Risk-Aware Planning by Extracting Uncertainty from Deep Learning-Based Perception

Maymoonah I. Toubeh

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Engineering

Pratap R. Tokekar, Chair
A. Lynn Abbott
Paul E. Plassmann

December 7, 2018
Blacksburg, Virginia

Keywords: Risk-Aware Planning, Uncertainty Approximation, Deep Learning

Risk-Aware Planning by Extracting Uncertainty from Deep Learning-Based Perception

Maymoonah I. Toubeh

(ABSTRACT)

The integration of deep learning models and classical techniques in robotics is constantly creating solutions to problems once thought out of reach. The issues arising in most models that work involve the gap between experimentation and reality, with a need for strategies that assess the risk involved with different models when applied in real-world and safety-critical situations. This work proposes the use of Bayesian approximations of uncertainty from deep learning in a robot planner, showing that this produces more cautious actions in safety-critical scenarios. The case study investigated is motivated by a setup where an aerial robot acts as a "scout" for a ground robot when the below area is unknown or dangerous, with applications in space exploration, military, or search-and-rescue. Images taken from the aerial view are used to provide a less obstructed map to guide the navigation of the robot on the ground. Experiments are conducted using a deep learning semantic image segmentation, followed by a path planner based on the resulting cost map, to provide an empirical analysis of the proposed method. The method is analyzed to assess the impact of variations in the uncertainty extraction, as well as the absence of an uncertainty metric, on the overall system with the use of a defined factor which measures surprise to the planner. The analysis is performed on multiple datasets, showing a similar trend of lower surprise when uncertainty information is incorporated in the planning, given threshold values of the hyperparameters in the uncertainty extraction have been met.

Risk-Aware Planning by Extracting Uncertainty from Deep Learning-Based Perception

Maymoonah I. Toubeh

(GENERAL AUDIENCE ABSTRACT)

Deep learning (DL) is the phrase used to refer to the use of large hierarchical structures, often called neural networks, to approximate semantic information from data input of various forms. DL has shown superior performance at many tasks, such as several forms of image understanding, often referred to as computer vision problems. Deep learning techniques are trained using large amounts of data to map input data to output interpretation. The method should then perform correct input-output mappings on new data, different from the data it was trained on.

Robots often carry various sensors from which it is possible to make interpretations about the environment. Inputs from a sensor can be high dimensional, such as pixels given by a camera, and processing these inputs can be quite tedious and inefficient given a human interpreter. Deep learning has recently been adopted by roboticists as a means of automatically interpreting and representing sensor inputs, like images. The issue that arises with the traditional use of deep learning is twofold: it forces an interpretation of the inputs even when an interpretation is not applicable, and it does not provide a measure of certainty with its outputs. Many techniques have been developed to address this issue with deep learning. These techniques aim to produce a measure of uncertainty associated with DL outputs, such that even when an incorrect or inapplicable output is produced, it is accompanied with a high level of uncertainty.

To explore the efficacy and applicability of these uncertainty extraction techniques, this thesis looks at their use as applied to part of a robot planning system. Specifically, the input to the robot planner is an overhead image taken by an unmanned aerial vehicle (UAV) and the output is a path from a set start and goal position to be taken by an unmanned ground vehicle (UGV) below. The image is passed through a deep learning portion of the system that performs what is called semantic segmentation, mapping each pixel to a meaningful class, on the image. Based on the segmentation, each pixel is given a cost proportionate to the perceived level of safety associated with that class. A cost map is thus formed on the entire image, from which traditional robotics techniques are used to plan a path from start to goal.

A comparison is performed between the risk-neutral case which uses the conventional DL method and the risk-aware case which uses uncertainty information accompanying the modified DL technique. The overall effects on the robot system are envisioned by observing a metric called the surprise factor, where a high surprise factor signifies a poor prediction of the actual cost associated with a path. The risk-neutral case is shown to have a higher surprise factor than the proposed risk-aware setup, both on average and in safety-critical case studies.

Dedication

To Yousef, Danyah, Ibrahim, and Ghazi.

Acknowledgments

The material is based upon work supported by the National Science Foundation under Grant number 1566247.

This work was also supported by the William E. Webber Fellowship.

Contents

List of Figures ix

List of Tables xii

1 Introduction 1

1.1 Deep Learning . . . 3

1.1.1 Training Algorithms . . . 5

1.1.2 Regularization . . . 6

1.1.3 Architectures . . . 7

1.2 Uncertainty Extraction from Deep Learning . . . 8

1.2.1 Bayesian Neural Networks . . . 10

1.2.2 Ensemble Methods . . . 10

1.2.3 Stochastic Regularization Methods . . . 11

1.3 Planning . . . 13

1.3.1 Mapping with Semantic Segmentation . . . 13

1.3.2 Risk-Aware Methods . . . 14

2 Risk-Aware Planning 16

2.1 Uncertainty Extraction . . . 18

2.1.1 Deep Learning Model 19

2.2 A* Planner 21

3 Results and Discussion **25**

3.1 Qualitative Analysis 26

3.2 Quantitative Results 32

4 Conclusions and Future Work **40**

4.1 Lessons Learned 40

4.2 Future Work 42

4.2.1 Risk Metrics 42

4.2.2 End-to-End Planner 44

Bibliography **49**

List of Figures

1.1 Basic illustration of a neural network 4

2.1 The overall flow of information in a robot planner that relies in part on a deep learning model in (a) the risk-neutral and (b) the risk-aware cases. . . . 17

2.2 Example semantic segmentation produced by the Bayesian SegNet model trained on Aeroscapes showing (a) an input image from the Aeroscapes test set with (b) its handcrafted ground truth segmentation, alongside (c) the resulting output segmentation, and (d) the model uncertainty associated with the output, with darker regions signifying higher uncertainty. 22

2.3 Example semantic segmentation produced by the Bayesian SegNet model trained on Aeroscapes showing (a) an input image from the Kentland dataset with (b) its handcrafted ground truth segmentation, alongside (c) the resulting output segmentation, and (d) the model uncertainty associated with the output, with darker regions signifying higher uncertainty. 23

3.1 Qualitative results on an Aeroscapes image showing the path planned from start to goal given (1) handcrafted ground truth image segments, (2) DL model segmentation alone, and (3) DL model segmentation with uncertainty. 28

3.2 Qualitative results on a Kentland image showing the path planned from start to goal given (1) handcrafted ground truth image segments, (2) DL model segmentation alone, and (3) DL model segmentation with uncertainty. 30

3.3 Effect of varying λ on the surprise factor given the Kentland image and start and goal positions from Figure 3.2. 31

3.4 Effect of varying the number of stochastic passes on the surprise factor given the Kentland image with start and goal positions from Figure 3.2. 32

3.5 Uncertainty predictions with varying the number of stochastic passes given the Kentland image from Figure 3.2, where darker regions signify higher uncertainty. 33

3.6 Example Kentland image with no corresponding ground truth class for "water", so the DL model classifies that region as the ambiguous class "background". 35

3.7 Effect of varying λ on the surprise factor averaged over 20 Kentland images with one random start and goal position each. 36

3.8 Effect of varying λ on the surprise factor averaged over five Kentland images with diverse cost maps given one random start and goal position each. . . . 37

3.9 Effect of varying the number of stochastic passes on the runtime (microseconds) averaged over 20 Kentland images. 38

3.10 Effect of varying the number of stochastic passes on the surprise factor averaged over 20 Kentland aerial images, given one random start and goal position each. 39

3.11 Effect of varying the number of stochastic passes on the surprise factor averaged over five Kentland images with diverse cost maps, given one random start and goal position each. 39

4.1 The overall flow of information in a robot planner which relies entirely on a DL model for its learning and decision making in (a) the risk-neutral and (b) risk-aware cases. 45

4.2 Reward per episode of learning using DQN for cartpole displaying (a) the risk-neutral traditional method and (b) the risk-aware method that incorporates uncertainty. 47

4.3 Histogram of rewards during learning using DQN for cartpole displaying (a) the risk-neutral traditional method and (b) the risk-aware method that incorporates uncertainty. 47

List of Tables

2.1 The fixed costs assigned to each segmentation class from the Aeroscapes dataset to be used in A* search. These costs are hand-designed to generate qualitative examples that demonstrate the utility of the proposed approach. In the real world, classes that are not navigable (e.g. fence, car, bike) will be assigned infinite cost. 16

3.1 The surprise factor for the example in Figure 3.1, given ground truth, risk-neutral, and risk-aware cost. 27

3.2 The surprise factor for the example in Figure 3.2, given ground truth, risk-neutral, and risk-aware cost. 29

3.3 The average surprise factor for 20 Kentland images and five randomly chosen start and goal positions each, given ground truth, risk-neutral, and risk-aware cost. 35

3.4 The average surprise factor for five Kentland images with diverse cost maps and five randomly chosen starting and goal positions each, given ground truth, risk-neutral, and risk-aware cost. 35

4.1 DQN cartpole statistics comparing risk-neutral and risk-aware methods. . . 48

List of Abbreviations

BNN Bayesian Neural Network

BP Backpropagation

CNN Convolutional Neural Network

DL Deep Learning

DQN Deep Q-Learning Network

DRL Deep Reinforcement Learning

GP Gaussian Process

GPU Graphical Processing Unit

HRI Human Robot Interaction

MDP Markov Decision Process

MI Mutual Information

ML Machine Learning

MSE Mean Square Error

POMDP Partially Observable Markov Decision Process

ReLU Rectified Linear Unit

RL Reinforcement Learning

SGD Stochastic Gradient Descent

SLAM Simultaneous Localization and Mapping

UAV Unmanned Aerial Vehicle

UGV Unmanned Ground Vehicle

VaR Value at Risk

Chapter 1

Introduction

Recent advances in deep learning (DL) training algorithms, paired with significant improvements in hardware, have shown potential in many fields, including robotics. From enabling robot systems to navigate using high-dimensional image inputs, to allowing tractable trial-and-error robot learning both in simulation and reality; deep learning is everywhere. However, as promising as applications of DL to robot planning may seem, the potential of the positive impact they may have on real-world scenarios is inevitably proportionate to their interpretability and applicability to imperfect environments. As a step towards risk-aware robotic systems that utilize the powers of DL, the work presented in this thesis combines methods of approximating uncertainty in DL with robot planners that are partially reliant on an otherwise black-box DL approach. With the acquired Bayesian uncertainty estimates, the system produces an explicable metric and can therefore be altered to accommodate for risks associated with safety-critical actions in the environment.

In machine learning, the main focus typically involves improving accuracy of a given method on a benchmark dataset that is agreed upon to be challenging enough. However, in robotics, it is important that designs that work in simulation can also function in reality, with a fail-safe well placed in the event that negative physical impacts may be involved. In order to redirect a robot's actions in the event of failure, the failure must first be detected. Currently, many DL models are being used in robot systems and have provided significant improvements in overall performance, but the majority of these works neglect questions regarding

vulnerabilities of a system that does not have the capacity to signal its own failures. The focus on performance in a controlled laboratory setting is understandable from the perspective of innovative researchers who are approaching novel problems that do not have solutions yet. However, for problems that already have solutions, it is important that the solutions are applicable to a real-world setting if we are ever to see them positively contributing to our daily lives. In safety-critical situations such as driving or manufacturing, it is important that robot systems are able to identify their own limits.

The work in this thesis proposes the use of approximations of uncertainty in deep learning to direct a robot planner towards more risk-aware decision making for safety-critical situations. The main contributions of this work are as follows:

- An efficient risk-aware framework is proposed for planning systems which already use deep learning for perception.
- A risk-aware cost function is defined as a means of interpreting uncertainty of a deep learning model's prediction in a meaningful way to a subsequent robot planner.
- A metric called the *surprise factor* is defined to provide a means of comparing what the robot planner expects with reality.
- An empirical analysis is performed with the purpose of portraying the impact of variations in the hyperparameters used in the uncertainty extraction, or the absence of an uncertainty metric, on the overall planning system. The surprise factor and runtime are observed, showing that the tuning of certain hyperparameters may be more important than others, especially in a resource constrained setting like robotics.
- The proposed risk-aware method is shown to produce more consistent and safe results, where the conventional risk-neutral method proves to be dangerous and possibly

catastrophic, given that threshold values of the hyperparameters in the uncertainty extraction are met.

Portions of the work leading towards this thesis were initially published in the following:

M. Toubeh and P. Tokekar, "Risk-Aware Planning by Extracting Uncertainty from Deep Learning-Based Perception," in *AAAI 2018 Fall Symposium on Reasoning and Learning in Real-World Systems for Long-Term Autonomy*, 2018.

The remainder of this chapter presents important background information on deep learning, Bayesian approaches in DL, and risk-aware planning. The intent is to provide a concise but thorough understanding of the tools being utilized in the machine learning and robotics communities, as well as present areas the two have benefited from each other. Some of the unanswered questions regarding the use of DL in robot systems are posed for this thesis to address.

