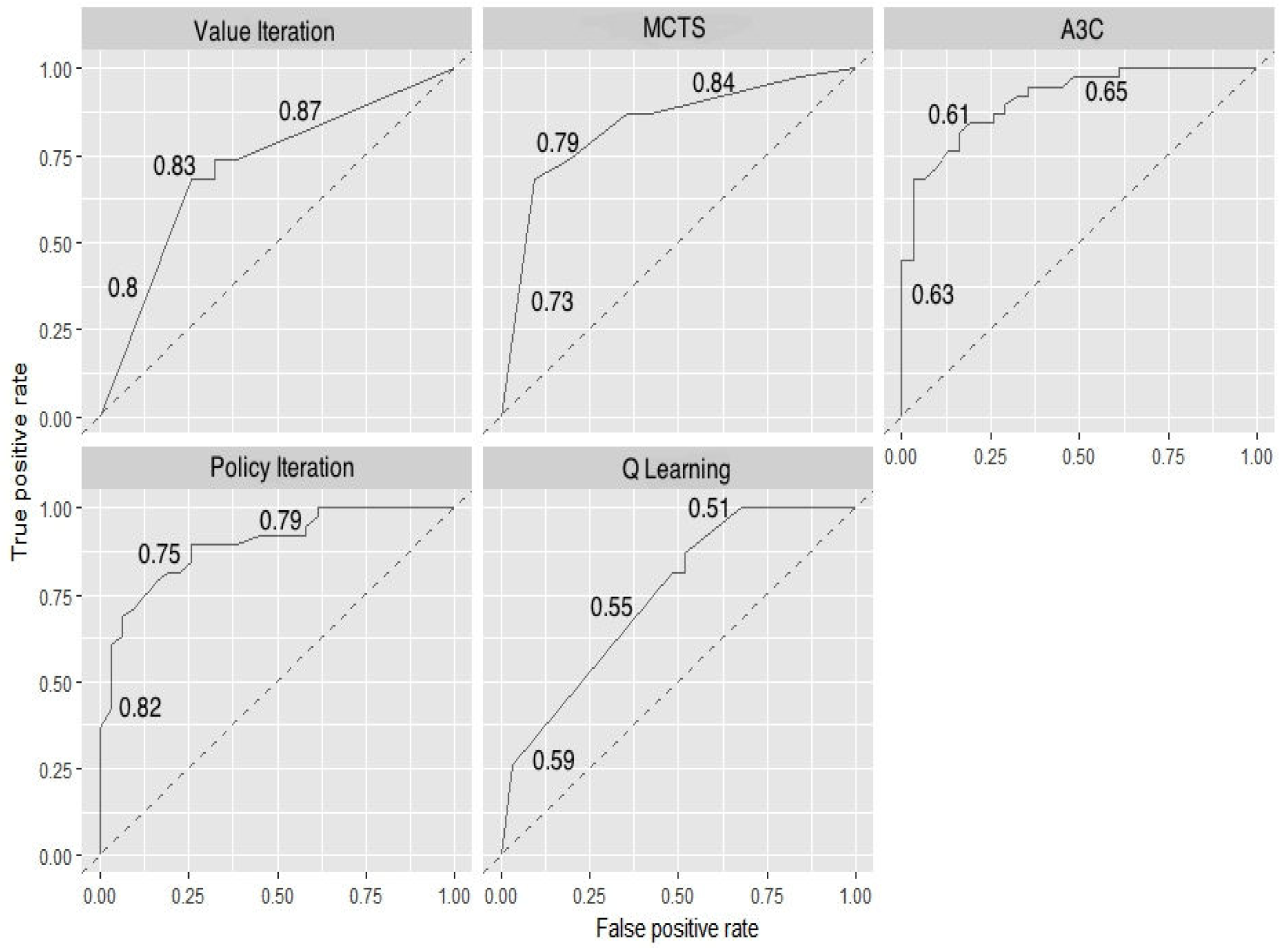


Figure 5.1: ROC curves for Experiment A as a representation.

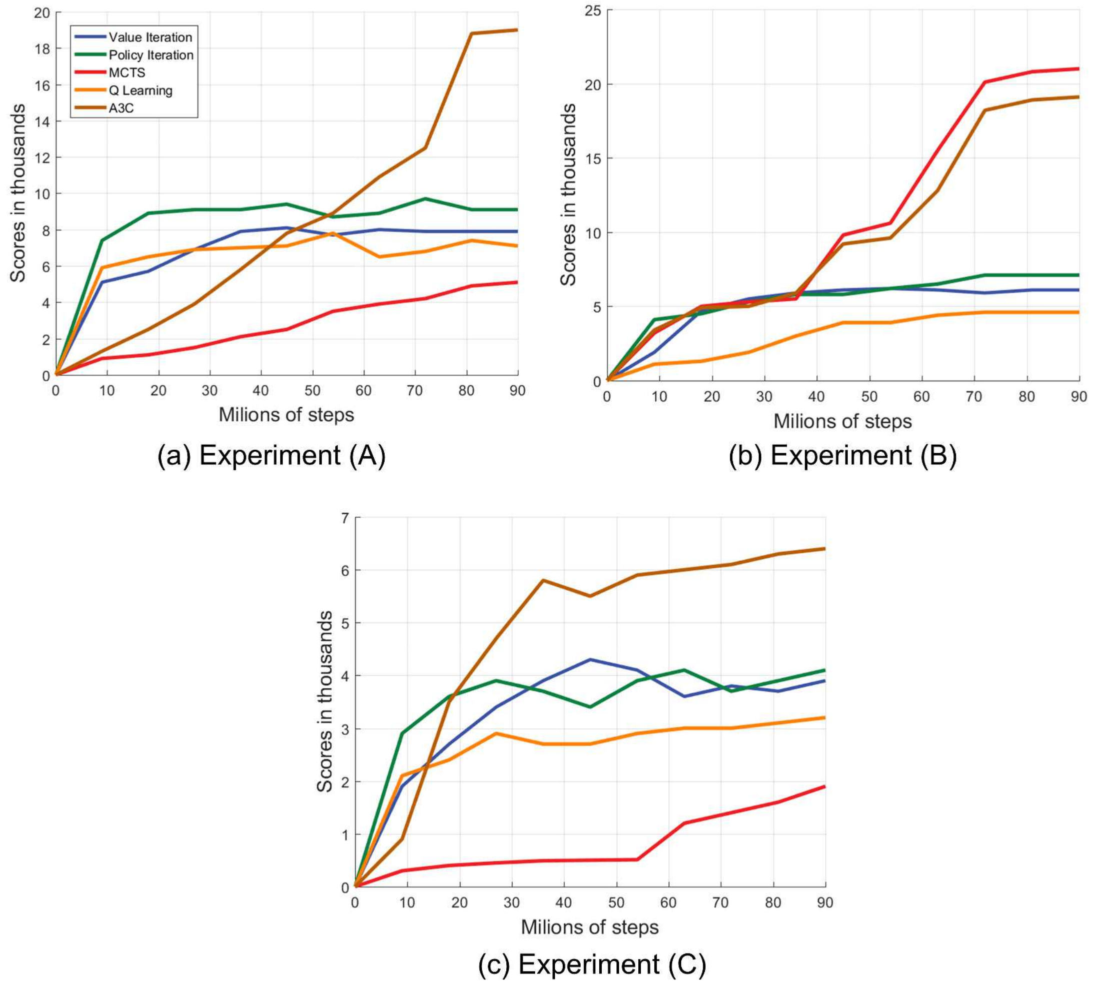
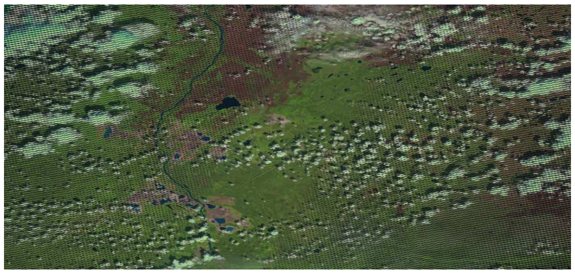
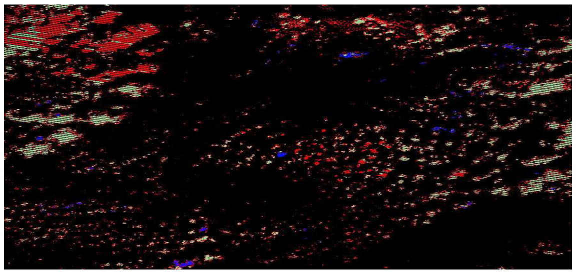
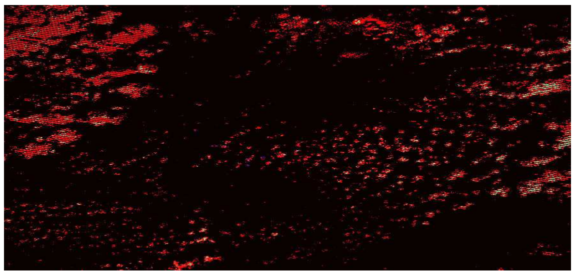
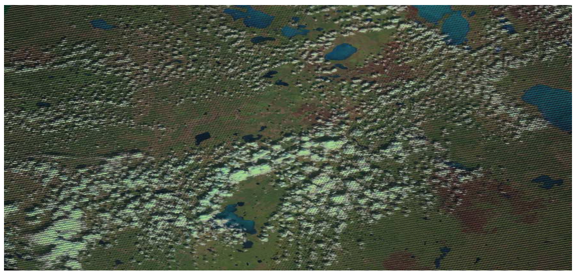


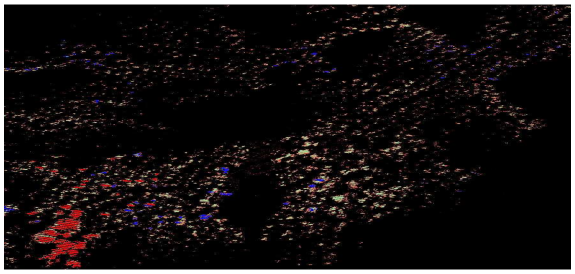
Figure 5.2: Run Time plot for all the experiments. All the curves share the legend as in (a). Each step is a permissible action taken by the corresponding algorithm. The Y axis denotes the sum of rewards accumulated by the agent (score) after the corresponding number of steps (X axis).


