Algoritmen op basis van machinaal leren voor gestructureerde besluitvorming

Machine Learning Algorithms for Structured Decision Making

Rein Houthooft

Promotor: prof. dr. ir. F. De Turck
Proefschrift ingediend tot het behalen van de graad van
Doctor in de ingenieurswetenschappen: computerwetenschappen

Vakgroep Informatietechnologie
Voorzitter: prof. dr. ir. B. Dhoedt
Faculteit Ingenieurswetenschappen en Architectuur
Academiejaar 2016 - 2017

UNIVERSITEIT
GENT

ISBN 978-90-8578-986-4
NUR 980
Wettelijk depot: D/2017/10.500/21

Ghent University
Faculty of Engineering and Architecture
Department of Information Technology

imec
Internet Technology and Data Science Lab

Supported by a Doctoral Fellowship of the
Research Foundation – Flanders (FWO)

Dissertation for acquiring the grade of
Doctor of Computer Science Engineering

Acknowledgments

Het is zover, na een kleine drie jaar dien ik mijn doctoraatsthesis in. Ik zou hier nooit geraakt zijn zonder de gulle steun van enkele bijzondere mensen. Een welgemeende dankuwel,

aan Krystle, dat je me doorheen mijn studies bent blijven steunen. Zonder jou zou dit me nooit gelukt zijn;

aan mijn familie, die me alle mogelijkheden bood om te doen wat ik wou, met vallen en opstaan;

aan Filip, dat je mij de kans tot wetenschappelijk onderzoek gaf, alsook alle vrijheid tot het bewandelen van mijn eigen onderzoekspad;

to everyone at the Berkeley AI Research lab, thanks for your support, especially Pieter, that you gave me the chance to be part of your group;

to everyone at OpenAI, for the great research environment and support through computing resources;

aan het Fonds voor Wetenschappelijk Onderzoek – Vlaanderen (FWO), welke via zowel een aspirantenmandaat als een reisbeurs mijn onderzoek ondersteunden, zodat ik dit volledig op mijn eigen manier kon inkleuren;

to all members of the examination board, for your constructive feedback;

aan iedereen aan de Universiteit Gent, op bureau, in de wandelgangen, het waren mooie tijden. Tevens bijzondere dank aan Brecht, voor de vele gezellige koffies.

Rein Houthooft
februari 2017

Table of Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

AE	autoencoder
BASS	Basic Abstraction of the Screenshots
BNN	Bayesian neural network
CRF	conditional random field
CNN	convolutional neural network
DQN	Deep Q-networks
ELU	exponential linear unit
ERWR	episodic reward-weighted regression
GPU	graphics processing unit
KL	Kullback-Leibler
LSH	locality-sensitive hashing
MAP	maximum a posteriori
MBIE	model-based interval estimation
MDP	Markov decision process
MRF	Markov random field
POMDP	partially-observable Markov decision process
ReLU	rectified linear unit
RL	reinforcement learning
SGD	stochastic gradient descent or subgradient descent
SSVM	structural support vector machine
TRPO	Trust Region Policy Optimization
UCB	upper confidence bound
VIME	Variational Information Maximizing Exploration

Summary

Machine Learning Algorithms for Structured Decision Making

This dissertation explores the realm of endowing artificial systems with the intelligence to learn how to make structured decisions. The first part focuses on structured decision making in which decisions are correlated in the spatial domain, while the second part focuses on making temporally-structured decisions.

Tackling pattern recognition problems in areas such as computer vision, bioinformatics, and speech recognition is often helped by taking into account task-specific statistical relations between output variables. These predicted variables can be regarded as decisions generated by the system at hand. When outputs are generated by taking into account their interrelations, we call it structured prediction. In particular, the first part of this thesis investigates the application of a class of methods called structural support vector machines (SSVMs), and proposes several fundamental extensions.

First, these models are investigated in light of overcoming a key challenge in the realization of autonomous vehicles, namely the machine's ability to perceive its surrounding environment. Structured prediction models are applied to the vehicle's input camera data stream in order to obtain a semantic segmentation, in which each image pixel is assigned a particular semantic class, such as 'road' or 'object'. For coherent segmentation, an SSVM is modeled to encode label distributions conditioned on the input images, taking into account visual contextual cues between adjacent pixel regions within a second-order neighborhood.

SSVMs are nonprobabilistic models that optimize a joint input-output function through margin-based learning. The structured model is formulated as an energy-based function using feature-dependent unary potentials, and pairwise potentials that differentiate between first- and second-degree

region neighbors. After optimizing this model through max-margin learning, based on a semantic loss function, efficient classification is realized via graph cuts inference using α-expansion. It is shown through quantitative and qualitative analyses that by taking into account contextual relations between pixel segmentation regions within a second-degree neighborhood, spurious segmentation label assignments are filtered out, leading to highly accurate semantic segmentations for outdoor scenes. Furthermore, we investigate these models in an autonomous vehicle use case, while also proposing an end-to-end segmentation method using deep convolutional neural networks as an alternative model.

Because SSVMs generally disregard the interplay between unary and interaction factors during the training phase, final parameters are suboptimal. Moreover, its factors are often restricted to linear combinations of input features, limiting its generalization power. To improve prediction accuracy, we improve SSVMs through: (i) Joint inference and learning by integration of back-propagation and loss-augmented inference in SSVM subgradient descent; (ii) Extending SSVM factors to neural networks that form highly nonlinear functions of input features. This model departs from the traditional bifurcated approach in which a unary classifier is trained independently from the structured predictor.

Results on a complex image segmentation task show that end-to-end SSVM training, and/or using neural factors, leads to more accurate predictions than conventional subgradient descent and N-slack cutting plane training. Furthermore, a deep convolutional implementation of this method is applied in an autonomous vehicle perception use case. The results indicate that the proposed model serves as a foundation for more advanced structured models, e.g., by using latent variables, learned feature representations, or complexer connectivity structures.

The second part of this dissertation investigates predictions that are correlated in the temporal domain. More specifically, we investigate reinforcement learning (RL), in which an agent tries to achieve a specific goal in an initially unknown environment. Because accomplishing this goal requires more than a single action, they can be seen as temporally-structured decisions since each action affects the next one. Several fundamentally new methods are proposed for deep RL—reinforcement learning using deep neural networks as function approximators—which are applied to robot locomotion learning as well as autonomous video game playing.

In particular, we notice that scalable and effective exploration remains a key challenge in deep RL. While there are methods with optimality guarantees in the setting of discrete state and action spaces, these methods cannot be applied in high-dimensional deep RL scenarios. As such, most contemporary RL relies on simple heuristics such as ϵ-greedy exploration or adding Gaussian noise to the controls. Therefore, the agent tends to lock prematurely onto decision sequences that are suboptimal, rather than taking exploratory actions to potentially improve its reward in the long term.

This dissertation introduces Variational Information Maximizing Exploration (VIME), an exploration strategy based on maximization of information gain about the agent's belief of environment dynamics. It is a curiosity-driven exploration strategy for continuous control tasks. We propose a practical implementation, using variational inference in Bayesian neural networks (BNNs) that efficiently handles continuous state and action spaces. Variational inference is used to approximate the posterior distribution of a BNN that represents the environment dynamics. VIME modifies the Markov decision process (MDP) reward function, and can be applied with several different underlying RL algorithms. We demonstrate that VIME achieves significantly better performance compared to heuristic exploration methods across a variety of continuous control tasks and algorithms, including environments with very sparse rewards.

Alternatives to curiosity-based exploration are count-based exploration algorithms. These methods are known to perform near-optimally when used in conjunction with tabular RL methods for solving small discrete MDPs. It is generally thought that count-based methods cannot be applied in high-dimensional state spaces, since most states will only occur once. Recent deep RL exploration strategies, such as VIME, are able to deal with high-dimensional continuous state spaces through complex heuristics, often relying on optimism in the face of uncertainty or intrinsic motivation.

We describe a surprising finding: A simple generalization of the classic count-based approach can reach near state-of-the-art performance on various high-dimensional and/or continuous deep RL benchmarks. States are mapped to hash codes, which allows to count their occurrences with a hash table. These counts are then used as a reward bonus, according to the classic count-based exploration theory. We find that simple hash functions can achieve surprisingly good results on many challenging tasks.

Furthermore, it is shown that a domain-dependent learned hash code may further improve these results. Detailed analysis reveals important aspects of a good hash function: (i) Having appropriate granularity; (ii) Encoding information relevant to solving the MDP. This exploration strategy achieves near state-of-the-art performance on robot locomotion tasks and Atari 2600 games, while providing a simple yet powerful baseline for solving MDPs that require considerable exploration.

Samenvatting

Algoritmen op basis van machinaal leren voor gestructureerde besluitvorming

Deze scriptie onderzoekt het ontwikkelen van artificiële intelligente systemen die in staat zijn gestructureerde beslissingen te nemen. Het eerste deel van deze thesis focust op het maken van gestructureerde beslissingen waarbij deze gecorreleerd zijn in het ruimtelijke domein. Het tweede deel handelt over temporeel gestructureerde beslissingname.

Het oplossen van patroonherkenningsproblemen in domeinen zoals computervisie, bio-informatica en spraakherkenning gebeurt vaak best door het in rekening brengen van statistische relaties tussen outputvariabelen. Indien outputs gegenereerd worden met het in acht nemen van hun onderlinge relaties, wordt dit gestructureerde predictie genoemd. In het bijzonder onderzoekt het eerste deel van deze thesis de applicatie van een klasse methoden genaamd structural support vector machines (SSVMs) en beschrijft het hiervoor verschillende fundamentele uitbreidingen.

Eerst worden deze SSVM modellen onderzocht bij het overwinnen van een belangrijke hindernis voor de realisatie van autonome voertuigen, namelijk de mogelijkheid van de machine om zijn omgeving semantisch waar te nemen. Camera-input van het voertuig wordt aan deze gestructureerde predictiemodellen gevoed met als doel een semantische segmentatie van de omgeving te bekomen. Hierbij wordt elke afbeeldingspixel een bepaalde semantische klasse toegekend, zoals bijvoorbeeld ‘rijweg’ of ‘object’. Om een coherente segmentatie te bekomen wordt een SSVM gemodelleerd die de labeldistributie, geconditioneerd op de inputafbeeldingen, encodeert. Dit model houdt rekening met visuele context tussen de verschillende pixelregio’s binnen een tweede-orde aangrenzendheid.

SSVMs zijn niet-probabilistische modellen die een gezamenlijke input-output functie optimaliseren via margegebaseerde leertechnieken. Het gestructureerde model wordt geformuleerd als een energiegebaseerde functie, gebruikmakende van kenmerkafhankelijke unaire potentialen, en paarsgewijze potentialen welke differentiëren tussen eerste- en tweede-orde aangrenzendheid. Nadat dit model geoptimaliseerd is via maximum-marge leertechnieken, met behulp van een semantische objectieffunctie, kan efficiënte classificatie bekomen worden via graafsnedegebaseerde inferentie. Dit werk toont aan dat het in acht nemen van contextuele relaties tussen pixel segmentatieregionen binnen een tweede-orde aangrezendheid foutieve segmentatielabels uitfiltert, wat tot hoogst precieze semantische segmentaties leidt. Verder onderzoeken we deze methoden in een use case omtrent autonome agrarische voertuigen, waarbij een eind-tot-eind segmentatiemethode, gebruikmakende van diepe convolutionele neurale netwerken, als alternatief wordt voorgesteld.

Omdat SSVMs in het algemeen niet gebruik maken van de wisselwerking tussen unaire en interactiefactoren gedurende hun trainingsfase, zijn de finale parameterwaarden suboptimaal. Bovendien zijn de factoren vaak beperkt tot lineaire combinaties van de inputkenmerken, wat het generalisatievermogen sterk beperkt. Om de voorspellingsprecisie te verhogen, verbeteren we SSVMs door: gezamenlijke inferentie- en leertechnieken via de integratie van terugpropagatie en loss-augmented inferentie in SSVM subgradient descent; (ii) het uitbreiden van SSVM factoren naar neurale netwerken, welke hoogst nonlineaire factoren van de inputkenmerken voorstellen. Dit model stapt af van de traditionele tweedelige aanpak waarin de unaire factoren onafhankelijk van de interactiefactoren getraind worden in de gestructureerde predictor.

Resultaten op een complexe afbeeldingsegmentatietaak tonen aan dat het eind-tot-eind trainen van een SSVM, en/of het gebruik van neurale factoren, tot een hogere accuraatheid leidt dan conventionele subgradient descent-methoden en N-slack cutting plane-leertechnieken. Verder passen we een diepe convolutionele implementatie van de voorgestelde methode toe in een use case omtrent perceptie in autonome voertuigen. De resultaten demonstreren dat het model kan dienen als het fundament voor meer geavanceerde gestructureerde modellen, bijvoorbeeld door integratie met latente variabelen, geleerde kenmerkrepresentaties of meer complexe connectiviteitsstructuren.

Het tweede deel van deze scriptie onderzoekt gestructureerde voorspellingen in het tijddomein. Meer bepaald wordt reinforcement learning (RL) onderzocht, waarbij een agent tracht een bepaald opgelegd doel te bereiken in een initieel onbekende omgeving. Omdat dit doel bereiken meerdere acties vereist, kunnen we de actiesequenties interpreteren als een gestructureerde beslissing in de tijd. Elke actie beïnvloedt namelijk de volgende. We introduceren fundamenteel nieuwe methoden voor diepe RL—reinforcement learning die gebruik maakt van diepe neurale netwerken als functiebenaderigsmethoden. Deze methoden worden toegepast in het leren van voortbewegingspatronen in robotica en het autonoom spelen van videospellen.

In het bijzonder merken we op dat schaalbare en effectieve exploratie een sleutelprobleem is in diepe RL. Hoewel er methoden met optimaliteitsgaranties bestaan in het geval van discrete staat- en actieruimten, kunnen deze methoden moeilijk aangewend worden in hoogdimensionale scenario's. Bijgevolg houden huidige RL technieken vooral vast aan eenvoudige heuristieken zoals ϵ-greedy exploratie of het toevoegen van Gaussische ruis aan de actie-outputs. Hierdoor heeft de agent de neiging vroegtijdig vast te raken in suboptimale beslissingssequenties, in plaats van exploratie-acties te ondernemen die mogelijk de totale beloning op lange termijn verhogen.

Deze thesis introduceert Variational Information Maximizing Exploration (VIME), een exploratiestrategie gebaseerd op de maximalisatie van informatiewinst omtrent het geloof van de agent in zijn model van de omgevingsdynamica. Het is een curiositeitsgedreven exploratie-algoritme voor continue controletaken. Er wordt een praktische implementatie voorgesteld op basis van variationele inferentie in Bayesiaanse neurale netwerken, welke efficiënt omgaan met continue staat- en actieruimten. Variationele inferentie wordt aangewend bij het benaderen van de posteriordistributie van het netwerk dat de omgevingsdynamica modelleert. VIME past de beloningsfunctie van het Markov beslissingsprocess (MDP) aan en kan gebruikt worden met tal van onderliggende RL algoritmen. We tonen aan dat VIME significant beter presteert dan heuristische exploratie-algoritmen over een waaier aan continue controletaken en RL algoritmen, inclusief taken met zeer schaarse beloningen.

Een alternatief voor curiositeitsgebaseerde exploratie zijn frequentiegebaseerde exploratie-algoritmen. Deze algoritmen staan erom bekend om bijna-optimaal te zijn in combinatie met tabulaire RL methoden voor het oplossen van kleine discrete MDPs. Over het algemeen wordt gedacht dat frequentiegebaseerde methoden niet aangewend kunnen worden in hoogdimensionale staatruimten, aangezien de agent in dit geval de meeste staten maar eenmaal tegenkomt. Recente diepe RL exploratiestrategieën, zoals VIME, zijn in staat met hoogdimensionale continue staatruimten om te gaan via complexe heuristieken, vaak gebruikmakend van concepten zoals optimisme in geval van onzekerheid of intrinsieke motivatie.

Deze thesis beschrijft een verrassende ontdekking: een simpele genera-

lisatie van de klassieke frequentiegebaseerde aanpak is in staat bijna-state-of-the-art performantie te behalen op meerdere hoogdimensionale en/of continue diepe RL benchmarks. Staten worden op hashcodes geprojecteerd, wat toelaat hun frequentie bij te houden via een hashtabel. Deze frequenties worden nadien gebruikt als beloningsbonus, naargelang de klassieke theorie rond frequentiegebaseerd exploreren. We ontdekken dat simpele hashfuncties bijzonder goede resultaten behalen op verscheidene uitdagende taken.

Verder wordt aangetoond dat een domeinafhankelijke geleerde hashcode deze resultaten nog kan verbeteren. Gedetailleerde analyse toont het belang aan van een goede hashfunctie. Deze moet namelijk: (i) een aangepaste granulariteit hebben; (ii) informatie encoderen omtrent de op te lossen MDP. Deze exploratiestrategie bereikt bijna-state-of-the-art performantie in zowel het leren van voortbewegingspatronen in robotica als het automoom spelen Atari 2600 spellen. Dit terwijl het een eenvoudige maar sterke vergelijkingsbasis vormt voor het oplossen van MDPs waarin significantie exploratie noodzakelijk is.

Chapter 1

Introduction

Artificial intelligence (AI) is taking a leading role in shaping our future. Rather than programming computers directly for task solving, a paradigm shift is taking place towards data-driven approaches, in which computers are programmed to *learn* how to solve a task. Being able to teach machines is extremely valuable, since the majority of tasks are simply too complex to be explicitly understood by the programmer. The field of machine learning studies artificial systems that learn from data, by extracting underlying patterns and structure from examples. Often, this is done through the optimization of mathematical models, for which well-fitting parameter values are sought. Whereas the previous generations of AI systems, such as expert systems, had problems in dealing with input uncertainty, these learned models naturally cope with noisy scenarios.

Designing systems that learn how to make complex decision, in which decisions are structured or correlated, enables many applications. Examples are placing web advertisements, ranking search results, translating sentences, robotics, generating text-to-speech, semantic segmentation, etc. The focus of this work is centered around structured decision making both in the spatial and temporal domain. As such, the first part of this dissertation (Chapters 2 and 3) investigates multi-output prediction, in which the predictions affect each other spatially. These models are applied in the domain of semantic image segmentation to enable visual perception in autonomous vehicles. The second part of this dissertation (Chapters 4 and 5) investigates structured decision making in time. Herein reinforcement learning is researched, which studies agents that learn how to accomplish a goal in an initially unknown environment. Doing this requires the agent to take sequential decisions, which affect each other in the temporal domain. The developed methods are applied in autonomous video game playing and learning robot locomotion.

1.1 Graphical Models

The concepts presented in this dissertation are based on building models of the world that can relate observed measurements with quantities that we care about. For example, given an image in which pixel regions are missing, infer their missing values (cfr. Chapters 2 and 3). Or, given a sequence of actions and joint angles generated by a robot that learns to walk, infer the next action to take (cfr. Chapters 4 and 5). This section introduces the most important concepts of graphical models needed to understand this dissertation. For additional background information, the reader is referred to [Koller and Friedman, 2009].

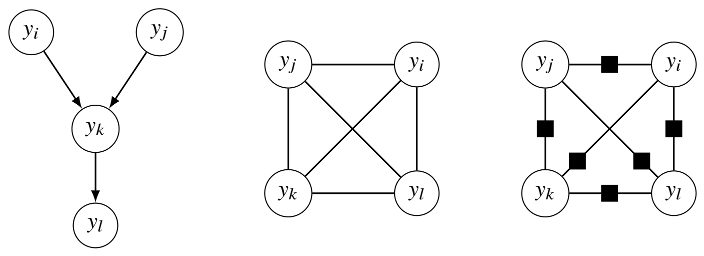


Figure 1.1: Illustration of a directed (left) and undirected (middle, right) graphical model. The rightmost figure represents a factor graph that encodes pairwise relations.

Graphical models form a language to efficiently encode interactions between variables. Doing so allows these models to connect both known observations and variables with unknown values. Often it is impossible to derive these values with certainty, which is where graphical models come into play. By encoding a conditional or joint probability distribution, unknown values can be estimated. It is possible to make the distinction between directed and undirected graphical models.

Directed graphical models, such as the one depicted in Figure 1.1 (left), specify a probability distribution $p(y)$ with $y \in \mathcal{Y}$. This is done through specification of a directed acyclic graph $G = (\mathcal{V}, \mathcal{E})$, which effectively encodes the statistical independencies between the random variables that make up the nodes of this graph. As such directed graphical models specify a factorized probability distribution

$$p(y) = \prod_{i \in \mathcal{V}} p\Big(y_i | y_{\mathrm{pa}_{G(i)}}\Big), \tag{1.1}$$

in which $y_{\mathrm{pa}_{G(i)}}$ represent the parent nodes of node i in G. For example, Figure 1.1 (left) specifies the probability distribution

$$p(y) = p(y_l | y_k) p(y_k | y_i, y_j) p(y_i) p(y_j). \tag{1.2}$$

A second class consists of the undirected graphical models, or Markov random fields (MRFs) [Nowozin and Lampert, 2011], as illustrated in Figure 1.1 (middle). These models define an undirected graph $G = (\mathcal{V}, \mathcal{E})$ which specifies the factorized distribution

$$p(y) = \frac{1}{Z} \prod_{C \in \mathcal{C}(G)} \psi_C(y_C), \quad Z = \sum_{y \in \mathcal{Y}} \prod_{C \in \mathcal{C}(G)} \psi_C(y_C), \tag{1.3}$$

in which $\mathcal{C}(G)$ denotes the set of cliques in G and Z is a normalization constant that ensures that a valid probability distribution is obtained. For example, Figure 1.1 (middle) defines the factorized probability distribution

$$p(y) = \frac{1}{Z} \prod_{A \in 2^{\{i,j,k,l\}}} \psi_A(y_A). \tag{1.4}$$

An alternative way to specify an MRF is by using a factor graph. A factor graph makes explicit the factorization of the probability distribution. It is a tuple $(\mathcal{V}, \mathcal{E}, \mathcal{F})$, composed of nodes, edges, and factors. Herein, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{F}$. If we define the scope of a factor to be $N(\mathcal{F}) = \{i \in \mathcal{V} : (i, \mathcal{F}) \in \mathcal{E}\}$, namely the nodes to which it connects, then the distribution factorizes as

$$p(y) = \frac{1}{Z} \prod_{F \in \mathcal{F}} \psi_F(y_{N(F)}). \tag{1.5}$$

An example is shown in Figure 1.1 (right), in which only pairwise interactions are considered.

Different types of graphical models exist, but they all have in common that they specify a family of probability distributions by means of a graph. The various types differ by the allowed graph structure and the conditional independence assumptions encoded in the graph. Given a graphical model we can think of it as a filter for probability distributions. Only those distributions pass that satisfy all conditional independencies the graph encodes. As such, a graphical model specifies a family of probability distributions rather than a single one [Nowozin and Lampert, 2011].

1.2 Deep Learning

Next to graphical models, many of the methods described in this dissertation make use of deep learning. Often these methods can be described as graphical models in which parts are modeled by deep neural networks. Deep learning models are capable of learning data representations at multiple abstraction levels. These levels are layers of nonlinear functions of which the output forms the input of the next layer. As such, they are able to extract a set of hierarchical features from input data. These models, often called deep neural networks, are trained using a training dataset, in which the internal parameters are perturbed such that an objective function is optimized. This section introduces the most important concepts of deep learning in order to understand this dissertation. For additional background information, the reader is referred to [Goodfellow et al., 2016].

The most straight-forward example is the feed-forward neural network, in which each layer l consists of a set of neurons (units) that first compute an affine transformation of the neuron outputs of the previous layer, after which a nonlinear function is applied element-wise. The coefficients of this affine transformation are tunable parameters. A neural network can thus be specified through the recursive relation

$$x_l = f(W_l x_{l-1} + b_l), \tag{1.6}$$

in which $f(\cdot)$ is an element-wise activation function, $W \in \mathbb{R}^{m \times p}$, and $b \in \mathbb{R}^p$. Examples of activation functions are sigmoidal functions, tanh functions, and rectified linear units.

A particular class of feed-forward neural networks is the convolutional neural network (CNN). These networks differ in the way each layer is connected to the following. Due to its restricted connectivity pattern, the number of tunable weights is limited, making it biased towards certain data types. These data types are generally data in which salient features are translation invariant, for example images, in which an object can appear anywhere within the image boundaries. An illustration is shown in Figure 1.2 which depicts a convolutional autoencoder, a model that attempts to reconstruct its input, in this case a single-channel image. The input passes through various transformations, depicted from left to right, applying (possibly transposed) convolutions and nonlinear transformations.

CNNs are described by the following recursive equation:

$$X_l^{(p)} = f\left(\sum_k W_l^{(k,p)} * X_{l-1}^{(k)} + b_l^{(p)}\right), \tag{1.7}$$

with $*$ the 2-dim convolution operator, and $W_l^{(k,p)}$ the learned parameters (filters) for each input feature map $X_{l-1}^{(k)}$ and output feature map $X_l^{(p)}$. Each output feature map $X_l^{(p)}$ is the sum of the convolution operations between each of the input feature maps $X_{l-1}^{(k)}$ and the corresponding filter $W_l^{(k,p)}$. The advantage is that the matrix multiplication is replaced by a convolution, which causes the weights of each output neuron in a particular output feature map to be shared. This makes the number of parameters to be significantly lower, countering overfitting behavior and lowering the memory requirements for storing the model.

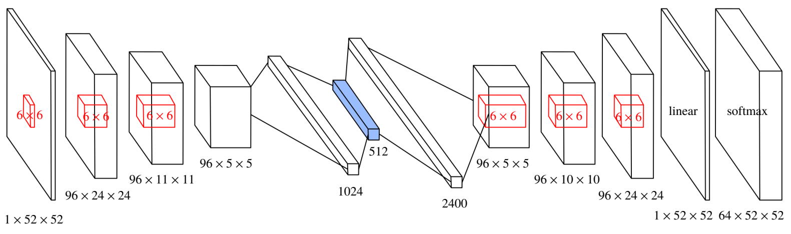


Figure 1.2: Illustration of a deep convolutional neural network, more specifically a convolutional autoencoder, as used in Chapter 5. Herein, 6 × 6 convolutions are used with a stride of 2.

Most deep learning models are trained by some form of stochastic gradient descent (SGD). Gradient descent-based methods optimize an objective function L, e.g., negative log-likelihood $L = -\sum_i \log p(x_i; w, b)$ or mean-squared error in the supervised learning case, by perturbing all network parameters jointly in a direction that improves the loss value on a particular training dataset. It is called stochastic since the dataset is often split up into multiple minibatches, which are processed sequentially. More specifically, these methods update the weights and biases of the neural network as follows:

$$w_n \leftarrow w_n - \alpha \frac{\partial L}{\partial w_n}, \quad b_n \leftarrow b_n - \alpha \frac{\partial L}{\partial b_n}. \tag{1.8}$$

The parameters are altered according to the partial derivative of the loss function L to each of its parameters, rescaled by a factor $\alpha \in \mathbb{R}_{>0}$, called the step size. This method of perturbing neural network parameters forms the basis of many methods described in this thesis, such as optimizing the policy in reinforcement learning through policy gradients (Chapter 4), optimizing the structural support vector machine objective function when using neural factors (Chapter 3), or training convolutional unary classification methods (Chapter 2).