1.1 Deep Learning

Deep Learning (DL) is used to describe machine learning approaches that utilize multi-layered processing structures, such as neural networks, inspired by the human brain configuration, though quite unlike it. Neural networks with more than three layers are considered "deep", which would apply to most modern structures. Although DL has been around for years, the hardware capacity and proper training algorithms were not available until recently. Therefore, in the past decade, ample research has emerged in both the machine learning and robotics communities employing deep learning for planning needs.

The use of DL is often synonymous with the use of neural networks (NN). Figure 1.1 depicts a very simplified representation of a neural network for illustrative purposes [19]. Neural

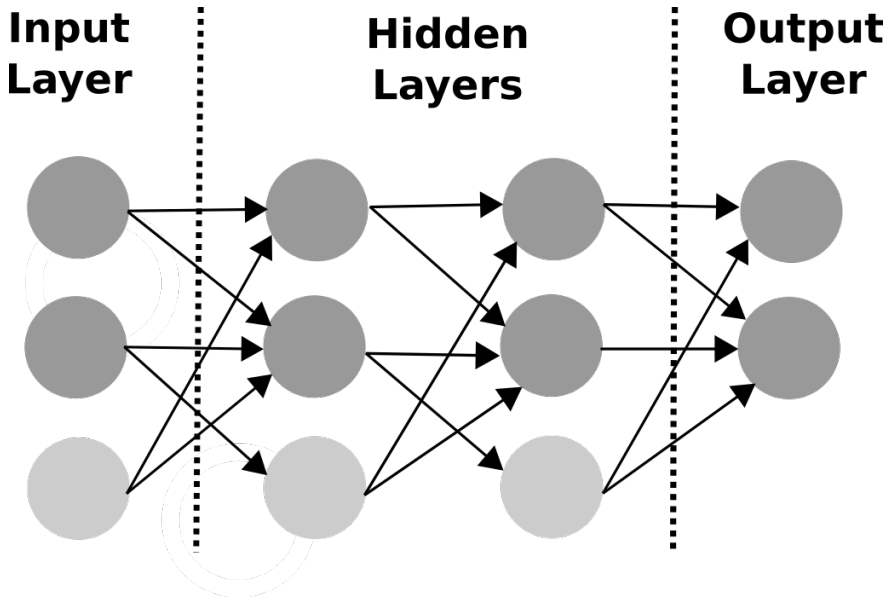


Figure 1.1: Basic illustration of a neural network

networks are composed of nodes, called neurons, which perform a mathematical activation function in layers: input, hidden, and output, as shown in Figure 1.1. The connections from each layer to the next are weighted, and these weights are adjusted through a training algorithm. Training is done to achieve a mapping from the input to the desired output, such that when new inputs are presented, the network can provide reasonable results. The network might also include bias neurons used for scaling to help with generalization, which refers to the extensive ability to map unseen inputs to correct outputs.

Activation functions are used to establish bounds for the output of a group of neurons in a layer, which is essential in creating a relationship between the inputs and desired outputs [19]. Input layers typically do not have activations, whereas the most common activations for hidden layers and the output layer are the rectified linear unit (ReLU) and the softmax, respectively. The ReLU activation is defined by the following equation [19]:

$$\sigma(x) = max(0, x) \tag{1.1}$$

The softmax is commonly used to select one out of several categories, especially in classification tasks. The equation for the softmax is as follows [19]:

$$\sigma(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}} \tag{1.2}$$

In the softmax function, the output of one neuron i is dependent on all neurons j in the same layer, where N is the total number of neurons in that layer. For this reason, the softmax is often inaccurately assumed to depict a probability distribution over the outputs. However, mathematically, it is simply a normalization of deterministic point estimate outputs, such that the sum of all softmax outputs is one [12].

1.1.1 Training Algorithms

During training, the weights of the connections between neurons are adjusted so that the actual output is made to match the desired output. A neural network can be trained using various training algorithms, the most common being the backpropagation algorithm, where the weights of the neural network are updated using the following equation [19]:

$$\Delta w_i = -\epsilon \frac{\partial E}{\partial w_i} + \alpha \Delta w_{i-1} \tag{1.3}$$

Here w_i is the weight of the current iteration, w_{i-1} is the weight of the previous iteration, α is the momentum, ϵ is the learning rate, and $\frac{\partial E}{\partial w_i}$ is the gradient [19]. The gradient here is the partial derivative of the error function at the weight's current value. The sign of the

gradient tells the NN how to change the weight. A zero gradient shows that the weight is not contributing to the error [19]. Negative gradient indicates that the weight should be increased to achieve a lower error [19]. Positive gradient implies that the weight should be decreased to achieve a lower error [19]. Several different error functions exist. One common error function is the Mean Square Error (MSE) defined by Equation 1.4 [19]:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 \tag{1.4}$$

This equation compares the actual output y with the expected output $\hat{y}$ for a number of training elements N [19]. Other error functions include cross entropy cosine proximity [7]. Several training algorithms have also evolved from backpropagation, including stochastic gradient decent (SGD) and Adam [7].

1.1.2 Regularization

Many tools have been developed for the purpose of generalization, a large class of which fall under the umbrella of regularization. Regularization is applied during the training of a neural network in order to avoid overfitting the model to its training set. An overfit model will not produce reasonable results on new data, implying that it has simply memorized the training data and not instead learned features that are intrinsically meaningful and transferable. At the top of the list of most commonly used regularizers is dropout [36]. Dropout can be thought of as effectively removing certain nodes from a neural network during training to increase the robustness of the network to variations in its structure, which can translate to a robustness to variations in input [18]. In most modern structures, neurons are "dropped out" by simply multiplying their weights by zero [18]. This type of dropout is often referred to as Bernoulli dropout, which resembles multiplying the weights of the network by a Bernoulli

random variable that takes on the value of either zero or one in a set proportion. The dropout rate is the proportion of neurons in the network that are multiplied by zero, where a 20% dropout rate implies that 20% of the non-output neurons are set to zero. During test time, to accommodate for the use of dropout during training, common practice involves weight averaging of all network weights proportionate to the dropout rate.

1.1.3 Architectures

A neural network's architecture is defined by the structure it takes and the direction of information flow within the structure. Although different architectures have been found to be more useful in solving different problems, the type of problem at hand alone does not uniquely determine the architecture that will work best [11]. A lot of trial and error is involved in the selection of the hyperparameters of a neural network, even though some common intuitions are shared among machine learning researchers. Generally, the more hidden layers in a neural network, the more complex it is, but the computational complexity usually also adds to the computational time. Processing deeper structures is made possible with newer hardware capacities that allow for massive parallelization, which has in turn made hand-engineered feature extraction obsolete. Large neural networks are capable of recognizing contextual information on their own, not without the correct hardware and often the large amounts of time needed for data collection and training. Finding a balance between the various trade-offs involved in a deep learning model's design choices is a core challenge for neural network designers.

Structures used in deep learning are multi-layered and usually involve feedforward information flow. A feedforward neural network propagates information from the inputs to the outputs in a forward direction only. Some networks may include backward information flow

through the use of what are called recurrent connections between their neurons. In a recurrent connection the data can flow not only in the forward direction, but also backwards. Certain restrictions may be applied by the designer. The processing of recurrent connections can be challenging, as they are prone to create endless loops [19]. Many different techniques are used to prevent this, including the use of context neurons. The purpose of the use of a recurrent neural network is to allow for the handling of data as it occurs over time [19].

Perhaps the most revolutionary structure, especially in the field of computer vision, is the convolutional neural network (CNN). A CNN simulates the features of the biological eye by utilizing the overlapping fields of input, which provides it with the ability to process scale, rotation, and noise. In order to utilize a CNN on non-visual data, the designer must find a way to encode the data so that it can mimic the properties of visual data [19]. Most modern structures use convolutional layers within the model. In an encoder-decoder network, convolutional layers are exploited to encode spatial information into features, then the decoder utilizes spatial information lost by retrieving it from previous layers [2]. This is found to be especially useful with tasks that involve learning fine-grained representations, like semantic image segmentation, which rely on both high-level features and low-level spatial information. This thesis utilizes a state-of-the-art deep learning architecture for experimentation, but it is important to note that the methods utilized for uncertainty extraction are theoretically proven to be architecture agnostic [12].

1.2 Uncertainty Extraction from Deep Learning

Deep learning is known for its data-driven rather than algorithmic learned representations [27]. A DL hierarchical structure can learn directly from data with little to no handcrafted features or learning variables [18]. However, this often comes at the expense of

interpretability of the learning outcomes. Deep neural networks can even misrepresent data outside the training distribution, giving predictions that are incorrect without providing a clear measure of certainty associated with the result [12]. The outputs of a deep neural network are generally point estimates of the parameters and predictions present, so they do not provide a meaningful measure of correlation to the overall data distribution the network was trained on. For this reason, deep learning models are considered deterministic functions, often called "black-boxes", unlike probabilistic models which inherently depict uncertainty information. Our work utilizes modern methods of uncertainty extraction from deep learning models, specifically those that do not interfere with the overall structure or training process [13]. The extraction of uncertainty information, as opposed to the reliance on point estimates, is crucial in safety-critical applications, such as robot navigation in an urban setting. If a robot planner encounters out-of-distribution data at test time, it is preferable that the system provides an uncertainty metric to allow for more accurate interpretation of a forced point estimate.

Although modern DL models are usually considered black-boxes due to their mathematical nature, recent work has initiated theoretically grounded understandings of them. Such works investigate the integration of deep learning techniques with information theoretical and statistical approaches for the purpose of calculating model uncertainties [12]. It is these practical methods of quantifying risk associated with DL models that are utilized in the proposed planning systems of this thesis.

Several prior works exist which have extracted uncertainty information from a deep learning model, mainly using Bayesian neural networks, ensemble methods, and methods that utilize stochastic regularization techniques like dropout [12, 34]. These works have also produced meaningful contributions to real-world applications, such as gas turbine control, camera relocalization, and robotic collision avoidance [10, 20, 22]. Our work seeks to extend the

most suitable of these approaches to robot path planning.

1.2.1 Bayesian Neural Networks

Some of the earliest attempts to bind the reasoning of probabilistic models, such as Gaussian processes, with deep learning (DL) is seen in Bayesian neural networks (BNN) [12, 24]. Unlike the DL models used in modern practice which do not depict uncertainty, a BNN produces an output that is a probability distribution over its predictions. Probability distributions are placed over the weights of a BNN, making it an approximation of a Gaussian process as the number of weights tends to infinity. Uncertainty can be extracted as a statistical measure, such as variance or entropy, over this output distribution in order to capture how confident the model is with its prediction. However, BNN require a larger number of parameters to be trained, with less practical training methods available for them. A recent example of a Bayesian neural network applied to a stochastic dynamic system utilizes a smaller model and trains by minimizing alpha-divergences [10]. The system dynamics are learned using the BNN and are fed into a model-based reinforcement learner for control, with application to a gas turbine. Due to the complexities involved in the design and training of a BNN, complexities in the model's structure and outputs are typically compromised, which can be limiting in real-world robotic applications.

1.2.2 Ensemble Methods

Since the practicality of Bayesian neural networks is questionable, approximations of these structures have arisen, including ensemble methods. One such recent method is referred to as the bootstrapped neural network, where several deep learning (DL) models are trained on subsets of the larger dataset sampled with replacement. The underlying concept behind

this method is that the different models will agree in high density areas and disagree in low density areas of the complete dataset. The outputs of the separate models combine to form a probability density function from which uncertainty of a particular prediction can be measured. This method is theoretically sound; however, it is not ideal for applications that are faced with time and resource constraints, such as robotics.

A recent work proposes using bootstrapped deep Q-learning networks quantifying uncertainty to direct the learning process towards more efficient exploration, which is a key issue in reinforcement learning [34]. The technique involves sampling the past experiences in Q-learning, rather than taking the whole sequence, in order to form an estimate of the Q-value at a given state. Sampling is also meant to scale better to large state-spaces, which are common in robotics. A similar approach is used to improve a form of Q-learning [21]. The work demonstrates that using bootstrapped uncertainty to direct data collection produces faster learning that is also less expensive, tedious, or likely to lead to physical damage to a real robot.

1.2.3 Stochastic Regularization Methods

Most recently, stochastic regularization methods common in deep learning (DL) have been used to approximate Bayesian neural network models without changes to the ready structures being used or their training process [12]. Regularization methods are used as a means to avoid overfitting of a DL model to its training set, so that it generalizes to data that is similar enough but not exactly the same. This ensures the learning model has not simply memorized the training data, but has actually learned something meaningful about the data that will translate to a slightly different setting. At a high level, stochastic regularization techniques work by introducing randomness in the training process to increase the robustness

of the model to noise. Dropout is one such popular method that is inspired by probabilistic interpretations of deep learning models which consider activation nonlinearity a cumulative distribution function [5]. In its traditional use, dropout is activated during training, in which case the weights of a deep learning model are randomly multiplied by zero or one in a certain predefined proportion. In the conventional approach, at test time, weight averaging by the percentage of dropout applied during training is performed on the trained model, which then leads to point estimate results.