(a) Satellite Image of July 26 - Experiment A


(b) MCTS for experiment A


(c) A3C for Experiment A


(d) Satellite image of April 27 - Experiment C


(e) MCTS for Experiment C

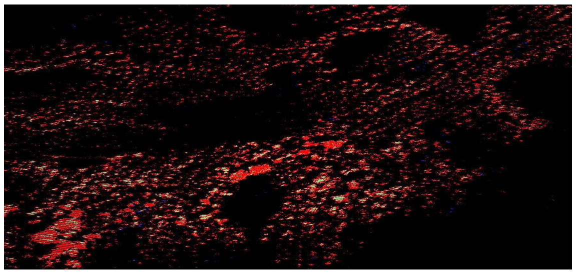
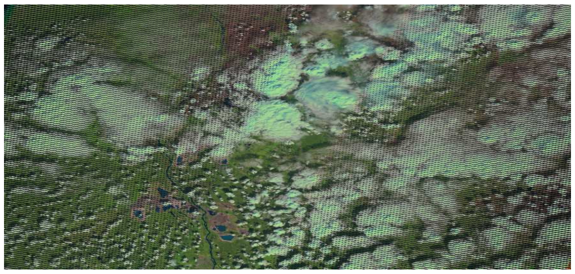
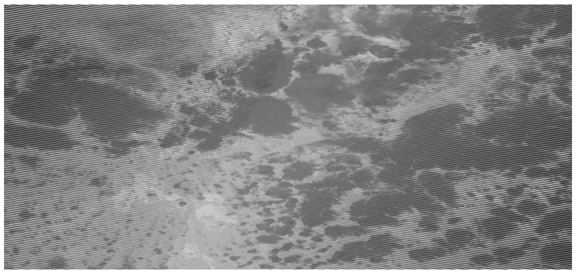
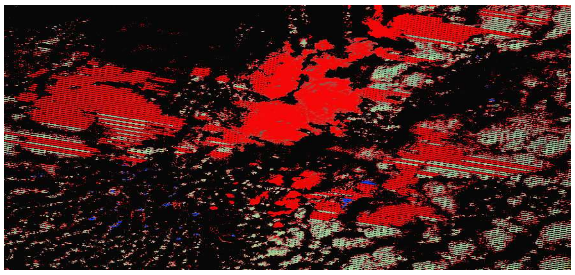
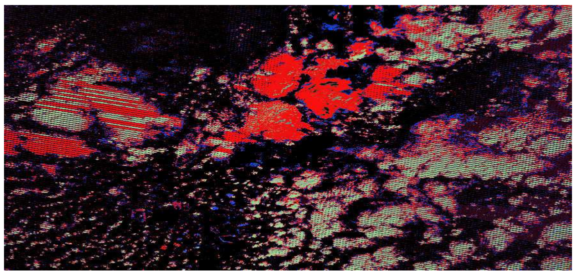

(f) A3C for Experiment C

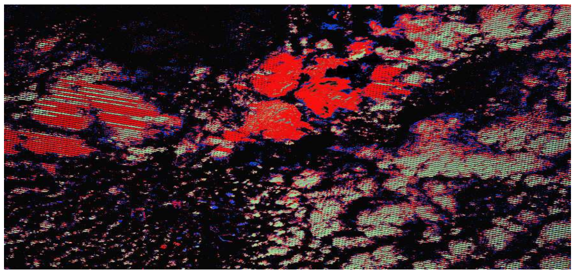
Figure 5.3: Results for experiments from the best two algorithms MCTS and A3C for filling in an (intermediary state) in the training region (A) and applying that policy in a different region (D). Red pixels were on fire and classified correctly (True Positives), blue pixels were incorrectly classified as burning (FP), white pixels were incorrectly classified as not burning (FN) and black pixels were correctly classified as not burning (TN).

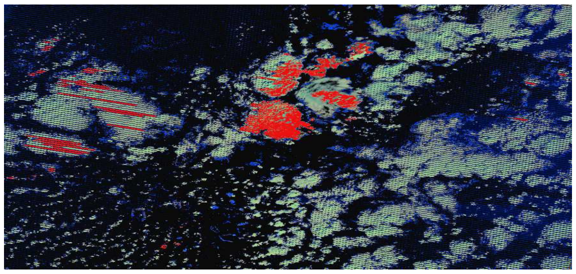

(a) Satellite Image of August 11

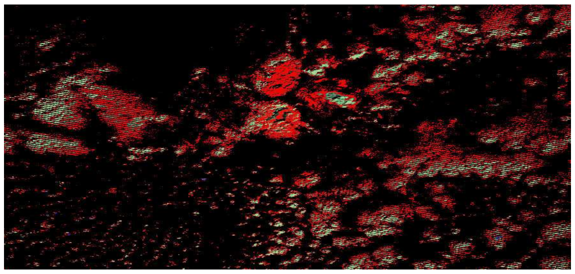

(b) Thermal Image of August 11


(c) Gaussian Processes


(d) Value Iteration


(e) Policy Iteration


(f) Q Learning


(g) MCTS

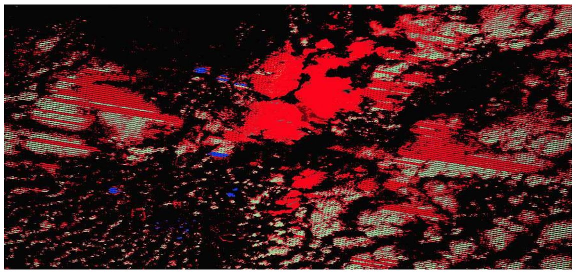

(h) A3C

Figure 5.4: Results for Experiment (B) from all algorithms showing performance on the prediction of the next state directly after the training data.

Method	Rich. Fire(A)	Rich. Fire(B)
GP	62.4%	50.8%
VI	72.2%	25.4%
PI	73.3%	38.2%
QL	67.2%	10.4%
MCTS	61.3%	**60.2**%
A3C	**87.3**%	53.2%

Table 5.2: Average Accuracy for each algorithm on the different test scenarios. The highest value in each experiment are bolded.

Method	Rich. Fire(A)	Rich. Fire(B)
GP	0.674	0.538
VI	0.752	0.287
PI	0.755	0.412
QL	0.693	0.144
MCTS	0.654	0.652
A3C	0.901	0.563

Table 5.3: Average Precision for each algorithm on the different test scenarios.

Method	Rich. Fire(A)	Rich. Fire(B)
GP	0.604	0.486
VI	0.655	0.204
PI	0.684	0.334
QL	0.652	0.096
MCTS	0.573	0.541
A3C	0.811	0.486

Table 5.4: Average Recall for each algorithm on the different test scenarios.

Method	(C)	(D)	(E)	(F)
GP	60.5%	47.9%	45.3%	**20.5** %
V.I	88.5%	68.4%	30.1%	6.4%
P.I	89.3%	67.8%	35.8%	8.9%
Q.L	84.2%	61.4%	26.4%	5.3%
MCTS	65.3%	55.7%	49.7%	5.8%
A3C	**90.1**%	**81.8**%	**50.8**%	13.4%

Table 5.5: Average Accuracy of each algorithm trained on the Richardson Fire but applied on the Fort McMurray fire for different time duration. The highest value in each experiment are bolded.

Method	(C)	(D)	(E)	(F)
GP	0.663	0.512	0.532	0.278
V.I	0.925	0.719	0.374	0.091
P.I	0.941	0.691	0.399	0.112
Q.L	0.876	0.654	0.301	0.071
MCTS	0.711	0.601	0.532	0.089
A3C	0.942	0.841	0.542	0.182

Table 5.6: Average Precision of each algorithm trained on the Richardson Fire but applied on the Fort McMurray fire for different time duration.