1.3 Spatially-Structured Decision Making

This section introduces structured decision making in which the decisions are spatially correlated, as researched in Chapters 2 and 3. This dissertation specifically investigates these types of methods in light of their application to visual perception for autonomous vehicles.

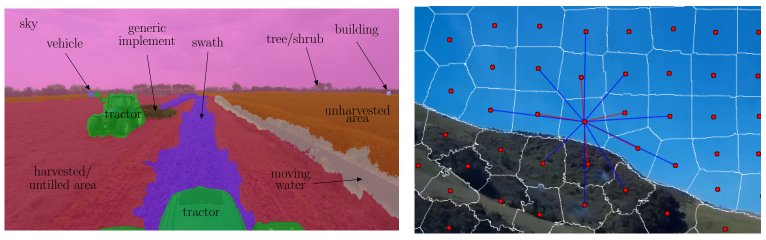


Figure 1.3: Illustration of image segmentation as applied in an agricultural autonomous vehicle setting (left) and an image over-segmentation that also depicts the first- and second-order adjacency neighborhood of a particular pixel cluster (right)

The lack of structured elements in the environment, such as walls, forms a big challenge for autonomous outdoor vehicles [Reina et al., 2012]. Rather than identifying which areas are traversable or nontraversable, or detecting objects using bounding boxes [Sivaraman and Trivedi, 2013, Bernini et al., 2014], we attempt to obtain intelligent environment sensing through assignment of semantic classes to each visual input pixel. Higher-level processes can then use this information to determine how the vehicle should act in order to attain a particular goal.

Two important streams in computer vision can be identified: segmentation models and bounding box models. The latter attempts to detect objects by drawing rectangles around them, which tends to break down when highly irregular regions have to be identified, such as sky or vegetation [Nowozin and Lampert, 2011]. Segmentation models handle this naturally since they label each individual pixel or pixel cluster. Since irregular shapes dominate outdoor scenes, we focus on the segmentation approach in our quest for intelligent autonomous vehicle perception.

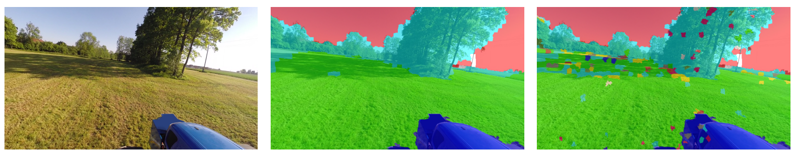


Figure 1.4: Illustration of image segmentation applied to the perception system of an agricultural autonomous vehicle: original image (left), segmentation through structured prediction (middle), segmentation through traditional independent prediction (right)

One approach to segmentation is by making use of a graphical model that embodies interactions between adjacent over-segmentation regions, i.e., coherent pixel clusters, conditionally-dependent on the input image. An example of a such an over-segmentation is show in Figure 1.3 (right). The ultimate target is to obtain a segmentation by coloring in each cluster with a particular semantic class as shown in Figure 1.4 (middle). Herein, a machine learning system was trained to segment camera images originating from autonomous agricultural vehicles into various outdoor classes.

In traditional supervised machine learning a system is trained to output a single scalar value, for example the type of object that is recognized in

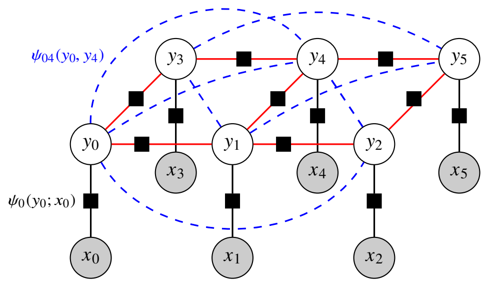


Figure 1.5: Structured output prediction as a graphical model: making spatially-structured decisions. The gray nodes represent observed/known variables, while the white nodes unknown variables, whose values we aim to determine.

an image. In our particular use case, this would correspond to training a system to output a correct class for each pixel or pixel cluster individually. An example is shown in Figure 1.4, which depicts the original image on the left, and the segmentation through traditional supervised learning on the right. Structured prediction models allow for arbitrary structured outputs, such as a graph or vector. These models have proven useful in tasks such as part-of-speech tagging and optical character recognition, where taking into account contextual cues and predicting all output variables at once is beneficial. In Figure 1.4, we see the improvement of using structured prediction in the middle image, over the traditional approach on the right.

A widely used framework is the conditional random field (CRF) [Koller and Friedman, 2009], a type of MRF that models the output probability distribution considering statistical conditional dependencies between input and output variables, as well as between output variables mutually. However, many tasks only require 'most-likely' predictions, which led to the rise of nonprobabilistic approaches. Rather than optimizing the Bayes' risk, these models minimize a structured loss, allowing the optimization of performance indicators directly [Nowozin and Lampert, 2011]. One such model is the structural support vector machine (SSVM) [Tsochantaridis et al., 2005] in which a generalization of the hinge loss to multiclass and multilabel prediction is used.

An example of a graphical model representing a CRF is depicted in Figure 1.5 as a factor graph. Herein, the gray nodes represent observed variables, while the white nodes are predictions/decisions made by the system. This model can be described by the following equation

$$p(y|x) = \frac{1}{Z(x)} \prod_{i \in \mathcal{V}} \psi_i(y_i, x_i) \prod_{(i,j) \in \mathcal{E}} \psi_{ij}\big(y_i, y_j\big). \tag{1.9}$$

The relations between the various decisions are modeled by pairwise factors $\psi_{ij}\big(y_i, y_j\big)$, which are functions that output real numbers, whose values indicate the relation strength. The unary factors $\psi_i(y_i, x_i)$ relate observations to predictions, indicating the relation between input features and output variables. Relating this to the example used previously, the different regions in Figure 1.3 (right) form the observations, the gray nodes x_i. The class assigned to each of them is represented by the decisions, the white nodes y_i. A coherent segmentation can best be obtained by predicting all variables at once, computing $\arg\max_y p(y|x)$, such as in Figure 1.4 (middle), which is exactly what the SSVM model does.

1.4 Temporally-Structured Decision Making

The second part of this dissertation focuses on structured decision making in which temporal correlations between prediction variables are considered, as researched in Chapters 4 and 5. This section briefly introduces the field of reinforcement learning, which forms a basis for these chapters. In particular, these chapters focus on the application of sequential decision making to learning robot locomotion and autonomous video game playing.

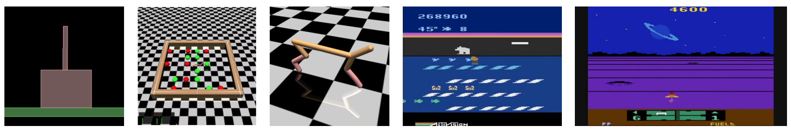


Figure 1.6: Illustrations of the tasks used in the experiments in Chapters 4 and 5. The left three figures represent continuous control tasks, while the right two figure represent Atari 2600 games.

Reinforcement learning (RL) studies how an agent can learn to maximize a cumulative reward in a previously unknown environment, which it learns about through experience. The agent could be any object with the

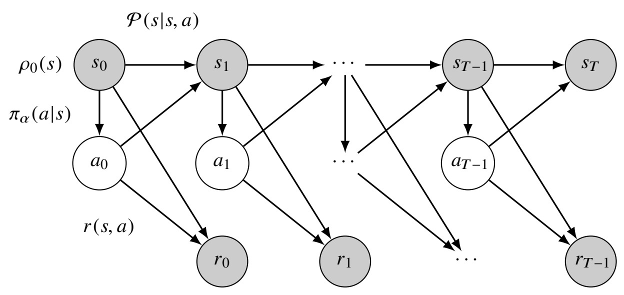


Figure 1.7: RL as a graphical model: making structured decisions in time. The agent decides which action to take based on $\pi_\alpha(a|s)$, a reward is given according to $r(s,a)$, after which the next state is generated according to $\mathcal{P}(s'|s,a)$. When s_T is reached, the episode terminates.

ability to influence its environment. Figure 1.6 shows several environments, such as balancing a pole on a cart, in which the agent controls the cart velocity, and learning stable walking behavior to collect items, in which the agent controls the joint torques of a robot. The environment is the set of possible states the agent can be in, which could be the robot joint angles or the position of the remaining objects in the collection task. As such, the agent has the ability to change the environment state through its decisions. The goal of the agent is specified through a reward function, which could for example reward the agent in making forward progress when learning to walk.

A core concept in RL is the Markov decision process (MDP), which describes an agent interacting with an environment [Sutton, 1990]. This concept is defined as the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \rho_0, \gamma, T)$. Herein, we consider (i) a state space $\mathcal{S} \subseteq \mathbb{R}^n$ defining the set of possible environment states, (ii) an action space $\mathcal{A} \subseteq \mathbb{R}^m$ defining the set of possible agent actions, (iii) a transition function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, generating a new state $s_{t+1} \in \mathcal{S}$ by taking action $a_t \in A$ in state $s_t \in \mathcal{S}$, and (iv) a reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ defining a reward $r(s_t, a_t)$ given when taking an action a_t in state s_t. Ultimately, the goal is to learn a policy $\pi : \mathcal{S} \to \mathcal{A}$ that allows sequential decision making for the agent reach its target.

This dissertation focuses specifically on episodic RL, in which the agent lifetime is broken up into episodes with finite number of states, actions, and rewards. The episode ends when a terminal state s_T is reached, with T

being the time horizon. We assume a parametrized policy π_α with the goal of maximizing $\mu(\pi_\alpha)$, the expected (discounted) return

$$\mu(\pi_\alpha) = \mathbb{E}_{\tau \sim p(\tau;\alpha)} \left[\sum_{t=0}^{T-1} \gamma^t r(s_t, a_t) \right], \tag{1.10}$$

where $\tau = (s_0, a_0, \ldots)$ denotes a whole trajectory, with $s_0 \sim \rho_0(s)$, $a_t \sim \pi_\alpha(a_t|s_t)$, and $s_{t+1} \sim \mathcal{P}(s_{t+1}|s_t, a_t)$. The parameter $\gamma \in (0, 1]$ is a discount factor often introduced to deal with long time horizons. Assuming an initial state distribution $\rho_0 : \mathcal{S} \rightarrow \mathbb{R}$, we can represent the agent behavior as a time-recurrent graphical model, which is shown unrolled in Figure 1.7. As such, the policy learned can be interpreted as a sequential decision making procedure with temporally-correlated decisions.

To illustrate this theory, imagine a robot that learns how to walk through reinforcement learning. The initial distribution could be a probability mass centered around some specific set of initial joint angles. Each state would defined as a vector with elements between 0 and 2π. The actions could be the joint torques, which would be real-valued vectors. Based on the underlying robot dynamics model, these torques would modify the angle settings. Since we work in the discrete time domain, these actions could for example be executed at 30 Hz. In order to learn to walk, a possible reward function would be the distance traversed so far. The final state can be defined as reaching a particular distance, or when the robot body hits the floor. The policy itself might be a neural network that takes as input the joint angle vector and outputs the mean and variance of a multivariate Gaussian distribution. Actions could then be sampled from this distribution, at a rate of 30 times per second.

One class of RL methods used very often in this thesis are policy gradient methods. These methods estimate the gradient of the expected return as defined in Eq. (1.10),

$$\begin{aligned} \nabla_\alpha \mu(\pi_\alpha) &= \nabla_\alpha \mathbb{E}_{\tau \sim p(\tau;\alpha)} \left[\sum_{t=0}^{T-1} \gamma^t r(s_t, a_t) \right] \\ &= \mathbb{E}_{\tau \sim p(\tau;\alpha)} \left[\nabla_\alpha \log p(\tau; \alpha) \sum_{t=0}^{T-1} \gamma^t r(s_t, a_t) \right], \end{aligned} \tag{1.11}$$

using a score function gradient estimator [Sutton, 1998], since the likelihood ratio trick tells us that

$$\begin{aligned}\nabla_\alpha \mathbb{E}_{x\sim p(x;\alpha)}\left[f(x)\right] &= \nabla_\alpha \int_{\mathbf{x}} p(x;\alpha) f(x) \mathrm{d}x \\ &= \int_{\mathbf{x}} \nabla_\alpha p(x;\alpha) f(x) \mathrm{d}x = \int_{\mathbf{x}} p(x;\alpha) \nabla_\alpha \log p(x;\alpha) f(x) \mathrm{d}x \\ &= \mathbb{E}_{x\sim p(x;\alpha)}\left[\nabla_\alpha \log p(x;\alpha) f(x)\right]. \qquad (1.12)\end{aligned}$$

Moreover, the probability $p(\tau;\alpha)$ in Eq. (1.11) can be expanded as

$$\begin{aligned}p(\tau;\alpha) = \rho(s_0)\pi(a_0|s_0;\alpha)\mathcal{P}(s_1|s_0,a_0)\ldots \\ \pi(a_{T-1}|s_{T-1};\alpha)\mathcal{P}(s_T|s_{T-1},a_{T-1}), \qquad (1.13)\end{aligned}$$

according to the factorization in Figure 1.7. When computing $\log p(\tau;\alpha)$, these multiplications become sums, hence when computing the gradient, the terms that are independent of the parameter α become zero. Since the transition dynamics probability distribution and the initial state distribution are both independent from α, these are dropped. As such,

$$\nabla_\alpha \log p(\tau;\alpha) = \nabla_\alpha \log \pi(a_0|s_0;\alpha) + \ldots + \nabla_\alpha \log \pi(a_{T-1}|s_{T-1};\alpha). \qquad (1.14)$$

Therefore, we can replace Eq. (1.11) by

$$\nabla_\alpha \mu(\pi_\alpha) = \mathbb{E}_{\tau\sim p(\tau;\alpha)}\left[\sum_{t=0}^{T-1} \nabla_\alpha \log \pi(a_t|s_t;\alpha) \sum_{t=0}^{T-1} \gamma^t r(s_t,a_t)\right]. \qquad (1.15)$$

As such, we can simply approximate this expectation by drawing samples according to the environment and policy, and average them. This allows the optimization of the policy parameters through sampling, without requiring access to the transition dynamics distribution that is often unknown. This forms the foundation of policy gradient methods as used throughout Chapters 4 and 5.

1.5 Scientific Challenges and Contributions

This section summarizes the research challenges and scientific contributions made in the different chapters of this dissertation.

Chapter 2: Structured Output Prediction for Semantic Perception in Autonomous Vehicles A key challenge in the realization of autonomous vehicles is the machine's ability to perceive its surrounding environment. This task is tackled through a model that partitions vehicle camera input into distinct semantic classes, by taking into account visual contextual cues. The use of structured machine learning models is investigated, which not only allows for complex input, but also arbitrarily structured output.

Towards this goal, an outdoor scene dataset is constructed with accompanying fine-grained image labelings. For coherent segmentation, a structured predictor is modeled to encode label distributions conditioned on the input images. After optimizing this model through max-margin learning, based on an ontological loss function, efficient classification is realized via graph cuts inference using α-expansion. Analyses demonstrate that by taking into account contextual relations between pixel segmentation regions within a second-degree neighborhood, spurious label assignments are filtered out, leading to highly accurate semantic segmentations for outdoor scenes.

Additionally, we propose the use of a convolutional neural network as a unary classifier with a structured predictor, as well as the use of an end-to-end segmentation method. These models are enhanced through transfer learning using features from an independent classification task.

Chapter 3: Integrated Inference and Learning of Neural Factors in Structural Support Vector Machines Tackling pattern recognition problems in areas such as computer vision, bioinformatics, speech or text recognition is often done best by taking into account task-specific statistical relations between output variables. In structured prediction, this internal structure is used to predict multiple outputs simultaneously, leading to more accurate and coherent predictions. Structural support vector machines (SSVMs) are nonprobabilistic models that optimize a joint input-output function through margin-based learning.

Because SSVMs generally disregard the interplay between unary and interaction factors during the training phase, final parameters are suboptimal. Moreover, its factors are often restricted to linear combinations of input features, limiting its generalization power. To improve prediction accuracy, this chapter proposes: (i) Joint inference and learning by integration of back-propagation and loss-augmented inference in SSVM subgradient descent; (ii) Extending SSVM factors to neural networks that form highly nonlinear functions of input features.

Image segmentation benchmark results demonstrate improvements over conventional methods in terms of accuracy, highlighting the feasibility of end-to-end SSVM training with neural factors. This is reinforced by segmentation results of a deep convolutional implementation of the proposed model on autonomous vehicle visual data.

Chapter 4: Variational Information Maximizing Exploration Scalable and effective exploration remains a key challenge in reinforcement learning (RL). While there are methods with optimality guarantees in the setting of discrete state and action spaces, these methods cannot be applied in high-dimensional deep RL scenarios. As such, most contemporary RL relies on simple heuristics such as epsilon-greedy exploration or adding Gaussian noise to the controls.

This chapter introduces Variational Information Maximizing Exploration (VIME), an exploration strategy based on maximization of information gain about the agent's belief of environment dynamics. We propose a practical implementation, using variational inference in Bayesian neural networks which efficiently handles continuous state and action spaces. VIME modifies the Markov decision process (MDP) reward function, and can be applied with several different underlying RL algorithms. We demonstrate that VIME achieves significantly better performance compared to heuristic exploration methods across a variety of continuous control tasks and algorithms, including tasks with very sparse rewards.

Chapter 5: A Study of Count-Based Exploration for Deep Reinforcement Learning Count-based exploration algorithms are known to perform near-optimally when used in conjunction with tabular RL methods for solving small discrete MDPs. It is generally thought that count-based methods cannot be applied in high-dimensional state spaces, since most states will only occur once. Recent deep RL exploration strategies are able to deal with high-dimensional continuous state spaces through complex heuristics, often relying on optimism in the face of uncertainty or intrinsic motivation.

In this chapter, we describe a surprising finding: a simple generalization of the classic count-based approach can reach near state-of-the-art performance on various high-dimensional and/or continuous deep RL benchmarks. States are mapped to hash codes, which allows to count their occurrences with a hash table. These counts are then used as a reward bonus, according to the classic count-based exploration theory. We find that simple hash functions can achieve surprisingly good results on many challenging tasks.

Furthermore, we show that a domain-dependent learned hash code may further improve these results. Detailed analysis reveals important aspects of a good hash function: (i) Having appropriate granularity; (ii) Encoding information relevant to solving the MDP. This exploration strategy achieves near state-of-the-art performance on both continuous control tasks and Atari 2600 games, while providing a simple yet powerful baseline for solving MDPs that require considerable exploration.

1.5.1 Publications

The research results obtained during my doctoral studies have been published in scientific journals and presented at a series of international conferences. The following sections provide an overview of the work published.

1.5.1.1 Conference Proceedings

Tang, H., Houthooft, R., Foote, D., Stooke, A., Chen, X., Duan, Y., Schulman, J., De Turck, F., and Abbeel, P. (2016). #Exploration: A study of count-based exploration for deep reinforcement learning. In *Deep Reinforcement Learning Workshop at NIPS*.

Houthooft, R., Chen, X., Duan, Y., Schulman, J., De Turck, F., and Abbeel, P. (2016). VIME: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems 29 (NIPS)*, pages 1109–1117, Barcelona, Spain.

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. (2016). InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems 29 (NIPS)*, pages 2172–2180, Barcelona, Spain.

Duan, Y., Chen, X., Houthooft, R., Schulman, J., and Abbeel, P. (2016). Benchmarking deep reinforcement learning for continuous control. In

Proceedings of the 33rd International Conference on Machine Learning (ICML), pages 1329–1338, New York, USA.

Houthooft, R., De Boom, C., Verstichel, S., Ongenae, F., and De Turck, F. (2016). Structured output prediction for semantic perception in autonomous vehicles. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI)*, Phoenix, Arizona, USA.

Houthooft, R., Sahhaf, S., Tavernier, W., De Turck, F., Colle, D., and Pickavet, M. (2015). Robust geometric forest routing with tunable load balancing. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, pages 1382–1390, Hong Kong, P.R. China.

Houthooft, R., Sahhaf, S., Tavernier, W., De Turck, F., Colle, D., and Pickavet, M. (2014). Fault-tolerant greedy forest routing for complex networks. In *Proceedings of the 6th International Workshop on Reliable Networks Design and Modeling (RNDM)*, pages 1–8, Barcelona, Spain.

De Backere, F., Hanssens, B., Heynssens, R., Houthooft, R., Zuliani, A., Verstichel, S., Dhoedt, B., and De Turck, F. (2014). Design of a security mechanism for RESTful Web service communication through mobile clients. In *Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS)*, pages 1–6, Krakow, Poland.

1.5.1.2 Journals

Houthooft, R. and De Turck, F. (2016). Integrated inference and learning of neural factors in structural support vector machines. *Pattern Recognition*, 59:292–301.

Houthooft, R., Ruyssinck, J., van der Herten, J., Stijven, S., Couckuyt, I., Gadeyne, B., Ongenae, F., Colpaert, K., Decruyenaere, J., Dhaene, T., and De Turck, F. (2015). Predictive modelling of survival and length of stay in critically ill patients using sequential organ failure scores. *Artificial Intelligence in Medicine*, 63(3):191 – 207.

Houthooft, R., Sahhaf, S., Tavernier, W., De Turck, F., Colle, D., and Pickavet, M. (2015). Optimizing robustness in geometric routing via embedding redundancy and regeneration. *Networks*, 66(4):320–334.

1.5.1.3 Patent Applications

Houthooft, R., Verstichel, S., Debilde, B., and Foster, C. A controller for a working vehicle. E.U. Patent Application No. 16177346.0 - 1905. U.S. Patent Application No. 15/199,090. Filed 30 June 2016.

References

[Bernini et al., 2014] Bernini, N., Bertozzi, M., Castangia, L., Patander, M., and Sabbatelli, M. (2014). Real-time obstacle detection using stereo vision for autonomous ground vehicles: A survey. In *Proceedings of the IEEE 17th International Conference on Intelligent Transportation Systems (ITSC)*, pages 873–878.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. The MIT Press.

[Koller and Friedman, 2009] Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press.

[Nowozin and Lampert, 2011] Nowozin, S. and Lampert, C. H. (2011). Structured learning and prediction in computer vision. *Foundations and Trends in Computer Graphics and Vision*, 6(3–4):185–365.

[Reina et al., 2012] Reina, G., Milella, A., and Underwood, J. (2012). Self-learning classification of radar features for scene understanding. *Robotics and Autonomous Systems*, 60(11):1377–1388.

[Sivaraman and Trivedi, 2013] Sivaraman, S. and Trivedi, M. M. (2013). Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis. *IEEE Transactions on Intelligent Transportation Systems*, 14(4):1773–1795.

[Sutton, 1990] Sutton, R. S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the 7th International Conference on Machine Learning (ICML)*, pages 216–224.

[Sutton, 1998] Sutton, R. S. (1998). *Introduction to reinforcement learning*. The MIT Press.

[Tsochantaridis et al., 2005] Tsochantaridis, I., Joachims, T., Hofmann, T., and Altun, Y. (2005). Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484.

Chapter 2

Structured Output Prediction for Semantic Perception in Autonomous Vehicles

Rein Houthooft, Cedric Boom, Stijn Verstichel, Femke Ongenae & Filip De Turck, *Structured Output Prediction for Semantic Perception in Autonomous Vehicles*, In Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16), Phoenix, Arizona, USA, 2016.

Includes changes and extensions not released at the time of publication.

Ghent University – imec, CNH Industrial

⋆ ⋆ ⋆

A key challenge in the realization of autonomous vehicles is the machine's ability to perceive its surrounding environment. This task is tackled through a model that partitions vehicle camera input into distinct semantic classes, by taking into account visual contextual cues. The use of structured machine learning models is investigated, which not only allow for complex *input, but also arbitrarily structured output. Towards this goal, an outdoor scene dataset is constructed with accompanying fine-grained image labelings. For coherent segmentation, a structured predictor is modeled to encode label distributions conditioned on the input images. After optimizing this model through max-margin learning, based on an ontological loss function, efficient classification is realized via graph cuts inference using alpha-expansion. Analyses demonstrate that by taking into account contextual relations between pixel segmentation regions within a second-degree neighborhood, spurious label assignments are filtered out, leading to highly accurate semantic segmentations for outdoor scenes. This applied research chapter thus i) applies structured prediction methods in an autonomous vehicle project for visual understanding of the environment, and ii) sets forth the background and implementation of the methods that form a basis for the algorithms developed in Chapter 3.*

2.1 Introduction

Outdoor settings present many challenges to autonomous vehicle operation due to the lack of structured elements in the environment, such as walls or doors [Reina et al., 2012]. Beyond the identification of traversable areas and object detection [Sivaraman and Trivedi, 2013, Bernini et al., 2014], environmental sensing can be approached by determining a semantic class for each point of the environment, enabling further high-level processing such as automatic vehicle steering based on road segments. In this chapter, we present a semantic perception model for autonomous vehicles based on image segmentation.

In computer vision, two main trends can be discerned, namely segmentation models and bounding box models. The latter attempt at identifying objects by drawing bounding rectangles. Although this approach performs very well at recognizing objects with a regular shape, its performance drops significantly when irregular regions, such as vegetation or sky, have to be identified [Nowozin and Lampert, 2011]. Segmentation models are not hindered by irregular shapes as they label each individual image pixel. Because outdoor scenes consist largely of irregular shapes, the segmentation approach fits best for autonomous vehicles.

Many segmentation techniques are based on a graphical model that embodies interactions between neighboring over-segmentation regions, i.e.,

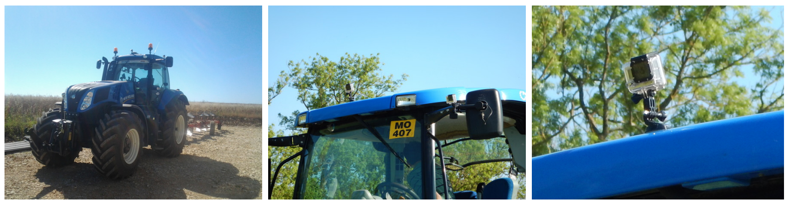


Figure 2.1: One of the vehicles used to capture the dataset images; cameras are mounted on the roof top center and mirrors.

coherent pixel clusters, conditionally-dependent on the input image (see [Nowozin and Lampert, 2011] for an overview). We propose a structured prediction model that takes into account contextual relations between both first- and second-degree region neighbors by means of distinct interaction functions. This model is optimized according to an ontological loss function via max-margin learning. Towards our goal of autonomous vehicle perception, a dataset is constructed in which recorded camera images are labeled in a fine-grained manner. Effective and efficient inference is possible by means of the α-expansion [Boykov et al., 2001] algorithm, yielding accurate semantic image segmentations.

2.2 Data Acquisition

Structured prediction methods are applied specifically in an autonomous agricultural vehicle use case, using alternative data, methods, and improvements that could not be shown publicly at the time of publication in [Houthooft et al., 2016].