In order to form a distribution over the outputs of a model trained using a stochastic regularization technique such as dropout, the regularization is activated at test time, producing stochastic estimates with multiple passes of the same input through the model [13]. The multiple stochastic passes are then averaged to form a mean estimate, as opposed to a point estimate. Uncertainty can be extracted given statistical or information theoretical metrics over the probability density function formed by the varying outputs of the stochastic passes.

One recent work applies Bayesian approximation using dropout to a pre-existing model-based reinforcement learning algorithm [14]. The authors replace the original Gaussian processes with a deep neural network, showing better quantification of uncertainty over longer periods of learning. In another work, the Bayesian approximation of uncertainty with dropout is used to assist in camera relocalization for landmark detection in a SLAM problem [22].The uncertainty estimate is used to approximate the localization error with no additional handcrafted parameterizations. In a similar work, the same Bayesian approximation approach to uncertainty is applied to semantic segmentation for improved learning and test time estimation [23]. In yet another work, the authors combine bootstrapped neural networks with stochastic regularization methods to avoid catastrophic or harsh collisions during robot training for collision avoidance [20]. They show that their method effectively minimizes dangerous collisions during training, while also showing comparable performance to baselines

without explicit account for uncertainty. A more recent approach proposes the use of a Mixture Density Model (MDM) as a replacement for the stochastic passes, where a single pass saves time in comparison to multiple in a robotics setting [6]. However, this comes at the expense of training a separate MDM network for the task of uncertainty extraction.

1.3 Planning

Robot motion planning entails finding policies for representing the robot and its environment and for searching a path through the represented environment [28]. Typically, the process is divided into two tasks: perception and planning. Perception refers to how the robot perceives its environment and itself within the environment. To do this, the robot must interpret its sensory inputs into a meaningful representation, with the use of tools such as symbols, graphs, maps, and features. The planning is then performed on the resulting interpretations of the environment. Several classical techniques exist for planning, some involving a systematic search performed using the representation of the robot and its environment [26]. Joint perception and planning is also possible, and this is usually referred to as active perception [1, 3]. However, this thesis focuses on the case where perception and planning are separated, with a deep learning model used for perception and a classical method used for path planning.

1.3.1 Mapping with Semantic Segmentation

Many robot planning applications involve the use of information taken from images, through a computer vision technique, to assist or guild the planning. One of the common ways perceptions are formed on images is through semantic segmentation, where each pixel is classified into a meaningful category. Semantic segmentation is useful for creating detailed

maps of a robot's environment for navigation. Pixel-wise semantic segmentation is one tool that lies under the broader umbrella of semantic mapping, which refers to providing a qualitative description of a robot's surroundings to augment its navigation and planning [25]. Semantic mapping provides a basis for human robot interaction (HRI) and increases the interpretability of a robot's navigation and planning choices.

Unmanned aerial vehicle (UAV) assisted navigation of an unmanned ground vehicle (UGV) often utilizes more flexible, and less obstructed, viewpoints from the air to form a map for the ground robot. The UAV-UGV cooperative setup is being used in Mars rovers to better navigate previously uncharted territories [32, 33]. Military, search-and-rescue, and radiation detection operations also benefit from this duo [9]. The crucial task given images or other sensory data provided by the aerial robot is to build a meaningful map of the below environment that can be used to assist the ground robot. Semantic segmentation can be used in order to form a semantic map given images provided by an aerial robot [9].

1.3.2 Risk-Aware Methods

In the robotics community, an emphasis has been placed on methods that work in a controlled experimental setup, but more recently risk-aware methods aim to ensure that these systems are also safe in the real-world. As a relatively new application to robotics, techniques have been adapted from the fields of statistics and machine learning. Common statistical methods of accommodating risk include altering the optimization criterion so that it becomes risk-sensitive [17].

When performing robot planning and decision making, a cost function, or conversely a utility function, is defined to quantify the desirability of certain outcomes as compared to others. The cost function is often associated with outcomes which are stochastic in nature. Common

practice when searching for an optimal decision is to take the expected value over the cost distribution as a fair estimate of the value of the decision [29]. Although this average value is convenient, it is considered risk-neutral, as it does not account for outcomes that can be undesirable or even catastrophic in safety critical scenarios. At the opposite end of the spectrum is taking the worst-case assessment of the stochastic outcomes, which yields a highly conservative planner that is considered completely risk-averse [29]. Ideally, a robot should be risk-aware, lying in between the two extremes, such that it has the ability to quantify risks and act rationally and consistently under uncertainty.

Utilizing learning-based approaches, some research attempts to approach this problem by explicitly learning cues to the undesired risky behavior. Deep learning approaches are used for robotic accident anticipation and risky region localization, where the risk is implicitly encoded in a dataset of collisions [38]. A single video frame is shown to predict collisions using a more vast dataset of explicit collisions [16]. However, these methods are limited by the specialized cases they are applied to. For more general risk-awareness, risk metrics, such as variance in the prediction which is used in this work, can be more readily applied to already existing risk-neutral systems.

Chapter 2

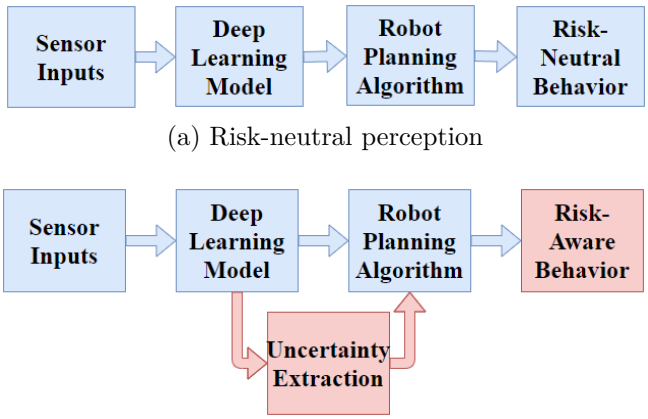
Risk-Aware Planning

The experimental setup of this thesis is inspired by a previous work that uses the overhead imagery provided by an unmanned aerial vehicle (UAV) as input to an image segmentation algorithm, which is then used to assist the navigation of an unmanned ground vehicle (UGV) [9]. The UAV acts as a "scout" by flying ahead of the UGV. The individual pixels of the overhead orthorectified imagery are then classified (into categories such as "road", "person", "car", etc.). Each category is assigned a cost (given in Table 2.1) which is used to determine a path for the ground robot to follow.

Figure 2.1 shows a high-level schematic of the proposed risk-aware approach by contrasting it with the traditional, risk-neural approach. Unlike the previous work, here, in addition to performing the semantic segmentation of the image, the uncertainty in the segmentation is also extracted. The measure of uncertainty is then used to manipulate the navigation away from low confidence regions. The navigation portion of the robot planner in this case is not a deep learning (DL) model, but a classical method, A* search. A cost function is mapped

Softmax	1	2	3	4	5	6	7	8	9	10	11	12	N/A
Class	Background	Person	Bike	Car	Drone	Boat	Animal	Obstacle	Construction	Vegetation	Road	Sky	Unlabeled
Cost	2	14	13	9	7	8	12	10	11	5	1	15	17
Color													

Table 2.1: The fixed costs assigned to each segmentation class from the Aeroscapes dataset to be used in A* search. These costs are hand-designed to generate qualitative examples that demonstrate the utility of the proposed approach. In the real world, classes that are not navigable (e.g. fence, car, bike) will be assigned infinite cost.



(a) Risk-neutral perception

(b) Risk-aware perception

Figure 2.1: The overall flow of information in a robot planner that relies in part on a deep learning model in (a) the risk-neutral and (b) the risk-aware cases.

onto each semantic class (see Table 2.1). Therefore, the uncertainty in the segmentation corresponds to uncertainties in the cost, for which A* search is sufficient. If the uncertainties in the transition function are to be considered instead, a Markov decision process (MDP) would be more useful. Considering uncertainty is contrast to trusting the outputs of the DL portion of the system invariably, which could lead to catastrophic outcomes if a point estimate outlier is produced given an input that is considered out-of-distribution to the training data. In the proposed approach, an uncertainty metric can be used to calibrate the robot's plan based on the level of confidence in the DL model predictions.

The results of the segmentation given by any DL model cannot guarantee complete accuracy in all settings. Variations in lighting, angle, or objects present in an image can contribute to inaccurate predictions. There will always be a prediction when a DL model is involved, as the model will force an estimate even when it does not make sense to. A good measure to account for this risk associated with DL outputs being used in the robot planner is to evaluate the certainty associated with the DL result. One practical method is using dropout, which is already being used as a regularizer during training.

In the risk-neutral case, the pixel classification is taken as is from the DL model and assigned a cost accordingly. For the proposed risk-aware method, the cost is evaluated by adding the uncertainty value, multiplied by some factor, to the risk-neutral cost assignment. Specifically, the cost of pixel p is given by,

$$C(p) = L(p) + \lambda \hat{V}(p), \tag{2.1}$$

where $L(p)$ is the cost associated with the semantic class that is predicted for p (given in Table 2.1) and $\hat{V}(p)$ is the uncertainty value extracted by dropout. In practice, $\hat{V}(p)$ does not need to be variance; it can be another statistical measure of uncertainty taken over the distribution provided by the stochastic passes. λ is a weighting parameter. The risk-neutral case corresponds to $\lambda = 0$ and the risk-aware case corresponds to $\lambda > 0$.

2.1 Uncertainty Extraction

Algorithm 1 shows a breakdown of uncertainty extraction given an input image. First, the stochastic outputs are generated for a number of stochastic passes, giving a softmax value for each pixel each time. For each pass, Bernoulli dropout is activated on the trained network, effectively multiplying random neuron weights by zero or one in a set proportion. The softmax outputs O are averaged over all stochastic passes. The maximum value of this average softmax is taken as the output class prediction Ind. Variance is computed over the stochastic passes for each output class, then the average of V over all classes produces a single value for each pixel. The value $\hat{V}$ is considered the uncertainty in that pixel's prediction.

The uncertainty value is computed as the average variance across all segment classes for a particular pixel. The higher the average variance, the less confident the deep learning model is with its prediction. Therefore, it is intuitive to incorporate this information along with the original prediction when planning a path for navigation, especially in a safety-critical

Algorithm 1 Uncertainty Extraction

Input: image I

Output: class of average prediction for each pixel $Ind(p)$, average variance for each pixel $\hat{V}(p)$

$p \leftarrow$ pixels in image I
for $t = 1$ to number of stochastic passes **do**
    $O(p, c, t) \leftarrow$ softmax output of stochastic pass t
end for
$\hat{O}(p, c) \leftarrow$ average $O(p, c, t)$ over stochastic passes
$Ind(p) \leftarrow$ argmax of softmax in $\hat{O}(p, c)$
for $c = 1$ to number of classes **do**
    $V(p, c) \leftarrow$ variance in $O(p, c, t)$ over stochastic passes for class c
end for
$\hat{V}(p) \leftarrow$ average $V(p, c)$ over classes

environment such as a road.

2.1.1 Deep Learning Model

To demonstrate the utility of the proposed method of extracting uncertainty from a deep learning (DL) model and using the uncertainty metric to guide the robot's planning, various experiments involving this setup are portrayed. The experiments consist mainly of a DL semantic image segmentation model from which uncertainty information is deduced, along with an A* search path planner for navigation based on the segmentation cost map. A dataset of aerial images is used to train the DL model, and then the model is tested on a different dataset to illustrate the robustness of the method to anomalies outside the training set.

We use the Bayesian SegNet to perform semantic segmentation of every pixel in the input image [23]. The Bayesian SegNet is first trained for the segmentation task using the predefined encoder-decoder architecture, along with the pretrained weights of the VGG16 image

classification network [23]. Since the model is already well adapted for image classification, less further training is needed for per-pixel segmentation, in comparison to starting with random model weighting. A batch size of 2 is used to fit an 8GB Nvidia GeForce GTX 1080 GPU.

Since the motivation for our work is inspired by prior applications in UAV-UGV coordination, where an autonomous aerial vehicle is responsible for providing a less obstructed map for the UGV below it, we trained the Bayesian SegNet on a realistic dataset from the aerial viewpoint. For this, the Aeroscapes dataset of 3269 diverse aerial images taken with a fleet of drones is used for training [31]. Aeroscapes is prelabeled to consist of 12 ground truth classes that correspond to 12 separate softmax outputs in the trained model. The classes and their softmax correspondence, as well as the color legend used for visualizations, are shown in Table 2.1. The dataset is first divided into training, testing, and validation sets, allotting 1963, 653, and 653 images for each, respectively. The test accuracy is computed for each 360x480 pixel image, resized to fit the Bayesian SegNet inputs, using the following equation:

$$test\ accuracy = \frac{number\ of\ pixels\ correctly\ classified - total\ number\ of\ pixels}{total\ number\ of\ pixels} \tag{2.2}$$

The value for each image is then averaged over the 653 image test set to give a final test accuracy of 80.04%.