Method	(C)	(D)	(E)	(F)
GP	0.541	0.314	0.342	0.132
V.I	0.731	0.575	0.231	0.032
P.I	0.702	0.589	0.289	0.075
Q.L	0.755	0.561	0.213	0.032
MCTS	0.491	0.423	0.395	0.024
A3C	0.735	0.723	0.402	0.078

Table 5.7: Average Recall of each algorithm trained on the Richardson Fire but applied on the Fort McMurray fire for different time duration.

based approaches like Value and Policy Iteration. In a spatial domain the environment has a high influence on the spread of the agent. Thus, the strong consideration of the model of the spatial environment helps the model based approaches.

The output images in Figure 5.4(g)(h) show the two most successful algorithms MCTS and A3C on the region on which they were trained, filling in fire predictions between the start state and the reward target. Looking at the output images in Figure 5.3(e)(f) we see these two algorithms applied to the Fort McMurray Fire (similar region, different year and not part of the training data) at the same number of days forward after the start state. We can see that the model trained on one set images applies quite well to a different start state it has never seen, but A3C has much fewer false positives than MCTS. Figure 5.4 visualizes the results of all the RL algorithms on predicting the next time step after the training data on the Richardson fire, here A3C clearly produces the most accurate prediction.

For all the average accuracies reported in Tables 5.2 and 5.5 the maximum standard deviation upon repetition for all algorithms was about 5%.

Somewhat surprisingly, MCTS outperforms A3C in experiment (B) for forward prediction of the next 16 day period on the same location as the training data. In general this experiment seems to be harder for most algorithms as one wrong move, while rolling forward misses a chunk of burned cells and completely moves in the wrong direction. Also, as a commonplace prediction is inherently harder than interpolation. All the algorithms struggled to get the direction of burn right in this experiment especially in the later stages of the fire. It seems that the MCTS rollouts have fit a better model to the sense of real world ”time” to predict the extent of a fire in the next time period. As can be seen for the results for (C) and (D) in Table 5.5, the A3C algorithm outperforms all others on prediction when the learned policy for one location is applied to another location in the same region. MCTS does significantly worse in this challenge, meaning its model is not as good at generalizing the essentials of the policy for transfer to a new location even though it seems to have a better model of dynamics.

Note that the accuracy of all algorithms reduce with increasing time after the start of a fire as can be see by looking at experiments (C)-(F) in Table 5.5. This is unsurprising, as predicting steps into the future is inherently harder. This is due to the fact that later fires are larger, more intense and spread faster, making learning more difficult. In the first time step all algorithms do very well as it is easier to predict the first few moves of the fire. The accuracy goes down rapidly and in the last time step it is only around 10%.

It is also interesting to note that even though MCTS starts of with a low accuracy compared to the other methods in the first time step, its cautious approach causes its accuracy to decrease much slower than other methods so that it seems tied with A3C when predicting three steps ahead. Similarly, the supervised GP method which loses out in most comparisons does far better for the hardest problem of predicting four time steps (about two months) ahead after all other methods have degraded. The GP approach is purely spatial pattern learning, with no element of dynamics. So one possible explanation is that, GP's with only a spatial model and no notion of time loses out when changes over time are relevant, but once we are so far ahead after the start of the fire a notion of time actually proves to be detrimental as the behaviour of a very intense fire is difficult to correlate with time and the GP's more accurate spatial model wins out. Researchers claim that it is possible to predict the next month into the fire season with the state of the art tools [51]. A research paper[48] on state of the art wildfire simulators mentions several disadvantages for each simulator as experiments are conducted into the fire season. Some of these tools were used in the same experiments as the Fort McMurray fire, where we obtained an average accuracy of about 20% when we are predicting one-two months into the fire season. The experiment (F) was actually a bit of a stretch for the RL algorithms too as it considers 2 months into the fire season, but we still conducted the same to understand relative performances.

Turning our attention to the running times (see Figure 5.2) we ran all experiments on similar scale problems running with the same system resources and training took between 45 - 50 hours for all algorithms. The average of total reward obtained (score) for every 9 million steps is plotted for every algorithm and experiment. Here, each step is defined as a distinct possible action taken by the algorithm. After observations for about 90 million steps, further observations are made. It is important to note that many researchers in the Reinforcement Learning domain publish runtime plots in terms of steps rather than time taken as it provides a better idea of performances ([7], [4], [47]). We have also followed the same approach here. There was almost a uniform standard deviation in scores of about 2000 for Experiment A and B and about a thousand in experiment C for all algorithms. As expected, the deviations, increase towards the end of the plot after many steps are taken. For Experiment A, we can see that the A3C algorithm is slower to start of as compared to the non deep algorithms, as expected, but over many number of steps it beats all other algorithms which flatten out early. MCTS does not flat out, but is painfully slow in improving itself. The trend is similar in Experiment C. For Experiment B, MCTS and A3C start of slightly slowly compared to the others but perform very well overall. In this case MCTS beats A3C after about 50 million steps. One unique aspect of A3C is that it can exploit multithreading of CPU cores rather than GPU acceleration. Each fire in our dataset could be run as its own thread allowing A3C use similar training time to obtain superior results. MCTS was the slowest algorithm we tried since it is not multithreaded and requires extra roll-out simulations and back propagation at every iteration.

The precision and recalls of all the experiments are also reported in Tables 5.3, 5.6, 5.4 and 5.7. The trend seems to be that the recall measures are lower than accuracy values and precision measures are greater than accuracy values, which shows that the false negatives are much greater than the false positives for most algorithms. This shows that the algorithms are cautious in predicting spread in general.

5.4 Summary

In this chapter, we considered the performances of different RL algorithms on the Alberta historical wild fires. Our results indicate that A3C is better at predicting spread dynamics at intermediate time steps and MCTS performs better while predicting the future spread. As the test data diverges from the training data and temporal changes become less relevant, such as when the intensity of fire increases, the fully supervised Gaussian process approach performs better than the RL algorithms. In the next chapter we consider additional testing domains and introduce a new algorithm that is a unique combination of preexisting RL algorithms.

Chapter 6

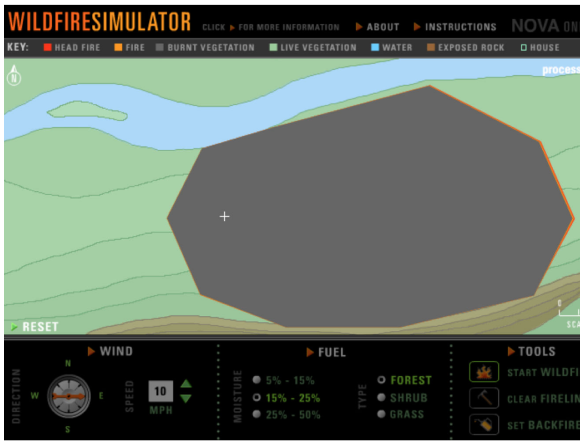
Combining the best of two worlds: MCTS-A3C Algorithm

From the results of the previous chapter, we saw that MCTS and A3C had distinct advantages on the forest fire domain. This gives us an intuitive idea to combine the advantages of both the algorithms and implement a new one which is expected to perform better than its sub components. We introduce a novel RL algorithm based on the previous results and also validate it on additional domains from both synthetic and real world data. In this chapter, we extend the experiments to include additional domains. One domain is an online simulator of a fire spread and the other is historical fires in Saskatchewan which has been compared to a physics based simulator Burn Probability 3 (BP3).

Note: This chapter is To Appear in the proceedings of the 30th Canadian Conference on Artificial Intelligence, Toronto. May 2018.

6.1 Testing Domains

In addition to the Alberta fire testing domain, two other testing domains are considered. The first one is the simulated wild fire domain, which does not belong to real fire events like the previous chapters, but simply simulates the spread of fire based on preset conditions. This domain is interesting as we can analyze performances on a cleaner set of input data set without the problem of pre-processing errors on satellite images. The other domain deals with a series of historical fires in Saskatchewan which has been used to analyze the performances of a physics-based simulator (BP3). The RL algorithms are implemented on the same data sets and their


WILDFIRESIMULATOR
CLICK ▶ FOR MORE INFORMATION
ABOUT
INSTRUCTIONS
NOVA ON
KEY: 🟥 HEAD FIRE 🟧 FIRE ⬛ BURNT VEGETATION 🟩 LIVE VEGETATION 🟦 WATER 🟫 EXPOSED ROCK □ HOUSE
N
process
RESET
WIND
DIRECTION
N
W
E
S
FUEL
TOOLS
START WILDFI
CLEAR FIRELIN
SET BACKFIRE

SPEED 10 MPH

MOISTURE ● 5% - 15% [✓ 15% - 25%] ● 25% - 50%

TYPE [✓ FOREST] ● SHRUB ● GRASS

Figure 6.1: Image of the fire simulator from the Nova

performances are compared against that of the physics based simulators.