As part of this use case, an autonomous agricultural vehicle segmentation dataset was created in collaboration with Case New Holland (CNH) Industrial. Camera video data was captured originating from ten different recording campaigns in the Styria region and Amstetten area in Austria, the Bologna province in Italy, West-Flanders in Belgium, Pinal County in Arizona, USA, the Indre and Lot-et-Garonne departments in France, the Thuringia region in Germany, and the North Brabant region in The Netherlands. To capture the data, different vehicles were equipped with GoPro Hero 3+ Black cameras, set to a resolution of 1920×1080 pixels and a

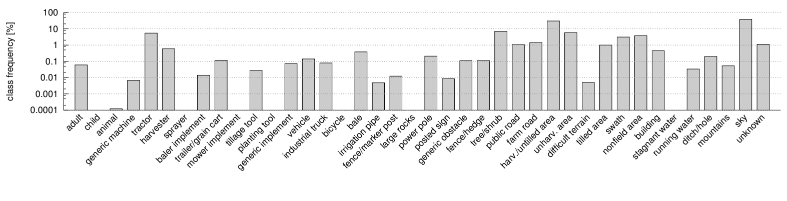


Figure 2.2: Class list and occurrence frequencies in the constructed dataset.

frame rate of 30 Hz. One of the vehicles used in the campaigns is shown in Figure 2.1. A wide-angle field-of-view was chosen to capture additional surroundings using the following camera angle settings: 69.5° vertical, 118.2° horizontal, and 133.6° diagonal, with a focal length of 14 mm. The resulting fisheye effect was corrected post-hoc through an inverse transformation.

The different captured videos were split into image sequences with a frequency of 1 Hz. From this set of images, for each measurement campaign, the most visibly informative and distinctive ones in terms of objects and perspective were selected. This resulted in a dataset of 595 training and 149 test images, which were labeled by an external party that was given labeling guidelines. For this task, a labeling tool was built that allows operators to load an image, segment it into over-segmentation regions, and assign classes to these regions. Some representative dataset images are shown in Figure 2.3, while Figure 2.2 shows the list of classes with their occurrence frequencies.

2.3 Preprocessing

Before the images are fed to the prediction model, they pass through a preprocessing pipeline in order to extract uniform feature representations. In this section, we first describe the region segmentation approach, after which we explain how features are extracted.

2.3.1 Region Over-segmentation

Classifying every pixel is computationally challenging. Therefore, we first segment the image into visually coherent regions, or superpixels, as shown in Fig. 2.5. This is done via the SLIC algorithm [Achanta et al., 2012],

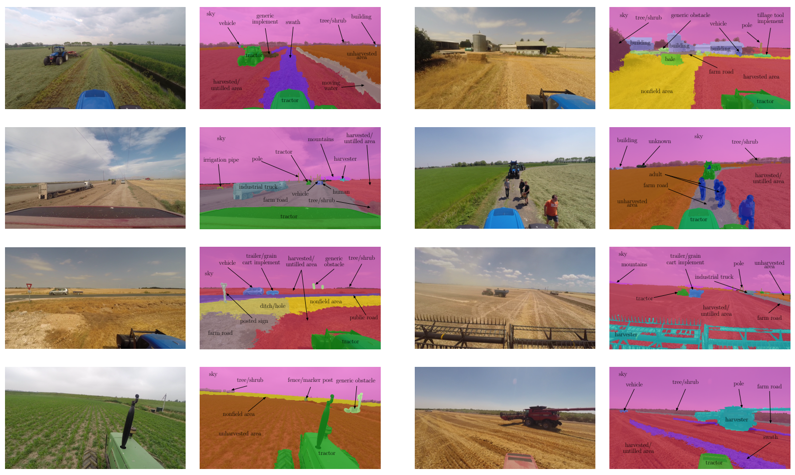


Figure 2.3: Illustation of dataset images (columns 1 and 3) and corresponding manual labelings (columns 2 and 4)

which clusters all pixels in a 5-dim space composed of the CIELab L-, a-, and b-values, and the pixel coordinates. First, a grid is defined on top of the image, whose nodes serve as starting positions for K distinct clusters. The clustering distance used is a weighted sum of both the distance in the (L, a, b)-space and the (x, y)-space. The weights allow for tuning the fuzziness of the region boundaries. By means of expectation-maximization, the clusters evolve until convergence.

2.3.2 Feature Extraction

The next step is the feature extraction process in which each distinct region is assigned a uniformly-sized feature vector based on both gradient and color information. Gradient information is extracted as 200-dim features through the DAISY [Tola et al., 2010] algorithm, while color information is modeled by the HSV-value of each pixel, leading to 3-dim color features. What we obtain now is a set of features for each image pixel for the whole training dataset.

Because the segmentation regions vary in size, the number of extracted features also varies. However, we want to obtain a uniform feature vector for each region, independent of its size. Therefore, after we have extracted features from all pixels, we apply k-means clustering to all extracted gradient features, and to all extracted color features from all image pixels. Gradient features are clustered into G clusters, while color features are clustered into C clusters.

To reduce the clustering complexity, the features are not extracted for each image pixel, but only for pixels that lie on a regular grid. The feature cluster centers now form so-called *words*. After these words have been identified, the extracted features within the same segmentation region are mapped to the closest word in terms of Euclidean distance in the feature space, forming bags-of-words.

Because the number of feature clusters, namely G and C, is fixed, a histogram can be built for each segmentation region that contains the frequency of occurrence of each of the words. Moreover, the relative position of the region center, i.e., the median of its pixel coordinates, is added as a feature. A uniform representation is obtained for each segmentation region by concatenating the two histograms and the center into a single vector in $\mathbb{R}^{(G+C+2)}$, which acts as the input to our prediction model. In contrast to the grid-based feature extraction process used for constructing the words of the bag-of-words model, the features used to construct the uniform feature vector for each region are densely sampled. In our case, we specifically used $G = 300$ and $C = 150$.

2.3.3 Convolutional Unary Classifier

A convolutional neural network (CNN) is used to learn representations of each superpixel. The CNN is trained in a supervised fashion, taking $3 \times 84 \times 84$ square windows around the median center of each superpixel as input, and outputs a probability distribution over all possible classes. This CNN has the following architecture. First, the $3 \times 84 \times 84$ windows are downscaled 3-channel 64×64 patches. Next, a convolutional layer of 16 $5{\times}5$ filters is used, after which a 2-dim max-pooling operation downsamples the 16 feature maps to 32×32 pixels. Hereafter, 2 additional convolutional layers with max-pooling operations are applied, with 32 and 64 3×3 filters respectively. Hereafter, 2 subsequent dense layers of 1024 units are used, which end in an 18-bin softmax output layer. Additionally, the previously mentioned bag-of-word features (452-dim) are added to the first dense layer.

The CNN architecture is visually described in Figure 2.4.

Adam [Kingma and Ba, 2015] is used as a learning scheme to minimize the cross-entropy, using minibatches of size 32. The cross-entropy term is weighted by the inverse of the square root of the class frequencies to correct for class imbalance. Batch normalization [Ioffe and Szegedy, 2015] is applied to every non-output layer; the layer weights are initialized according to [Glorot and Bengio, 2010]; a dropout [Srivastava et al., 2014] rate of 50% is applied to both dense layers; all nonlinearities are exponential linear units (ELUs) [Clevert et al., 2015]; label smoothing of 0.0003 is applied to the softmax output. Furthermore, the dataset is augmented through random mirroring over the vertical axis. The CNN softmax ouput probabilities are used as input features to the SSVM described in the following sections. Recently, related methods have been proposed by [Mostajabi et al., 2015].

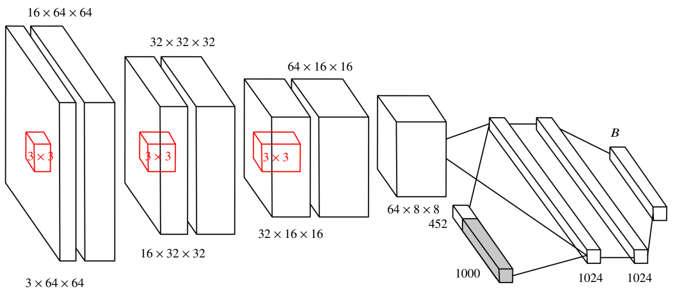


Figure 2.4: The convolutional neural network architecture used for the unary predictor which takes as input windows centered around each superpixel

To enhance the accuracy of the proposed unary classification model, transfer learning is used through leverage of a CNN that was pretrained for classification on the ImageNet dataset [Russakovsky et al., 2015]. In particular, we use the OverFeat CNN model [Sermanet et al., 2013], which classifies 3×232×232 images into 1000 distinct ImageNet classes. Although the majority of these classes do not correspond the classes in our dataset, the goal is to reuse certain learned features in the segmentation model. This can aid in lowering the overfitting behavior caused by the bias in the proposed agricultural dataset to particular types of images. The pretrained CNN model is applied to the same windows as used by the classifier described in the previous paragraph. The $3 \times 84 \times 84$ windows are scaled up to $3 \times 232 \times 232$ to match the pretrained CNN input dimensions. The class probabilities that are outputted for each superpixel act as an additional 1000-dim feature vector, which is fed into the first dense layer of the CNN model described in the previous paragraph. This is shown in Figure 2.4 as the gray-colored block.

2.4 Structured Output Prediction

Traditional machine learning models for classification predict a single output through a function $f : \mathcal{X} \to \mathbb{R}$. In contrast, structured prediction models [Nowozin and Lampert, 2011] are defined by a function $f : \mathcal{X} \to \mathcal{Y}$ of which the output is arbitrarily structured. In this work, this structure is shaped as a vector in $\mathcal{Y} = \mathcal{L}^V = \{1, \ldots, K\}^V$, with V the number of output variables to be predicted. In our use case, V is the number of regions present in the input image, as explained in Section 2.3.1. Structured models maximize a compatibility function $g : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ to obtain the predicted value $f(x)$, defined as

$$f(x) = \operatorname*{arg\,max}_{y \in \mathcal{Y}} g(x, y), \tag{2.1}$$

which is called inference. In this work, we use a linearly parametrized function $g(x, y) = \langle w, \Psi(x, y) \rangle$, in which w is learned from data and $\Psi(x, y)$ is a joint feature vector of both x and y. Because the domain $\mathcal{Y}$ is very large, as it is a combination of label assignments to all regions, Ψ has to be defined in such a way that underlying structures can be exploited. In this work, we ensure that Ψ identifies with the energy function of an SSVM [Nowozin and Lampert, 2011], for which efficient inference techniques exist that correspond to maximizing the compatibility g.

2.4.1 Structural Support Vector Machines

We first explain conditional random fields (CRFs), and afterwards the SSVM model as used in this chapter is derived. A CRF is a type of probabilistic graphical model, namely the conditional variant of a Markov random field. A graphical model defines a probability distribution in a compact way by making explicit dependencies between random variables. CRFs in particular define not the full joint distribution over all random variables, but the conditional distribution of the labels, given the input. This allows for tractable inference within the model [Nowozin and Lampert, 2011].

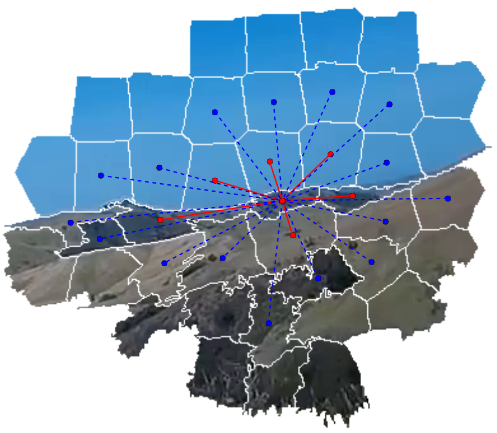


Figure 2.5: Neighborhood connectivity with both first-degree (solid red) and second-degree (dashed blue) neighbor connections for an actual region over-segmentation

A CRF models the conditional probability distribution $p(y|x)$, with x an observation, and y an assignment of labels in $\mathcal{Y}$. This distribution can be written as

$$p(y|x) = \frac{1}{Z(x)} \prod_{F \in \mathcal{F}} \psi_F(y_F, x_F), \text{ with} \tag{2.2}$$

$$Z(x) = \sum_{y \in \mathcal{Y}} \prod_{F \in \mathcal{F}} \psi_F(y_F, x_F) \tag{2.3}$$

called the partition function, which normalizes the distribution. Herein, $\mathcal{F}$ represents the set of factors in the CRF, which model relations between variables. The model as presented here can be split up into two types of factors, namely unary and pairwise factors, as follows:

$$p(y|x) = \frac{1}{Z(x)} \prod_{i \in \mathcal{V}} \psi_i(y_i, x_i) \prod_{(i,j) \in \mathcal{E}} \psi_{ij}\big(y_i, y_j\big), \tag{2.4}$$

for a CRF represented by a graph $G = (\mathcal{V}, \mathcal{E})$. Fig. 2.6 depicts the graphical model as used in this work, while Fig. 2.5 shows the second-order neighborhood connections for an actual image over-segmentation. The unary terms $\psi_i(y_i, x_i)$ represent relations between input features x_i and region labels y_i, while the pairwise terms $\psi_{ij}\big(y_i, y_j\big)$ model relations between regions. For example, when x_i is a green-colored region, then ψ_i is large if

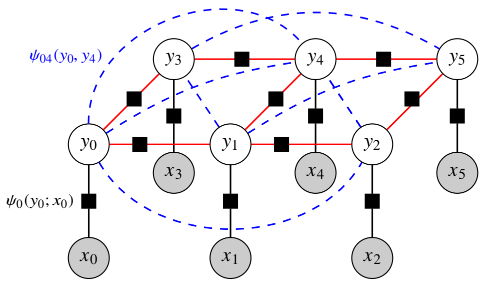


Figure 2.6: A CRF is shown with second-degree neighborhood connectivity graph, represented as a factor graph with observations x_i representing region feature vectors, and random variables representing label assignments y_i. The blocks represent the factors/potentials (not drawn for second-degree edges), while their incident nodes represent their arguments.

y_i = `grass` and low if y_i = `road`. For the pairwise factors, ψ_{ij} is high if $y_i = y_j$ and low otherwise. This leads to a smoothing effect by ensuring that proximal and similar regions favor equal labels. In our scenario, interlabel relations are linked together based on each region's first- and second-degree neighborhood[1], as will be explained later on.

Because the normalization function $Z(x)$ is difficult to compute [Nowozin and Lampert, 2011] and we are foremost interested in retrieving the most-likely labels, we convert the CRF into an energy-based model by dropping the normalization function $Z(x)$ and defining energy potentials as $E_F(y_F, x_F) = -\log \psi_F(y_F, x_F)$. By altering the CRF in this way, we obtain a model called a structural support vector machine (SSVM) [Taskar et al., 2003, Nowozin and Lampert, 2011], which defines the total energy as

$$E(y, x) = \sum_{i \in V} E_i(y_i, x_i) + \sum_{(i,j) \in \mathcal{E}} E_{ij}\left(y_i, y_j\right). \tag{2.5}$$

Computing $y^* = \arg\max_{y \in \mathcal{Y}} p(y|x)$, called maximum a priori (MAP) inference, requires maximization of $p(y|x)$, which is the same as minimizing $E(y, x)$. The factor energies are then linearly parametrized [Lucchi et al.,

[1]Second-degree neighbors are defined as neighbors of neighbors, based on a region connectivity graph.

2012] as

$$E_i(y_i, x_i; w^U) = \left\langle w^U, \psi_i(y_i, x_i) \right\rangle \text{ and} \tag{2.6}$$

$$E_{ij}(y_i, y_j; w^P) = \left\langle w^P, \psi_{ij}\left(y_i, y_j\right) \right\rangle, \tag{2.7}$$

with w^U and w^P the unary and pairwise parameters respectively. Constructing a weight vector $w = \left[\left(w^U\right)^\top, \left(w^P\right)^\top\right]^\top$, and similarly a feature vector $\Psi(y, x) = \left[\left(\Psi^U\right)^\top, \left(\Psi^P\right)^\top\right]^\top$, with Ψ^U and Ψ^P respectively the sum of all unary features $\psi_i\left(y_i, x_i\right)$ and the sum of all pairwise features $\psi_{ij}\left(y_i, y_j\right)$, leads to a linear parametrization of E :

$$E(y, x; w) = \langle w, \Psi(y, x) \rangle. \tag{2.8}$$

In this formulation, the unary features are defined as

$$\psi_i\left(y_i, x_i\right) = \left(h\left(x_i\right)^\top \left[y_i = m\right]\right)^\top_{(m \in \mathcal{L})}, \tag{2.9}$$

with $[\cdot]$ the Iverson brackets, and $h(x_i)$ the probabilistic output of a unary classification function.

To obtain a fine-grained segmentation of the image, e.g., to detect vehicles and signs from a distance, we choose a fine-grained region segmentation of 1000 regions. Because neighboring interactions span a low distance, this leads to noisy predictions. For this reason, both first- and second-degree neighboring regions are connected by means of pairwise potentials, as shown in Fig. 2.6. Contrary to the unary factors that depend linearly on the inputs $h(x_i)$, the pairwise factors are uniform for all region interactions. However, a different interaction factor is used for first- and second-degree connections, which allows for differentiation between short- and medium-distance interactions. This is encoded into the pairwise features, as defined by

$$\psi_{ij}(y_i, y_j) = \left(\pi(i, j)[y_i = m \wedge y_j = n]\right)^\top_{((m,n) \in \mathcal{L}^2)}, \tag{2.10}$$

with $\pi(i, j) = (1, 0)$ if the incident nodes of edge $(i, j) \in \mathcal{E}$ are first-degree neighbors, and $\pi(i, j) = (0, 1)$ if they are second-degree neighbors. This effectively encodes the separate pairwise table for both first- and second-degree links into the first and second column of w^P.

In Eq. (2.8), it can be noticed that $E(y, x; w)$ has a form similar to $g(x, y)$, the compatibility function g defined at the beginning of this section. Computing Eq. (2.1) is therefore equivalent to MAP inference in the SSVM model. Doing so allows us to avoid the intractable computation of g over all y-values in $\mathcal{Y}$ by the use of efficient SSVM MAP inference methods, as will be explained in Section 2.4.3.

2.4.2 Max-margin Learning

Training the SSVM means tuning the parameters w of the energy function based on a training dataset, such that its predictions generalize well on a test set. This can be done through so-called max-margin learning methods, using quadratic programming. In structured prediction, we minimize a regularized structured risk rather than the Bayes' risk, namely

$$R(w) + \frac{\lambda}{N} \sum_{n=1}^{N} \Delta(y^n, f(x^n)), \tag{2.11}$$

with $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$ the loss function for which holds $\forall y, y' \in \mathcal{Y} : \Delta(y, y') \geq 0$, $\Delta(y, y) = 0$, and $\Delta(y', y) = \Delta(y, y')$, $R(\cdot)$ a regularization function, λ the inverse regularization strength, and a training set $\{(x^n, y^n)\}_{n \in \{1,\ldots,N\}} \subset \mathcal{X} \times \mathcal{Y}$. Due to the piecewise nature of this function, caused by the loss function shape, gradient-based optimization techniques are unusable [Nowozin and Lampert, 2011]. Therefore, we minimize a convex upper bound of this function

$$w^* = \underset{w \in \mathbb{R}^D}{\arg\min} \left[R(w) + \frac{\lambda}{N} \sum_{n=1}^{N} u(y^n, x^n; w) \right], \tag{2.12}$$

with $u(\cdot, \cdot)$ a function that extracts the maximal difference between the prediction loss and the energy loss for a data sample (x^n, y^n), defined as

$$u(y^n, x^n; w) = \\ \max_{y \in \mathcal{Y}} \left[\Delta(y, y^n) - (E(y, x^n; w) - E(y^n, x^n; w)) \right]. \tag{2.13}$$

In this work, we optimize this objective function by means of the N-slack cutting plane method [Joachims et al., 2009] with L_2-regularization, which reformulates the above optimization problem as

$$w^* = \operatorname*{arg\,min}_{w,\xi_1,\ldots,\xi_N} \left[\frac{1}{2} \|w\|^2 + \frac{\lambda}{N} \sum_{i=1}^{N} \xi_i \right], \text{ s.t.}$$
$$E(y, x^n; w) - E(y^n, x^n; w) \geq \Delta(y, y^n) - \xi_n, \quad (2.14)$$

with $n \in \{1, \ldots, N\}$ and $y \in \mathcal{Y}$. Herein, ξ_n are slack variables, which are introduced to allow for linearly punished constraint violation. By doing this, the maximization function in Eq. (2.13) is translated into linear constraints. Moreover, the objective function becomes quadratic in w. This allows quadratic optimization, for which various optimization libraries exist. However, a downside is the large number of constraints due to the high dimensionality of $\mathcal{Y}$.

To counter this, we use cutting plane optimization [Joachims et al., 2009], in which a working set of constraints W is utilized. Cutting plane optimization starts by solving the optimization problem with $W = \emptyset$, and iteratively adds additional constraints. Only those constraints which are maximally violated, for each training sample (x^n, y^n), are added to W. This allows fast optimization at the start of the procedure, combined with strong results.

For $\Delta(\cdot, \cdot)$, we chose a weighted ontological loss function based on the distance between classes in an ontology, as

$$\Delta(y, y^n) = \frac{1}{V} \sum_{i=1}^{V} \theta_{y_i^n} \delta\left(y_i^n, y_i\right), \quad (2.15)$$

with V the number of segmentation regions. The weights $\theta_{y_i^n}$ are set to the inverse of the square root of label frequency in the training set, normalized over all labels, to correct for some class imbalance. The function $\delta : \mathcal{L} \times \mathcal{L} \rightarrow \mathbb{R}_{\geq 0}$ represents the distance along the spanning tree defined by the ontology. As such, the loss of misclassification between two classes that are conceptually far apart is high.

2.4.3 Reasoning

In order to infer a set of labels that maximally correspond to the regions of an input image, we have to calculate $f(x)$, as defined in Eq. (2.1). However, brute-force calculation of $f(x)$ is intractable as it involves due to the combinatorial complexity of the set of possible labelings, we rely on using tractable approximate reasoning. An inference technique called α-expansion [Boykov et al., 2001] is used. α-expansion breaks up the energy minimization problem of Eq. (2.1), based on Eq. (2.5), into sequential subproblems. In a subproblem, SSVM nodes have the possibility to alter their label y_i to a different (but fixed) label α.

Each subproblem is converted into an auxiliary graph of which the nodes and edges are constructed in such a manner, that a graph cut results in an assignment of labels in which y_i remains constant, or changes to α. Because the edge weights are set to the SSVM energies, finding a minimal cut corresponds to a labeling change that results in a minimal energy value (for a particular α switch). Solving the subproblems sequentially for different α-values, yields an approximately optimal labeling. A more in-depth analysis, including energy formulation requirements and restrictions, is given in [Boykov et al., 2001, Nowozin and Lampert, 2011].

2.4.4 End-to-end Segmentation

This section propses an alternative approach that avoids the use of an SSVM model completely. Here, a CNN is modeled to take as input a complete $3 \times 640 \times 320$ image in order to output a full segmentation. This avoids the need for an over-segmentation preprocessing method, as described in Section 2.3.1. Its advantage is that the segmentations are no longer limited by errors made by the SLIC procedure. A possible disadvantage is that less prior structure is imposed through both the superpixel method and the SSVM connectivity graph. The proposed model is a type of convolutional autoencoder with the following architecture, which is also visually described in Figure 2.7. First a stack of convolutional layers is applied to the 3-channel input images, composed of respectively 16, 32, 64, and 128 filters of size 3×3. After each convolutional layer, a 2-dim max-pooling operation is applied, halving the size of each set of feature maps.

After the convolutional stack, 2 dense layers of respectively 512 and 8000 units are used, which feed into the stack of transposed convolutional layers. This stack first reshapes the set of 8000 units into a tensor of size $10 \times 20 \times 40$, after which layers of respectively 128, 64, 32, and 16 filters of size 3×3 are used. After each of these layers, a 2-dim upscaling layer is applied, doubling the size of each feature map. The final feature map

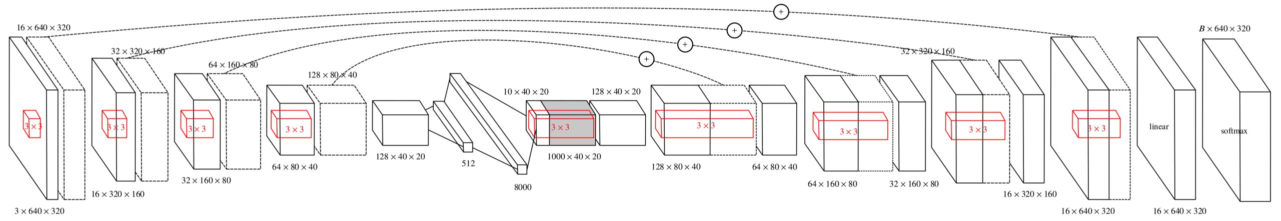


Figure 2.7: End-to-end convolutional segmentation architecture; the circled plus connections represent the skip-layer connections; the gray box represents the OverFeat CNN class probabilities according to the window extraction process.

is fed into a convolutional layer of 16 3×3 filters, after which a linear transformation is applied, which connects to a final 18-bin softmax layer.

To ensure sufficiently detailed segmentations, skip-layer connections are added that connect the convolutional layers, before their max-pooling operation, to their transposed convolutional counterparts. As such, the input of each transposed convolutional layer is expanded through concatenation. This is made clear in Figure 2.7 by the horizontal dashed connections with the circled plus symbol. All nonlinear transformations are composed of ELUs [Clevert et al., 2015]; the learning scheme Adam [Kingma and Ba, 2015] optimizes the cross-entropy augmented with L_2-regularization using minibatches of size 4. The cross-entropy is weighted by the inverse of the square root of the class frequencies. The training set is augmented through random image mirroring over the vertical axis. Recently a related method has been proposed by [Long et al., 2015] and [Ronneberger et al., 2015].

Similar to Section 2.3.3, transfer learning is used to leverage of information learned on the ImageNet [Russakovsky et al., 2015] dataset. Again, we make use of the OverFeat pretrained CNN model [Sermanet et al., 2013]. However, this time the model is slided with a fixed stride over 2x upscaled input images and applied to the resulting $3 \times 232 \times 232$ windows. This transformation results in a 1000-dim class probability vector extracted along a grid of 20×40 points. As such, a tensor of size $1000 \times 20 \times 40$ is formed, which is concatenated to the reshaped tensor of the 8000 units of the second dense layer, resulting in a feature map of size $1010 \times 20 \times 40$, shown in Figure 2.7 by the gray-colored block. The transposed convolutional stack of layers can now make use of transferred external knowledge in order to make more informed decisions about the segmentation outputs.