An example result on an image from the Aeroscapes test set is presented in Figure 2.2. The trained segmentation model is also tested on a different aerial dataset taken from the authors of a previous UAV-UGV cooperative radiation detection work [9]. The dataset consists of 262 low flying UAV images taken at Kentland Farms at Virginia Tech, and it is labeled with

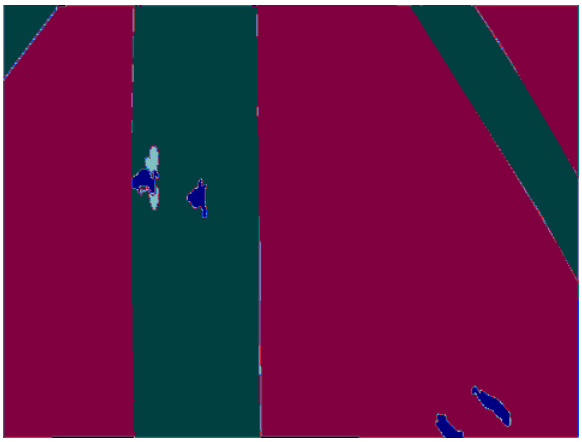
only four of the ground truth labels in Aeroscapes: road, vegetation, construction, and car. A result given an image from the Kentland dataset is shown in Figure 2.3. Here, images from the Kentland dataset provide a means of testing the model on data for which it is not specifically trained, but similar performance is expected due to the similar view points and content. The railroad tracks at the top left of the image seen in Figure 2.3a, not included in the ground truth human labeling, are classified by the network as mostly "contruction", and that area shows higher uncertainty. The area in the bottom center also shows higher uncertainty, where the cars are labeled in a mix of "background" and "bike". Errors are common, and expected, in a varying or real-world setting, but the uncertainty metric should provide a means of detecting such errors.

The dropout rate is set to 50%, where half of the neurons in the six central encoder-decoder layers are set randomly to zero, consistent with the qualitative suggestion in [23]. Dropout is activated at training time as a regularizer and at test time to approximate the posterior over the output segmentation results. From the approximate posterior, the uncertainty metric is then extracted for use in the robot planner.

2.2 A* Planner

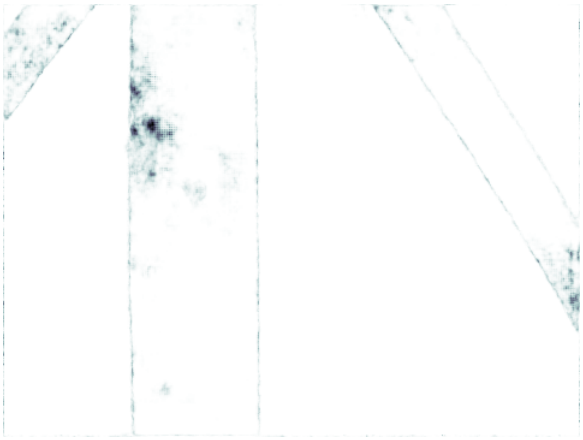
After the DL model is trained to produce reasonable results on images similar to the training set, the next step involves using the pixel segmentations as input to the A* search planner. In order to use semantic segmentation in robot planning, a cost function must be defined over the segmentation result to be used in the planning. The most common method of translating segment identity into a real number cost involves some quantification of the segmentation class traversability combined with the robot's speed. For example, in the previous work inspiring our problem setup, segment traversability is proportionate to the

(a) Input Aeroscapes overhead image



(b) Handcrafted ground truth segmentation

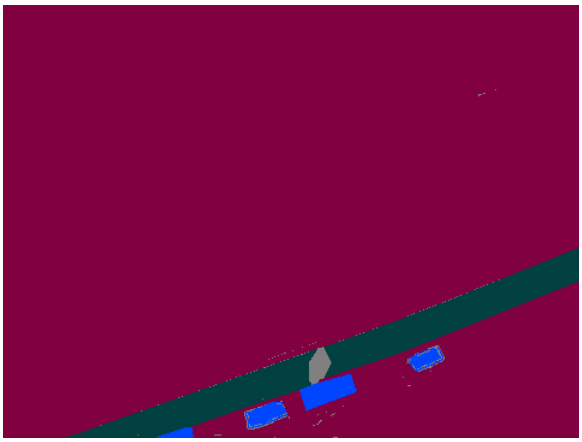
(c) Deep learning model segmentation



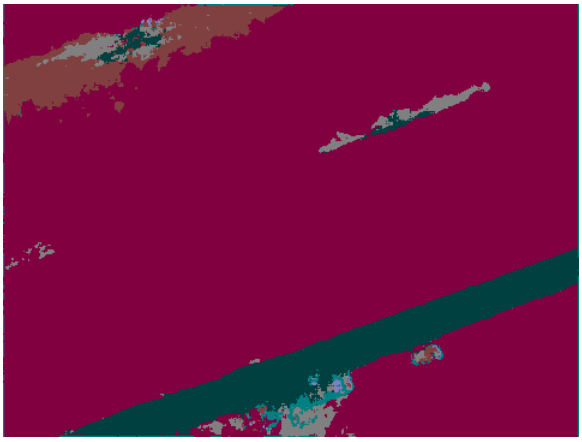
(d) Deep learning model uncertainty

Figure 2.2: Example semantic segmentation produced by the Bayesian SegNet model trained on Aeroscapes showing (a) an input image from the Aeroscapes test set with (b) its handcrafted ground truth segmentation, alongside (c) the resulting output segmentation, and (d) the model uncertainty associated with the output, with darker regions signifying higher uncertainty.

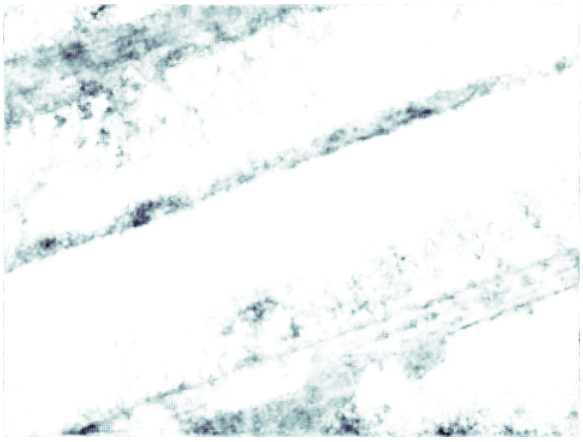
(a) Input Kentland overhead image



(b) Handcrafted ground truth segmentation



(c) Deep learning model segmentation



(d) Deep learning model uncertainty

Figure 2.3: Example semantic segmentation produced by the Bayesian SegNet model trained on Aeroscapes showing (a) an input image from the Kentland dataset with (b) its handcrafted ground truth segmentation, alongside (c) the resulting output segmentation, and (d) the model uncertainty associated with the output, with darker regions signifying higher uncertainty.

power consumption of the UGV based on prior experience [9]. However, for simplicity, here, the costs of the segment classes are set to a fixed value, as shown in Table 2.1. Once a cost is assigned, a start position and a goal position are defined in the input image, and the planning algorithm produces a path based on the cost assigned to the predicted pixel identification.

Chapter 3

Results and Discussion

In order to perform a qualitative analysis of the risk-neutral and risk-aware methods, the *surprise factor* is calculated to compare the expected path cost with the actual path cost by subtracting the two, then normalizing by the actual path cost for scaling, as shown in Equation 3. The path cost is found by summing up the cost associated with every pixel along the path. We use the predicted class of every pixel to determine the expected cost and the ground truth class of every pixel to determine the actual cost. If a path passes through pixels that the DL model classifies with low uncertainty, then we expect the predicted classes to be largely the same as the actual cost, thereby given a low surprise factor. On the other hand, if the predicted classes are wrong, then the surprise will be high.

$$surprise\ factor = \frac{|expected\ cost - actual\ cost|}{actual\ cost} \tag{3.1}$$

Unlike Shannon entropy, the surprise factor defined here is proportionate to the value of a path (based on the cost function) and not how probable the path is. Shannon's theory of information addresses the accuracy with which a model depicts the data it is meant to, but not the semantic and subjective dimensions of the data [4, 35]. In our work, we are concerned with the value associated with a prediction, and not only the accuracy of the prediction. A path cost prediction with a higher subjective value contributes more significantly to the

surprise factor, but not its entropy. This formulation of the surprise factor is consistent with a risk-aware setting, where not all predictions have the same risk value. That being said, underestimation and overestimation of the path cost is treated symmetrically, given the absolute difference and division by the actual cost. If penalizing underestimates to a higher degree is desirable, the sign of the difference or division by the expected cost would address this.

For an example setup, this work utilizes a DL image segmentation model to generate a cost map used in an A* path planner. Both a qualitative and a quantitative empirical analysis is performed using the deep learning model which is trained to segment images from the aerial viewpoint with the Aeroscapes and Kentland datasets [9, 31].

3.1 Qualitative Analysis

To depict the usefulness of the proposed setup in safety-critical situations, two scenarios are provided, one from Aeroscapes and the other from Kentland. In each qualitative example, the start and goal positions are selected to portray the implications of an imperfect deep learning model prediction, especially in situations where a human or surrounding vehicles are involved.

Figure 3.1 shows results given by the deep learning (DL) model and the subsequent planner. In this image taken from the Aeroscapes dataset, the pedestrian near the top center is not sufficiently segmented by the DL model prediction shown in Figure 3.1c [31]. Incorporating uncertainty associated with this prediction before passing it to the planner produces a more reasonable and risk-aware resulting path, as seen in Figure 3.1f. Higher levels of uncertainty are visualized by darker spots in Figure 3.1e. To generate these results, five stochastic passes are used to generate the prediction of variance in the output distribution, which is used as

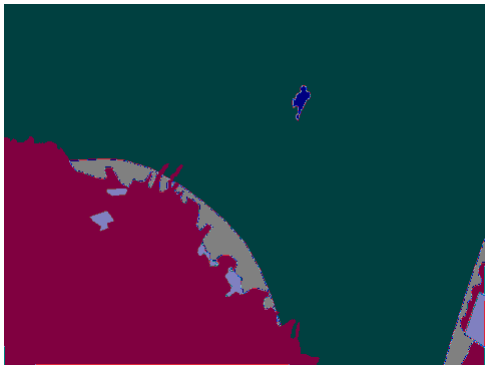
Measure	Ground Truth	Risk-Neural	Risk-Aware
Expected Cost	251	251	251
Actual Cost	251	430	251
Surprise Factor	0.000	0.416	0.000

Table 3.1: The surprise factor for the example in Figure 3.1, given ground truth, risk-neutral, and risk-aware cost.

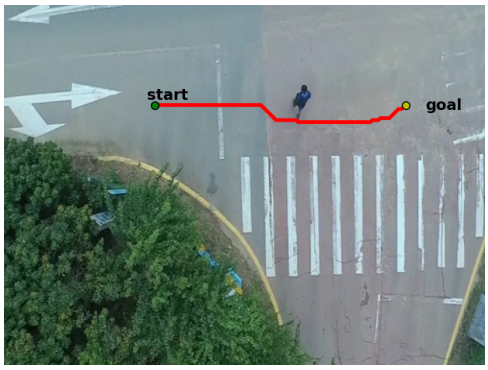
the uncertainty metric. The value of λ in Equation 2.1 is set to 10,000. λ here is relatively high, and this might be attributed to the DL model being trained on Aeroscapes and thus producing better predictions for images from the Aeroscapes test set. The pedestrian in Figure 3.1c is partially segmented as the correct class, which is a smaller prediction error compared to the coming example from a different dataset.

Table 3.1 shows the surprise factor in the three different planning scenarios of Figure 3.1: using ground truth segmentation, using DL model segmentation alone, and using DL model segmentation while taking into account its prediction uncertainty. When using DL segmentation alone, not accounting for model confidence increases the chance for a higher disparity between the expectation and reality, as seen in our example. On the other hand, in the risk-aware approach, the expectation better matches the reality and leads to lower surprise.