6.1.1 Simulated wild fires

For clean images and as a faster testing domain, we are using the wild fire simulator by Nova online [25]. This simulator is derived from Farsite which is a professional fire-modeling application [20]. The original simulator also takes into account the role of fire lines which is disabled by us as we are not studying the role of fire lines in this set of experiments. The environmental variables like wind and vegetation (fuel) can be preset by the user with the controls in the simulator. We used a variety of environment variable settings to create a realistic set of simulations. The fire spread with the given conditions is trained and tested with RL algorithms. Figure 6.1 shows the simulator at work. All the necessary state variables are obtained from the simulator outputs in this case.

6.1.2 Saskatchewan wild fires

A comparative study was made to demonstrate the performance of RL algorithms against the physics-based simulation in Burn-P3 (BP3). Physics-based simulations are the current state of the art in wild fire analysis literature. BP3 is a physics-based spatial fire simulation model which uses Prometheus fire growth engine to simulate the growth of the fire [18]. BP3 is capable of accepting different kinds of inputs that influence the spread of fire like weather, topography and fire ignition patterns [13]. The work by [50], which uses the BP3 for analysis of historical fires in central Saskatchewan ($54°26'55.0''$ N, $106°1'38.6''$ W) was taken as the basis for comparison. The data for the analysis in this paper is described in detail in [49]. The same data was used to analyze by us but with some exceptions. The ignition of the fire was given as geocoded inputs as against ignition grids mentioned in the above work. Also the Saskatchewan Environment Fire occurrence database was not used as this source is described to be unreliable [49] and the Canadian large fire database provides a reasonable alternative. For the experiment with Saskatchewan fires, vegetation rewards ranged from 0.7 to 0.1 based on fuel type flammability as mentioned in Canadian Fire Behavior Prediction system [72]. The data for the state variables come from the data sets described in [49] combined with the exceptions mentioned. Approximate ignition points are also present for Saskatchewan fires. The work of the paper [50] was reproduced using the minor modifications in the data with BP3. BP3 uses a large number of Monte-Carlo Simulations (called as iterations) to make the Burn Probability map. This motivates the use of MCTS in the RL algorithm. The burn probability in BP3 is calculated using the following formula

$$BP_i = b_i/N * 100$$

where b_i is the number of Monte Carlo Simulations needed to make cell i burn and N is the total number of iterations. This value represents the likelihood that a given cell will burn given the inputs like landscape, ignition, weather etc. The BP3 separates the burn probabilities into 7 regions with the burn probabilities increasing from 0 to 3. The ranges used in [50] to represent the burn probabilities were 0, 0-0.6, 0.6-1.2, 1.2-1.8, 1.8-2.4, 2.4-3, >3 .

6.2 Algorithms considered

In this series of experiments we considered the DQN, DQN with prioritized experience replay, A3C, MCTS, Gaussian Processes and the newly implemented MCTS-A3C algorithms. We are considering more Deep learning techniques in this chapter as we found that the performance of the classic algorithms were not as promising compared to the Deep Learning schemes like A3C. The classic ones like Value Iteration, Policy Iteration and Q Learning are left out in this chapter. MCTS seemed to have advantages in a particular experiment, so it is retained to analyze performances in an expanded set of testing environments. The DQN network and algorithm was released by Deep Mind [44] and it formed the basis for the subsequent set of Deep RL algorithms including A3C [43]. So these algorithms are also considered in this chapter. Again, GP was considered to provide a baseline like the previous chapter.

6.3 MCTS-A3C algorithm

This algorithm was coined to combine the advantages in physics based simulations (MCTS [50]) and RL approaches (A3C [65]) for analyzing spatially spreading processes. The approach is experimentally validated on fire spread prediction. The high level pseudo code is given in Algorithm 7. There is an MCTS search tree that contains an average reward value (X) and a visit count (N) for all the nodes. Each node is a cell on fire and has its own state space S. The fire is made to start at the ignition points and each node of the search tree comprises of a cell burning in the resolution of the source data (30m in case of Alberta fires and 300m in case of Saskatchewan fires). The selection step in the algorithm is given by the UCT strategy. To encourage exploration, random selection is also done with probability ϵ. The root node of the tree contains the ignition point and an action choice (selection step) leads to another cell being on fire, and this leads to child nodes containing the new ignited cell on fire. This continues till we reach the end of the tree with the visited nodes (leaf of the visited node tree). In the next expansion phase an unvisited node is randomly added to the search tree and we simulate forward using the A3C algorithm. The initial state for A3C gets 84*84 cells (as followed in [45]) surrounding the center ignited cell as its start state. Each worker of A3C is defined as an instance of fire with a distinct environment (84 *84 cells surrounding it) and related networks. The algorithm spins off a separate worker on a distinct thread every time a new flame is started in the search space (for every legal action of the preexisting fires).Then each worker propagates fire in its own local condition, obtains rewards and updates the global network. Next, the back propagation step increments the visit count and updates the reward values of each node. The algorithm is confined to stop after training happens for sufficient time (which is fixed as a constant for all algorithms). As discussed in the MCTS implementation, if the intensity of fire at a cell falls below 0.3, it is empirically determined to cease to burn and is stopped from spreading further (in case of A3C) and hence is removed from the search tree (in MCTS) until it is reignited. This is done to keep the branching factor down. The model architecture for A3C follows the scheme in [45].

6.4 Experimental Setup

All the experiments (A) - (F) are the same as ones carried out in the previous chapter. The statistical measures from the similar algorithms in the previous chapter are repeated in the results for readability. In experiment (G), we use the Nova simulator to generate wild fire data and predict forward based on the given situation. This is similar in approach to experiment (B) but uses cleaner data samples. Experiment (H) uses the Saskatchewan fire data from 1981 to 1992 for training the fire spread model. The fire spread for given ignition points were then analyzed

Algorithm 7 MCTS-A3C

```
procedure VOID MCTS-A3C(NODE ROOTNODE, TIME T)
    assign Time = 0, Time is tied to the system clock and tracks the total time the program is
    being executed.
    while Time $<=$ T do
        currentnode $\leftarrow$ rootnode
        while currentnode!=leafnode do
            lastnode $\leftarrow$ currentnode
            currentnode $\leftarrow$ SELECTION(currentnode)
        end while
        lastnode $\leftarrow$ EXPANSION(lastnode)
        SIMULATE using A3C
        while currentnode do
            Backpropogate(currentnode)
            currentnode $\leftarrow$ parent(currentnode)
        end while
    end while
end procedure
```

for the years from 1993 to 2002. We also compare to existing results using burn probabilities for 1993 from [50]. In the same way, Experiment (I) uses data from 1981 - 2002 and tests for 2003 - 2008. This is done to test the performances on an experiment having more training samples and less prediction spreads. For both experiments (H) and (I), the RL algorithms and BP3 were run on a 300m resolution cells as this was the lowest resolution of data source grids (except ignition grids which are not used) for the cell based inputs in the experiments carried out in [50], and due to limitations in computation time. As done in previous chapters, all the experiments with each algorithm was repeated 20 times and average values are reported in the results.

In our experiment, the results from BP3 for every cell were translated into a binary with the last three ranges characterized as a burn and the first four ranges characterized as a no-burn. The number of iterations for BP3 were fixed to be 500 as suggested in the paper [50]. From analyzing the results from the BP3 we found that categorizing the middle range as a no-burn gives higher accuracy than categorizing it as a burn. The results of this experiment will be used to indicate the performance of RL against the physics based simulator.

We have compared the performance of our combined MCTS-A3C algorithm to the A3C, MCTS, DQN algorithm [45], DQN with prioritized experience replay (PER) [58] and also the Gaussian Process supervised classifier. The discount rate γ was 0.9 as the goal is to make the

Experiment	MCTS	A3C	DQN	DQN(PER)	MCTS-A3C
A	0.80	0.96	0.79	0.80	**0.97**
B	0.82	0.80	0.75	0.75	**0.91**
C	0.71	0.90	0.85	0.85	**0.91**
D	0.69	0.87	0.70	0.73	**0.89**
E	0.67	0.77	0.62	0.69	**0.87**
F	0.60	0.63	0.61	0.65	**0.75**
G	0.88	0.84	0.87	0.88	**0.92**
H	0.69	0.83	0.75	0.80	**0.88**
I	0.76	0.83	0.79	0.85	**0.89**

Table 6.1: Average AUC for all the methods

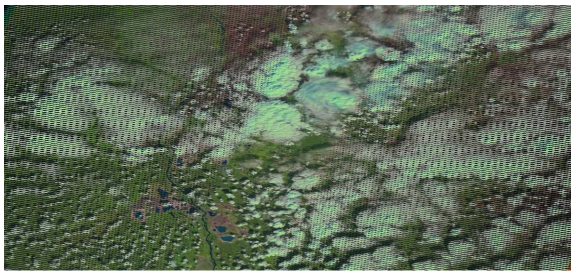
agent strive for a long term reward and the exploration for the ϵ greedy policy was kept as 0.05, which is on the lines of exploration rates in previous works ([43], [45]). An hour of training yields an average of 90000 observations across all algorithms and experiments. Training of algorithms happen with inputs time stamped with the number of days from the start of the fire season. Algorithms are forced to stop when time goes above a pre-defined time for test cases (based on experiments).