2.5 Results and Discussion

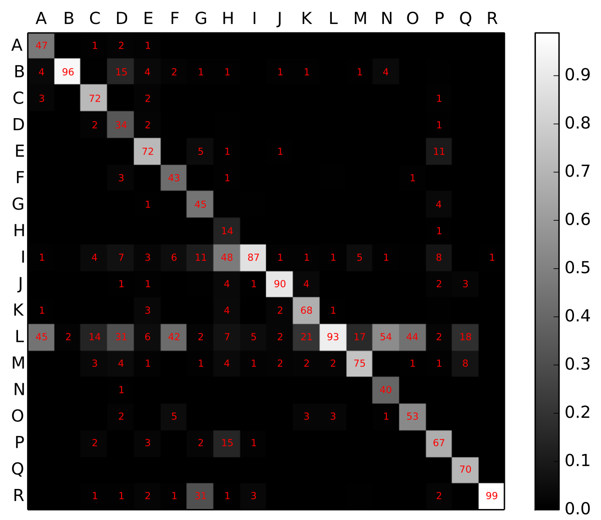
Quantitative results for both models on the test dataset are shown in Figures 2.9 and 2.8 through confusion matrices. The total set of classes in the dataset was translated into a set of 18 most interesting class groups. In these matrices, each row i represents pixels that have a ground truth label of class i, while each column j represents predictions made by the model as class j. Therefore, in a confusion matrix, at position (i, j), the number of pixels classified as j but actually belonging to class i is shown. In this work, rather than showing the actual pixel counts at each position (i, j), we show this count divided by the row sum (in %), which forms the recall confusion matrix. Hence, the values in each row sum to 100%[2]. The diagonal values of the matrix can be interpreted as the per-class accuracy values, while the off-diagonal values represent the percentage of pixels (belonging to a particular class) mispredicted as another class. If we would interpret the ground truth labels as model predictions, a matrix would be obtained with only values of 100% on its diagonal, and 0% everywhere else. Moreover, we show the precision confusion matrix, which is composed of the actual pixel counts divided by the column sum (in %).

These results reveal that the mistakes made by both methods are logically interpretable, e.g., mistaking types of land or types of vehicle, while their accuracy results are very similar. When looking at the average accuracy values, the unary classification model achieves an average precision of 64.7% and an average recall of 67.7%. The end-to-end model achieves an average precision of 62.6% and an average recall of 67.7%. The global pixel-wise accuracy, which is the fraction of correctly classified pixels, is 91.2% for the SSVM method and 90.9% for the end-to-end method.

Since the quantitative results of both methods are very similar, illustrative qualitative results on the test dataset are shown in Figures 2.10 and 2.11. On the one hand, these results reveal that the SSVM method tends to overshoot when the over-segmentations are inaccurate, while the end-to-end method suffers from no such limitation. On the other hand, when the superpixels do align, they nicely delineate objects, leading to highly accurate and coherent segmentations. This is true even when the actual ground truth labels do not correctly delineate the objects. Although the quantitative results of both models are very similar, it can be noticed that the end-to-end segmentation method is capable of correctly predicting objects that are very far away, which is its main advantage over the over-segmentation method.

The results of both models highlight their capability to operate in highly cluttered environments by achieving accurate image segmentations. Such predictions could form a basis for steering autonomous vehicles towards particular types of land, e.g., unharvested fields, or to avoid dangerous areas, e.g., public roads. The accurate segmentation of trees and shrubbery can aid in marking field boundaries or in avoiding collisions, as does the segmentation of objects in general, e.g., power poles. Segmenting sky regions can prove to be helpful in horizon estimation, or to avoid running

[2]Note that due to rounding errors in the figures, the sum may not be exactly 100%.



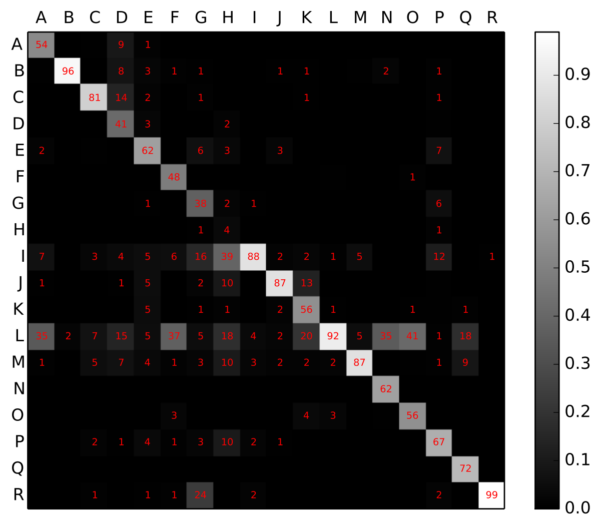


Figure 2.8: SSVM with a CNN unary classifier (top) and end-to-end segmentation model (bottom): pixel-wise precision results on the autonomous agricultural vehicle test dataset, described as a confusion matrix. Class legend: person (A), tractor (B), harvester (C), implement (D), moving object (E), nonmoving object (F), power pole (G), fence/hedge (H), tree/shrubbery (I), public road (J), farm road (K), harvested untilled area (L), unharvested area (M), tilled area (N), swath (O), building (P), water (Q), sky (R).

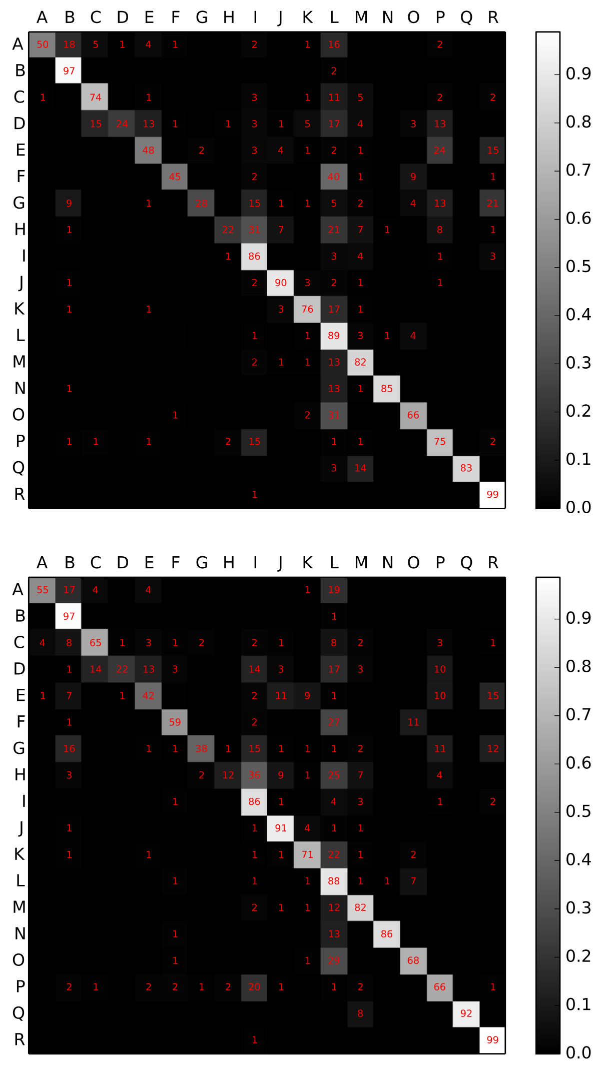


Figure 2.9: SSVM with a CNN unary classifier (top) and end-to-end segmentation model (bottom): pixel-wise recall results on the autonomous agricultural vehicle test dataset, described as a confusion matrix. Class legend: person (A), tractor (B), harvester (C), implement (D), moving object (E), nonmoving object (F), power pole (G), fence/hedge (H), tree/shrubbery (I), public road (J), farm road (K), harvested untilled area (L), unharvested area (M), tilled area (N), swath (O), building (P), water (Q), sky (R).

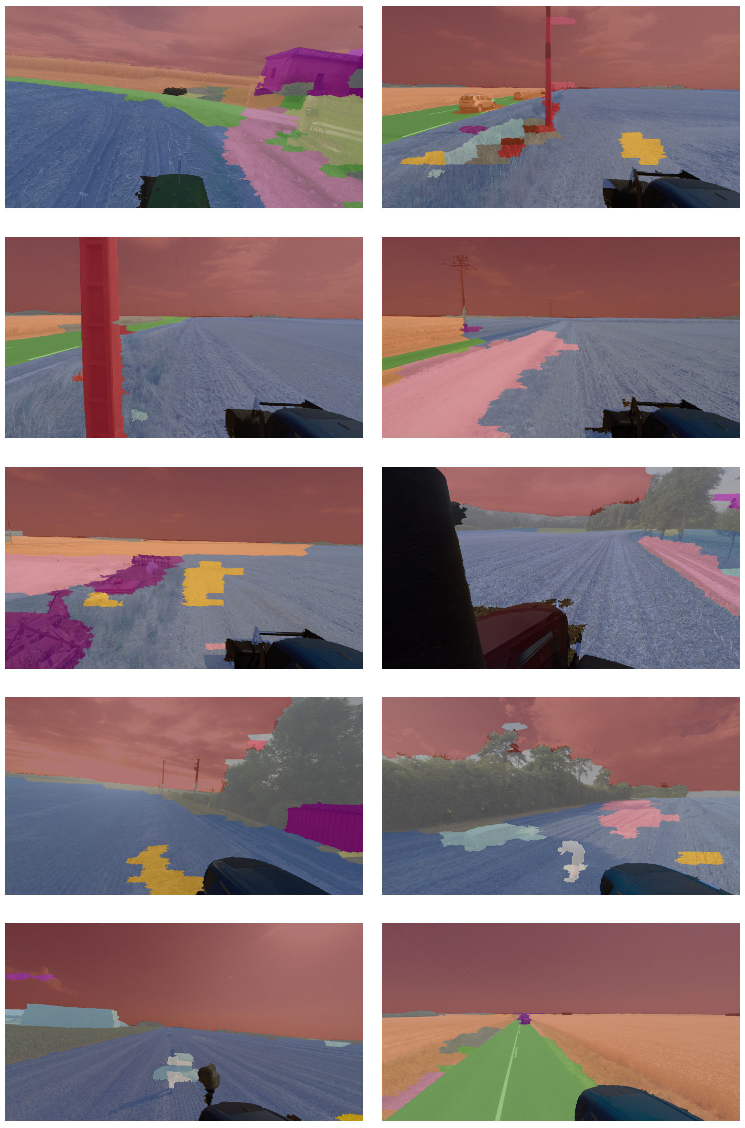


Figure 2.10: SSVM combined with the unary CNN classifier: illustration of segmented test images from the autonomous agricultural vehicle dataset; visible class legend: public road (green), sky (red), private road (pink), unharvested area (orange), human (white), tractor (black), building or implement (purple), swath (yellow), harvester (bright yellow), harvested area (blue), trees/shrubbery (gray), generic nonmoving object (cyan), power pole (dark red), car (dark orange), fence/hedge (khaki)

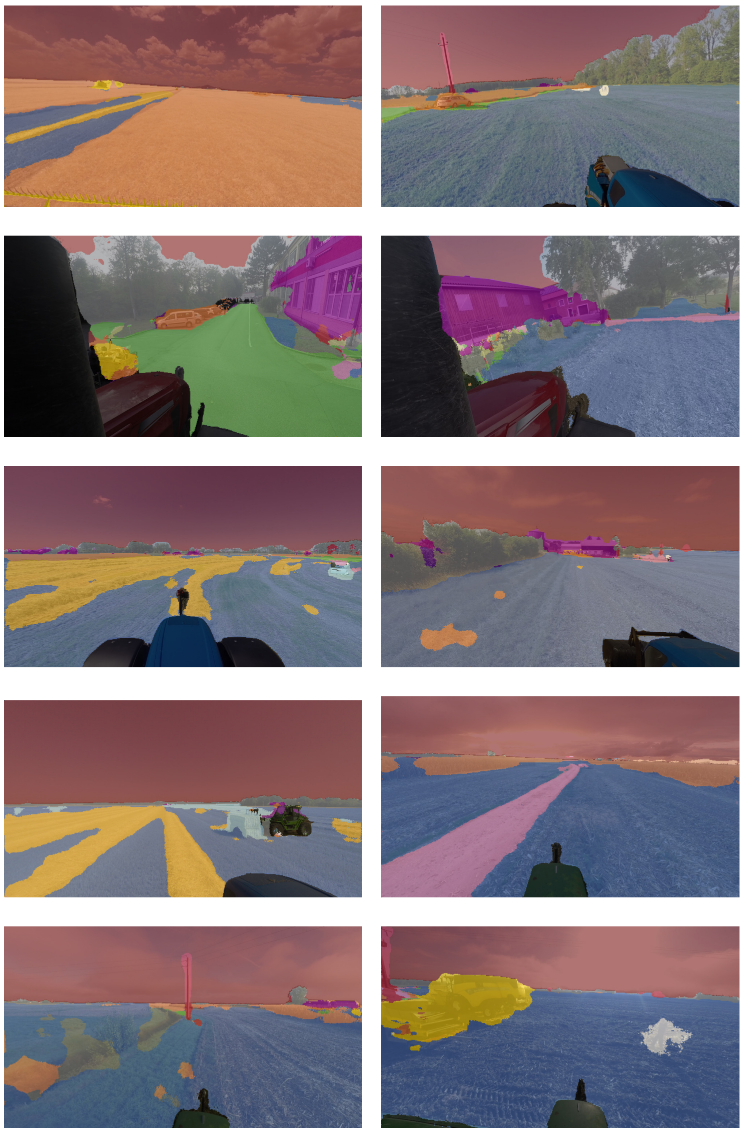


Figure 2.11: End-to-end segmentation: illustration of segmented test images from the autonomous agricultural vehicle dataset; visible class legend: public road (green), sky (red), private road (pink), unharvested area (orange), human (white), tractor (black), building or implement (purple), swath (yellow), harvester (bright yellow), harvested area (blue), trees/shrubbery (gray), generic nonmoving object (cyan), power pole (dark red), car (dark orange), fence/hedge (khaki)

other expensive algorithms on image regions of which we know a priori that they do not contain important information, e.g., when trying to localize particular objects.

2.6 Related Work

This section explores literature related to perception in outdoor autonomous vehicles and outdoor scene understanding, and how our work differs from previous approaches.

Byun et al. [Byun et al., 2015] used a Markov random field for predicting road regions and obstacles that fall out of the perception sensor range. Their approach relies on manually engineered unary and pairwise potential functions. In contrast, our work makes use of structured learning through structural support vector machines (SSVMs) to simultaneously tune all graphical model potentials, allowing the model to adapt to previously recorded data. Bosch et al. [Bosch et al., 2007] investigated the use of segmentation in outdoor environments by means of a probabilistic pixel map and fuzzy classifiers. In contrast, we discard probabilities and focus on the model's discriminative aspect by using an energy-based formulation of our model. Armbrust et al. [Armbrust et al., 2009] built an off-road autonomous vehicle with the goal of operating in highly vegetated terrain. Their system integrates multiple manually-engineered sensor processing systems, and is responsible for traversable region and object detection, omitting any form of semantics. Kelly et al. [Kelly et al., 2006] and Leonard et al. [Leonard et al., 2008] similarly focused on an integration of sensor processing systems in which traversable regions are detected. Geiger et al. [Geiger et al., 2014] proposed a combined scene flow, vanishing point, and scene segmentation approach for 3-dim traffic understanding using geometric features. Contrary to their approach, we focus on semantic scene understanding through an SSVM, which enables accurate segmentations based on contextual information.

Several authors tackle the problem of outdoor scene segmentation by means of convolutional neural networks [Hadsell et al., 2009, Sermanet et al., 2009, Hadsell et al., 2007]. Kuthirummal et al. [Kuthirummal et al., 2011] built a traversable region map using a 3-dim grid. Other work proposed the use of radar for traversable region detection [Reina et al., 2012, Milella et al., 2014, Milella et al., 2011], however, this does not permit high-level reasoning about objects and their semantics. Furthermore, Nourani-Vatani et al. [Nourani-Vatani et al., 2015] propose the use of SSVMs with a hierarchical loss function for seafloor imagery classification in autonomous underwater vehicles. In contrast, we focus on segmentation rather than classification.

Several works study agricultural autonomous vehicles. Closely related is the work of [Rouveure et al., 2012], in which a model for ambient awareness in autonomous tractors is proposed, within the overarching QUAD-AV project. Using a multisensory approach based on stereovision, laser scanning, thermography, and microwave radar, they focus specifically on obstacle detection by classifying the surroundings as either traversable or non-traversable. In contrast, rather than identifying traversable regions through a combination of 2-dim and 3-dim data, our model segments 2-dim images into a multitude of semantic classes. In light of the same project, [Reina and Milella, 2012] and [Milella et al., 2013] propose a two-stage terrain classifier in which geometry-based information, e.g., the estimated terrain slope, is first used to classify terrain into broad categories, which are refined in a second stage using a color-based classifier. They propose a self-learning model in order to avoid the tedious task of manual labeling. A similar method is proposed in [Reina et al., 2012] for generic rural environments using a radar sensor, as well as in [Milella and Reina, 2014], using a multi-baseline camera while relying only on geometric classification. Rather than using independent classification on grid cells defined on top of the 3-dim point cloud or 2-dim image, we use a structured prediction model to classify irregular image regions simultaneously. This allows for coherent and fine-grained image segmentation for an arbitrary set of classes. Within the same QUAD-AV project, [Bellone et al., 2013a, Bellone et al., 2013b] propose RGB-D vision-based terrain analysis using normal vector estimation through principal component analysis of point cloud data, in order to assess traversability, suitable for outdoor environments. We obtain traversability assessment indirectly through semantic segmentation of the ground/field, in which traversable regions are identified as being part of a subset of classes, e.g., harvested, tilled, and unharvested field.

[Weiss and Biber, 2010] propose a classification model of the environment using a stochastic automaton to describe relations between classes. To classify the environment, they employ a predefined pattern database against which laser data patterns are matched in order to determine where the autonomous vehicle is located in the field. In their other work [Weiss and Biber, 2011], they propose a plant detection system based on 3-dim laser scanner data. Herein, they employ a clustering approach, while we use machine learning to allow the vision system to be trained in a supervised fashion. In [Weiss et al., 2010], the same authors propose a supervised learning approach for plant recognition. In contrast to this work, our goal is to obtain a classification of every pixel, rather than identifying specific objects.

An earlier segmentation study is presented by [Subramanian et al., 2006]. Herein a rudimentary form of segmentation is used to segment trees on the side of citrus grove alleyways. Their model is, however, tuned for this specific setting, making it difficult to generalize their results to significantly different settings. The application of our model is not restricted to a particular type of agricultural environment. [Guijarro et al., 2011] propose a segmentation approach for plants, e.g., barley, corn, and cereal, and soil in agricultural images. Their model can be viewed as a component in a larger machine vision system, but cannot be used for generic segmentation in which we want to detect a set of arbitrary classes. [Xue et al., 2012] propose a machine vision method for autonomous navigation in cornfield rows using background subtraction to segment the plants from the field. The processed images were used for detecting rows, which are fed to a fuzzy logic controller that steers the vehicle. [Åstrand and Baerveldt, 2005] propose machine vision for plant row recognition using the Hough transform in order to guide autonomous vehicles. Similarly, [Bergerman et al., 2012] investigate row detection and row trajectory planning in autonomous orchard vehicles. In general, these methods are tailored to very specific settings, while our goal is to obtain a method usable in generic agricultural environments.

Another related work is [Rovira-Más et al., 2008] in which stereovision, localization sensor information, and inertial information is used to build a 3-dim map of the environment in autonomous vehicles. Such works could prove to be an asset in our pursuit towards autonomous agricultural vehicles by projecting the segmentation output onto a map built through their approach. [Moreno et al., 2014] study laser-finder mapping applied to autonomous spraying vehicles, relying on GPS. Contrary to their work, we avoid any form of sensor calibration.

Other recent works of [Kraus et al., 2013] and [Kayacan et al., 2015] focus on lower-level tasks in agricultural robotics, such as vehicle dynamics control in reference trajectory tracking. [Godoy et al., 2012] similarly focus on vehicle control, albeit in a distributed fashion. These works are orthogonal to our study as we focus on semantic perception. [Oksanen, 2015] evaluates the performance of an autonomous wheat sowing vehicle developed in previous studies [Oksanen and Linkolehto, 2013]. The vehicle drives along a predefined polyline trajectory, composed of multiple waypoints, by making use of inertial sensors and a GPS signal. An interesting outcome of this study is that being heavily reliant on GPS is not sufficiently robust for autonomous operation, due to signal loss and shadowing. We argue that intelligent vision can aid in robustifying autonomous behavior.

Most of these studies are tailored to a specific task, heavily relying on the programming of background knowledge. It is hard to envision how these systems can generalize to significantly different environments. The goal of our work is to allow for robust vehicle guidance through intelligent vision that only relies on real-time video data. By training the vision model through machine learning, using a diverse set of data, our model generalizes to different settings. The output of the final processing stage can be used for further high-level reasoning, e.g., to verify whether the current position, inferred from detected objects, matches the GPS signal and ground map.

2.7 Conclusions

Advanced perception systems that understand the environment are essential in enabling autonomous vehicles. We modeled a structured output machine learning model that takes into account visual contextual cues between over-segmentation regions within a second-order neighborhood. The structured model is formulated as an energy-based function using feature-dependent unary potentials, and pairwise potentials that differentiate between first- and second-degree region neighbors. After optimization by means of max-margin learning, most-probable label assignments are obtained via graph cuts inference using α-expansion. Results indicate that using second-degree contextual information allows for a high labeling accuracy by better filtering out spurious labels. The next chapter will investigate how to integrate the SSVM's high segmentation coherence with the detailed output of the end-to-end model.

References

[Achanta et al., 2012] Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., and Susstrunk, S. (2012). SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282.

[Armbrust et al., 2009] Armbrust, C., Braun, T., Föhst, T., Proetzsch, M., Renner, A., Schäfer, B.-H., and Berns, K. (2009). RAVON–the robust autonomous vehicle for off-road navigation. In *Proceedings of the IARP International Workshop on Robotics for Risky Interventions and Environmental Surveillance*, pages 12–14.

[Åstrand and Baerveldt, 2005] Åstrand, B. and Baerveldt, A.-J. (2005). A vision based row-following system for agricultural field machinery. *Mechatronics*, 15(2):251–269.

[Bellone et al., 2013a] Bellone, M., Messina, A., and Reina, G. (2013a). A new approach for terrain analysis in mobile robot applications. In *Proceedings of the IEEE International Conference on Mechatronics (ICM)*, pages 225–230.

[Bellone et al., 2013b] Bellone, M., Reina, G., Giannoccaro, N. I., and Spedicato, L. (2013b). Unevenness point descriptor for terrain analysis in mobile robot applications. *International Journal of Advanced Robotic Systems*, 10.

[Bergerman et al., 2012] Bergerman, M., Singh, S., and Hamner, B. (2012). Results with autonomous vehicles operating in specialty crops. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1829–1835.

[Bernini et al., 2014] Bernini, N., Bertozzi, M., Castangia, L., Patander, M., and Sabbatelli, M. (2014). Real-time obstacle detection using stereo vision for autonomous ground vehicles: A survey. In *Proceedings of the IEEE 17th International Conference on Intelligent Transportation Systems (ITSC)*, pages 873–878.

[Bosch et al., 2007] Bosch, A., Muñoz, X., and Freixenet, J. (2007). Segmentation and description of natural outdoor scenes. *Image and Vision Computing*, 25(5):727 – 740.

[Boykov et al., 2001] Boykov, Y., Veksler, O., and Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239.

[Byun et al., 2015] Byun, J., Na, K.-i., Seo, B.-s., and Roh, M. (2015). Drivable road detection with 3D point clouds based on the MRF for intelligent vehicle. In *Proceedings of the 9th International Conference on Field and Service Robotics (FSR)*, pages 49–60.

[Clevert et al., 2015] Clevert, D.-A., Unterthiner, T., and Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (ELUs). *arXiv preprint arXiv:1511.07289*.

[Geiger et al., 2014] Geiger, A., Lauer, M., Wojek, C., Stiller, C., and Urtasun, R. (2014). 3D traffic scene understanding from movable platforms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(5):1012–1025.

[Glorot and Bengio, 2010] Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference Artificial Intelligence and Statistics (AISTATS)*, pages 249–256.

[Godoy et al., 2012] Godoy, E. P., Tangerino, G. T., Tabile, R. A., Inamasu, R. Y., and Porto, A. J. V. (2012). Networked control system for the guidance of a four-wheel steering agricultural robotic platform. *Journal of Control Science and Engineering*.

[Guijarro et al., 2011] Guijarro, M., Pajares, G., Riomoros, I., Herrera, P., Burgos-Artizzu, X., and Ribeiro, A. (2011). Automatic segmentation of relevant textures in agricultural images. *Computers and Electronics in Agriculture*, 75(1):75–83.

[Hadsell et al., 2007] Hadsell, R., Erkan, A., Sermanet, P., Ben, J., Kavukcuoglu, K., Muller, U., and LeCun, Y. (2007). A multi-range vision strategy for autonomous offroad navigation. In *Proceedings of the 13th IASTED International Conference on Robotics and Applications (RA)*.

[Hadsell et al., 2009] Hadsell, R., Sermanet, P., Ben, J., Erkan, A., Scoffier, M., Kavukcuoglu, K., Muller, U., and LeCun, Y. (2009). Learning long-range vision for autonomous off-road driving. *J. Field Robot.*, 26(2):120–144.

[Houthooft et al., 2016] Houthooft, R., De Boom, C., Verstichel, S., Ongenae, F., and De Turck, F. (2016). Structured output prediction for semantic perception in autonomous vehicles. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI)*.

[Ioffe and Szegedy, 2015] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 448–456.

[Joachims et al., 2009] Joachims, T., Finley, T., and Yu, C.-N. J. (2009). Cutting-plane training of structural SVMs. *Machine Learning*, 77(1):27–59.

[Kayacan et al., 2015] Kayacan, E., Kayacan, E., Ramon, H., and Saeys, W. (2015). Towards agrobots: Identification of the yaw dynamics and trajectory tracking of an autonomous tractor. *Computers and Electronics in Agriculture*, 115:78–87.

[Kelly et al., 2006] Kelly, A., Stentz, A., Amidi, O., Bode, M., Bradley, D., Diaz-Calderon, A., Happold, M., Herman, H., Mandelbaum, R., Pilarski, T., et al. (2006). Toward reliable off road autonomous vehicles operating in challenging environments. *International Journal of Robotics Research*, 25(5-6):449–483.

[Kingma and Ba, 2015] Kingma, D. and Ba, J. (2015). Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

[Kraus et al., 2013] Kraus, T., Ferreau, H., Kayacan, E., Ramon, H., Baerdemaeker, J. D., Diehl, M., and Saeys, W. (2013). Moving horizon estimation and nonlinear model predictive control for autonomous agricultural vehicles. *Computers and Electronics in Agriculture*, 98:25–33.

[Kuthirummal et al., 2011] Kuthirummal, S., Das, A., and Samarasekera, S. (2011). A graph traversal based algorithm for obstacle detection using lidar or stereo. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3874–3880.