Shown in Figure 3.2 are qualitative results using the same DL model and subsequent planner on an image taken from the Kentland dataset [9]. In this image, from an entirely new dataset of images taken at similar aerial viewpoints to that which the DL model was trained on, the "vegetation" and "road" are better segmented than the "car" instances, as shown in Figure 3.2c. The cars are classified as a mix of the "obstacle" and "animal" classes, not showing consistency in that region of the image. When the uncertainty is computed using dropout, with five stochastic passes, the area corresponding to the cars shows high variance, depicted by the darker regions in Figure 3.2e. However, even in the region to the top left where "road" is consistently misclassified as "background", the uncertainty is high. This

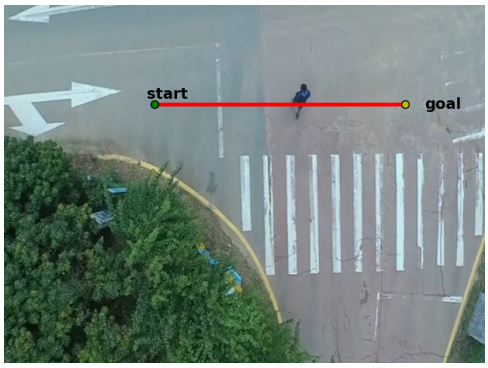


(a) Handcrafted ground truth segmentation

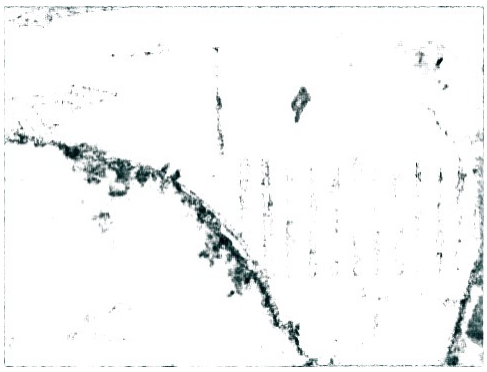


(b) Planning based on ground truth segmentation

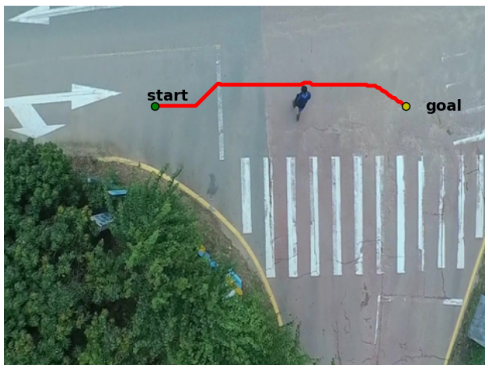
(c) DL model segmentation



(d) Planning based on DL model segmentation alone



(e) Uncertainty of DL model segmentation



(f) Planning based on DL model segmentation with uncertainty

Figure 3.1: Qualitative results on an Aeroscapes image showing the path planned from start to goal given (1) handcrafted ground truth image segments, (2) DL model segmentation alone, and (3) DL model segmentation with uncertainty.

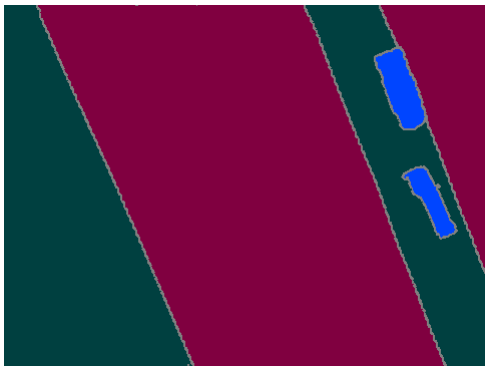
Measure	Ground Truth	Risk-Neutral	Risk-Aware
Expected Cost	246	246	310
Actual Cost	246	939	445
Surprise Factor	0.000	2.817	0.303

Table 3.2: The surprise factor for the example in Figure 3.2, given ground truth, risk-neutral, and risk-aware cost.

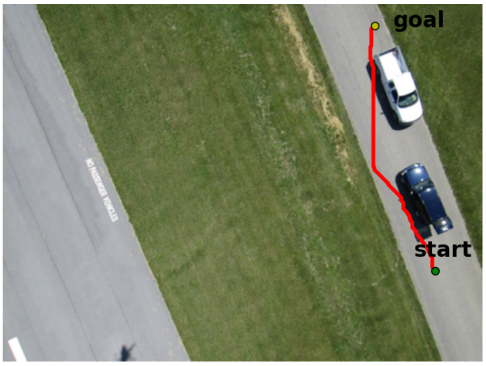
shows that the uncertainty metric is not merely a function the average prediction, but it provides valuable information about the quality of the prediction in its own right. With the DL model segmentation alone, the path from the selected start and goal positions does not entirely avoid the cars. A more risk-aware path is produced when uncertainty information is passed to the planner along with the segmentation, as seen in Figure 3.2f. The value of λ from Equation 2.1 is set to 1,500 in this example, which is much lower than the value used in the previous Aeroscapes example. This suggests that the method used for extracting uncertainty is more sensitive to unknown cases, such as an input that is outside the DL model's training distribution, and so the scale factor λ can be calibrated to a lower value for such data.

Table 3.2 shows the surprise factor in three planning scenarios of Figure 3.2. Although there is still a disparity between the expected and actual costs when using the proposed risk-aware method, the disparity is significantly lower than in the conventional risk-neutral case. The surprise factor depicts the significance, where the risk-neutral path collides with the vehicles sufficiently many times to produce a surprise factor of 2.8 as compared to the risk-aware path which produces a surprise of 0.3.

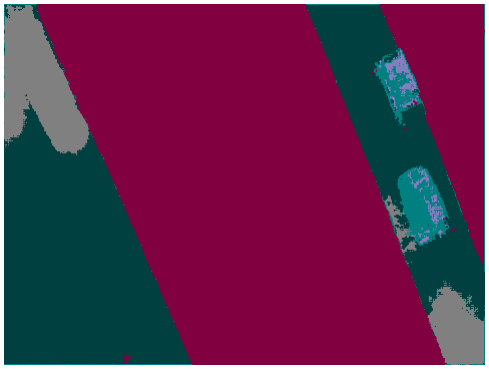
Figure 3.3 shows the effects of varying the value of lambda on the surprise factor given the Kentland aerial image and start and goal position shown in Figure 3.2. The paths from the same start and goal positions are computed for the risk-neutral case where $\lambda = 0$ and for the risk-aware cases where $\lambda > 0$. The values of λ are taken at intervals of 10 from zero to



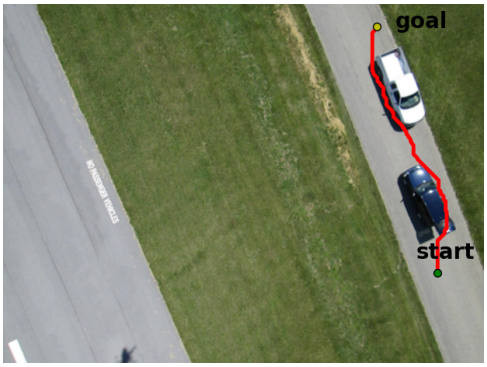
(a) Handcrafted ground truth segmentation



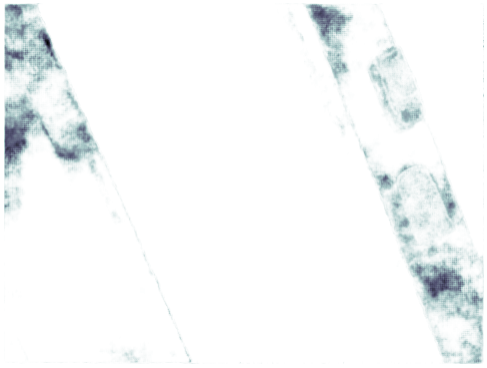
(b) Planning based on ground truth segmentation



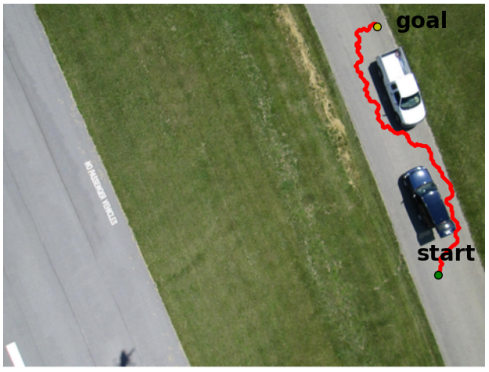
(c) DL model segmentation



(d) Planning based on DL model segmentation alone

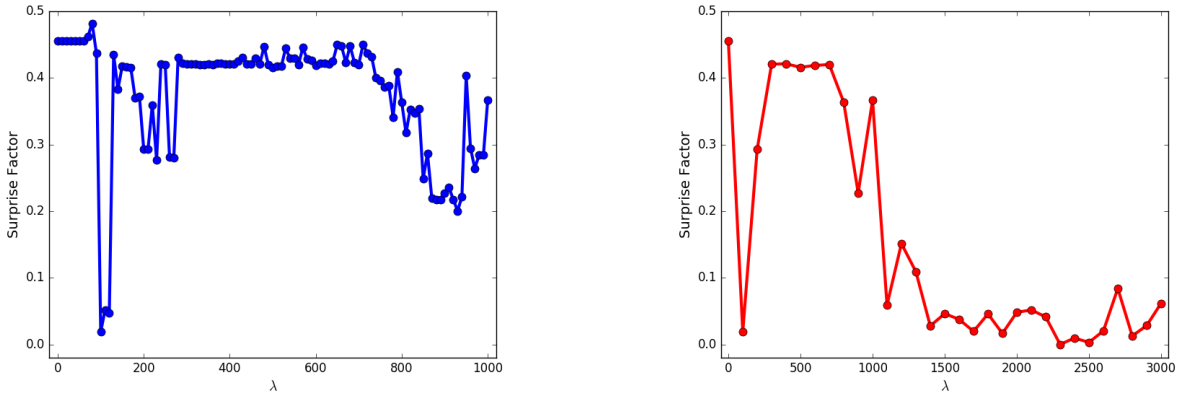


(e) Uncertainty of DL model segmentation



(f) Planning based on DL model segmentation with uncertainty

Figure 3.2: Qualitative results on a Kentland image showing the path planned from start to goal given (1) handcrafted ground truth image segments, (2) DL model segmentation alone, and (3) DL model segmentation with uncertainty.



(a) Intervals of 10

(b) Intervals of 100

Figure 3.3: Effect of varying λ on the surprise factor given the Kentland image and start and goal positions from Figure 3.2.

1,000, then intervals of 100 from zero to 3,000, in an attempt to find the best value of λ for the specific test case.

When λ is small, the surprise factor is large. This is consistent with previous findings, since $\lambda = 0$ corresponds to the risk-neutral case. As λ increases, the surprise factor decreases finally converging to a fixed value. This is because, once λ is sufficiently large, increasing λ further does not change the path produced as output significantly (except for a few pixels). In fact, for very large λ, the path found will be the minimum uncertainty path since the second term in Equation 2.1 dominates the first term. Therefore, the surprise factor remains largely the same.

Figure 3.4 shows the effects of varying the number of stochastic passes on the surprise factor given the Kentland aerial image and start and goal positions shown in Figure 3.2. The number of stochastic samples used in the uncertainty extraction has an effect on the resulting uncertainty metric, illustrated in Figure 3.5. Darker regions in Figure 3.5 depict a higher variance value given the stochastic samples for that individual pixel. Although the prediction of the uncertainty metric seems to become more fine grained, with higher variance