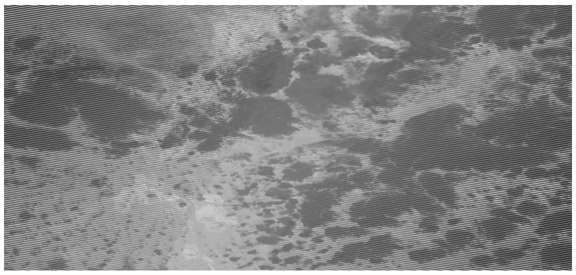
For all the experiments, to determine if a cell was on fire, a threshold value beyond which a cell must burn was determined for the value function to balance true positives and false positives. Similar to the previous chapter, the area under the ROC curves are tabulated in Table 6.1 for each algorithm for all the different experiments. The AUC values seen in Table 6.1 corroborates well with the accuracies seen in Table 6.2 in general. The result from the algorithms in the form of burned areas are compared against the actual scenario, using the satellite images or simulator output corresponding to the predicted time step.

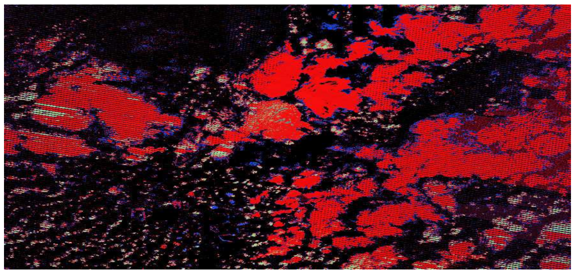
6.5 Results

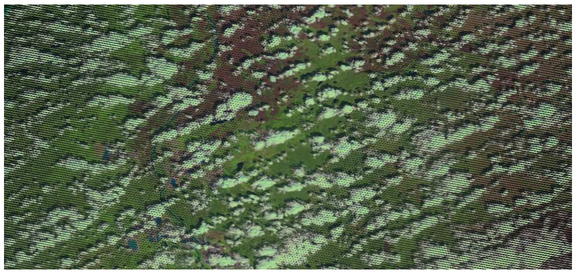
For all the images in Figure 6.2, the color scheme is same as that followed in Figure 4.4(b).

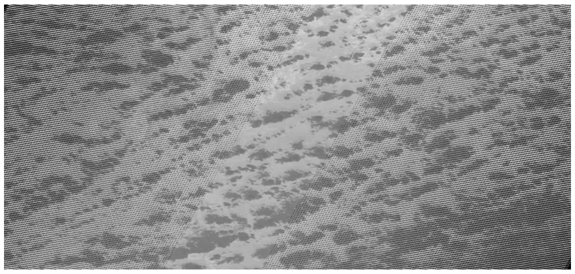
Referring to Table 6.2 we can notice some general trends in the accuracies for all the experiments from (A) to (I). MCTS-A3C algorithm beats all the remaining algorithms in terms of the accuracy. The general rule seems to be MCTS-A3C $>$ A3C $>$ DQN with PER $>$ DQN $>$ GP $>$ MCTS with some exceptions. The deep learning based approaches perform better than MCTS

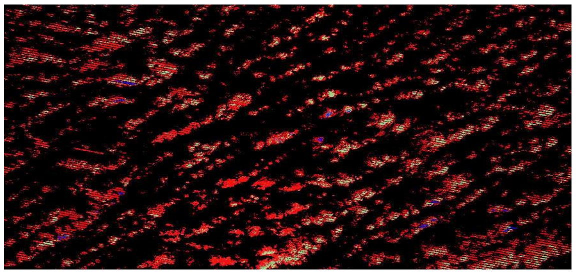

(a) Satellite Image of August 11 (Richarson)

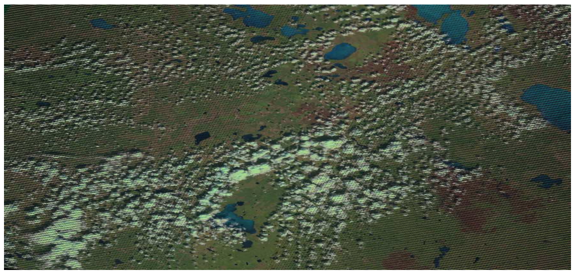

(b) Thermal Image of August 11 (Richarson)


(c) MCTS-A3C for experiment B


(d) Satellite Image for July 10 (Richarson)


(e) Thermal Image of July 10 (Richarson)


(f) MCTS-A3C for experiment A


(g) Satellite image of 27 April (Mcmurray)

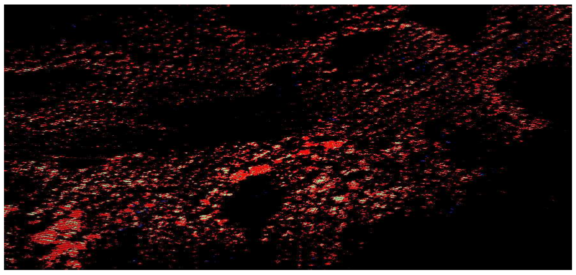

(h) MCTS-A3C for experiment C

Figure 6.2: Results for Alberta fire experiments

Experiment	MCTS	A3C	DQN	DQN(PER)	GP	MCTS-A3C
A	61.3%	87.3%	73.2%	79.4%	62.4%	92.4%
B	60.2%	53.2%	49.5%	51.8%	50.8%	90.8%
C	65.3%	90.1%	79.7%	80.8%	60.5%	90.3%
D	55.7%	81.8%	68.3%	75.9%	47.9%	83.4%
E	49.7%	50.8%	45.6%	48.5%	45.3%	57.8%
F	5.8%	13.4%	9.4%	10.8%	20.5%	50.6%
G	80.7%	84.2%	81.5%	82.7%	83.4%	95.8%
H	51.3%	58.5%	52.1%	53.4%	51.7%	65.8%
I	60.1%	67.2%	60.5%	62.7%	62.4%	73.8%

Table 6.2: Average Accuracy (as a percentage) for each algorithm on the different test scenarios.

Experiment	MCTS	A3C	DQN	DQN(PER)	GP	MCTS-A3C
A	0.654	0.901	0.798	0.843	0.674	0.965
B	0.652	0.563	0.543	0.556	0.538	0.932
C	0.711	0.942	0.845	0.865	0.663	0.945
D	0.601	0.841	0.732	0.812	0.512	0.889
E	0.532	0.542	0.501	0.555	0.532	0.612
F	0.089	0.182	0.210	0.245	0.278	0.583
G	0.877	0.891	0.885	0.893	0.901	0.982
H	0.598	0.683	0.601	0.655	0.599	0.732
I	0.655	0.754	0.698	0.701	0.698	0.801

Table 6.3: Average Precision for each algorithm on the different test scenarios.

Experiment	MCTS	A3C	DQN	DQN(PER)	GP	MCTS-A3C
A	0.573	0.811	0.655	0.701	0.604	0.801
B	0.541	0.486	0.401	0.423	0.486	0.812
C	0.491	0.735	0.712	0.737	0.541	0.798
D	0.423	0.723	0.601	0.677	0.314	0.744
E	0.395	0.402	0.341	0.391	0.342	0.497
F	0.024	0.078	0.054	0.034	0.132	0.397
G	0.699	0.718	0.724	0.701	0.716	0.828
H	0.382	0.391	0.395	0.375	0.405	0.567
I	0.554	0.591	0.523	0.574	0.501	0.614

Table 6.4: Average Recall for each algorithm on the different test scenarios.

Ex	MCTS	A3C	DQN	DQN(PER)	GP	MCTS-A3C
A	59.32 - 63.27	85.24 - 89.35	71.09 - 75.30	77.55 - 81.24	60.25 - 64.54	90.47 - 94.32
B	58.4 - 61.99	51.49 - 54.90	47.87 - 51.12	49.47 - 54.12	48.74 - 52.85	88.82 - 92.77
C	63.94 - 66.65	88.04 - 92.15	77.94 - 81.45	79.52 - 82.07	58.96 - 62.03	88.59 - 92.00
D	54.03 - 57.36	80.00 - 83.59	66.98 - 69.61	74.23 - 77.56	45.62 - 50.17	81.16 - 85.63
E	48.29 - 51.10	49.22 - 52.37	43.89 - 47.30	47.14 - 49.85	43.15 - 47.44	56.04 - 59.55
F	3.65 - 7.94	11.95 - 14.84	7.16 - 11.63	8.74 - 12.85	18.70 - 22.29	48.97 - 52.22
G	79.56 - 81.83	82.49 - 85.9	79.7 - 83.29	81.77 - 83.62	82.12 - 84.67	94.26 - 97.33
H	49.59 - 53.00	57.44 - 59.55	50.74 - 53.45	51.42 - 55.37	50.6 - 52.79	63.95 - 67.64
I	58.56 - 61.63	65.92 - 68.47	58.79 - 62.20	60.9 - 64.49	60.51 - 64.28	72.39 - 75.20

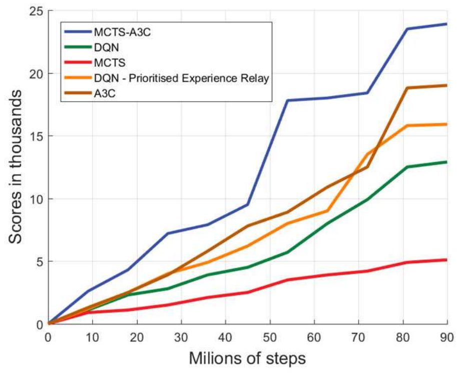
Table 6.5: Confidence Interval with Z distribution for all the mean accuracies with 95% confidence. All the numbers are in percentages.