[Leonard et al., 2008] Leonard, J., How, J., Teller, S., Berger, M., Campbell, S., Fiore, G., Fletcher, L., Frazzoli, E., Huang, A., Karaman, S., et al. (2008). A perception-driven autonomous urban vehicle. *Journal of Field Robotics*, 25(10):727–774.

[Long et al., 2015] Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440.

[Lucchi et al., 2012] Lucchi, A., Li, Y., Smith, K., and Fua, P. (2012). Structured image segmentation using kernelized features. In *Proceedings of the 12th European Conference on Computer Vision (ECCV)*, pages 400–413.

[Milella and Reina, 2014] Milella, A. and Reina, G. (2014). 3D reconstruction and classification of natural environments by an autonomous vehicle using multi-baseline stereo. *Intelligent Service Robotics*, 7(2):79–92.

[Milella et al., 2013] Milella, A., Reina, G., and Foglia, M. M. (2013). A multi-baseline stereo system for scene segmentation in natural environments. In *Proceedings of the IEEE International Conference on Technologies for Practical Robot Applications (TePRA)*.

[Milella et al., 2011] Milella, A., Reina, G., Underwood, J., and Douillard, B. (2011). Combining radar and vision for self-supervised ground segmentation in outdoor environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 255–260.

[Milella et al., 2014] Milella, A., Reina, G., Underwood, J., and Douillard, B. (2014). Visual ground segmentation by radar supervision. *Robotics and Autonomous Systems*, 62(5):696–706.

[Moreno et al., 2014] Moreno, F.-A., Cielniak, G., and Duckett, T. (2014). Evaluation of laser range-finder mapping for agricultural spraying vehicles. In *Towards Autonomous Robotic Systems: 14th Annual Conference*, pages 210–221.

[Mostajabi et al., 2015] Mostajabi, M., Yadollahpour, P., and Shakhnarovich, G. (2015). Feedforward semantic segmentation with zoom-out features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3376–3385.

[Nourani-Vatani et al., 2015] Nourani-Vatani, N., López-Sastre, R., and Williams, S. (2015). Structured output prediction with hierarchical loss functions for seafloor imagery taxonomic categorization. In *Proceddings of the 7th Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA)*, pages 173–183.

[Nowozin and Lampert, 2011] Nowozin, S. and Lampert, C. H. (2011). Structured learning and prediction in computer vision. *Foundations and Trends in Computer Graphics and Vision*, 6(3–4):185–365.

[Oksanen, 2015] Oksanen, T. (2015). Accuracy and performance experiences of four wheel steered autonomous agricultural tractor in sowing operation. In *Proceedings of the 9th International Conference on Field and Service Robotics (FSR)*, pages 425–438.

[Oksanen and Linkolehto, 2013] Oksanen, T. and Linkolehto, R. (2013). Control of four wheel steering using independent actuators. In *Proceedings of the 4th IFAC Conference on Modelling and Control in Agriculture, Horticulture and Post Harvest Industry*, pages 159–163.

[Reina and Milella, 2012] Reina, G. and Milella, A. (2012). Towards autonomous agriculture: automatic ground detection using trinocular stereovision. *Sensors*, 12(9):12405–12423.

[Reina et al., 2012] Reina, G., Milella, A., and Underwood, J. (2012). Self-learning classification of radar features for scene understanding. *Robotics and Autonomous Systems*, 60(11):1377–1388.

[Ronneberger et al., 2015] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *Proceedings of the 18th International Conference on Medical*

Image Computing and Computer-Assisted Intervention (MICCAI), pages 234–241.

[Rouveure et al., 2012] Rouveure, R., Nielsen, M., Petersen, A., Reina, G., Foglia, M., Worst, R., Seyed-Sadri, S., Blas, M., Faure, P., Milella, A., et al. (2012). The QUAD-AV project: Multi-sensory approach for obstacle detection in agricultural autonomous robotics. In *Proceedings of the International Conference of Agricultural Engineering (CIGR-AgEng)*.

[Rovira-Más et al., 2008] Rovira-Más, F., Zhang, Q., and Reid, J. F. (2008). Stereo vision three-dimensional terrain maps for precision agriculture. *Computers and Electronics in Agriculture*, 60(2):133–143.

[Russakovsky et al., 2015] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252.

[Sermanet et al., 2013] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. (2013). OverFeat: Integrated recognition, localization and detection using convolutional networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

[Sermanet et al., 2009] Sermanet, P., Hadsell, R., Scoffier, M., Grimes, M., Ben, J., Erkan, A., Crudele, C., Miller, U., and LeCun, Y. (2009). A multirange architecture for collision-free off-road robot navigation. *Journal of Field Robotics*, 26(1):52–87.

[Sivaraman and Trivedi, 2013] Sivaraman, S. and Trivedi, M. M. (2013). Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis. *IEEE Transactions on Intelligent Transportation Systems*, 14(4):1773–1795.

[Srivastava et al., 2014] Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

[Subramanian et al., 2006] Subramanian, V., Burks, T. F., and Arroyo, A. (2006). Development of machine vision and laser radar based autonomous vehicle guidance systems for citrus grove navigation. *Computers and Electronics in Agriculture*, 53(2):130–143.

[Taskar et al., 2003] Taskar, B., Guestrin, C., and Koller, D. (2003). Max-margin markov networks. In *Advances in Neural Information Processing Systems 16 (NIPS)*, pages 25–32.

[Tola et al., 2010] Tola, E., Lepetit, V., and Fua, P. (2010). DAISY: An efficient dense descriptor applied to wide baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(5):815–830.

[Weiss and Biber, 2010] Weiss, U. and Biber, P. (2010). Semantic place classification and mapping for autonomous agricultural robots. In *Proceedings of IROS Workshop on Semantic Mapping and Autonomous Knowledge Acquisition*.

[Weiss and Biber, 2011] Weiss, U. and Biber, P. (2011). Plant detection and mapping for agricultural robots using a 3D LIDAR sensor. *Robotics and autonomous systems*, 59(5):265–273.

[Weiss et al., 2010] Weiss, U., Biber, P., Laible, S., Bohlmann, K., and Zell, A. (2010). Plant species classification using a 3D lidar sensor and machine learning. In *Proceedings of the 9th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 339–345.

[Xue et al., 2012] Xue, J., Zhang, L., and Grift, T. E. (2012). Variable field-of-view machine vision based row guidance of an agricultural robot. *Computers and Electronics in Agriculture*, 84:85–91.

Chapter 3

Integrated Inference and Learning of Neural Factors in Structural Support Vector Machines

Rein Houthooft & Filip De Turck, *Integrated Inference and Learning of Neural Factors in Structural Support Vector Machines*, Pattern Recognition, 59, pp. 292–301, 2016.

Includes changes and extensions not released at the time of publication.

Ghent University – imec, CNH Industrial

⋆ ⋆ ⋆

Tackling pattern recognition problems in areas such as computer vision, bioinformatics, speech or text recognition is often done best by taking into account task-specific statistical relations between output variables. In structured prediction, this internal structure is used to predict multiple outputs simultaneously, leading to more accurate and coherent predictions. Structural support vector machines (SSVMs) are nonprobabilistic models *that optimize a joint input-output function through margin-based learning. Because SSVMs generally disregard the interplay between unary and interaction factors during the training phase, final parameters are suboptimal. Moreover, its factors are often restricted to linear combinations of input features, limiting its generalization power. To improve prediction accuracy, this chapter proposes: (i) Joint inference and learning by integration of back-propagation and loss-augmented inference in SSVM subgradient descent; (ii) Extending SSVM factors to neural networks that form highly nonlinear functions of input features. Image segmentation benchmark results demonstrate improvements over conventional SSVM training methods in terms of accuracy, highlighting the feasibility of end-to-end SSVM training with neural factors.*

3.1 Introduction

In traditional machine learning, the output consists of a single scalar, whereas in structured prediction, the output can be arbitrarily structured. These models have proven useful in tasks where output interactions play an important role. Examples are image segmentation, part-of-speech tagging, and optical character recognition, where taking into account contextual cues and predicting all output variables at once is beneficial. A widely used framework is the conditional random field (CRF), which models the statistical conditional dependencies between input and output variables, as well as between output variables mutually. However, many tasks only require 'most-likely' predictions, which led to the rise of nonprobabilistic approaches. Rather than optimizing the Bayes' risk, these models minimize a structured loss, allowing the optimization of performance indicators directly [Nowozin and Lampert, 2011]. One such model is the structural support vector machine (SSVM) [Tsochantaridis et al., 2005] in which a generalization of the hinge loss to multiclass and multilabel prediction is used.

A downside to traditional SSVM training is the bifurcated training approach in which *unary factors* (dependencies of outputs on inputs), and *interaction factors* (mutual output dependencies) are trained sequentially. A unary classification model is optimized, while the interactions are trained post-hoc. However, this two-phase approach is suboptimal, because the errors made during the training of the interaction factors cannot be accounted for during training of the unary classifier. Another limitation is that SSVM factors are linear feature combinations, restricting the SSVM's generalization power. We propose to extend these linearities to highly nonlinear functions by means of multilayer neural networks, to which we refer as *neural factors*. Towards this goal, subgradient descent is extended by combining loss-augmented inference with back-propagation of the SSVM objective error into both unary and interaction neural factors. This leads to better generalization and more synergy between both SSVM factor types, resulting in more accurate and coherent predictions.

Our model is empirically validated by means of the complex structured prediction task of image segmentation on the MSRC-21, KITTI, and SIFT Flow benchmarks. The results demonstrate that integrated inference and learning, and/or using neural factors, improves prediction accuracy over conventional SSVM training methods, such as N-slack cutting plane and subgradient descent optimization [Nowozin and Lampert, 2011]. Furthermore, we demonstrate that our model is able to perform on par with current state-of-the-art segmentation models on the MSRC-21 benchmark.

3.2 Related Work

Although the combination of neural networks and structured or probabilistic graphical models dates back to the early '90s [Bottou et al., 1997, Bridle, 1989], interest in this topic is resurging. Several recent works introduce nonlinear unary factors/potentials into structured models. For the task of image segmentation, Chen et al. [Chen et al., 2015a] train a convolutional neural network as a unary classifier, followed by the training of a dense random field over the input pixels. Similarly, Farabet et al. [Farabet et al., 2013] combine the output maps of a convolutional network with a CRF for image segmentation, while Li and Zemel [Li and Zemel, 2014] propose semisupervised maxmargin learning with nonlinear unary potentials. Contrary to these works, we trade the bifurcated training approach for integrated inference and training of unary and interactions factors. Several works [Collobert et al., 2011, Morris and Fosler-Lussier, 2008, Prabhavalkar and Fosler-Lussier, 2010, Yu et al., 2009] focus on linear-chain graphs, using an independently trained deep learning model whose output serves as unary input features. Contrary to these works, we focus on more general graphs. Other works suggest kernels towards nonlinear SSVMs [Lucchi et al., 2012, Bertelli et al., 2011]; we approach nonlinearity by representing SSVM factors by arbitrarily deep neural networks.

Do and Artières [Do and Artières, 2010] propose a CRF in which potentials are represented by multilayer networks. The performance of their linear-chain probabilistic model is demonstrated by optical character and speech recognition using two-hidden-layer neural network outputs as unary potentials. Furthermore, joint inference and learning in linear-chain models is also proposed by Peng et al. [Peng et al., 2009], however, the application to more general graphs remains an open problem [Müller, 2014]. Contrary to these works, we popose a nonprobabilistic approach for general graphs by also modeling nonlinear interaction factors. More recently, Schwing and Urtasun [Schwing and Urtasun, 2015] train a convolutional network as a unary classifier jointly with a fully-connected CRF for the task of image segmentation, similar to [Tompson et al., 2014, Krähenbühl and Koltun, 2013]. Chen et al. [Chen et al., 2015b] advocate a joint learning and reasoning approach, in which a structured model is probabilistically trained using loopy belief propagation for the task of optical character recognition and image tagging. Other related work includes Domke [Domke, 2013] who uses relaxations for combined message-passing and learning.

Other related work aiming to improve conventional SSVMs are the works of Wang et al. [Wang et al., 2013] and Lin et al. [Lin et al., 2015], in which a hierarchical part-based model is proposed for multiclass object recognition and shape detection, focusing on model reconfigurability through compositional alternatives in And-Or graphs. Liang et al. [Liang et al., 2015] propose the use of convolutional neural networks to model an end-to-end relation between input images and structured outputs in active template regression. Xu et al. [Xu et al., 2014] propose the learning of a structured model with multilayer deformable parts for action understanding, while Lu et al. [Lu et al., 2015] propose a hierarchical structured model for action segmentation.

Many of these works use probabilistic models that maximize the negative log-likelihood, such as [Do and Artières, 2010, Peng et al., 2009]. In contrast, this chapter takes a nonprobabilistic approach, wherein an SSVM is optimized via subgradient descent. The algorithm is altered to backpropagate SSVM loss errors, based on the ground truth and a loss-augmented prediction into the factors. Moreover, all factors are nonlinear functions, allowing the learning of complex interaction patterns.

3.3 Methodology

In this section integrated inference and back-propagation is explained for nonlinear unary factors. Finally, this notion is generalized into an SSVM model using only neural factors which are optimized by an alteration of subgradient descent.

As explained in Section 2.4.1, structured prediction is performed as

$$f(x) = \operatorname*{arg\,max}_{y \in \mathcal{Y}} g(x, y; w). \tag{3.1}$$

This is called inference, i.e., obtaining the most-likely assignment of labels, which is similar to maximum-a-posteriori (MAP) inference in probabilistic models. Because of the combinatorial complexity of the output space $\mathcal{Y}$, the maximization problem in Eq. (3.1) is NP-hard [Chen et al., 2015b]. Hence, it is important to impose on g some kind of regularity that can be exploited for inference. This can be done by ensuring that g corresponds to a nonprobabilistic factor graph, for which efficient inference techniques exist [Nowozin and Lampert, 2011]. In Chapter 2, g is linearly parametrized as a product of a weight vector w and a joint feature function $\varphi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^D$.

Commonly, g decomposes as a sum of unary and interaction factors[1], in which $\varphi = \left[(\varphi_U)^\top, (\varphi_I)^\top\right]^\top$. The functions φ_U and φ_I are then sums over all individual joint input-output features of the nodes $\psi_i(y, x)$ and interactions $\psi_{ij}(y, x)$ of the corresponding factor graph [Nowozin and Lampert, 2011, Lucchi et al., 2012]. For example in the use case of Section 5.3, nodes are image regions, while interactions are connections between regions, each with their own joint feature vector. Data samples (x, y) are conform this graphical structure, i.e., x is composed of unary features x^U and interaction features x^I. Moreover, the unary and interaction parameters are generally concatenated as $w = [(w_U)^\top, (w_I)^\top]^\top$.

In this formulation, the unary features are defined as

$$\psi_i(y_i, x_i) = \left(\epsilon_i(x)^\top [y_i = m]\right)^\top_{(m \in \mathcal{L})}, \tag{3.2}$$

while the interaction features for 2nd-order (edges) interactions are defined as

$$\psi_{ij}(y_i, y_j) = \left(\xi_{ij}(x)[y_i = m \wedge y_j = n]\right)^\top_{((m,n) \in \mathcal{L}^2)}, \tag{3.3}$$

[1]Maximizing g corresponds to minimizing the state of a nonprobabilistic factor graph, which factorizes into a product of factors. However, by operating in the log-domain, the state decomposes as a sum of factors.

with $\epsilon_i(x)$ the unary features corresponding to node i and $\xi_{ij}(x)$ the interaction features corresponding to interaction (edge) (i, j). Similarly, higher-order interaction features can be incorporated by extending this matrix into higher-order combinations of nodes, according to the interactions. Rather than optimizing an empirical estimate of the structured risk in Eq (2.11), a continuous and convex upper bound is defined (cfr. Eqs. (2.12) and (2.13)):

$$L(w) = \frac{1}{2}\|w\|^2 + \frac{\lambda}{N}\sum_{n=1}^{N} \max\left\{\ell(x^n, y^n; w), 0\right\}, \text{ with} \tag{3.4}$$

$$\ell(x^n, y^n; w) = \max_{y\in\mathcal{Y}} \left[\Delta(y^n, y) - g(x^n, y^n; w) + g(x^n, y; w)\right], \tag{3.5}$$

which can be minimized effectively by solving $\arg\min_{w\in\mathbb{R}^D} L(w)$ through numerical optimization [Zhang, 2004].

3.3.1 Integrated Back-propagation and Inference

Traditional SSVM training methods optimize a joint parameter vector of the unary and interaction factors. However, they restrict these parameters to linear combinations of input features, or allow limited nonlinearity through the addition of kernels. The objective function in case of arbitrary non-linear factors is often hard to optimize, as many numerical optimization methods require a convex objective function formulation. For example, N-slack cutting plane training requires the conversion of the max-operation in Eq. (3.5) to a set of $N|\mathcal{Y}|$ linear constraints for its quadratic programming procedure [Joachims et al., 2009]; block-coordinate Frank-Wolfe SSVM optimization [Lacoste-julien et al., 2013] assumes linear input dependencies; the structured perceptron similarly assumes linear parametrization [Collins, 2002]; and dual coordinate descent focuses on solving the dual of the linear L_2-loss in SSVMs [Chang and Yih, 2013].

Subgradient descent minimization, as described in [Nowozin and Lampert, 2011, Shor et al., 1985], is a flexible tool for optimizing Eq. (3.4) as it naturally allows error back-propagation. This algorithm alternates between two steps. First,

$$z^n = \arg\max_{y\in\mathcal{Y}}[\Delta(y^n, y) + \langle w, \varphi(x^n, y)\rangle] \tag{3.6}$$

is calculated for all N training samples, which is called the loss-augmented inference or prediction step, derived from Eq. (3.5). In this chapter, general

Algorithm 3.1: Integrated SSVM subgradient descent with neural unary and linear interaction factors

```
Input: number of iterations $T$; learning rate curve $\mu/(t_0 + t)$; inverse
       regularization strength $\lambda$; training samples
       $\{(x^n, y^n)\}_{n \in \{1,\dots,N\}}$
Output: optimized parameters $\theta \in \mathbb{R}^K$ and $w \in \mathbb{R}^L$
1  Initialize $w$ to $\vec{0}$ and $\theta$ according to [Glorot and Bengio, 2010]; the
     output layer weights are initialized to 0.
2  for $1 \le t \le T$ do
3      for $1 \le n \le N$ do
4          $z^n \leftarrow \arg\max_{y \in \mathcal{Y}} [\Delta(y^n, y) + \langle w, \varphi_I(x^n, y) \rangle + f(x^n, y; \theta)]$, as
             loss-augmented prediction in Eq. (3.6).
5          if $\Delta(y^n, y) - g(x^n, y^n; \theta, w) + g(x^n, z^n; \theta, w) > 0$, according
             to max-operation in Eq. (3.4) then
6              Standard SSVM subgradient computation as defined
                 in [Nowozin and Lampert, 2011],
                 $\frac{\partial L_n}{\partial w}(\theta, w) \leftarrow w + \lambda(\varphi_I(x^n, z^n) - \varphi_I(x^n, y^n))$
7              Gradient computation as in Eq. (3.10),
                 $\nabla_\theta L_n(\theta, w) \leftarrow \theta + \lambda(\nabla_\theta f(x^n, z^n; \theta) - \nabla_\theta f(x^n, y^n; \theta))$
8          else
9              $\nabla_\theta L_n(\theta, w) \leftarrow \theta$  and  $\frac{\partial L_n}{\partial w}(\theta, w) \leftarrow w$
10         end
11     end
12     Update linear interaction factors as
           $$w \leftarrow w - \frac{\mu}{t_0 + t} \frac{1}{N} \sum_{n=1}^{N} \frac{\partial L_n}{\partial w}(\theta, w)$$
13     Update neural unary factors via back-propagation as
           $$\theta \leftarrow \texttt{backprop}\left( \frac{1}{N} \sum_{n=1}^{N} \nabla_\theta L_n(\theta, w) \right)$$
14 end
```

inference for determining Eq. (3.1) is approximated via the α-expansion [Boykov et al., 2001] algorithm (also see Section 2.4.3), whose effectiveness has been validated through extensive experiments [Peng et al., 2013]. Loss-augmented prediction as in Eq. (3.6) is incorporated into this procedure by adding the loss term $\eta(y_i^n)[y_i^n \neq y_i]$ to the unary factors.

Second, these z-values are used to calculate a subgradient[2] of Eq. (3.4) as

$$\frac{1}{N}[w + \lambda\left(\varphi(x^n, z^n) - \varphi(x^n, y^n)\right)] \tag{3.7}$$

for each sample (x^n, y^n), in order to update w. Traditional SSVMs assume that $g(x, y; w) = \langle w, \varphi(x, y)\rangle$ in which φ is a predefined joint input-output feature function. Commonly, this joint function is made up of the outputs of a nonlinear 'unary' classifier $C : \mathcal{X} \to [0, 1]^{|\mathcal{L}|}$, such that $\varphi_U(x, y)$ becomes $\varphi_U(C(x), y)$ [Houthooft et al., 2016]. This classifier is trained upfront, based on the different unary inputs corresponding to each node in the underlying factor graph. Due to the linear definition of g, the SSVM model is learning linear combinations of these classifier outputs as its unary factors. In general, the interaction factors are not trained through a separate classifier, and are thus linear combinations of the interaction features directly.

We propose to replace the pretraining of a nonlinear unary classifier, and the transformation of its outputs through linear factors, by the direct optimization of nonlinear unary factors. In particular, the unary part of g is represented by a sum f of outputs of an adapted neural network which models factor values. To achieve this, the loss-augmented prediction step defined in Eq. (3.6) is altered to

$$z^n = \arg\max_{y \in \mathcal{Y}}[\Delta(y^n, y) + \langle w, \varphi_I(x^n, y)\rangle + f(x^n, y; \theta)], \tag{3.8}$$

in which φ_I represents the joint interaction feature function as described at the beginning Section 4.2 and Eq. (3.3). Eq. (3.8) is calculated similarly to Eq. (3.6) through α-expansion by encoding the loss term into the unary factors.

The compatibility function thus becomes

$$g(x, y; \theta, w) = \langle w, \varphi_I(x, y)\rangle + f(x, y; \theta). \tag{3.9}$$

[2] $v \in \mathbb{R}^D$ is a subgradient of $f : \mathbb{R}^D \to \mathbb{R}$ in a point p_0 if $f(p) - f(p_0) \geq \langle v, p - p_0\rangle$. Due to its piecewise continuous nature, Eq. (3.4) is nondifferentiable in some points, hence we are forced to rely on subgradients.

The calculation of $\frac{\partial L}{\partial w}$, originally defined as the subderivative of the objective function in Eq. (3.4), remains unaltered. However, we can no longer assume that $\frac{\partial L}{\partial \theta}$ conforms to the definition of a subgradient due to its nonconvexity. However, we can calculate

$$\nabla_\theta L(\theta, w) = \theta + \frac{\lambda}{N} \sum_{n \in \mathcal{N}} \left(\nabla_\theta f\left(x^n, z^n; \theta\right) - \nabla_\theta f\left(x^n, y^n; \theta\right) \right), \quad (3.10)$$

with $\mathcal{N}$ the set of indices corresponding to training samples for which $\ell(x^n, y^n; w) > 0$ in Eq. (3.5), for a particular loss-augmented prediction z^n. In case $\ell(x^n, y^n; w) = 0$, we set $\nabla_\theta L = \theta$. This gradient incorporates the loss-augmented prediction of Eq. (3.8) and is back-propagated through the underlying network to adjust each element of θ. The altered subgradient descent method is shown in Algorithm 3.1. Herein, L_n represents the objective function for the n-th training sample, i.e., $L_n(\theta, w) = \frac{1}{N}[\frac{1}{2}\|w\|^2 + \lambda(\Delta(y^n, z^n) - g(x^n, y^n; \theta, w) + g(x^n, z^n; \theta, w))]$.

In contrast to gradient descent, subgradient methods [Nowozin and Lampert, 2011, Shor et al., 1985] do not guarantee the lowering of the objective function value in each step. Therefore, the current best value $L_*^{(t)} = \min\{L_*^{(t-1)}, L(w^{(t)})\}$ is memorized in each iteration t, along with the corresponding parameter values (w_*, θ_*). As such, the objective value L_* decreases at each step as $L_*^{(t)} = \min\{L(w^{(1)}), \ldots, L(w^{(t)})\}$. This update rule is omitted from Algorithm 3.1 to improve readability.

Because the loss terms in Eq. (3.5) are no longer affine input transformations due to the introduced nonlinearities of the neural network, we can no longer assume Eq. (3.4) to be convex, as is the case for conventional SSVMs. Although theoretical guarantees can be made for the convergence of (sub)gradient methods for convex functions [Nedić and Bertsekas, 2001], and particular classes of nonconvex functions [Bagirov et al., 2013], no such guarantees can be made for arbitrary nonconvex functions [Ngiam et al., 2011]. The problem of optimizing highly nonconvex functions is studied extensively in neural network gradient descent literature. However, it has been demonstrated that nonconvex objectives can be minimized effectively due to the high dimensionality of the neural network parameter space [Pascanu et al., 2014]. Dauphin et al. [Dauphin et al., 2014] show that saddle points are much likelier than local minima in multilayer neural network objective landscapes. In particular, the ratio of saddle points to local minima increases exponentially with the parameter dimensionality. Several methods exists to avoid these these saddle points, e.g., momentum [Sutskever et al., 2013]. Furthermore, Dauphin et al. [Dauphin et al., 2014] show, based on random matrix theory, that the existing local minima are very close to the global minimum of the objective function. This can be understood intuitively as the probability that all directions surrounding a local minimum lead upwards is very small, making local minima not an issue in general. The empirical results presented in Section 3.4.2 reinforce this believe by demonstrating that the regularized objective function can still be minimized effectively, as we achieve accurate predictions.

As described in Algorithm 3.1, the (sub)gradient is defined over whole data samples, which each consist of multiple nodes. f thus models the unary part of the compatibility function g, which is a sum of the V_n unary factors. Therefore, the function $f(x, y; \theta)$ decomposes as a sum of *neural unary factors*

$$f(x, y; \theta) = \sum_{i=1}^{V_n} f^*\left(x_i^U; \theta\right)_{y_i}, \tag{3.11}$$

with x^U the unary features in x. The nonlinear function $f^* : \mathcal{X} \to \mathbb{R}^{|\mathcal{L}|}$ is a multiclass multilayer neural network parametrized by $\theta \in \mathbb{R}^K$, whose inputs are features corresponding to the V_n different nodes. It forms a template for the neural unary factors. In this network $f^*(x_i^U; \theta)$, the softmax-function is removed from the output layer, such that it matches the unary factor range $\mathbb{R}^{|\mathcal{L}|}$. The argument y of the joint feature function is used as an index y_i to select a particular output unit.