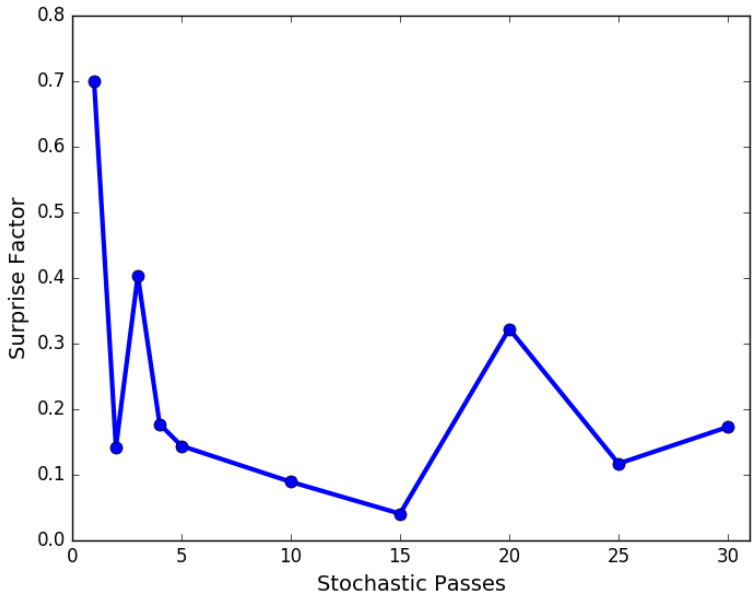


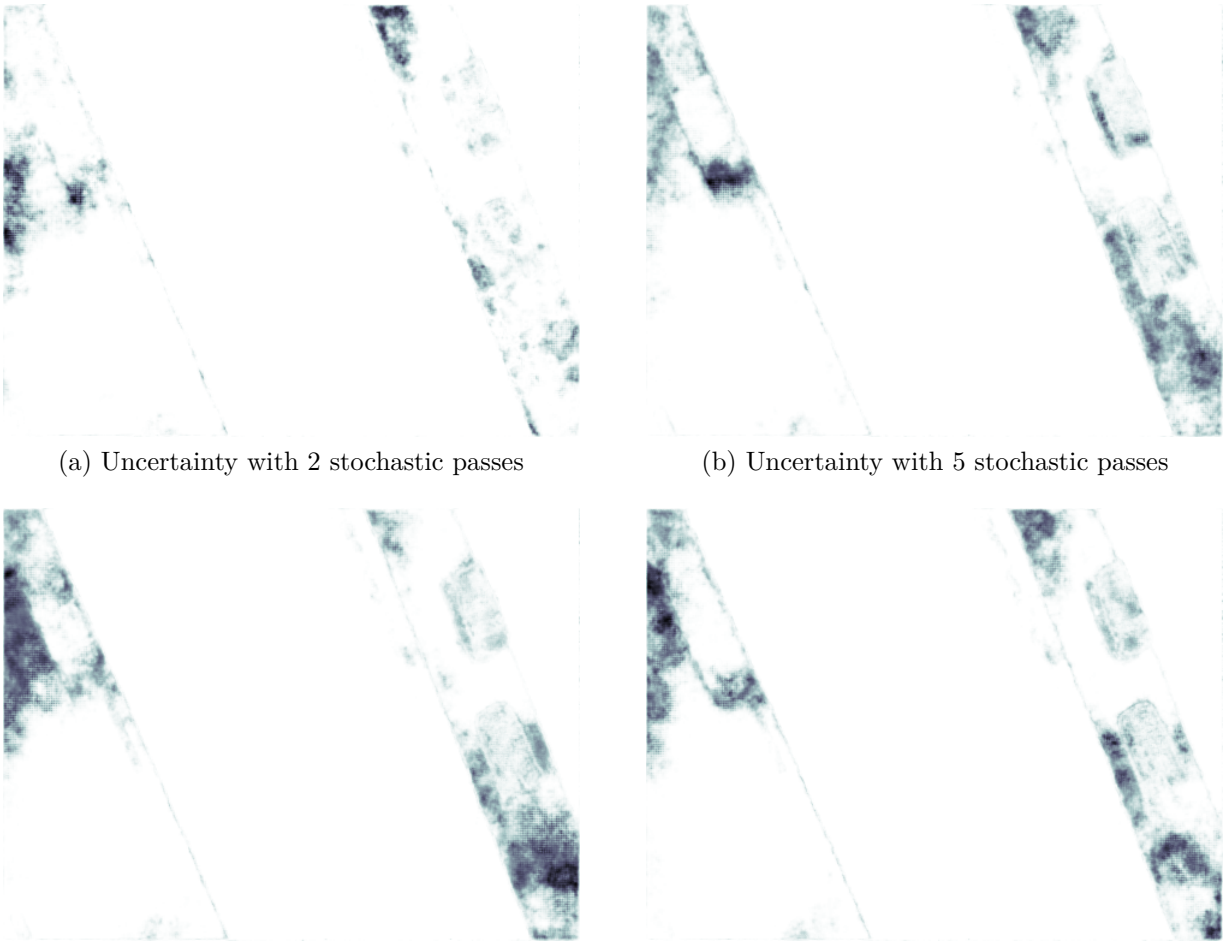
Figure 3.4: Effect of varying the number of stochastic passes on the surprise factor given the Kentland image with start and goal positions from Figure 3.2.

concentrations near the edges of a misclassified region, the benefits of increased computation diminishes after about five stochastic passes for this particular example.

3.2 Quantitative Results

To evaluate the proposed method more thoroughly, a quantitative analysis is performed on 20 selected images from the Kentland dataset. For each image, one to five random start and goal positions are selected for the empirical analysis. When a set of points is selected, the random pair is kept consistent throughout the given experiment.

Table 3.3 shows the trend for the average surprise factor calculated over the 20 images selected from the Kentland dataset with 5 randomly sampled starting and goal positions for each image. The trend observed is consistent with the single qualitative example in



(a) Uncertainty with 2 stochastic passes

(b) Uncertainty with 5 stochastic passes

(c) Uncertainty with 10 stochastic passes

(d) Uncertainty with 30 stochastic passes

Figure 3.5: Uncertainty predictions with varying the number of stochastic passes given the Kentland image from Figure 3.2, where darker regions signify higher uncertainty.

Table 3.2. The risk-neutral case still shows a higher overall surprise factor, although the disparity between the risk-neutral and risk-aware case is significantly lower than that in Table 3.2. The average case does again portrays the underestimation of the expected cost for the risk-neutral case.

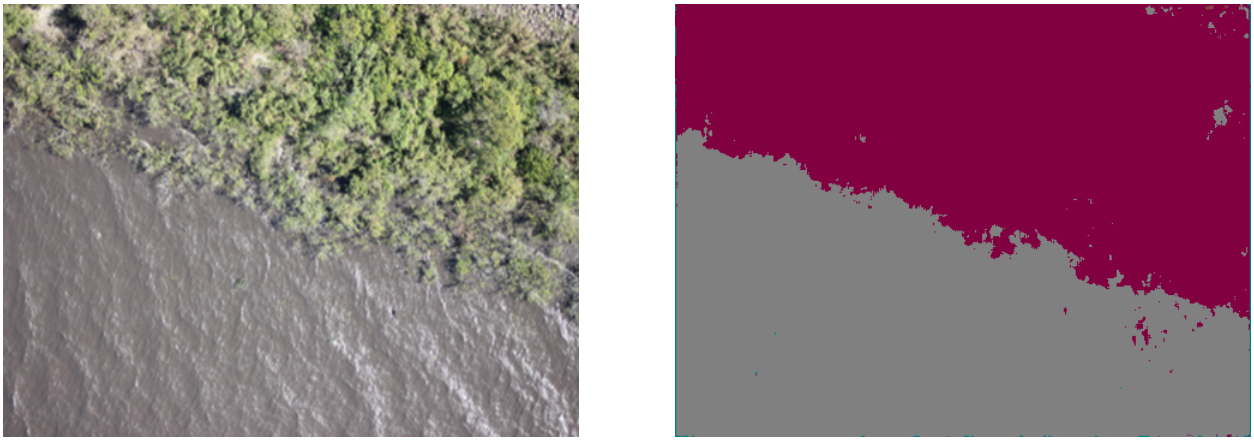
The results are very much dependent on the cost assignment selected for the semantic classes shown in Table 2.1. For example, the "background" class is semantically ambiguous, and is assigned by the DL model to several misclassified cases. The misclassifications include those with a ground truth label of "car" or even a class not provided, like "water". One such example is shown in Figure 3.6. The tendency of the DL model to classify something as "background" when it is unsure, suggests that a higher cost should be assigned to such a class.

Another possible reason the difference in risk-neutral and risk-aware surprise factor is not as large in the average case is the selection of the Kentland images. 20 images were selected at random, but not all images show interesting or diverse cost maps, even in ground truth. Images that contain few to none of the higher cost classes, corresponding to higher risk associated with those classes, will not show a significant change in surprise factor. Selecting five images from the set of 20 with the highest variation in classes, shows a more significant trend of lower surprise for the risk-aware case, as seen in Table 3.4. The same five randomly selected start and goal positions are considered for only five of the 20 images. Here, the risk-neutral case underestimates the expected cost with a higher disparity, resulting in a surprise factor of 0.43. This is compared to the risk-aware case, where the expected cost is underestimated to a lesser degree, but the surprise factor is lower with an average value of 0.28.

Figure 3.7 shows the effect of varying the λ parameter in Equation 2.1 on the surprise factor. For the same 20 images from the Kentland dataset, one random start and goal position is

Measure	Ground Truth	Risk-Neutral	Risk-Aware
Actual Cost	752.990	687.470	700.920
Expected Cost	752.990	1015.010	958.680
Surprise Factor	0.000	0.330	0.304

Table 3.3: The average surprise factor for 20 Kentland images and five randomly chosen start and goal positions each, given ground truth, risk-neutral, and risk-aware cost.



(a) Kentland image (b) Classification of water as "background"

Figure 3.6: Example Kentland image with no corresponding ground truth class for "water", so the DL model classifies that region as the ambiguous class "background".

Measure	Ground Truth	Risk-Neutral	Risk-Aware
Expected Cost	815.720	671.480	682.520
Actual Cost	815.720	1247.919	990.200
Surprise Factor	0.000	0.430	0.282

Table 3.4: The average surprise factor for five Kentland images with diverse cost maps and five randomly chosen starting and goal positions each, given ground truth, risk-neutral, and risk-aware cost.

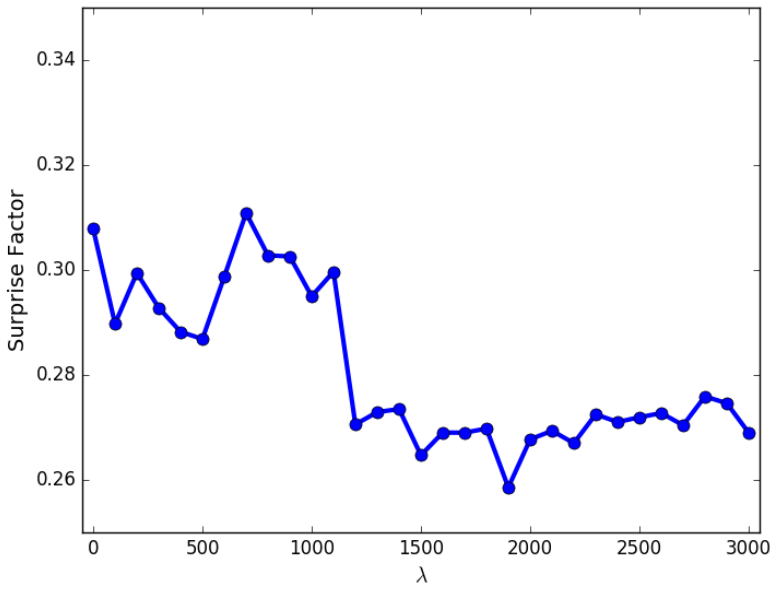


Figure 3.7: Effect of varying λ on the surprise factor averaged over 20 Kentland images with one random start and goal position each.

sampled for each image. Since the surprise factor is averaged over 20 images with the issues mentioned before, being the tendency of the DL model to classify unknown classes with the "background" label and low diversity cost maps, five of the 20 images are selected again to observe any changes to the surprise factor. The five images are chosen for the higher variation in the classes they portray, and the surprise factor is again observed for the same randomly sampled start and end positions for each given varying values of λ, shown in Figure 3.8. The effect of varying lambda is more significant for the five images with a more diverse cost map, and the surprise factor drops by a larger margin, from about 0.30 to 0.05. However, in both cases, increasing λ after a certain point does not produce any further reduction in surprise. The path found after λ reaches a threshold value, about 1200 in this experiment, will be the minimum uncertainty path since the second term in Equation 2.1 dominates the first term. Therefore, for larger λ the surprise factor remains almost the same.

The number of stochastic passes is also varied between one and 100. The runtime is observed

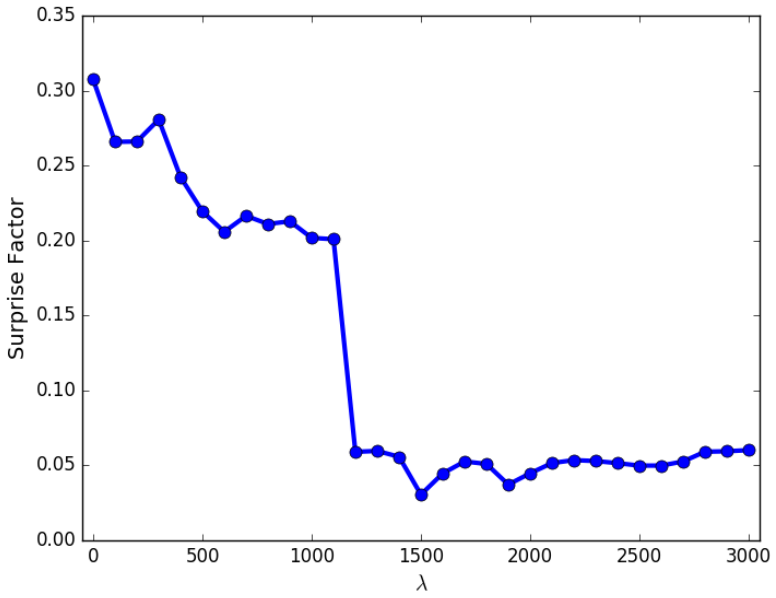


Figure 3.8: Effect of varying λ on the surprise factor averaged over five Kentland images with diverse cost maps given one random start and goal position each.

for one to five stochastic passes at intervals of one, then from five to 100 at intervals of five. This configuration is chosen due to the memory constraints on the GPU being used, the 8GB Nvidia GeForce GTX 1080, which has a maximum capacity for parallelizing five stochastic passes at once. Figure 3.9 shows the effects of increasing the number of stochastic passes during the uncertainty extraction on the runtime from neural network input to output. The average runtime is calculated for the same 20 Kentland images for each number of stochastic passes. It is clear that the runtime is magnified drastically after the maximum number of parallizable stochastic passes, five here, is reached.