Experiment	Percentage increase
A	5.84%
B	50.83%
C	0.22%
D	1.95%
E	13.77%
F	59.48%
G	13.42%
H	12.47%
I	9.82%

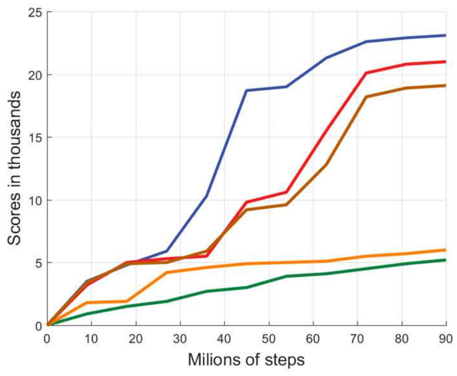
Table 6.6: Percentage increase in accuracy when using MCTS-A3C as opposed to the next best algorithm in each experiment.

as well as the supervised GP approach in most experiments. As elaborated in the last chapter, MCTS has advantages in some experiments such as (B) as it is able to fit the experiments with limited data set that predicts forward in time better than other algorithms. A3C on the other hand is able to do well in experiments that predict in the middle of a time step (like (A)) and also in experiments that transfer the learned policy to a new test domain ((C)). Thus, a combination of these 2 naturally does well in all test domains. MCTS-A3C even beats the best algorithms in the experiments (A) and (B) by some distance. Again, as explained in the previous chapter, as a general rule the accuracies for all experiments decrease from (C) to (F) as we keep moving into the fire season. Now the fire becomes progressively more intense and erratic, which makes prediction tougher. Still, the MCTS-A3C beats all the other algorithms considered in these experiments. The Q Learning based deep learning approaches like DQN and DQN with PER did not perform well in general. With a cross comparison to results in the earlier chapter, we can see their performances were comparable to the classic algorithms and sometimes even not as good as the classic ones in similar experiments. This reinforces the idea that value/policy function based methods are more suitable for this domain as they are model based methods in comparison with model free Q learning based methods. The reasons for the performance of the GP algorithm for the Alberta experiments was discussed in the last chapter. The supervised GP algorithm performed slightly better than MCTS and had comparable performance to the Q learning based Deep methods for the new experiments (G,H and I). Its simple spatial model seems to have some advantage over tree search and DQN algorithms for experiments with larger and/or cleaner data sets as well.

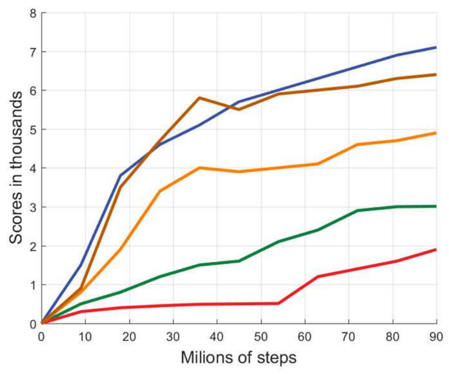
For the simulated fire experiments in experiment (G), we see that the MCTS-A3C has a clear



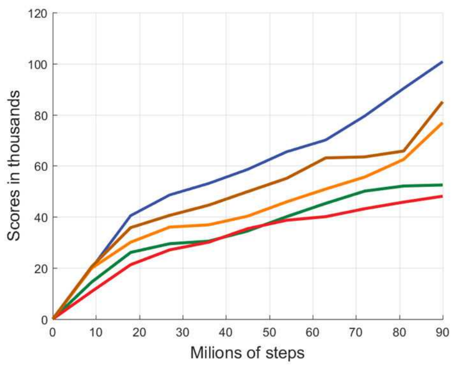
(a) Alberta fires (A)



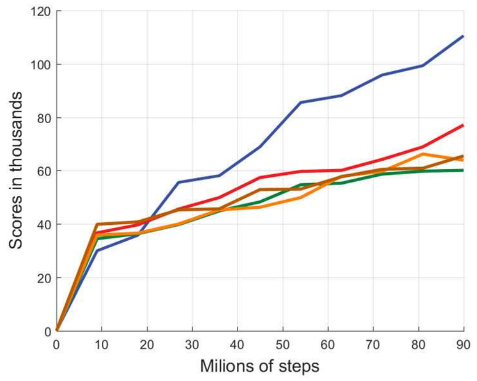
(b) Alberta fires (B)



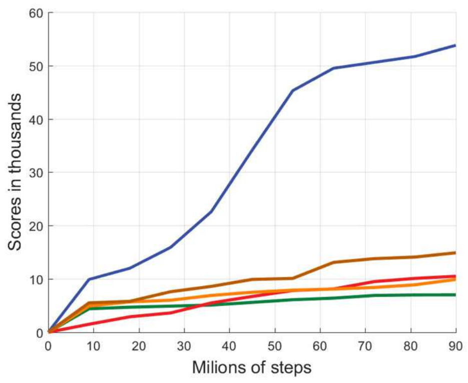
(c) Alberta fires (C)



(d) Saskatchewan fires (H)



(e) Saskatchewan fires (I)



(f) Simulated fires (G)

Figure 6.3: Run Time plots for all the Experiments. All the graphs share the same legend as in (a). Each step is a permissible action taken by the corresponding algorithm.

EcoRegion	BP3 (H)	RL (H)	BP3 (I)	RL(I)
Boreal Transition	33.7%	68.4%	36.7%	69.3%
Mid-boreal Lowland	**53.5%**	**49.1%**	56.5%	69.1%
Mid-boreal Upland	59.2%	75.0%	57.5%	79.2%
Churchill river upland	62.1%	70.5%	66.3%	77.8%

Table 6.7: Average Accuracy values for the different eco-regions in the area of Saskatchewan for the experiment (H) and (I). Only the MCTS- A3C (RL) algorithm and the BP3 physics simulator are considered for this comparison.

advantage over all the other algorithms. This experiment is quite distinct from the rest of the experiments as it has cleaner data, test domains that match most closely to the training data sets, and larger amount of training data. The performances of all algorithms are pretty good in this test domain as expected but overall the general rule of performances mentioned earlier holds well in this domain too. In the experiments related to the Saskatchewan fires seen in (H) and (I) we can see that all the algorithms have lesser average accuracies. This is because there is a lot of inaccuracies in the source data [50] for experiments (H) and (I) and also because we compare several years forward from the training data set for the experiment (H). As more data samples are available for training, performances are better in the experiment (I) as compared to (H). The performance of the RL algorithms in the experiments (G, H and I) proves that it is scalable as it works well with additional data sets and a larger spatial extent as well.

The precision and recalls of all the experiments are also reported in Tables 6.3 and 6.4. The measures follow the general trend of precision and recall values seen in the previous chapter.

The scores as a function of number of steps taken by the algorithms are shown in Figures 6.3. In the case of the simulated fire experiment, MCTS-A3C clearly races ahead of all the other algorithms and the other algorithms have almost comparable learning rates. In the Alberta fire experiments, A3C alone is a close match but MCTS-A3C still finds higher scores as time goes on. On the larger experiments (H) and (I) when all algorithms have access to a longer stream of data and more training time the gain of MCTS-A3C is still clear. For all these curves, we have plotted the average scores as done in the previous chapter and the standard deviation of experiments A,B and C are same as discussed in the last chapter. Experiment (G) saw a maximum standard deviation of 2500 for all the algorithms. Experiment (H) and (I) saw a maximum deviation of 10000. The deviations were almost uniform.

The accuracies we obtained for the Saskatchewan fire (Table 6.7) using BP3 correspond well with detailed landscape burn proportions reported in [50]. The best RL algorithm, MCTS-A3C is used for this comparison. Our algorithm has a higher overall average accuracy in comparison to the BP3 outputs and also it does better in most eco-regions. The Mid-boreal Lowland region for experiment (H) is the only region in which the BP3 beats the RL algorithm. This region is highlighted in [50] as having a much higher uncertainty due to large inaccuracies in the source fuel data. This shows that RL algorithm falls short of the physics based simulator only when the source data sets are affected too much by distortions and noise. But even then we can see that in experiment (I) the RL algorithm beats the performance of the BP3 in the same region. In general the RL does better in (I) as compared to (H). Thus, having enough data samples makes RL algorithm better than physics based simulators. This is a clear advantage in the present times as much more cleaner data sets are available in terms of hyper spectral sensors and drone images during forest wild fires as compared to the last couple of decades.