3.3.2 Neural Interaction Factors

In this section we extend the notion of nonlinear factors beyond the integration of the training of a unary classifier. We now also replace the linear interaction part $\langle w, \varphi_I(x, y) \rangle$ of the compatibility function g with a function $h(x, y; \gamma)$ that decomposes as a sum of *neural interaction factors*

$$h(x, y; \gamma) = \sum_{i=1}^{E_n} h^*\left(x_i^I; \gamma\right)_{\mathcal{N}_i(y)}, \tag{3.12}$$

with x^I the interaction features in x, $\mathcal{N}_i(y)$ the combination of node labels in the i-th interaction, and E_n the number of interactions in the n-th training

Algorithm 3.2: Integrated SSVM subgradient descent with both unary and interaction neural factors

```
Input: number of iterations $T$; learning rate; inverse regularization
       strength $\lambda$; training set $\{(x^n, y^n)\}$
Output: optimized parameters $\theta \in \mathbb{R}^K$ and $\gamma \in \mathbb{R}^M$
1  Initialize $\theta$ and $\gamma$ according to [Glorot and Bengio, 2010]; the
     weights of the output layers are initialized to 0.
2  for $1 \leq t \leq T$ do
3      for $1 \leq n \leq N$ do
4          $z^n \leftarrow \arg\max_{y \in \mathcal{Y}} [\Delta(y^n, y) + f(x^n, y; \theta) + h(x^n, y; \gamma)]$
5          if $\Delta(y^n, y) - g(x^n, y^n; \theta, \gamma) + g(x^n, z^n; \theta, \gamma) > 0$ then
6              $\nabla_\theta L_n(\theta, \gamma) \leftarrow \theta + \lambda\Big(\nabla_\theta f(x^n, z^n; \theta) - \nabla_\theta f(x^n, y^n; \theta)\Big)$
               $\nabla_\gamma L_n(\theta, \gamma) \leftarrow \gamma + \lambda\Big(\nabla_\gamma h(x^n, z^n; \gamma) - \nabla_\gamma h(x^n, y^n; \gamma)\Big)$
7          else
8              $\nabla_\theta L_n(\theta, \gamma) \leftarrow \theta$ and $\nabla_\gamma L_n(\theta, \gamma) \leftarrow \theta$
9          end
10     end
11     $\theta \leftarrow \texttt{backprop}\left(\frac{1}{N}\sum_{n=1}^{N} \nabla_\theta L_n(\theta, \gamma)\right)$ and
                                 $\gamma \leftarrow \texttt{backprop}\left(\frac{1}{N}\sum_{n=1}^{N} \nabla_\gamma L_n(\theta, \gamma)\right)$
12 end
```

sample. The function $h^* : \mathcal{X} \rightarrow \mathbb{R}^{|\mathcal{L}|^Q}$ is parametrized by $\gamma \in \mathbb{R}^M$, and forms a template for the interaction factors. Herein, Q depends on the interaction order, e.g., $Q = 2$ in the Section 5.3 use case as connections between nodes are then edges. Interaction factors are generally not trained upfront. However, neural interaction factors are useful as they can extract complexer interaction patterns, and thus transcend the limited generalization power of linear combinations. In image segmentation for example, interaction features consisting of vertical gradients and a 90°-angle can indicate that the two connected nodes belong to the same class. The loss-augmented inference step in Eq. (3.8) is now adapted to

$$z^n = \arg\max_{y \in \mathcal{Y}} [\Delta(y^n, y) + f(x^n, y; \theta) + h(x^n, y; \gamma)], \tag{3.13}$$

while the compatibility function becomes

$$g(x, y; \theta, \gamma) = f(x, y; \theta) + h(x, y; \gamma). \tag{3.14}$$

The two distinct models f and h are trained in a similar fashion to the method described in Algorithm 3.1, as depicted in Algorithm 3.2. Notice that this method can easily be adjusted for batch or online learning by adapting and moving the weight updates at line 11 into the inner loop.

Like the unary function f^* in Eq. (3.11), $h^*\left(x_i^I; \gamma\right)$ is a multiclass multilayer neural network in which the top softmax-function is removed, shared among all E_n interaction factors. The output layer dimension matches the number of interaction label combinations, $|\mathcal{L}|^Q$ in the most general case. For example in image segmentation, for a problem with symmetric edge features, the number of output units in h^* is $\frac{1}{2}|\mathcal{L}|(|\mathcal{L}| + 1)$, which all represent different states for a particular interaction factor (in this case the interactions are undirected edges, thus $\mathcal{N}_i(y)$ consists of the i-th edge's incident nodes).

The resulting structured predictor no longer requires two-phase training in which linear interaction factors are combined with the upfront training of a unary classifier, whose output is transformed linearly into unary factor values. It makes use of highly nonlinear functions for all SSVM factors, by way of multilayer neural networks, using an integration of loss-augmented inference and back-propagation in a subgradient descent framework. This allows the factors to generalize strongly while being able to mutually adapt to each other's parameter updates, leading to more accurate predictions.

3.4 Experiments

In this section, our model is analyzed on the task of image segmentation. Herein, the goal is to label different image regions with a correct class label. This is cast into a structured prediction problem by predicting all image region class labels simultaneously. There is one unary factor in underlying SSVM graphical structure for every image region, while interactions represent edges between neighboring regions. First, our model is analyzed and its different variants are compared to conventional SSVM training schemes. Second, the best performing variant is compared with state-of-the-art segmentation approaches. Our model is implemented as an extension of PyStruct [Müller and Behnke, 2014], using Theano [Bastien et al., 2012] for GPU-accelerated neural factor optimization.

3.4.1 Experimental setup

The model analysis experiments are executed on the widely-used MSRC-21 benchmark [Shotton et al., 2009], which consists of 276 training, 59 validation, and 256 testing images. This benchmark is sufficiently complex with its 21 classes and noisy labels, and focuses on object delineation as well as irregular background recognition. Furthermore, the experiments are executed on the KITTI benchmark [Ros et al., 2015] consisting of 100 training and 46 testing images, augmented with 49 training images of Kundu et al. [Kundu et al., 2014]. This latter benchmark consists of 11 classes, but we drop the 3 least frequently-occurring ones as they are insufficiently represented in the dataset. Finally, the same experiment is repeated for a larger dataset, namely the SIFT Flow benchmark [Liu et al., 2011], consisting of 33 classes with 2488 training and 200 testing images.

All image pixels are clustered into ±300 regions using the SLIC [Achanta et al., 2012] superpixel algorithm. For each region, gradient (DAISY [Tola et al., 2010]) and color (in HSV-space) features are densely extracted. These features are transformed two times into separate bags-of-words via minibatch k-means clustering (once 60 gradient and 30 color words, once 10 and 5 words). The unary input vectors form (60 + 30)-dim concatenations of the first two bags-of-words. The model's connectivity structure links together all neighboring regions via edges. The edge/interaction input vectors are based on concatenations of the second set of bags-of-words. Both (10 + 5)-dim input vectors of the edge's incident regions are concatenated into a $(2 \times (10 + 5))$-dim vector. Moreover, two edge-specific features are added, namely the distance and angle between adjacent superpixel centers, leading to $(2 \times (10 + 5) + 2)$-dim interaction feature vectors.

Factors are trained with (regular) momentum, using a learning rate curve $\frac{\mu}{t_0+t}$, with μ and t_0 parameters, and t the current training iteration number as used in Algorithms 3.1 and 3.2. The regularization, learning rate, and momentum hyperparameter values are tuned using a validation set by means of a coarse- and fine-grained grid search over the parameter spaces, yielding separate settings for the unary and pairwise factors. The

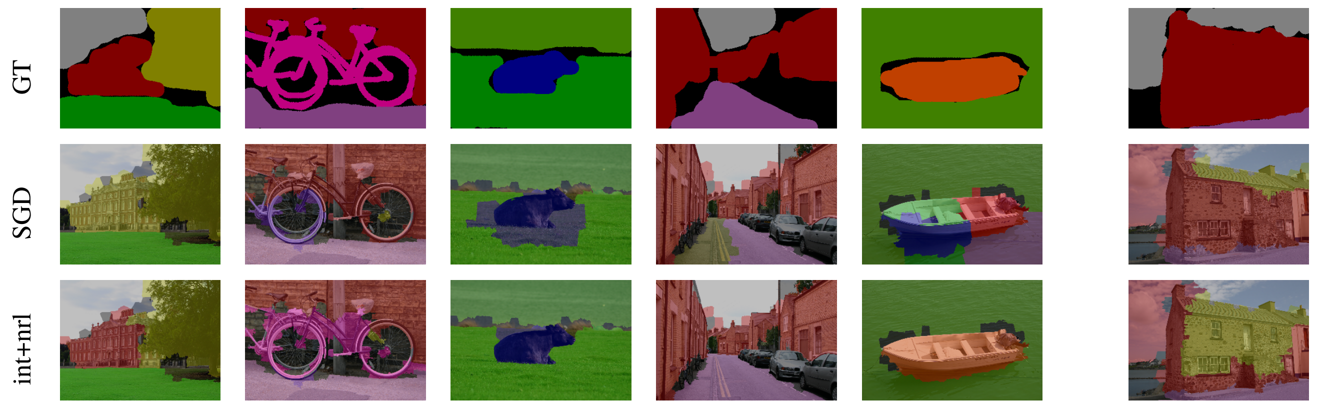


Figure 3.1: Illustrative examples of the performance of SGD and int+nrl on several MSRC-21 test images. Integrated training with neural factors improves classification accuracy over subgradient descent. The last column presents a case in which our model fails to outperform SGD.

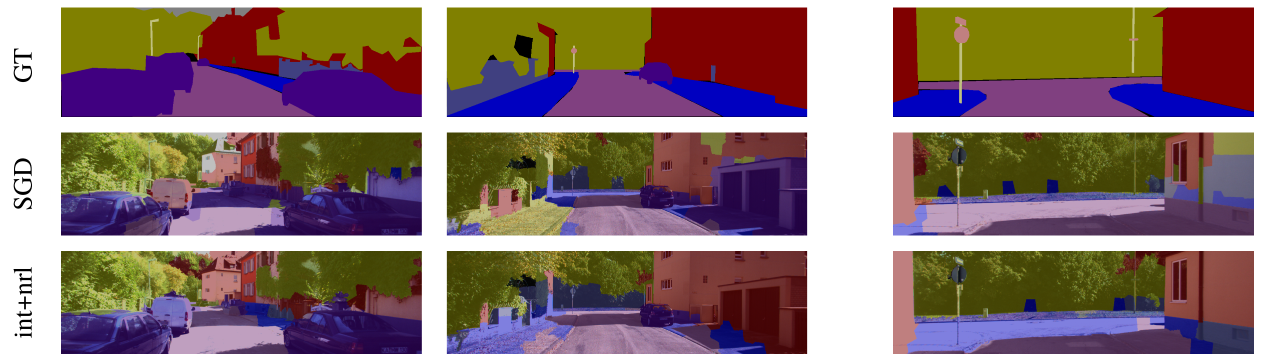


Figure 3.2: Illustrative examples of the performance of SGD and int+nrl on several KITTI test images. Integrated training with neural factors improves classification accuracy over subgradient descent. The last column presents a case in which our model fails to outperform SGD.

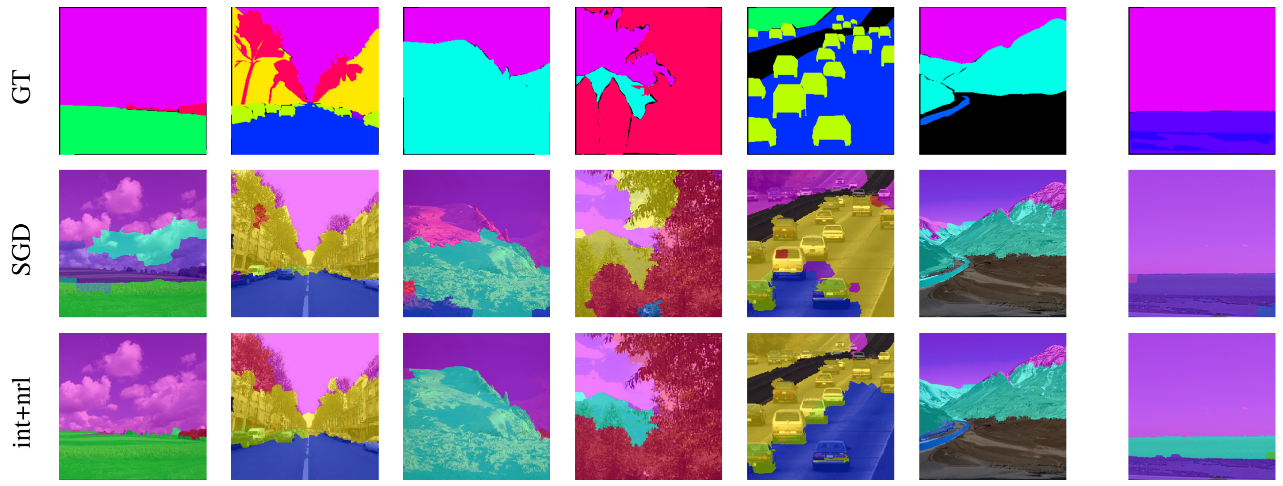


Figure 3.3: Illustrative examples of the performance of SGD and int+nrl on several SIFT Flow test images. Integrated training with neural factors improves classification accuracy over subgradient descent. The last column presents a case in which our model fails to outperform SGD.

linear parameters w are initialized to 0, while the neural factor parameters θ and γ are initialized according to [Glorot and Bengio, 2010], except for the top layer weights which are set to 0. The class weights are set to correct for class imbalance. The model is trained using CPU-parallelized loss-augmented prediction, while the neural factors are trained using GPU parallelism.

The following models are compared: unary-only (unary), N-slack cutting plane training (CP) with delayed constraint generation, subgradient descent (SGD)[3], integrated training with neural unary and linear interaction factors (int+lin), bifurcated training with neural interaction factors (bif+nrl), and integrated training with neural unary and neural interaction factors (int+nrl).

Multiclass logistic regression is used as unary classifier, trained with gradient descent by cross-entropy optimization. All unary neural factors contain a single hidden layer with 256 tanh-units, for direct comparison of integrated learning with upfront logistic regression training. The interaction neural factors contain a single hidden layer of 512 tanh-units to elucidate the benefit of nonlinear factors, without overly increasing the model's capacity. The experiment is set up to highlight the benefit of integrated learning by restricting the unary factors to features insufficiently discriminative on their own. This deliberately leads to noisy unary classification, forcing the model to rely on contextual relations for accurate prediction. The interaction factors encode information about their incident region feature vectors to allow neural factors to extract meaningful patterns from gradient/color combinations. We deliberately encoded less information in the interaction features, such that the model cannot solely rely on interaction factors for accurate and coherent predictions.

3.4.2 Results and Discussion

Accuracy results on the MSRC-21 [Shotton et al., 2009] test images are presented in Table 3.1, while Figure 3.1 shows a handful of illustrative examples that compare segmentations attained by SGD with int+nrl. The results of the same experiment for the KITTI benchmark [Ros et al., 2015], augmented with additional training images Kundu et al. [Kundu et al., 2014], are shown in Table 3.2 and Figure 3.2. Qualitative results on the

[3]SGD uses bifurcated training with linear interactions, hence it could be named bif+lin.

Table 3.1: MSRC-21 class, pixel-wise, and class-mean test accuracy (in %) for different models

	building	grass	tree	cow	sheep	sky	aeropl.	water	face	car	bicycle	flower	sign	bird	book	chair	road	cat	dog	body	boat	**pixel**	**class**
unary	15	60	52	8	10	68	35	46	12	21	21	42	9	2	36	0	21	14	5	6	1	36.3	23.1
CP	44	**77**	61	48	21	85	60	69	51	70	63	54	49	16	87	21	41	47	6	16	**33**	59.4	48.5
SGD	49	67	71	39	64	80	81	67	35	74	60	42	19	2	88	**51**	53	38	4	31	26	59.2	49.6
int+lin	48	76	83	67	73	94	78	**67**	59	56	68	65	48	14	**95**	43	61	53	6	45	32	67.4	58.5
bif+nrl	46	74	79	51	51	92	**83**	64	76	64	67	50	53	9	83	34	42	42	0	47	22	62.7	53.7
int+nrl	53	**77**	86	61	73	**95**	**83**	60	**87**	**77**	72	**69**	**77**	27	85	29	**67**	46	0	57	26	70.1	62.3
int†+lin	46	67	80	47	69	83	79	60	35	66	63	53	10	2	89	43	66	**62**	4	45	17	61.2	51.7
3-layer	**62**	76	**87**	**68**	**77**	94	81	66	84	65	**75**	53	69	**33**	81	**51**	**67**	58	**30**	**64**	25	**71.6**	**65.1**

SIFT Flow [Liu et al., 2011] dataset are shown in Figure 3.3, while accuracy results are shown in Table 3.3.

The results show that unary-only prediction is very inaccurate (pixel-wise/class-mean accuracy of 36.3/23.1% for the MSRC-21 dataset, 53.8/42.8% for the KITTI dataset, and 44.7/7.5% for the SIFT Flow dataset). The reason for this is that unary features are not sufficiently distinctive to allow for differentiation between classes due to their low dimensionality. Accurate predictions are only possible by taking into account contextual output relations, demonstrated by the increased accuracy of CP (MSRC-21: 59.4/48.5%; KITTI: 61.5/46.7%; SIFT Flow: 62.5/13.8%) as well as SGD (MSRC-21: 59.2/49.6%; KITTI: 65.5/50.6%; SIFT Flow: 65.9/15.3%). These structured predictors learn linear relations between image regions, which allows them to correct errors originating from the underlying unary classifier. However, the unary factor's linear weights w have only limited capability for error correction in the opposite direction, due to the fact that the SSVM cannot alter the unary classifier parameters post-hoc.

Using an integrated training approach such as int+lin, in which the SSVM is trained end-to-end, improves accuracy (MSRC-21: 67.4/58.5%; KITTI: 70.2/57.8%; SIFT Flow: 70.2/15.6%) over the bifurcated procedures CP and SGD. Although neither the unary or interaction features are very distinctive, the integrated procedure updates parameters in such a way that both factor types have a unique discriminative focus. Their synergistic relationship ultimately results in higher accuracy. To better compare SGD (which uses 8, 21, and 33 logistic regression outputs as unary input features for the different benchmarks) with int+lin, we also depict the accuracy (MSRC-21: 61.2/51.7%; KITTI: 69.2/53.5%; SIFT Flow: 70.2/15.6%) of a model (int†+lin) with only 8, 21, and 33 unary hidden units for the KITTI, MSRC-21, and SIFT Flow dataset, rather than 256 units. The 2.0/2.1% (MSRC-21), 3.7/2.9% (KITTI), and 4.3/0.3% (SIFT Flow) increases in accuracy over SGD further illustrates the benefit of integrated learning and inference over conventional bifurcated SSVM training.

Another insight gained by the results is that accuracy increases when replacing linear interaction factors of conventional SSVMs with neural factors, i.e., int+nrl (MSRC-21: 70.1/62.3%; KITTI: 75.6/60.9%; SIFT Flow: 71.3/17.0%) and bif+nrl (MSRC-21: 62.7/53.7%; KITTI: 70.0/55.9%; SIFT Flow:68.8/16.1%) outperform int+lin (MSRC-21: 67.4/58.5%; KITTI: 70.2/57.8%; SIFT Flow: 70.3/16.2%) and SGD (MSRC-21: 59.2/49.6%;

Table 3.2: KITTI class, pixel-wise, and class-mean test accuracy (in %) for different models

	sky	building	road	sidewalk	fence	vegetation	pole	car	**pixel**	**class**
unary	75	63	59	29	8	71	0	38	53.8	42.8
CP	84	76	75	11	5	75	0	48	61.5	46.7
SGD	77	68	86	19	4	80	0	71	65.5	50.6
int+lin	86	76	82	42	23	81	**6**	67	70.2	57.8
bif+nrl	86	77	81	41	12	80	0	71	70.0	55.9
int+nrl	86	**83**	**88**	50	19	84	4	74	75.6	60.9
int‡+lin	81	76	85	22	12	82	0	70	69.2	53.5
3-layer	**90**	82	**88**	**55**	**28**	**87**	1	**78**	**77.6**	**63.6**

Table 3.3: SIFT Flow pixel-wise and class-mean test accuracy (in %) for different models

	pixel	**class**
unary	44.7	7.5
CP	62.5	13.8
SGD	65.9	15.3
int+lin	70.3	16.2
bif+nrl	68.8	16.1
int+nrl	71.3	17.0
int‡+lin	70.2	15.6
3-layer	**71.5**	**17.2**

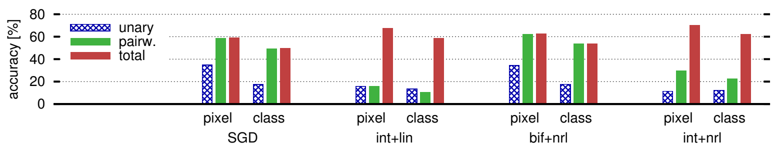


Figure 3.4: Visualization of the synergy between unary and interaction factors. In bifurcated training the interactions make unary factors redundant as these cannot be adapt to errors made by the interactions. In integrated training, combining both factor types leads to a higher accuracy as they can mutually adapt to each other's weight updates.

KITTI: 65.5/50.6%; SIFT Flow: 65.9/15.3%) respectively. This increase can be attributed to the higher number of parameters, as well as the added nonlinearities in combination with correct regularization. The model has greater generalization power, allowing the factors to extract more complex and meaningful interaction patterns. Neural factors offer great flexibility as they can be stacked to arbitrary depths. This leads to even higher generalization, as indicated by the increased accuracy (MSRC-21: 71.6/65.1%; KITTI: 77.6/63.6%; SIFT Flow: 71.5/17.2%) of the deeper 3-layer (int+nrl) model. Herein both unary and interaction factors are 3-hidden-layer neural networks consisting of 256 and 512 units (rectified linear units for MSRC-21 and KITTI and tanh units for SIFT Flow) in each layer respectively. Our model can thus easily be extended, for example by letting neural factors represent the fully-connected layer in convolutional neural networks. As such, it serves as a foundation for more complex structured models.

All methods converge within 600 epochs, with one epoch taking approximately 12.62 seconds for the MSRC-21 dataset, 4.35 seconds for the KITTI dataset, and 197.27 seconds on the SIFT Flow dataset for the int+nrl algorithm. The algorithm can be further optimized for speed by exploiting CPU parallelism and optimizing the data transfer between CPU and GPU when training the nonlinear factors in our model. These optimizations will be part of our future work.

Figure 3.4 illustrates the synergy between unary and interaction factors achieved through both integrated and bifurcated training, exercised on the MSRC-21 dataset. The bars depict model test accuracy when using only unary or pairwise factors, by setting either the pairwise or unary factors respectively to a zero factor value, thus $\langle w, \varphi_I(x,y)\rangle$ or $\langle w, \varphi_U(x,y)\rangle = 0 \; \forall y \in \mathcal{Y}$. Although the unary factors alone perform

Table 3.4: State-of-the-art comparison: MSRC-21 per-class, class-mean, and global pixel-wise test accuracy (in %) for different models

	building	grass	tree	cow	sheep	sky	aeropl.	water	face	car	bicycle	flower	sign	bird	book	chair	road	cat	dog	body	boat	**pixel**	**class**
neural factors	**76**	94	**94**	92	97	92	94	85	93	**88**	94	95	70	**78**	**97**	87	88	91	**78**	**88**	**63**	**88.9**	**87.4**
[Liu et al., 2015]	71	95	92	87	**98**	**97**	**97**	**89**	**95**	85	**96**	94	75	76	89	84	88	**97**	77	87	52	88.5	86.7
[Yao et al., 2012]	71	**98**	90	79	86	93	88	86	90	84	94	**98**	**76**	53	**97**	71	**89**	83	55	68	17	86.2	79.3
[Lucchi et al., 2013]	67	89	85	93	79	93	84	75	79	87	89	92	71	46	96	79	86	76	64	77	50	83.7	78.9
[Munoz et al., 2010]	63	93	88	84	65	89	69	78	74	81	84	80	51	55	84	80	69	47	59	71	24	78	71
[Gonfaus et al., 2010]	60	78	77	91	68	88	87	76	73	77	93	97	73	57	95	81	76	81	46	56	46	77	75
[Shotton et al., 2008]	49	88	79	**97**	97	78	82	54	87	74	72	74	36	24	93	51	78	75	35	66	18	72	67
[Lucchi et al., 2012]	41	77	79	87	91	86	92	65	86	65	89	61	76	48	77	**91**	77	82	32	48	39	73	70

well in bifurcated training, nearly all accuracy can be attributed to the interactions. A possible explanation is that both types essentially learn the same information. The interactions correct errors of the underlying classifier and ultimately make unary factors redundant. In integrated training, neither the unary or interaction factors alone attain a high accuracy, but the combination of both does.

We explain this synergistic relationship with an example: Unary factors assign to a region of class A, a second-to-highest factor value to class A, a highest value to class B, and a low value to class C. The interactions also assign a second-to-highest value to class A, but a highest value to class C, and a low value to class B. Independently both factors incorrectly predict the region of class A as belonging to class B or class C. However, when combined they correctly assign a highest value to class A. In the figure, bifurcated training only shows limited signs of factor synergy, as the optimization procedure is insufficiently able to steer unary and pairwise parameters in different directions, which causes them have a similar discriminative focus. This observation leads us to believe that integrated learning and inference results in higher accuracy by synergistic unary/interaction factor optimization. Both factor types are no longer optimized for independent accuracy, but mutually adapt to each other's parameter updates, which results in enhanced predictive power.

In addition to the previous experiments, the viability of our neural factor model is shown through comparison with the closely related work of Liu et al. [Liu et al., 2015] on the MSRC-21 dataset. Liu et al. make use of features extracted from square regions of varying size around each superpixel, through means of a pretrained convolutional neural network. We compare our model with theirs using overfeat features [Sermanet et al., 2013], trained on individual regions. Furthermore, the model settings have been altered with respect to the previous experiments. More specifically, 1,000 SLIC superpix els are utilized for the over-segmentation preprocessing step, enforcing superpixel connectivity and merging any superpixel with a surface area below a particular threshold. DAISY gradient and HSV color features are extracted according to a regular lattice, and clustered via minibatch k-means clustering. Next, the same type of features are extracted for each individual pixel, leading to unary and pairwise factor feature vectors. Moreover, the (x, y)-position of the superpixel (median-based) center is included in the unary feature vectors, while the distance and angle between the two superpixel centers is encoded into the interaction feature vectors. The neural factors are represented by multilayer neural networks using tanh-units, trained according to our Algorithm 3.2, using conventional momentum and single image-sized batches per gradient update. Classes are balanced by weighing them with the inverse of the class frequency. The results are presented in Table 3.4, which indicate that our model is capable of performing on par with the current state-of-practice, when used in conjunction with more advanced methods, e.g., overfeat features. Moreover, similar to Liu et al. [Liu et al., 2015], we have added the scores of closely related methods for completeness, for which the results are shown below the horizontal line in Table 3.4.