The effects of varying the number of stochastic passes on the surprise factor is also observed. Figure 3.10 shows a similar trend to that of the qualitative example in Figure 3.4. Increasing the number of stochastic passes after a certain value, about five here, does not have a dramatic effect on the surprise factor. This is along with the the significant increase in computational time associated with a larger number of stochastic passes when they are not

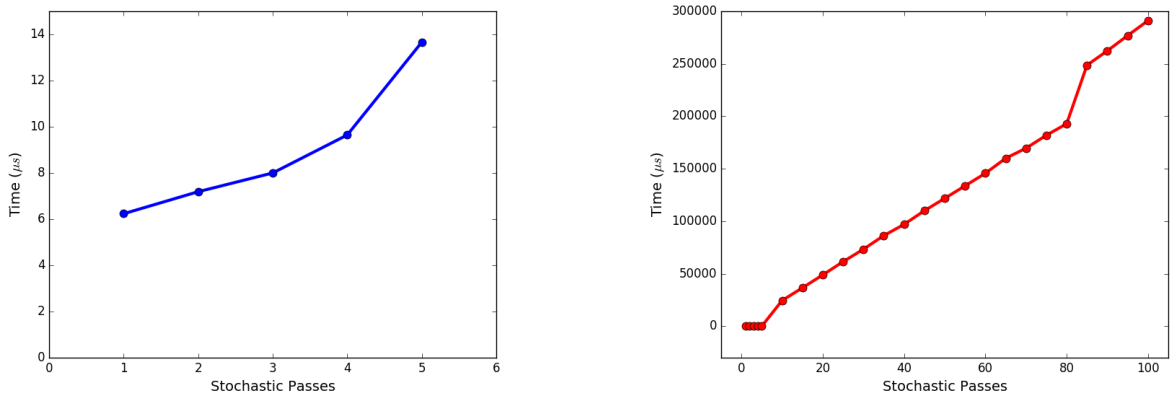


Figure 3.9: Effect of varying the number of stochastic passes on the runtime (microseconds) averaged over 20 Kentland images.

all computed in parallel. However, the 20 selected images and the selected cost function are shown to play an important role in displaying the effects on the surprise factor, so again, the five images with the most diverse cost maps are chosen. The effect of varying the number of stochastic passes is observed for the same start and goal positions for the five images. The surprise factor is shown to decrease at a higher rate between three and five stochastic passes, but increasing the number of stochastic passes after five still does not produce any further reduction in surprise factor. The surprise factor also shows higher variation across the number of passes, not converging to a small range, which may be attributed to the stochastic nature of the samples.

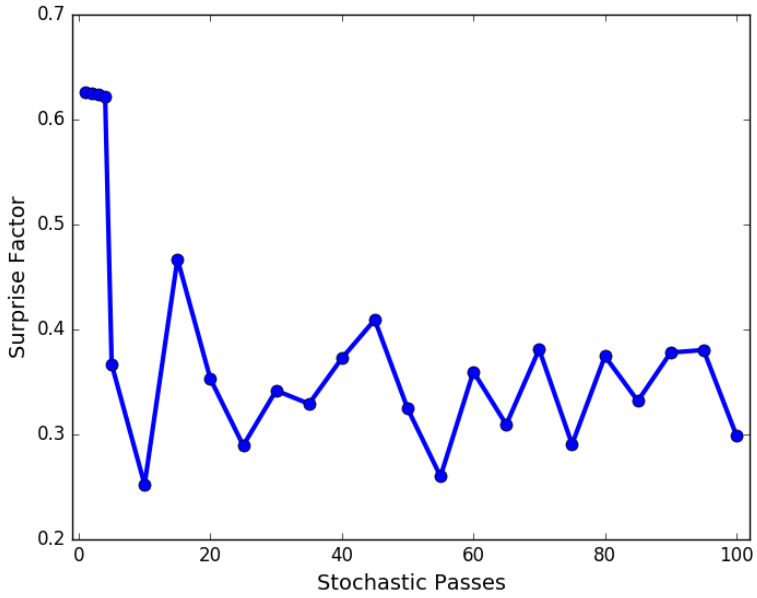


Figure 3.10: Effect of varying the number of stochastic passes on the surprise factor averaged over 20 Kentland aerial images, given one random start and goal position each.

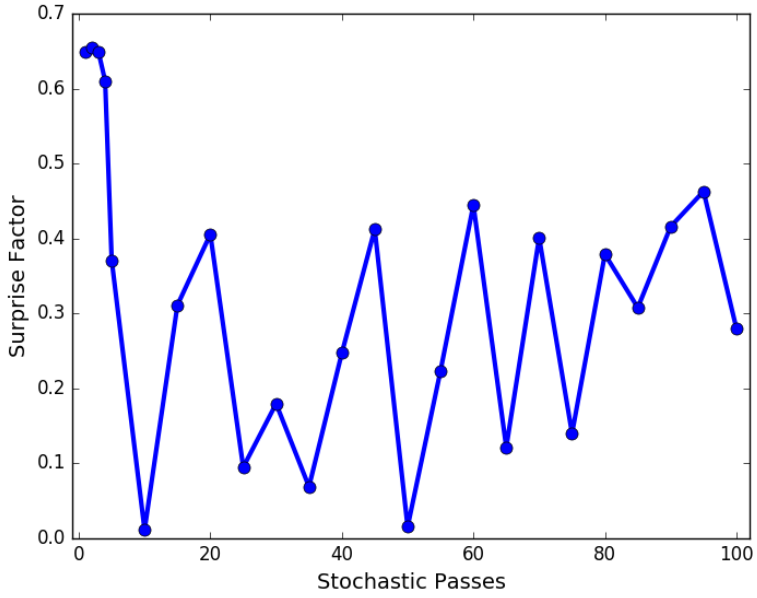


Figure 3.11: Effect of varying the number of stochastic passes on the surprise factor averaged over five Kentland images with diverse cost maps, given one random start and goal position each.

Chapter 4

Conclusions and Future Work

This work proposes a risk-aware approach to robot planning that already involves deep learning. Risk is quantified by the perception model prediction uncertainty in the planning process. When deep learning is used for perception as a portion of the planning loop, an understanding of confidence in deep learning (DL) estimates is useful. Uncertainty is extracted directly from the DL models utilizing dropout as a practical method, especially in resource constrained settings such as robotics. Promising results show that including uncertainty in a planner provides better predictability of actions, and even the avoidance of catastrophic actions in a safety-critical setting. Qualitative experiments and quantitative empirical analysis show the potential of this method for real-world systems by applying the method to unseen real-world data.

4.1 Lessons Learned

In this thesis, a risk-aware framework is developed and empirically analyzed to explore its potential in safety-critical robotics settings. Experiments involving a deep learning (DL) model image semantic segmentation, followed by an A* path planner, are used to illustrate the utility of the framework. A *surprise factor* is defined to compare the risk-aware setup with the standard risk-neutral case. To account for risk, uncertainty in the DL perception is considered as a part of the path planning cost function. It is found that the surprise factor

is lower, in high-risk test cases as well as on average, given the risk-aware setup if threshold values of hyperparameters in uncertainty extraction are reached.

Key components of the experiments are varied to observe the effects on the overall system. Hyperparameters in the uncertainty extraction process are analyzed for their impact on the efficacy and efficiency of the uncertainty metric being used to quantify risk. Dropout is used as a stochastic regularizer during DL training, then adapted to produce stochastic samples at test time from which a probability distribution is formed. The variance over the sampled posterior is used as the uncertainty metric.

First the effects of the weighting parameter λ in the proposed altered cost function given in Equation 2.1 is observed. With both a qualitative (anecdotal) and quantitative (average) analysis, the scaling factor λ plays an important role in calibrating the uncertainty metric over differing datasets. It is observed that λ must be sufficiently large in order to scale the uncertainty metric with the assigned prediction costs. Otherwise, the second term in Equation 2.1 is not considered by the planner. Once λ reaches a threshold value, uncertainty is adequately accounted for by the planner, and so no added benefit arises with increasing the value of λ any further. This is illustrated when the surprise factor is evaluated across a range of values for λ. However, since the uncertainty metric itself is not calibrated across several datasets in this work, when new data is being evaluated, it is advisable to set a relatively high λ as a precaution [15]. This is true in the face of evidence produced in this thesis displaying higher sensitivity of the uncertainty metric, and thus a need for lower λ, when data outside the training distribution is introduced.

The impact of the number of stochastic samples on the surprise factor and runtime is also studied. It is shown that like the effects of the scaling factor λ, the number of stochastic samples used to estimate the variance over the posterior has a threshold value, before which it produces an insufficient estimate. However, unlike λ, the number of samples has significant

adverse effects on the computational time, especially in a resource constrained setting like robotics. With the potential for overheating and added weight given larger GPU capacity, a relatively small GPU is used in experimentation, limiting the number of stochastic samples that can be taken in parallel to five. After five, the negative impact of further samples on runtime outweighs the benefits of a small reduction in surprise factor to the system.

The formulation of the original cost function, provided by a fixed predefined value for illustrative purposes in this thesis, is shown to have significant impact on the analysis. Ambiguous perception definitions, like the class "background" in aerial images, should be assigned higher costs for the high likelihood of their misclassification. In the empirical analysis, it is also observed that the proposed risk-aware method shows much higher potential given a diverse cost map (i.e. perception representation), with lower benefit but no added drawbacks in situations where the cost map is uniform.

4.2 Future Work

The main anticipated extensions of this work involve investigating different uncertainty metrics extracted from the same experiment setup presented in this thesis, along with exploring the end-to-end joint perception and planning setup. Preliminary experiments and results for the latter are provided below, as well as the open questions to be answered with further development.

4.2.1 Risk Metrics

In both the machine learning and robotics communities, in an attempt to make processes safer, risk metrics have been employed from other fields. The two main areas from which

risk metrics have been adapted are business and information theory.

From Business

Risk has been a long studied value in business, and as such, there exists a vast literature on the subject in the business community. One of the main risk measures used in business, which has also been adapted in robotics, is the value at risk (VaR), and more recently the conditional value at risk (CVaR). CVaR and VaR both quantify the costs that might be encountered in the tail of a cost distribution [8]. VaR is a measure of risk as the maximum cost that might be incurred with respect to a given confidence level. CVaR is the expected cost given with the cost being greater than or equal to the VaR at that confidence level. CVaR is found to have added theoretical and computational benefits to VaR [8]. CVaR is also considered more intuitive, as it places more emphasis on current costs rather than future costs [8]. Worse-case is another risk measure widely used in business planning, and later used in robotics as well. The worse-case criterion focuses on the portion of the probability distribution that amounts for the largest losses, which tends to overestimate risk [8]. Due to its theoretical and intuitive benefits, extracting and utilizing CVaR from the experimental setup used in this thesis may prove to be more useful than using the combination of mean and variance.

From Information Theory

Information theory itself is a well-established field of study [35], however the methods it has produced for the signal processing community have only recently been applied to machine learning. Such methods include concepts of uncertainty quantification that are model-agnostic. One such uncertainty metric is entropy [12, 13]. Entropy captures the average

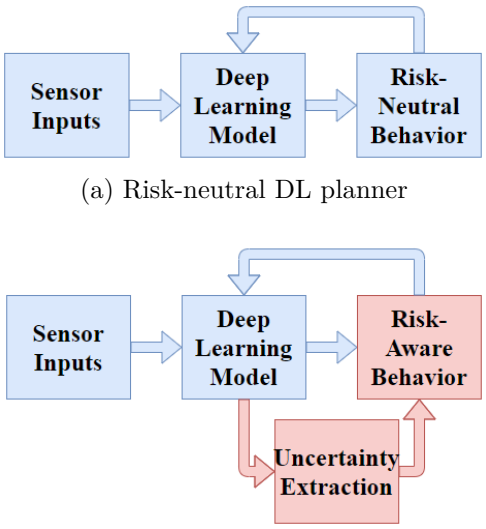
amount of information contained in a predictive distribution. Another commonly used value is a manipulation of various entropies in the formulation of mutual information provided by a model [12]. Mutual information is measured between the prediction and the posterior over model parameters, being how much one value informs of the other [12]. It is worth exploring the potential benefit of utilizing an information theoretic risk metric in our proposed risk-aware planner.