The 95% confidence interval with the Z distribution for all the mean accuracies are given in Table 6.5. These ranges prove that the statistics with which inferences are made are sound. These numbers increases the confidence in the earlier statistics. The lead of MCTS-A3C is still clear with these numbers. The Table 6.6 gives the percentage improvement in using MCTS-A3C for each experiment as opposed to the next best algorithm. An analysis similar to the statistical A/B testing is done using this table. For some experiments we can see an almost 50% absolute increase in the accuracy as compared to the next best algorithm. This lends a huge support for the new algorithm implemented as opposed to the other algorithms. For some experiments the absolute increase in accuracy is low but for many experiments the increase is more that 10%. This shows the strength of this algorithm in this domain.

6.6 Summary

This chapter proposes a hybrid Deep RL algorithm using strengths of MCTS and A3C which is particularly well suited for learning dynamics of spatially spreading systems such as forest fire spread. We demonstrated the effectiveness of the algorithm on a range of simulated and historical data comparing to previous works, alternative algorithms and the sub-components of our hybrid approach.

Chapter 7

Conclusions and Future Directions

In this chapter, we summarize the major research contributions of the thesis and outline some potential future directions that could be pursued from this research.

7.1 Future Work

As expounded in [39], forest fire prediction requires additional information consisting of fire-fighting intervention (such as fire fighting strategy and time elapsed), which are not taken into consideration in this study, as we chose study regions having very minimal to no fire fighting. In future work we aim to incorporate this kind of information as well as enrich the model by including more land characteristics such as moisture, slope and directional aspect as state variables in individual cells. We will also perform a wider comparison against different existing wildfire models algorithms such as those in [12]. Note that fire managers can use our present research outcomes to determine if the fire needs to be suppressed or not. It also helps them have learned models for particular regions that can predict future fire spreads given ignition points. They could also explore altering the source data with simulated fire lines to determine the behaviour of the fire. Additionally, since the fire simulators take a long time for analyzing a forest fire, the methods presented in this work can be used to augment such full scale simulators or used for a quick ad hoc analysis.

We also plan to investigate improvements to the structure of the Deep Neural Network policy representation to tailor it more closely to this kind of spatially spreading problem. For example, the relatively better behaviour of the supervised GP approach on distance predictions where the dynamics approximation may be hindrance, suggest trying a hybrid approaches, learning a pair of models, one temporal and atemporal to achieve the best of both.

Reachability analysis can be performed using methods similar to experiments in this study that goes backwards in time. This will applications in many domains apart from forest fires like agricultural sowing and harvesting operations. Analysis of variance (ANOVA) can be performed to determine which parameters of the model have the most impact on the accuracy.

The learned model could have captured some interesting patterns or rules that can be used to augment preexisting physics-based simulators. A hybrid RL-physics based model could be implemented and tested.

A severe challenge in the wildfire domain is the relative paucity of data compared to some other image analysis problems. While there are vast databases of satellite imagery, they are not all readily available and finding the very small proportion which contain forest wildfires is not straightforward. This study has looked at a pair of fires in a well known region but future work will use data from more locales and automate the process of data collection. We will also look at flooding as a similar but more data rich domain. This limited amount of image data is one reason we chose to use a state feature extraction approach rather than learning on the entire image directly. There simply may not be enough images to learn effectively. However, a filter based approach such as Convolutional Neural Networks (CNNs) applied directly to the images would be possible if we use small filters on the same scale as the local neighborhood we used in this study. We are currently trying this approach using a combination of CNNs as well as Recurrent Neural Networks to better encapsulate the effect of time on the fire.

7.2 Conclusion

In this work, we presented a novel approach for utilizing RL for learning forest wildfire spread dynamics directly from readily available satellite images. Our approach inverts the usual RL setup so that the dynamics of the MDP is a simple function of the fire spread actions being explored while the agent policy is a learned model of the dynamics of a complex spatially-spreading process. We also introduced a hybrid RL algorithm that performs well in such domains.

The intersection between the decision making tools of Artificial Intelligence, the pattern recognition tools of Machine Learning and the challenging datasets of sustainability domains offer a rich area for research. For the machine learning community our approach opens up new set of challenging and plentiful data sets for learning patterns of spatial change over time in the form of spatially spreading wildfires and a platform for experimenting with new Deep RL approaches on a challenging problem with high social impact.

We hope our work can lead to development of a comprehensive way of integrating Deep Learning and RL approaches to support the tasks of prediction, dynamics model learning and decision making in problems with SSP structure.

Our work would inspire new representations for spatial data and policies which can benefit theoretical as well as applied practitioners. Finally, the algorithmic approach demonstrated here could lead to more effective modelling and decision making tools for domain practitioners in forest wildfire management which we are exploring with collaborators.

References

[1] US Fire Administration. Fire statistics. https://www.usfa.fema.gov/data/statistics/. Accessed: 2018-01-01.

[2] Ahmad AA Alkhatib. A review on forest fire detection techniques. *International Journal of Distributed Sensor Networks*, 2014, 2014.

[3] K Angayarkkani and N Radhakrishnan. Efficient forest fire detection system: a spatial data mining and image processing based approach. *International Journal of Computer Science and Network Security*, 9(3):100–107, 2009.

[4] MIT BCS. Deep successor reinforcement learning. *channels*, 128(4x4):64, 2016.

[5] J Robert Beck and Edwarld K Shultz. The use of relative operating characteristic (roc) curves in test performance evaluation. *Archives of pathology & laboratory medicine*, 110(1):13–20, 1986.

[6] Dimitri P Bertsekas and John N Tsitsiklis. *Parallel and distributed computation: numerical methods*, volume 23. Prentice hall Englewood Cliffs, NJ, 1989.

[7] Charles Blundell, Benigno Uria, Alexander Pritzel, Yazhe Li, Avraham Ruderman, Joel Z Leibo, Jack Rae, Daan Wierstra, and Demis Hassabis. Model-free episodic control. *arXiv preprint arXiv:1606.04460*, 2016.

[8] Justin A Boyan and Andrew W Moore. Generalization in reinforcement learning: Safely approximating the value function. In *Advances in neural information processing systems*, pages 369–376, 1995.

[9] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[10] GF Byrne, PF Crapper, and KK Mayo. Monitoring land-cover change by principal component analysis of multitemporal landsat data. *Remote sensing of Environment*, 10(3):175–184, 1980.

[11] Canadaweather. Environment and natural resources. https://www.canada.ca/en/services/environment/weather.html. Accessed: 2017-9-30.

[12] M Castelli, L Vanneschi, and A Popovič. Predicting burned areas of forest fires: an artificial intelligence approach. *Fire ecology*, 11(1):106–118, 2015.

[13] Canadian Interagency Forest Fire Centre. Ahead of wildland fire. http://www.firegrowthmodel.ca/burnp3/overview_e.php. Accessed: 2017-05-05.

[14] Lawrence J Christiano. Solving the stochastic growth model by linear-quadratic approximation and by value-function iteration. *Journal of Business & Economic Statistics*, 8(1):23–26, 1990.

[15] Paulo Cortez and Anibal de Jesus Raimundo Morais. A data mining approach to predict forest fires using meteorological data. In *volume proceedings of the 13th Portuguese conference on Artificial Intelligence*, 2007.

[16] Matthew J Cracknell and Anya M Reading. Geological mapping using remote sensing data: A comparison of five machine learning algorithms, their response to variations in the spatial distribution of training data and the use of explicit spatial information. *Computers & Geosciences*, 63:22–33, 2014.

[17] Kristopher De Asis, J Fernando Hernandez-Garcia, G Zacharias Holland, and Richard S Sutton. Multi-step reinforcement learning: A unifying algorithm. *arXiv preprint arXiv:1703.01327*, 2017.

[18] Peter Englefield. Fire growth model. http://www.firegrowthmodel.ca/burnp3/overview_e.php. Accessed: 2017-11-25.

[19] Mark A. Finney, Jack D. Cohen, Sara S. McAllister, and W. Matt Jolly. On the need for a theory of wildland fire spread. *International Journal of Wildland Fire*, 22(1):25–36, 2013.

[20] Mark A Finney et al. *FARSITE, Fire Area Simulator–model development and evaluation*, volume 3. US Department of Agriculture, Forest Service, Rocky Mountain Research Station Ogden, UT, 1998.

[21] Mark A. Finney, Isaac C. Grenfell, Charles W. McHugh, Robert C. Seli, Diane Trethewey, Richard D. Stratton, and Stuart Brittain. A Method for Ensemble Wildland Fire Simulation. *Environmental Modeling and Assessment*, 16(2):153–167, nov 2011.

[22] N Forsell, F Garcia, and R Sabbadin. Reinforcement learning for spatial processes. In *18th World IMACS/MODSIM Congress*, pages 755–761, 2009.

[23] Marta Benito Garzón, Radim Blazek, Markus Neteler, Rut Sanchez De Dios, Helios Sainz Ollero, and Cesare Furlanello. Predicting habitat suitability with machine learning models: the potential area of pinus sylvestris l. in the iberian peninsula. *ecological modelling*, 197(3):383–393, 2006.