3.5 Conclusion

A structured prediction model that integrates back-propagation and loss-augmented inference into subgradient descent training of structural support vector machines (SSVMs) is proposed. This model departs from the traditional bifurcated approach in which a unary classifier is trained independently from the structured predictor. Furthermore, the SSVM factors are extended to neural factors, which allows both unary and interaction factors to be highly nonlinear functions of input features. Results on a complex image segmentation task show that end-to-end SSVM training, and/or using neural factors, leads to more accurate predictions than conventional subgradient descent and N-slack cutting plane training. Results show that our model serves as a foundation for more advanced structured models, e.g., by using latent variables, learned feature representations, or complexer connectivity structures.

3.A Application to Autonomous Vehicle Data

This addendum shows an extension to the original work in [Houthooft and De Turck, 2016]. We compare an end-to-end segmentation method, as described in Section 2.4.4, with a convolutional implementation of the end-to-end SSVM method described in this chapter. We call this implementation a *deep SSVM*, and apply it to the autonomous agricultural vehicle segmentation dataset introduced in Chapter 2.

3.A.1 Deep SSVM

Avoiding the need for an over-segmentation preprocessing method, similar to Section 2.4.4, we propose the use of an SSVM with an underlying grid-structured graphical model. This model connects each pixel to its four adjacent neighbors in the left, top, right, and bottom directions. It uses a unary energy function $f(x, y; \theta)$, as described in Eq. (3.11), modeled by a convolutional neural network (CNN) which outputs an energy value for each distinct class. The interaction energy function $h(x, y; \gamma)$, as described in Eq. (3.12), is modeled by the same CNN, except that it uses a different output branch that ends in two distinct output blocks of 18^2 values. Once 18^2 outputs for the horizontal connections (left and right), once 18^2 outputs for the vertical connections (top and bottom). The right column and bottom row are stripped away from both branch outputs, to match the actual grid layout. The CNN factors are composed of stack of convolutional layers, feeding into a set of dense layers, feeding into a stack of transposed convolutional layers, which output the SSVM energy values. The whole system is trained end-to-end using back-propagation and α-expansion for loss-augmented inference, as described in Algorithm 3.2. This means, rather than applying the unary and pairwise neural networks as presented in this chapter separately for each individual unary or pairwise connection, that the CNN outputs all SSVM energy values simultaneously.

The model has the following architecture, which is also shown visually in Figure 3.5. The SSVM graphical model is a grid that connects adjacent pixels, as described previously, consisting of $V_n = 160 \times 80$ nodes and $E_n = (160\times79)+(159\times160)$ undirected edges. It takes as input $3\times640\times320$ images downscaled to $3\times160\times80$, and outputs a 160×80 segmentation. This means that both the unary features x_i^U as well as the interaction features x_i^I

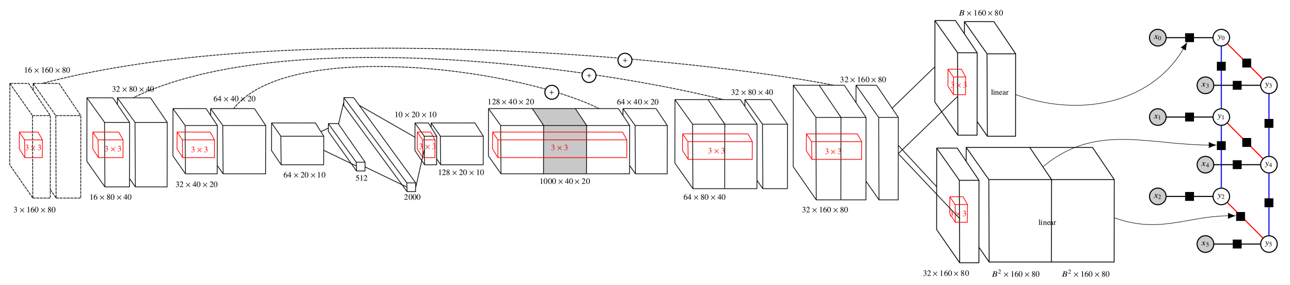


Figure 3.5: Deep SSVM using a convolutional architecture; the circled plus connections represent the skip-layer connections; the gray box represents the OverFeat CNN class probabilities according to the window extraction process.

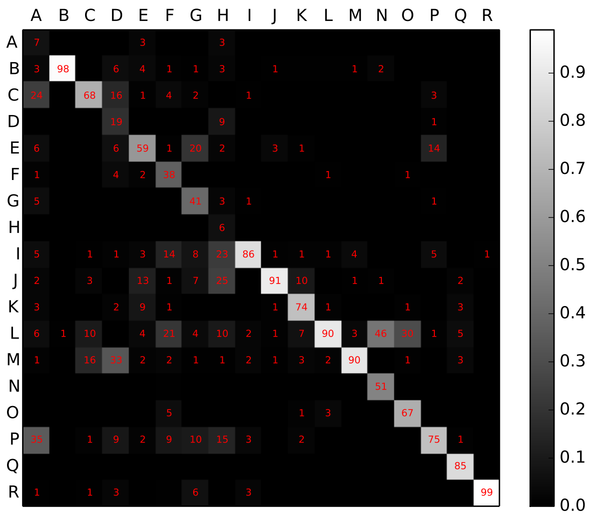
are the complete 3-channel input images. We specifically use downscaled images of 160×80 pixels to reduce the processing time, since our α-expansion algorithm implementation cannot be run parallely on a GPU. A possible solution to this is described in [Vineet and Narayanan, 2009].

The CNN consists of consecutive convolutional layers with 16, 32, and 64 filters of size 3×3. After each layer, a 2-dim max-pooling operation is applied, halving the size of each set of feature maps. Next, 2 dense layers of respectively 512 and 2000 units are used, which feed into the stack of transposed convolutional layers. The set of 2000 dense outputs is first reshaped into a tensor of size $10 \times 10 \times 20$, after which layers of respectively 128, 32, and 32 filters of size 3×3 are used. Each of these layers is followed by a 2-dim upscaling layer, doubling the size of each feature map.

The final feature map is fed into two parallel transposed convolutional layers, one with 18 3×3 filters and one with 2×18^2 filters. The first output branch feeds into the affine unary energy layer, with output size $18 \times 160 \times 80$, while the second branch feeds into an affine interaction energy layer, with output size $324 \times 160 \times 80$ (which is cropped to match the actual interaction grid). As done in Section 2.4.4, skip-layer connections are added that connect the convolutional layers, before their max-pooling operation, to their transposed convolutional counterparts. As such, the input of each transposed convolutional layer is expanded through concatenation. This is made clear in Figure 3.5 by the horizontal dashed connections with the circled plus symbol. All nonlinear transformations are composed of ELUs [Clevert et al., 2015]; the learning scheme Adam [Kingma and Ba, 2015] optimizes the objective function of Eq. (3.5), according to Algorithm 3.2, using minibatches of size 6. The loss function $\Delta(\cdot, \cdot)$ is weighted by the inverse of the class frequencies. Transfer learning is used as described in Section 2.4.4 through concatenation of the $1000 \times 20 \times 40$ feature tensor with tensor of feature maps after the first upscaling layer.

3.A.2 End-to-end Segmentation without SSVM

The deep SSVM is compared to an end-to-end segmentation model, as described in Section 2.4.4, with the same architecture as the deep SSVM. The architecture is identical to Figure 3.5, but without the lower branch, the graphical model (SSVM), and with a softmax output of 18 classes rather than an affine transformation (linear) into 18 unary energy values.



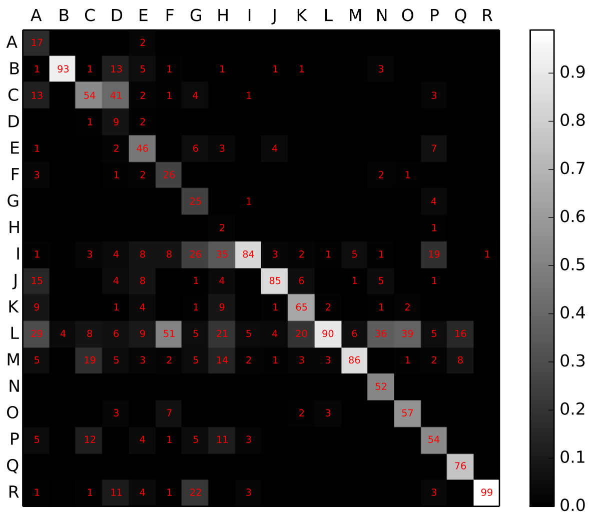
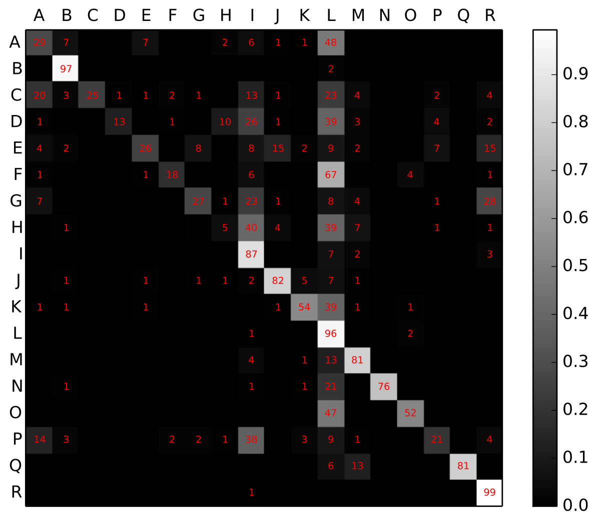


Figure 3.6: Deep SSVM (top) and end-to-end segmentation model (bottom): pixel-wise precision results on the autonomous agricultural vehicle test dataset, described as a confusion matrix. Class legend: person (A), tractor (B), harvester (C), implement (D), moving object (E), nonmoving object (F), power pole (G), fence/hedge (H), tree/shrubbery (I), public road (J), farm road (K), harvested untilled area (L), unharvested area (M), tilled area (N), swath (O), building (P), water (Q), sky (R).



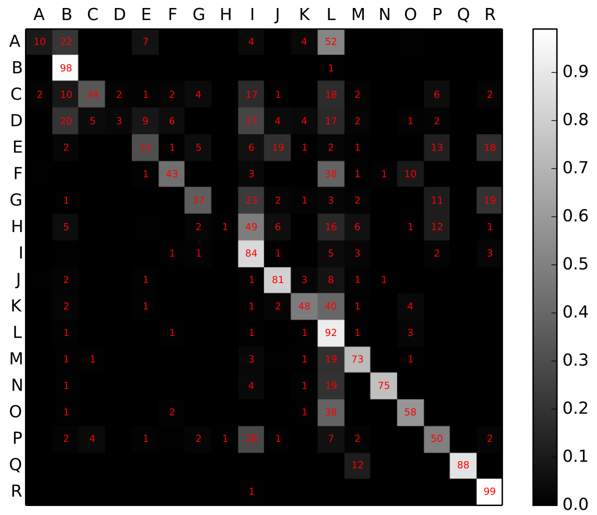


Figure 3.7: Deep SSVM (top) and end-to-end segmentation model (bottom): pixel-wise recall results on the autonomous agricultural vehicle test dataset, described as a confusion matrix. Class legend: person (A), tractor (B), harvester (C), implement (D), moving object (E), nonmoving object (F), power pole (G), fence/hedge (H), tree/shrubbery (I), public road (J), farm road (K), harvested untilled area (L), unharvested area (M), tilled area (N), swath (O), building (P), water (Q), sky (R).

3.A.3 Results and Discussion

This section demonstrates that the presented end-to-end SSVM training method can be used in conjunction with highly complex underlying neural factor models. The prediction accuracy between the deep SSVM and an end-to-end segmentation model that uses the exact same convolutional base is compared through precision and recall on the autonomous agricultural test dataset as introduced in Chapter 2. The results are shown in Figures 3.6 and 3.7 through confusion matrices. In these matrices, each row i represents pixels that have ground truth label j. As such, at position (i, j), the number of pixels classified as j but actually belonging to i is shown. Rather than showing the true pixel counts, we split up this matrix into a precision and a recall confusion matrix. The recall matrix (Figures 3.7) shows the original count divided by the row sum (in %), while the precision matrix (Figures 3.6) shows this count divided by the column sum (in %). Although recall is often treated as the actual accuracy, we also include precision results since it captures different predictive qualities of the models

Similar to Chapter 2, the mistakes made by both models are highly interpretable, for example incorrectly classifying 'fence/hedge' as 'tree/shrubbery'. On average, it can be noticed that the deep SSVM method attains a slightly lower recall (53.3% compared to 55.8%), but a higher precision (63.6% compared to 56.7%) than the end-to-end segmentation model. This means that if the deep SSVM model makes a prediction, on average, it is more likely to be correct than its competitor. However, this leads to a slightly lower detection rate for particular classes. When comparing these results to the results in Chapter 2, we see that the accuracy values of both models are lower. This is due to the downsampling process, which discards lots of information present in the original image. When looking at the global accuracy, which is the total number of correctly classified pixels, the end-to-end segmentation model attains an accuracy of 90.5%, while the deep SSVM scores slightly higher with 92.3%.

These results indicate that the deep SSVM model is capable of adding value to the segmentation process. However, the most important point to take away from these results is that they reinforce the conclusion made previously, namely that the end-to-end SSVM training method can serve as a foundation for highly complex underlying models.

References

[Achanta et al., 2012] Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., and Susstrunk, S. (2012). SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282.

[Bagirov et al., 2013] Bagirov, A. M., Jin, L., Karmitsa, N., Al Nuaimat, A., and Sultanova, N. (2013). Subgradient method for nonconvex nonsmooth optimization. *Journal of Optimization Theory and Applications*, 157(2):416–435.

[Bastien et al., 2012] Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I., Bergeron, A., Bouchard, N., Warde-Farley, D., and Bengio, Y. (2012). Theano: new features and speed improvements. In *NIPS Workshop Deep Learning and Unsupervised Feature Learning*.

[Bertelli et al., 2011] Bertelli, L., Yu, T., Vu, D., and Gokturk, B. (2011). Kernelized structural SVM learning for supervised object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2153–2160.

[Bottou et al., 1997] Bottou, L., Bengio, Y., and LeCun, Y. (1997). Global training of document processing systems using graph transformer networks. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 489–494.

[Boykov et al., 2001] Boykov, Y., Veksler, O., and Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239.

[Bridle, 1989] Bridle, J. S. (1989). Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In *Advances in Neural Information Processing Systems 2 (NIPS)*, pages 211–217.

[Chang and Yih, 2013] Chang, M.-W. and Yih, W.-t. (2013). Dual coordinate descent algorithms for efficient large margin structured prediction. *Transactions of the Association for Computational Linguistics*, 1:207–218.

[Chen et al., 2015a] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2015a). Semantic image segmentation with deep convolutional nets and fully connected CRFs. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

[Chen et al., 2015b] Chen, L.-C., Schwing, A. G., Yuille, A. L., and Urtasun, R. (2015b). Learning deep structured models. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pages 1785–1794.

[Clevert et al., 2015] Clevert, D.-A., Unterthiner, T., and Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (ELUs). *arXiv preprint arXiv:1511.07289*.

[Collins, 2002] Collins, M. (2002). Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8.

[Collobert et al., 2011] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

[Dauphin et al., 2014] Dauphin, Y. N., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., and Bengio, Y. (2014). Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in Neural Information Processing Systems 27 (NIPS)*, pages 2933–2941.

[Do and Artières, 2010] Do, T.-M.-T. and Artières, T. (2010). Neural conditional random fields. In *Proceedings of the 13th International Conference Artificial Intelligence and Statistics (AISTATS)*, pages 177–184.

[Domke, 2013] Domke, J. (2013). Structured learning via logistic regression. In *Advances in Neural Information Processing Systems 24 (NIPS)*, pages 647–655.

[Farabet et al., 2013] Farabet, C., Couprie, C., Najman, L., and LeCun, Y. (2013). Learning hierarchical features for scene labeling. *IEEE*

Transactions on Pattern Analysis and Machine Intelligence, 35(8):1915–1929.

[Glorot and Bengio, 2010] Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference Artificial Intelligence and Statistics (AISTATS)*, pages 249–256.

[Gonfaus et al., 2010] Gonfaus, J. M., Boix, X., Van de Weijer, J., Bagdanov, A. D., Serrat, J., and Gonzalez, J. (2010). Harmony potentials for joint classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3280–3287.

[Houthooft et al., 2016] Houthooft, R., De Boom, C., Verstichel, S., Ongenae, F., and De Turck, F. (2016). Structured output prediction for semantic perception in autonomous vehicles. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI)*.

[Houthooft and De Turck, 2016] Houthooft, R. and De Turck, F. (2016). Integrated inference and learning of neural factors in structural support vector machines. *Pattern Recognition*, 59:292–301.

[Joachims et al., 2009] Joachims, T., Finley, T., and Yu, C.-N. J. (2009). Cutting-plane training of structural SVMs. *Machine Learning*, 77(1):27–59.

[Kingma and Ba, 2015] Kingma, D. and Ba, J. (2015). Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

[Krähenbühl and Koltun, 2013] Krähenbühl, P. and Koltun, V. (2013). Parameter learning and convergent inference for dense random fields. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 513–521.

[Kundu et al., 2014] Kundu, A., Li, Y., Dellaert, F., Li, F., and Rehg, J. (2014). Joint semantic segmentation and 3D reconstruction from monocular video. In *Proceedings of the 13th European Conference on Computer Vision (ECCV)*, pages 703–718.

[Lacoste-julien et al., 2013] Lacoste-julien, S., Jaggi, M., Schmidt, M., and Pletscher, P. (2013). Block-coordinate Frank-Wolfe optimization for structural SVMs. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 53–61.

[Li and Zemel, 2014] Li, Y. and Zemel, R. (2014). High order regularization for semi-supervised learning of structured output problems. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pages 1368–1376.

[Liang et al., 2015] Liang, X., Liu, S., Shen, X., Yang, J., Liu, L., Lin, L., and Yan, S. (2015). Deep human parsing with active template regression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(12):2402–2414.

[Lin et al., 2015] Lin, L., Wang, X., Yang, W., and Lai, J.-H. (2015). Discriminatively trained and-or graph models for object shape detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(5):959–972.

[Liu et al., 2011] Liu, C., Yuen, J., and Torralba, A. (2011). Sift flow: Dense correspondence across scenes and its applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):978–994.

[Liu et al., 2015] Liu, F., Lin, G., and Shen, C. (2015). CRF learning with CNN features for image segmentation. *Pattern Recognition*, 48(10):2983–2992.

[Lu et al., 2015] Lu, J., Xu, R., and Corso, J. J. (2015). Human action segmentation with hierarchical supervoxel consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3762–3771.

[Lucchi et al., 2013] Lucchi, A., Li, Y., and Fua, P. (2013). Learning for structured prediction using approximate subgradient descent with working sets. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1987–1994.

[Lucchi et al., 2012] Lucchi, A., Li, Y., Smith, K., and Fua, P. (2012). Structured image segmentation using kernelized features. In *Proceedings* *of the 12th European Conference on Computer Vision (ECCV)*, pages 400–413.

[Morris and Fosler-Lussier, 2008] Morris, J. and Fosler-Lussier, E. (2008). Conditional random fields for integrating local discriminative classifiers. *IEEE/ACM Transactions on Audio, Speech, Language Processing*, 16(3):617–628.

[Müller, 2014] Müller, A. C. (2014). *Methods for Learning Structured Prediction in Semantic Segmentation of Natural Images*. PhD thesis, University of Bonn.

[Müller and Behnke, 2014] Müller, A. C. and Behnke, S. (2014). PyStruct - Learning structured prediction in Python. *Journal of Machine Learning Research*, 15:2055–2060.

[Munoz et al., 2010] Munoz, D., Bagnell, J. A., and Hebert, M. (2010). Stacked hierarchical labeling. In *Proceedings of the 11th European Conference on Computer Vision (ECCV)*, pages 57–70.

[Nedić and Bertsekas, 2001] Nedić, A. and Bertsekas, D. (2001). Convergence rate of incremental subgradient algorithms. In *Stochastic Optimization: Algorithms and Applications*, pages 223–264.

[Ngiam et al., 2011] Ngiam, J., Coates, A., Lahiri, A., Prochnow, B., Le, Q. V., and Ng, A. Y. (2011). On optimization methods for deep learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 265–272.

[Nowozin and Lampert, 2011] Nowozin, S. and Lampert, C. H. (2011). Structured learning and prediction in computer vision. *Foundations and Trends in Computer Graphics and Vision*, 6(3–4):185–365.

[Pascanu et al., 2014] Pascanu, R., Dauphin, Y. N., Ganguli, S., and Bengio, Y. (2014). On the saddle point problem for non-convex optimization. *arXiv preprint arXiv:1405.4604*.

[Peng et al., 2013] Peng, B., Zhang, L., and Zhang, D. (2013). A survey of graph theoretical approaches to image segmentation. *Pattern Recognition*, 46(3):1020 – 1038.

[Peng et al., 2009] Peng, J., Bo, L., and Xu, J. (2009). Conditional neural fields. In *Advances in Neural Information Processing Systems 22 (NIPS)*, pages 1419–1427.

[Prabhavalkar and Fosler-Lussier, 2010] Prabhavalkar, R. and Fosler-Lussier, E. (2010). Backpropagation training for multilayer conditional random field based phone recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5534–5537.

[Ros et al., 2015] Ros, G., Ramos, S., Granados, M., Bakhtiary, A., Vazquez, D., and Lopez, A. M. (2015). Vision-based offline-online perception paradigm for autonomous driving. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 231–238.

[Schwing and Urtasun, 2015] Schwing, A. G. and Urtasun, R. (2015). Fully connected deep structured networks. *arXiv preprint arXiv:1503.02351*.

[Sermanet et al., 2013] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. (2013). OverFeat: Integrated recognition, localization and detection using convolutional networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

[Shor et al., 1985] Shor, N. Z., Kiwiel, K. C., and Ruszcaynski, A. (1985). *Minimization methods for non-differentiable functions*. Springer-Verlag New York, Inc.

[Shotton et al., 2008] Shotton, J., Johnson, M., and Cipolla, R. (2008). Semantic texton forests for image categorization and segmentation. In *Proceedings of the IEEE Conference on Computer vision and pattern recognition (CVPR)*, pages 211–227.

[Shotton et al., 2009] Shotton, J., Winn, J., Rother, C., and Criminisi, A. (2009). Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *International Journal of Computer Vision*, 81(1):2–23.

[Sutskever et al., 2013] Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th international conference on machine learning (ICML)*, pages 1139–1147.

[Tola et al., 2010] Tola, E., Lepetit, V., and Fua, P. (2010). DAISY: An efficient dense descriptor applied to wide baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(5):815–830.

[Tompson et al., 2014] Tompson, J. J., Jain, A., LeCun, Y., and Bregler, C. (2014). Joint training of a convolutional network and a graphical model for human pose estimation. In *Advances in Neural Information Processing Systems 27 (NIPS)*, pages 1799–1807.

[Tsochantaridis et al., 2005] Tsochantaridis, I., Joachims, T., Hofmann, T., and Altun, Y. (2005). Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484.

[Vineet and Narayanan, 2009] Vineet, V. and Narayanan, P. (2009). Solving multilabel MRFs using incremental α-expansion on the GPUs. In *Proceedings of the 9th Asian Conference on Computer Vision (ACCV)*, pages 633–643.

[Wang et al., 2013] Wang, X., Lin, L., Huang, L., and Yan, S. (2013). Incorporating structural alternatives and sharing into hierarchy for multiclass object recognition and detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3334–3341.

[Xu et al., 2014] Xu, R., Chen, G., Xiong, C., Chen, W., and Corso, J. J. (2014). Compositional structure learning for action understanding. *arXiv preprint arXiv:1410.5861*.

[Yao et al., 2012] Yao, J., Fidler, S., and Urtasun, R. (2012). Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 702–709.

[Yu et al., 2009] Yu, D., Deng, L., and Wang, S. (2009). Learning in the deep-structured conditional random fields. In *NIPS Workshop Deep Learning for Speech Recognition and Related Applications*.

[Zhang, 2004] Zhang, T. (2004). Statistical behavior and consistency of classification methods based on convex risk minimization. *Annals of Statistics*, 32(1):56–85.

Chapter 4

Variational Information Maximizing Exploration

Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck & Pieter Abbeel, *VIME: Variational Information Maximizing Exploration*, In Advances in Neural Information Processing Systems (NIPS), pp. 1109–1117, Barcelona, Spain, 2016.

Makes use of the task and algorithm descriptions presented in

Yan Duan, Xi Chen, Rein Houthooft, John Schulman & Pieter Abbeel, *Benchmarking Deep Reinforcement Learning for Continuous Control*, In Proceedings of the 33rd International Conference on Machine Learning (ICML), pp. 1329–1338, New York City, USA, 2016.

UC Berkeley, OpenAI, Ghent University – imec

⋆ ⋆ ⋆

Scalable and effective exploration remains a key challenge in reinforcement learning (RL). While there are methods with optimality guarantees in the setting of discrete state and action spaces, these methods cannot be applied in high-dimensional deep RL scenarios. As such, most contemporary RL *relies on simple heuristics such as epsilon-greedy exploration or adding Gaussian noise to the controls. This chapter introduces Variational Information Maximizing Exploration (VIME), an exploration strategy based on maximization of information gain about the agent's belief of environment dynamics. We propose a practical implementation, using variational inference in Bayesian neural networks which efficiently handles continuous state and action spaces. VIME modifies the MDP reward function, and can be applied with several different underlying RL algorithms. We demonstrate that VIME achieves significantly better performance compared to heuristic exploration methods across a variety of continuous control tasks and algorithms, including tasks with very sparse rewards.*

4.1 Introduction

Reinforcement learning (RL) studies how an agent can maximize its cumulative reward in a previously unknown environment, which it learns about through experience. A long-standing problem is how to manage the trade-off between exploration and exploitation. In *exploration*, the agent experiments with novel strategies that may improve returns in the long run; in *exploitation*, it maximizes rewards through behavior that is known to be successful.

An effective exploration strategy allows the agent to generate trajectories that are maximally informative about the environment. For small tasks, this trade-off can be handled effectively through Bayesian RL [Ghavamzadeh et al., 2015] and PAC-MDP methods [Kakade et al., 2003, Kearns and Singh, 2002, Brafman and Tennenholtz, 2003, Auer, 2003, Pazis and Parr, 2013], which offer formal guarantees. However, these guarantees assume discrete state and action spaces. Hence, in settings where state-action discretization is infeasible, many RL algorithms use heuristic exploration strategies. Examples include acting randomly using ϵ-greedy or Boltzmann exploration [Mnih et al., 2015], and utilizing Gaussian noise on the controls in policy gradient methods [Schulman et al., 2015]. These heuristics often rely on random walk behavior which can be highly inefficient, for example Boltzmann exploration requires a training time exponential in the number of states in order to solve the well-known n-chain MDP [Osband et al., 2016b].

In between formal methods and simple heuristics, several works have proposed to address the exploration problem using less formal, but more expressive methods [Stadie et al., 2015, Osband et al., 2016a, Oh et al., 2015, Hester and Stone, 2015, Subramanian et al., 2016]. However, none of them fully address exploration in continuous control, as discretization of the state-action space scales exponentially in its dimensionality. For example, the Walker2D task [Duan et al., 2016] has a 26-dim state-action space. If we assume a coarse discretization into 10 bins for each dimension, a table of state-action visitation counts would require 10^{26} entries.