4.2.2 End-to-End Planner

Deep reinforcement learning (DRL) is an emerging contributor to exceptional robot planning, as an end-to-end robot system can be trained using trial and error in polynomial time. The applications of DRL could be enhanced by incorporating uncertainty information in the training process as well as in real-time, with some theoretical and experimental proof showing shorter training times and better overall performance [12]. The proposed system uses a DRL model for end-to-end planning, given high dimensional inputs such as images, and provides a realistic uncertainty measure at the output which can then be used to promote risk-aware behavior in training and at test time.

Experimental Setup

The preliminary experiment uses a game simulation software, in an end-to-end deep reinforcement learning (DRL) planner. Oftentimes, robotic settings are comparable to a game in which the robot desires to maximize its reward whilst taking into account the dynamics of the system. In a DRL framework, the robot system dynamics are implicitly encoded to the learning model involved through trial-and-error encounters with the environment. To avoid catastrophic errors and to produce more consistent, as opposed to erratic, learning,



(a) Risk-neutral DL planner

(b) Risk-aware DL planner

Figure 4.1: The overall flow of information in a robot planner which relies entirely on a DL model for its learning and decision making in (a) the risk-neutral and (b) risk-aware cases.

uncertainty associated with the predicted next move could provide useful information.

To extract the uncertainty metric, dropout is utilized, as in the experimental setup of this thesis. During training, the confidence measure is fed back through the DRL model to promote the learning of a more risk-aware policy. At test time, the uncertainty provides a means to quantify the risk associated with a chosen action.

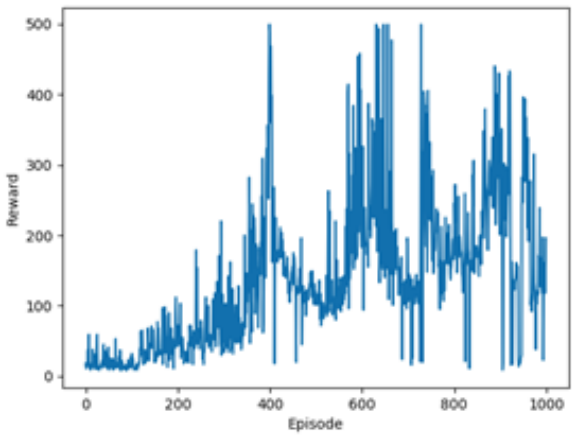
The end-to-end DRL system setup is initially tested on a cartpole game problem produced by the OpenAI Gym framework. The deep Q-learning network (DQN) is used for the DRL process [30]. No fixed training set is required in a DRL system, and so the DQN is initialized with random weights. The DQN is built using Keras, a high level deep learning library built on top of Google's Tensorflow [7]. At each phase of the learning process, the current state of the cartpole system is input to the DQN, and an estimate of Q-value of the next action to take is given as the output, based on the expected next state from prior experience. The adaptive ϵ-greedy algorithm is used to balance the amount of exploration done by the system at a given point in its learning [37]. More exploration is desirable at the early stages

of learning, but this can cause sporadic results which can be managed based on an added uncertainty metric. The uncertainty is weighted by some factor and added to the Q-value as it is fed back to the deep reinforcement learner to provide a better understanding of the model confidence, which would also manage exploration in an intuitive way.

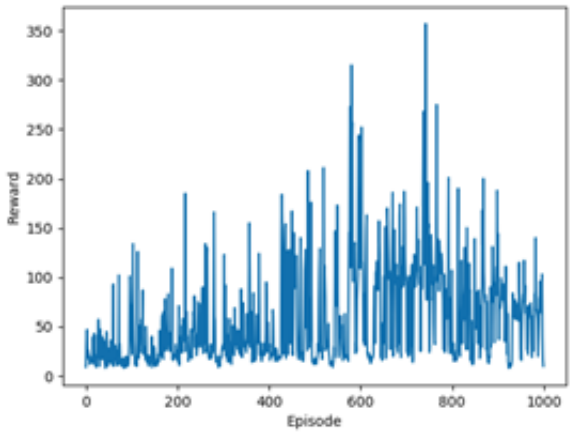
Preliminary Results

For the end-to-end planning system, the results are analyzed over a thousand learning episodes in the case of deep reinforcement learning (DRL) using the traditional deep Q-learning network (DQN) to solve the cartpole game problem. At each episode, the Q-value is given by the outputs of the DQN. Based on this Q-value an action is selected, which in the case of cartpole is to move either left or right to keep the pole upright. Once an action is taken, a reward is given by the cartpole system, being the amount of iterations the pole remains upright after a certain action. The reward is observed over several learning episodes and a statistical analysis is performed.

After training the DQN for 1,000 episodes, the overall learning is assessed in both the risk-neutral and risk-aware cases. The Q-value is considered alone in the decision for the next action in each episode for the risk-neutral original approach, while the uncertainty is weighted and added to the Q-value before considering the next action in our risk-aware method. As shown in the corresponding plots in Figures 4.2 and 4.3 and the statistical values displayed in Table 4.1, the average reward over all episodes is higher in the risk-neutral method. However, the reward also varies considerably greater in the risk-neutral method as compared to the risk-aware adaptation, as depicted in the overall variance. This suggests that the behavior of the DQN is much more predictable, and therefore more safely steered away from catastrophic low rewards at a higher rate, even though it returns lower average rewards. It would be interesting to observe the trends with a longer period of learning and using uncertainty

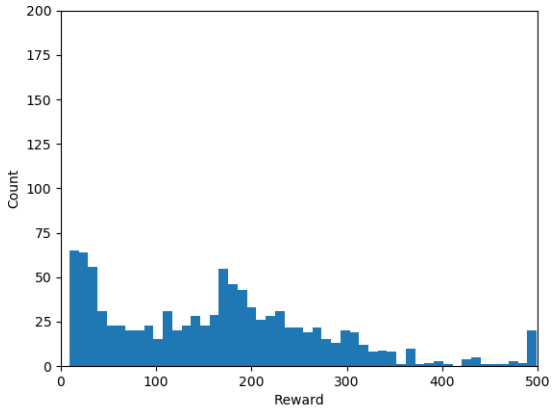


(a) Risk-neutral

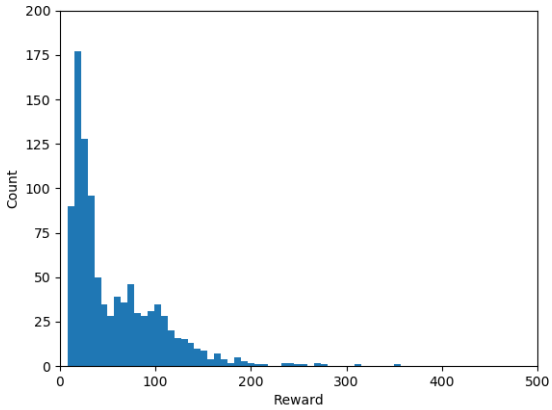


(b) Risk-aware

Figure 4.2: Reward per episode of learning using DQN for cartpole displaying (a) the risk-neutral traditional method and (b) the risk-aware method that incorporates uncertainty.



(a) Risk-neutral



(b) Risk-aware

Figure 4.3: Histogram of rewards during learning using DQN for cartpole displaying (a) the risk-neutral traditional method and (b) the risk-aware method that incorporates uncertainty.

metrics other than variance.

Reward	**Risk-Neutral**	**Risk-Aware**
Mean	136.238	56.988
Mode	18 (count = 14)	18 (count = 31)
Variance	10272.359	2320.114
Standard Deviation	101.353	48.168

Table 4.1: DQN cartpole statistics comparing risk-neutral and risk-aware methods.

Bibliography

[1] John Aloimonos, Isaac Weiss, and Amit Bandyopadhyay. Active vision. *International Journal of Computer Vision*, 1(4):333–356, 1988.

[2] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*, 2015.

[3] Ruzena Bajcsy. Active perception. 1988.

[4] Pierre Baldi. A computational theory of surprise. In *Information, Coding and Mathematics*, pages 1–25. Springer, 2002.

[5] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 2006.

[6] Sungjoon Choi, Kyungjae Lee, Sungbin Lim, and Songhwai Oh. Uncertainty-aware learning from demonstration using mixture density networks with sampling-free variance modeling. *arXiv preprint arXiv:1709.02249*, 2017.

[7] François Chollet et al. Keras. https://keras.io, 2015.

[8] Yinlam Chow, Mohammad Ghavamzadeh, Lucas Janson, and Marco Pavone. Risk-constrained reinforcement learning with percentile risk criteria. *Journal of Machine Learning Research*, 18:167–1, 2017.

[9] Gordon Christie, Adam Shoemaker, Kevin Kochersberger, Pratap Tokekar, Lance McLean, and Alexander Leonessa. Radiation search operations using scene understanding with autonomous uav and ugv. *Journal of Field Robotics*, 34(8):1450–1468, 2017.

[10] Stefan Depeweg, José Miguel Hernández-Lobato, Finale Doshi-Velez, and Steffen Udluft. Learning and policy search in stochastic dynamical systems with bayesian neural networks. *arXiv preprint arXiv:1605.07127*, 2016.

[11] Laurene V Fausett et al. *Fundamentals of Neural Networks: Architectures, Algorithms, and Applications*, volume 3.

[12] Yarin Gal. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, January 2017.

[13] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059, 2016.

[14] Yarin Gal, Rowan McAllister, and Carl Edward Rasmussen. Improving pilco with bayesian neural network dynamics models. In *Data-Efficient Machine Learning workshop, ICML*, 2016.

[15] Yarin Gal, Jiri Hron, and Alex Kendall. Concrete dropout. In *Advances in Neural Information Processing Systems*, pages 3581–3590, 2017.

[16] Dhiraj Gandhi, Lerrel Pinto, and Abhinav Gupta. Learning to fly by crashing. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 3948–3955. IEEE, 2017.

[17] Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. In *Journal of Machine Learning Research*, volume 16.

[18] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*, volume 1.

[19] Jeff Heaton. Artificial intelligence for humans, volume 3: Deep learning and neural networks; heaton research. *Inc.: St. Louis, MO, USA*, 2015.

[20] Gregory Kahn, Adam Villaflor, Vitchyr Pong, Pieter Abbeel, and Sergey Levine. Uncertainty-aware reinforcement learning for collision avoidance. *arXiv preprint arXiv:1702.01182*, 2017.

[21] Gabriel Kalweit and Joschka Boedecker. Uncertainty-driven imagination for continuous deep reinforcement learning. In *Conference on Robot Learning*, pages 195–206, 2017.

[22] Alex Kendall and Roberto Cipolla. Modelling uncertainty in deep learning for camera relocalization. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 4762–4769. IEEE, 2016.

[23] Alex Kendall, Vijay Badrinarayanan, and Roberto Cipoll. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *arXiv preprint arXiv:1511.02680*, 2015.

[24] Igor Kononenko. Bayesian neural networks. *Biological Cybernetics*, 61(5):361–370, 1989.

[25] Ioannis Kostavelis and Antonios Gasteratos. Semantic mapping for mobile robotics tasks: A survey. *Robotics and Autonomous Systems*, 66:86–103, 2015.

[26] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

[27] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436, 2015.

[28] Janno Johan Maria Lunenburg, Sebastiaan Antonius Maria Coenen, Gerrit JL Naus, Marinus Jacobus Gerardus van de Molengraft, and Maarten Steinbuch. Motion planning for mobile robots: A method for the selection of a combination of motion-planning algorithms. *IEEE Robotics & Automation Magazine*, 23(4):107–117, 2016.

[29] Anirudha Majumdar and Marco Pavone. How should a robot assess risk? towards an axiomatic theory of risk in robotics. *arXiv preprint arXiv:1710.11040*, 2017.

[30] 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, 2015.

[31] Ishan Nigam, Chen Huang, and Deva Ramanan. Ensemble knowledge transfer for semantic segmentation. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1499–1508. IEEE, 2018.

[32] Petter Nilsson, Sofie Haesaert, Rohan Thakker, Kyohei Otsu, Cristian-Ioan Vasile, Ali-akbar Agha-mohammadi, Richard M Murray, and Aaron D Ames. Toward specification-guided active mars exploration for cooperative robot teams.

[33] Masahiro Ono, Thoams J Fuchs, Amanda Steffy, Mark Maimone, and Jeng Yen. Risk-aware planetary rover operation: Autonomous terrain classification and path planning. In *Aerospace Conference, 2015 IEEE*, pages 1–10. IEEE, 2015.

[34] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. *Advances in Neural Information Processing Systems*, pages 4026–4034, 2016.

[35] Claude Elwood Shannon. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423, 1948.

[36] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[37] Michel Tokic. Adaptive ε-greedy exploration in reinforcement learning based on value

differences. In *Annual Conference on Artificial Intelligence*, pages 203–210. Springer, 2010.

[38] Kuo-Hao Zeng, Shih-Han Chou, Fu-Hsiang Chan, Juan Carlos Niebles, and Min Sun. Agent-centric risk assessment: Accident anticipation and risky region localization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, page 6, 2017.