[24] Sylvain Gelly and David Silver. Combining online and offline knowledge in uct. In *Proceedings of the 24th International Conference on Machine learning*, pages 273–280. ACM, 2007.

[25] Rick Groleau. Fire wars wildfire simulator. http://www.pbs.org/wgbh/nova/fire/simulation.html. Accessed: 2017-11-30.

[26] Matthew C Hansen and Thomas R Loveland. A review of large area monitoring of land cover change using landsat data. *Remote sensing of Environment*, 122:66–74, 2012.

[27] Robert J. Hijmans. World clim global climate data - free climate data for ecological modeling and gis. http://worldclim.org/node/1. Accessed: 2017-10-15.

[28] Rachel M. Houtman, Claire A. Montgomery, Aaron R. Gagnon, David E. Calkin, Thomas G. Dietterich, Sean McGregor, and Mark Crowley. Allowing a wildfire to burn: Estimating the effect on future fire suppression costs. *International Journal of Wildland Fire*, 22(7):871–882, 2013.

[29] Neal Jean, Marshall Burke, Michael Xie, W Matthew Davis, David B Lobell, and Stefano Ermon. Combining satellite imagery and machine learning to predict poverty. *Science*, 353(6301):790–794, 2016.

[30] Paul Johnston, Joel Kelso, and George J. Milne. Efficient simulation of wildfire spread on an irregular grid. *International Journal of Wildland Fire*, 17(5):614, 2008.

[31] Jon E Keeley. Fire intensity, fire severity and burn severity: a brief review and suggested usage. *International Journal of Wildland Fire*, 18(1):116–126, 2009.

[32] Anders Knudby, Ellsworth LeDrew, and Alexander Brenning. Predictive mapping of reef fish species richness, diversity and biomass in Zanzibar using IKONOS imagery and machine-learning techniques. *Remote Sensing of Environment*, 114(6):1230–1241, 2010.

[33] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

[34] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer Berlin Heidelberg, 2006.

[35] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[36] Miroslav Kubat, Robert C Holte, and Stan Matwin. Machine learning for the detection of oil spills in satellite radar images. *Machine learning*, 30(2-3):195–215, 1998.

[37] Guillaume Lample and Devendra Singh Chaplot. Playing fps games with deep reinforcement learning. In *AAAI*, pages 2140–2146, 2017.

[38] Jin Li, Andrew D Heap, Anna Potter, and James J Daniell. Application of machine learning methods to spatial interpolation of environmental variables. *Environmental Modelling & Software*, 26(12):1647–1659, 2011.

[39] K Malarz, S Kaczanowska, and K Kuakowski. Are forest fires predictable? *International Journal of Modern Physics C*, 13(08):1017–1031, 2002.

[40] David L Martell. A Review of Recent Forest and Wildland Fire Management Decision Support Systems Research. *Current Forestry Reports*, 1(2):128–137, 2015.

[41] Sean Mcgregor, Rachel Houtman, Hailey Buckingham, Claire Montgomery, Ronald Metoyer, and Thomas G Dietterich. Fast Simulation for Computational Sustainability Sequential Decision Making Problems. In *Proceedings of the 4th International Conference on Computational Sustainability*, pages 5–7, 2016.

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

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

[44] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[45] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[46] Claire Montgomery. Fire: An Agent and a Consequence of Land Use Change. In Joshua M. Duke and JunJie Wu, editors, *The Oxford Handbook of Land Economics*, chapter 13, pages 281–301. Oxford University Press, 2014.

[47] Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc Bellemare. Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 1054–1062, 2016.

[48] George D Papadopoulos and Fotini-Niovi Pavlidou. A comparative review on wildfire simulators. *IEEE systems Journal*, 5(2):233–243, 2011.

[49] Marc-André Parisien, KG Hirsch, SG Lavoie, JB Todd, VG Kafka, et al. Saskatchewan fire regime analysis. *Natural Resources Canada, Canadian Forest Service, Northern Forestry Centre Information Report NOR-X-394.(Edmonton, AB)*, 2004.

[50] Marc-André Parisien, VG Kafka, KG Hirsch, JB Todd, SG Lavoie, PD Maczek, et al. Mapping wildfire susceptibility with the burn-p3 simulation model. *Natural Resources Canada, Canadian Forest Service, Northern Forestry Centre Edmonton (AB)*, 2005.

[51] Mark Finney Patricia Andrews and Mark Fischetti. Predicting wildfires. `https://www.fs.fed.us/rm/pubs_other/rmrs_2007_andrews_p001.pdf`. Accessed: 2018-04-08.

[52] Guangxiong Peng, Jing Li, Yunhao Chen, Abdul Patah Norizan, and Liphong Tay. High-resolution surface relative humidity computation using modis image in peninsular Malaysia. *Chinese Geographical Science*, 16(3):260–264, 2006.

[53] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[54] Carl Edward Rasmussen and Christopher KI Williams. *Gaussian processes for machine learning*, volume 1. MIT press Cambridge, 2006.

[55] Friedrich Recknagel. Applications of machine learning to ecological modelling. *Ecological Modelling*, 146(1):303–310, 2001.

[56] Mitchel Resnick. *Turtles, termites, and traffic jams: Explorations in massively parallel microworlds*. Mit Press, 1997.

[57] N Naga Saranya and M Hemalatha. Integration of machine learning algorithm using spatial semi supervised classification in fwi data. In *Advances in Engineering, Science and Management (ICAESM), 2012 International Conference on*, pages 699–702. IEEE, 2012.

[58] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint*, abs/1511.05952, 2015.

[59] Vivek Sehgal, Lise Getoor, and Peter D Viechnicki. Entity resolution in geospatial data integration. In *Proceedings of the 14th annual ACM international symposium on Advances in Geographic Information Systems*, pages 83–90. ACM, 2006.

[60] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[61] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 10 2017.

[62] Imas Sukaesih Sitanggang and Mohd Hasmadi Ismail. Classification model for hotspot occurrences using a decision tree method. *Geomatics, Natural Hazards and Risk*, 2(2):111–121, 2011.

[63] Bradly C Stadie, Sergey Levine, and Pieter Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*, 2015.

[64] Brian J Stocks, TJ Lynham, BD Lawson, ME Alexander, CE Van Wagner, RS McAlpine, and DE Dube. Canadian forest fire danger rating system: an overview. *The Forestry Chronicle*, 65(4):258–265, 1989.

[65] Sriram Ganapathi Subramanian and Mark Crowley. Learning forest wildfire dynamics from satellite images using reinforcement learning. In *Conference on Reinforcement Learning and Decision Making*, Ann Arbor, MI, USA., 2017.

[66] US Geological Survey. Earth explorer. https://earthexplorer.usgs.gov/. Accessed: 2016-05-07.

[67] R S Sutton and A G Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[68] Richard S Sutton. Introduction: The challenge of reinforcement learning. In *Reinforcement Learning*, pages 1–3. Springer, 1992.

[69] Canadian Wildland Fire Information System. National wildland fire situation report. http://cwfis.cfs.nrcan.gc.ca. Accessed: 2017-10-26.

[70] Incident Information System. Incident information syatem. https://inciweb.nwcg.gov/incident/5670. Accessed: 2018-01-04.

[71] USGS. Landsat data policy. https://landsat.usgs.gov/sites/default/files/documents/Landsat_Data_Policy.pdf. Accessed: 2018-04-06.

[72] CE Van Wagner, BJ Stocks, BD Lawson, ME Alexander, TJ Lynham, and RS McAlpine. Development and structure of the Canadian forest fire behavior prediction system. *Forestry Canada Fire Danger Group Information Report ST-X-3.(Ottawa, ON)*, 1992.

[73] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, May 1992.

[74] C.J Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK, 1989.

[75] Andrea Woo and Carrie Tait. Up to 90,000 evacuated from fort mcmurray. *The Globe and Mail*, May 4 2016.

[76] Zhang Yongzhong, Z-D Feng, Han Tao, Wu Liyu, Li Kegong, and Duan Xin. Simulating wildfire spreading processes in a spatially heterogeneous landscapes using an improved cellular automaton model. In *Geoscience and Remote Sensing Symposium, 2004. IGARSS'04. Proceedings. 2004 IEEE International*, volume 5, pages 3371–3374. IEEE, 2004.

[77] Guoqiang Peter Zhang. Neural networks for classification: a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 30(4):451–462, 2000.

[78] Jia-Hua Zhang, Feng-Mei Yao, Cheng Liu, Li-Min Yang, and Vijendra K Boken. Detection, emission estimation and risk prediction of forest fires in china using satellite sensors and simulation models in the past three decadesan overview. *International journal of environmental research and public health*, 8(8):3156–3178, 2011.

[79] Nevin Lianwen Zhang and Weihong Zhang. Speeding up the convergence of value iteration in partially observable markov decision processes. *Journal of Artificial Intelligence Research*, 14:29–51, 2001.

[80] Xin Zhang, Jintian Cui, Weisheng Wang, and Chao Lin. A study for texture feature extraction of high-resolution satellite images based on a direction measure and gray level co-occurrence matrix fusion algorithm. *Sensors*, 17(7):1474, 2017.