This chapter proposes a curiosity-driven exploration strategy, making use of information gain about the agent's internal belief of the dynamics model as a driving force. This principle can be traced back to the concepts of *curiosity* and *surprise* [Schmidhuber, 1991, Sun et al., 2011, Itti and Baldi, 2005]. Within this framework, agents are encouraged to take actions that result in states they deem surprising—i.e., states that cause large updates to the dynamics model distribution. We propose a practical implementation of measuring information gain using variational inference. Herein, the agent's current understanding of the environment dynamics is represented by a Bayesian neural network (BNN) [Graves, 2011, Blundell et al., 2015]. We also show how this can be interpreted as measuring compression improvement, a proposed model of curiosity [Schmidhuber, 2010].

In contrast to previous curiosity-based approaches [Stadie et al., 2015, Storck et al., 1995], our model scales naturally to continuous state and action spaces. The presented approach is evaluated on a range of continuous control tasks, and multiple underlying RL algorithms. Experimental results show that VIME achieves significantly better performance than naïve exploration strategies.

4.2 Methodology

In Section 4.2.1, we establish notation for the subsequent equations. Next, in Section 4.2.2, we explain the theoretical foundation of curiosity-driven exploration. In Section 4.2.3 we describe how to adapt this idea to continuous control, and we show how to build on recent advances in variational inference for Bayesian neural networks (BNNs) to make this formulation practical. Thereafter, we make explicit the intuitive link between compression improvement and the variational lower bound in Section 4.2.4. Finally, Section 4.2.5 describes how our method is practically implemented.

4.2.1 Preliminaries

This chapter assumes a finite-horizon discounted Markov decision process (MDP), defined by $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \rho_0, \gamma, T)$, in which $\mathcal{S} \subseteq \mathbb{R}^n$ is a state set, $\mathcal{A} \subseteq \mathbb{R}^m$ an action set, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_{\geq 0}$ a transition probability distribution, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ a bounded reward function, $\rho_0 : \mathcal{S} \to \mathbb{R}_{\geq 0}$ an initial state distribution, $\gamma \in (0, 1]$ a discount factor, and T the horizon. States and actions viewed as random variables are abbreviated as S and A. The presented models are based on the optimization of a stochastic policy $\pi_\alpha : \mathcal{S} \times \mathcal{A} \to \mathbb{R}_{\geq 0}$, parametrized by α. Let $\mu(\pi_\alpha)$ denote its expected discounted return:

$$\mu(\pi_\alpha) = \mathbb{E}_\tau \left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t) \right], \tag{4.1}$$

where $\tau = (s_0, a_0, \ldots)$ denotes the whole trajectory, with

$$s_0 \sim \rho_0(s_0), \tag{4.2}$$

$$a_t \sim \pi_\alpha(a_t|s_t), \text{ and} \tag{4.3}$$

$$s_{t+1} \sim \mathcal{P}(s_{t+1}|s_t, a_t). \tag{4.4}$$

4.2.2 Curiosity

Our method builds on the theory of curiosity-driven exploration [Schmidhuber, 1991, Schmidhuber, 2010, Storck et al., 1995, Sun et al., 2011], in which the agent engages in systematic exploration by seeking out state-action regions that are relatively unexplored. The agent models the environment dynamics via a model $p(s_{t+1}|s_t, a_t; \theta)$, parametrized by the random variable Θ with values $\theta \in \mathbf{\Theta}$. Assuming a prior $p(\theta)$, it maintains a distribution over dynamic models through a distribution over θ, which is updated in a Bayesian manner (as opposed to a point estimate). The history of the agent up until time step t is denoted as $\xi_t = \{s_1, a_1, \ldots, s_t\}$. According to curiosity-driven exploration [Sun et al., 2011], the agent should take actions that maximize the reduction in uncertainty about the dynamics. This can be formalized as maximizing the sum of reductions in entropy

$$\sum_t \left(H(\Theta|\xi_t, a_t) - H(\Theta|S_{t+1}, \xi_t, a_t) \right), \tag{4.5}$$

through a sequence of actions $\{a_t\}$. According to information theory, the individual terms equal the mutual information between the next state distribution S_{t+1} and the model parameter Θ, namely $I\left(S_{t+1};\Theta|\xi_t,a_t\right)$. Therefore, the agent is encouraged to take actions that lead to states that are maximally informative about the dynamics model. Furthermore, we note that

$$I\left(S_{t+1};\Theta|\xi_t,a_t\right) = \\ \mathbb{E}_{s_{t+1}\sim\mathcal{P}(\cdot|\xi_t,a_t)}\left[D_{\mathrm{KL}}[p(\theta|\xi_t,a_t,s_{t+1})\,\|\,p(\theta|\xi_t)]\right], \quad (4.6)$$

the KL divergence from the agent's new belief over the dynamics model to the old one, taking expectation over all possible next states according to the true dynamics $\mathcal{P}$. This KL divergence can be interpreted as *information gain*.

If calculating the posterior dynamics distribution is tractable, it is possible to optimize Eq. (4.6) directly by maintaining a belief over the dynamics model [Sun et al., 2011]. However, this is not generally the case. Therefore, a common practice [Kolter and Ng, 2009, Stadie et al., 2015] is to use RL to approximate planning for maximal mutual information along a trajectory $\sum_t I\left(S_{t+1};\Theta|\xi_t,a_t\right)$ by adding each term $I\left(S_{t+1};\Theta|\xi_t,a_t\right)$ as an *intrinsic reward*, which captures the agent's surprise in the form of a reward function. This is practically realized by taking actions $a_t \sim \pi_\alpha(s_t)$ and sampling $s_{t+1} \sim \mathcal{P}(\cdot|s_t,a_t)$ in order to add $D_{\mathrm{KL}}[p(\theta|\xi_t,a_t,s_{t+1})\,\|\,p(\theta|\xi_t)]$ to the external reward. The trade-off between exploitation and exploration can now be realized explicitly as follows:

$$r'(s_t,a_t,s_{t+1}) = r(s_t,a_t) + \eta D_{\mathrm{KL}}[p(\theta|\xi_t,a_t,s_{t+1})\,\|\,p(\theta|\xi_t)], \quad (4.7)$$

with $\eta \in \mathbb{R}_+$ a hyperparameter controlling the urge to explore. In conclusion, the biggest practical issue with maximizing information gain for exploration is that the computation of Eq. (4.7) requires calculating the posterior $p(\theta|\xi_t,a_t,s_{t+1})$, which is generally intractable.

4.2.3 Variational Bayes

We propose a tractable solution to maximize the information gain objective presented in the previous section. In a purely Bayesian setting, we can derive the posterior distribution given a new state-action pair through Bayes' rule as

$$p(\theta|\xi_t,a_t,s_{t+1}) = \frac{p(\theta|\xi_t)p(s_{t+1}|\xi_t,a_t;\theta)}{p(s_{t+1}|\xi_t,a_t)}, \quad (4.8)$$

with $p(\theta|\xi_t, a_t) = p(\theta|\xi_t)$ as actions do not influence beliefs about the environment [Sun et al., 2011]. Herein, the denominator is computed through the integral

$$p(s_{t+1}|\xi_t, a_t) = \int_{\mathbf{\Theta}} p(s_{t+1}|\xi_t, a_t; \theta) p(\theta|\xi_t) d\theta. \tag{4.9}$$

In general, this integral tends to be intractable when using highly expressive parametrized models (e.g., neural networks), which are often needed to accurately capture the environment model in high-dimensional continuous control.

We propose a practical solution through variational inference [Hinton and Van Camp, 1993]. Herein, we embrace the fact that calculating the posterior $p(\theta|\mathcal{D})$ for a data set $\mathcal{D}$ is intractable. Instead we approximate it through an alternative distribution $q(\theta;\phi)$, parameterized by ϕ, by minimizing $D_{\mathrm{KL}}[q(\theta;\phi) \,\|\, p(\theta|\mathcal{D})]$. This is done through maximization of the *variational lower bound* $L[q(\theta;\phi), \mathcal{D}]$:

$$L[q(\theta;\phi), \mathcal{D}] = \mathbb{E}_{\theta\sim q(\cdot;\phi)}\left[\log p(\mathcal{D}|\theta)\right] - D_{\mathrm{KL}}[q(\theta;\phi) \,\|\, p(\theta)]. \tag{4.10}$$

Rather than computing information gain in Eq. (4.7) explicitly, we compute an approximation to it, leading to the following total reward:

$$r'(s_t, a_t, s_{t+1}) = r(s_t, a_t) + \eta D_{\mathrm{KL}}[q(\theta;\phi_{t+1}) \,\|\, q(\theta;\phi_t)], \tag{4.11}$$

with ϕ_{t+1} the updated and ϕ_t the old parameters representing the agent's belief. Natural candidates for parametrizing the agent's dynamics model are Bayesian neural networks (BNNs) [Graves, 2011], as they maintain a distribution over their weights. This allows us to view the BNN as an infinite neural network ensemble by integrating out its parameters:

$$p(y|x) = \int_{\mathbf{\Theta}} p(y|x;\theta) q(\theta;\phi) d\theta. \tag{4.12}$$

In particular, we utilize a BNN parametrized by a fully factorized Gaussian distribution [Blundell et al., 2015]. Practical BNN implementation details are deferred to Section 4.2.5, while we give some intuition into the behavior of BNNs in Appendix 4.A.

4.2.4 Compression

It is possible to derive an interesting relationship between compression improvement—an intrinsic reward objective defined in [Schmidhuber, 2007], and the information gain of Eq. (4.6). In [Schmidhuber, 2007], the agent's curiosity is equated with compression improvement, measured through

$$C(\xi_t;\phi_{t-1}) - C(\xi_t;\phi_t), \tag{4.13}$$

where $C(\xi;\phi)$ is the description length of ξ using ϕ as a model. Furthermore, it is known that the negative variational lower bound can be viewed as the description length [Graves, 2011]. Hence, we can write compression improvement as

$$L[q(\theta;\phi_t),\xi_t] - L[q(\theta;\phi_{t-1}),\xi_t]. \tag{4.14}$$

In addition, an alternative formulation of the variational lower bound in Eq. (4.10) is given by

$$\log p(\mathcal{D}) = \overbrace{\int_{\Theta} q(\theta;\phi)\log\frac{p(\theta,\mathcal{D})}{q(\theta;\phi)}d\theta}^{L[q(\theta;\phi),\mathcal{D}]} + D_{\mathrm{KL}}[q(\theta;\phi)\,\|\,p(\theta|\mathcal{D})]. \tag{4.15}$$

Thus, compression improvement can now be written as

$$\begin{aligned}(\log p(\xi_t) - D_{\mathrm{KL}}[q(\theta;\phi_t)\,\|\,p(\theta|\xi_t)]) \\ - (\log p(\xi_t) - D_{\mathrm{KL}}[q(\theta;\phi_{t-1})\,\|\,p(\theta|\xi_t)])\,.\end{aligned} \tag{4.16}$$

If we assume that ϕ_t perfectly optimizes the variational lower bound for the history ξ_t, then $D_{\mathrm{KL}}[q(\theta;\phi_t)\,\|\,p(\theta|\xi_t)] = 0$, which occurs when the approximation equals the true posterior, i.e., $q(\theta;\phi_t) = p(\theta|\xi_t)$. Hence, compression improvement becomes

$$D_{\mathrm{KL}}[p(\theta|\xi_{t-1})\,\|\,p(\theta|\xi_t)]. \tag{4.17}$$

Therefore, optimizing for compression improvement comes down to optimizing the KL divergence from the posterior given the past history ξ_{t-1} to the posterior given the total history ξ_t. As such, we arrive at an alternative way to encode curiosity than information gain, namely

$$D_{\mathrm{KL}}[p(\theta|\xi_t)\,\|\,p(\theta|\xi_t, a_t, s_{t+1})], \tag{4.18}$$

its reversed KL divergence. In experiments, we noticed no significant difference between the two KL divergence variants. This can be explained as both variants are locally equal when introducing small changes to the parameter distributions. Investigation of how to combine both information gain and compression improvement is deferred to future work.

4.2.5 Implementation

The complete method is summarized in Algorithm 4.1. We first set forth implementation and parametrization details of the dynamics BNN. The BNN weight distribution $q(\theta;\phi)$ is given by the fully factorized Gaussian distribution [Blundell et al., 2015]:

$$q(\theta;\phi) = \prod_{i=1}^{|\boldsymbol{\Theta}|} \mathcal{N}\left(\theta_i|\mu_i;\sigma_i^2\right). \tag{4.19}$$

Hence, $\phi = \{\mu, \sigma\}$, with μ the Gaussian's mean vector and σ the covariance matrix diagonal. This is particularly convenient as it allows for a simple analytical formulation of the KL divergence. This is described later in this section. Because of the restriction $\sigma > 0$, the standard deviation of the Gaussian BNN parameter is parametrized as $\sigma = \log(1 + e^{\rho})$, with $\rho \in \mathbb{R}$ [Blundell et al., 2015].

Now the training of the dynamics BNN through optimization of the variational lower bound is described. The second term in Eq. (4.10) is approximated through sampling

$$\mathbb{E}_{\theta\sim q(\cdot;\phi)}\left[\log p(\mathcal{D}|\theta)\right] \approx \frac{1}{N}\sum_{i=1}^{N} \log p(\mathcal{D}|\theta_i), \tag{4.20}$$

with N samples drawn according to $\theta \sim q(\cdot;\phi)$ [Blundell et al., 2015]. Optimizing the variational lower bound in Eq. (4.10) in combination with the reparametrization trick is called stochastic gradient variational Bayes (SGVB) [Kingma et al., 2015] or Bayes by Backprop [Blundell et al., 2015]. Furthermore, we make use of the local reparametrization trick proposed in [Kingma et al., 2015], in which sampling at the weights is replaced by sampling the neuron pre-activations, which is more computationally efficient and reduces gradient variance. The optimization of the variational lower bound is done at regular intervals during the RL training process, by sampling $\mathcal{D}$ from a FIFO replay pool that stores recent samples (s_t, a_t, s_{t+1}). This is to break up the strong intratrajectory sample correlation which destabilizes learning in favor of obtaining i.i.d. data [Mnih et al., 2015]. Moreover, it diminishes the effect of compounding posterior approximation errors.

The posterior distribution of the dynamics parameter, which is needed to compute the KL divergence in the total reward function r' of Eq. (4.11),

Algorithm 4.1: Variational Information Maximizing Exploration

```
1  for each epoch $n$ do
2      for each timestep $t$ in each trajectory generated during $n$ do
3          Generate action $a_t \sim \pi_\alpha(s_t)$ and sample state
             $s_{t+1} \sim \mathcal{P}(\cdot|\xi_t, a_t)$, get $r(s_t, a_t)$.
4          Add triplet $(s_t, a_t, s_{t+1})$ to FIFO replay pool $\mathcal{R}$.
5          Compute intrinsic reward $D_{\mathrm{KL}}[q(\theta;\phi'_{n+1}) \,\|\, q(\theta;\phi_{n+1})]$
             through approximation $\nabla^\top H^{-1}\nabla$, following Eq. (4.26), or
             by optimizing Eq. (4.21) to obtain $\phi'_{n+1}$.
6          Divide $D_{\mathrm{KL}}[q(\theta;\phi'_{n+1}) \,\|\, q(\theta;\phi_{n+1})]$ by median of previous
             KL divergences.
7          Construct
             $r'(s_t, a_t, s_{t+1}) \leftarrow r(s_t, a_t) + \eta D_{\mathrm{KL}}[q(\theta;\phi'_{n+1}) \,\|\, q(\theta;\phi_{n+1})]$,
             following Eq. (4.11).
8      end
9      Minimize $D_{\mathrm{KL}}[q(\theta;\phi_n) \,\|\, p(\theta)] - \mathbb{E}_{\theta\sim q(\cdot;\phi_n)}[\log p(\mathcal{D}|\theta)]$
         following Eq. (4.10), with $\mathcal{D}$ sampled randomly from $\mathcal{R}$,
         leading to updated posterior $q(\theta;\phi_{n+1})$.
10     Use rewards $\{r'(s_t, a_t, s_{t+1})\}$ to update policy $\pi_\alpha$ using any
         standard RL method.
11 end
```

can be computed through the following minimization

$$\phi' = \arg\min_{\phi} \Big[\overbrace{\underbrace{D_{\mathrm{KL}}[q(\theta;\phi) \,\|\, q(\theta;\phi_{t-1})]}_{\ell_{\mathrm{KL}}(q(\theta;\phi))} - \mathbb{E}_{\theta\sim q(\cdot;\phi)}\left[\log p(s_t|\xi_t, a_t;\theta)\right]}^{\ell(q(\theta;\phi), s_t)} \Big], \tag{4.21}$$

where we replace the expectation over θ with samples $\theta \sim q(\cdot;\phi)$. Because we only update the model periodically based on samples drawn from the replay pool, this optimization can be performed in parallel for each s_t, keeping ϕ_{t-1} fixed. Once ϕ' has been obtained, we can use it to compute the intrinsic reward.

To optimize Eq. (4.21) efficiently, we only take a single second-order step. This way, the gradient is rescaled according to the curvature of the KL divergence at the origin. As such, we compute $D_{\mathrm{KL}}[q(\theta;\phi+\lambda\Delta\phi) \,\|\, q(\theta;\phi)]$, with the update step $\Delta\phi$ defined as

$$\Delta\phi = H^{-1}(\ell)\nabla_{\phi}\ell(q(\theta;\phi), s_t), \tag{4.22}$$

in which $H(\ell)$ is the Hessian of $\ell(q(\theta;\phi), s_t)$. Since we assume that the variational approximation is a fully factorized Gaussian, the KL divergence from posterior to prior has a particularly simple form:

$$D_{\mathrm{KL}}[q(\theta;\phi)\,\|\,q(\theta;\phi')] =$$
$$\frac{1}{2}\sum_{i=1}^{|\mathbf{\Theta}|}\left(\left(\frac{\sigma_i}{\sigma'_i}\right)^2 + 2\log\sigma'_i - 2\log\sigma_i + \frac{(\mu'_i-\mu_i)^2}{\sigma'^2_i}\right) - \frac{|\mathbf{\Theta}|}{2}. \tag{4.23}$$

Because this KL divergence is approximately quadratic in its parameters and the log-likelihood term can be seen as locally linear compared to this highly curved KL term, we approximate H by only calculating it for the term KL term $\ell_{\mathrm{KL}}(q(\theta;\phi))$. This can be computed very efficiently in case of a fully factorized Gaussian distribution, as this approximation becomes a diagonal matrix. Looking at Eq. (4.23), we can calculate the following Hessian at the origin. The μ and ρ[1] entries are defined as

$$\frac{\partial^2\ell_{\mathrm{KL}}}{\partial\mu_i^2} = \frac{1}{\log^2(1+e^{\rho_i})} \tag{4.24}$$

$$\frac{\partial^2\ell_{\mathrm{KL}}}{\partial\rho_i^2} = \frac{2e^{2\rho_i}}{(1+e^{\rho_i})^2}\frac{1}{\log^2(1+e^{\rho_i})}, \tag{4.25}$$

while all other entries are zero. Furthermore, it is also possible to approximate the KL divergence through a second-order Taylor expansion as $\frac{1}{2}\Delta\phi H\Delta\phi = \frac{1}{2}\left(H^{-1}\nabla\right)^{\top} H \left(H^{-1}\nabla\right)$, since both the value and gradient of the KL divergence are zero at the origin. This gives us

$$D_{\mathrm{KL}}[q(\theta;\phi+\lambda\Delta\phi)\,\|\,q(\theta;\phi)] \approx \tfrac{1}{2}\lambda^2\nabla_{\phi}\ell^{\top}H^{-1}(\ell_{\mathrm{KL}})\nabla_{\phi}\ell. \tag{4.26}$$

Note that $H^{-1}(\ell_{\mathrm{KL}})$ is diagonal, so this expression can be computed efficiently.

Instead of using the KL divergence $D_{\mathrm{KL}}[q(\theta;\phi_{t+1})\,\|\,q(\theta;\phi_t)]$ directly as an intrinsic reward in Eq. (4.11), we normalize it by division through

[1]Recall that the standard deviation is parametrized as $\sigma = \log(1+e^{\rho})$.

the average of the median[2] KL divergences taken over a fixed number of previous trajectories. Rather than focusing on its absolute value, we emphasize relative difference in KL divergence between samples. This accomplishes the same effect since the variance of KL divergence converges to zero, once the model is fully learned.

4.3 Experiments

In this section, we investigate (i) whether VIME can succeed in domains that have extremely sparse rewards, (ii) whether VIME improves learning when the reward is shaped to guide the agent towards its goal, and (iii) how η, as used in in Eq. (4.7), trades off exploration and exploitation behavior.

All experiments make use of the rllab [Duan et al., 2016] benchmark code base and the complementary continuous control tasks suite. The following tasks are part of the experimental setup: CartPole ($\mathcal{S} \subseteq \mathbb{R}^4$, $\mathcal{A} \subseteq \mathbb{R}^1$), CartPoleSwingup ($\mathcal{S} \subseteq \mathbb{R}^4$, $\mathcal{A} \subseteq \mathbb{R}^1$), DoublePendulum ($\mathcal{S} \subseteq \mathbb{R}^6$, $\mathcal{A} \subseteq \mathbb{R}^1$), MountainCar ($\mathcal{S} \subseteq \mathbb{R}^3$, $\mathcal{A} \subseteq \mathbb{R}^1$), locomotion tasks HalfCheetah ($\mathcal{S} \subseteq \mathbb{R}^{20}$, $\mathcal{A} \subseteq \mathbb{R}^6$), Walker2D ($\mathcal{S} \subseteq \mathbb{R}^{20}$, $\mathcal{A} \subseteq \mathbb{R}^6$), and the hierarchical task SwimmerGather ($\mathcal{S} \subseteq \mathbb{R}^{33}$, $\mathcal{A} \subseteq \mathbb{R}^2$). Performance is measured through the average return (not including the intrinsic rewards) over the trajectories generated (y-axis) at each iteration (x-axis). More specifically, the darker-colored lines in each plot represent the median performance over a fixed set of 10 random seeds while the shaded areas show the interquartile range at each iteration. Moreover, the number in each legend shows this performance measure, averaged over all iterations. The exact experimental setup is described in Appendix 4.B.

Domains with sparse rewards are difficult to solve through naïve exploration behavior because, before the agent obtains any reward, it lacks a feedback signal on how to improve its policy. This allows us to test whether an exploration strategy is truly capable of systematic exploration, rather than improving existing RL algorithms by adding more hyperparameters. Therefore, VIME is compared with heuristic exploration strategies on the following tasks with sparse rewards. A reward of +1 is given when the car escapes the valley on the right side in MountainCar; when the pole is pointed upwards in CartPoleSwingup; and when the cheetah moves forward

[2]The median is used as it is more robust to KL divergence outliers.

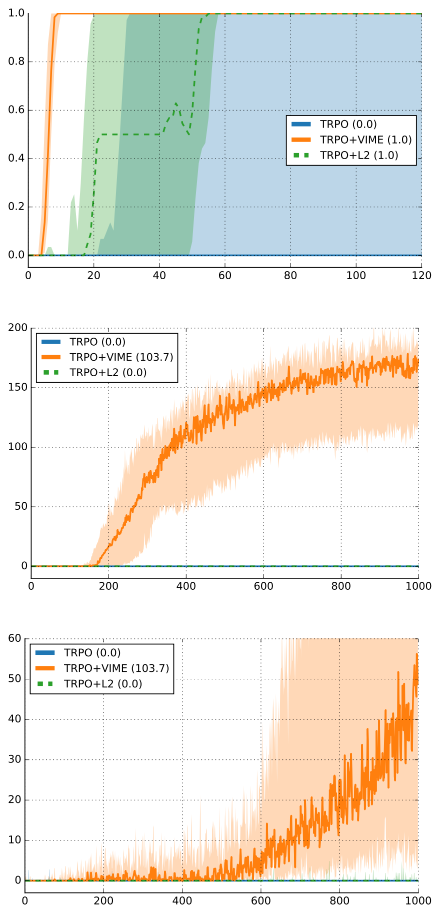
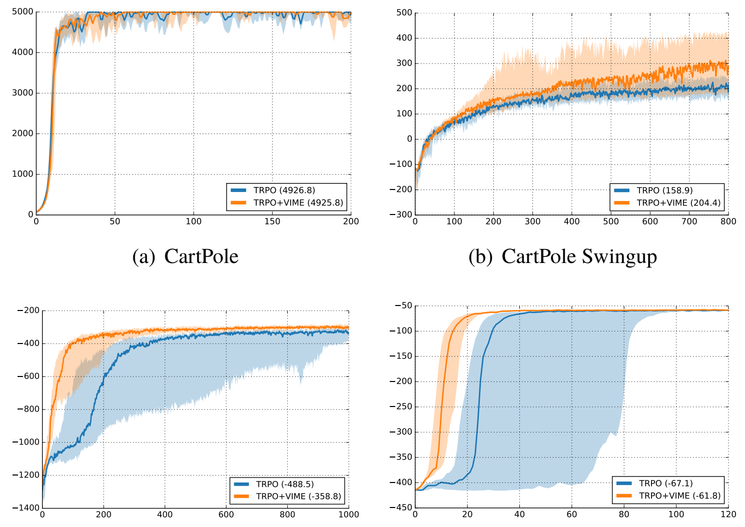


Figure 4.1: TRPO+VIME versus TRPO on tasks with sparse rewards



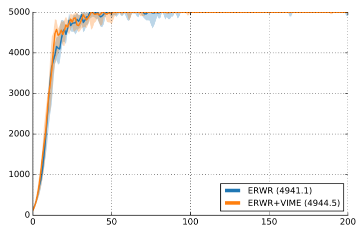
(a) CartPole (b) CartPole Swingup

(c) DoublePendulum (d) MountainCar

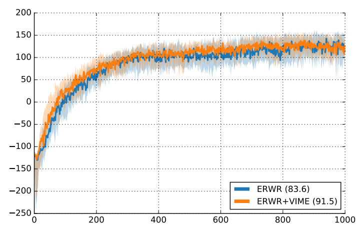
Figure 4.2: Performance of TRPO with (+VIME) and without exploration for different continuous control tasks

over five units in HalfCheetah. We compare VIME with the following baselines: only using Gaussian control noise [Duan et al., 2016] and using the ℓ^2 BNN prediction error as an intrinsic reward, a continuous extension of [Stadie et al., 2015]. TRPO [Schulman et al., 2015] is used as the RL algorithm, as it performs very well compared to other methods [Duan et al., 2016]. Figure 4.1 shows the performance results. We notice that using a naïve exploration performs very poorly, as it is almost never able to reach the goal in any of the tasks. Similarly, using ℓ^2 errors does not perform well. In contrast, VIME performs much better, achieving the goal in most cases. This experiment demonstrates that curiosity drives the agent to explore, even in the absence of *any* initial reward, where naïve exploration completely breaks down.

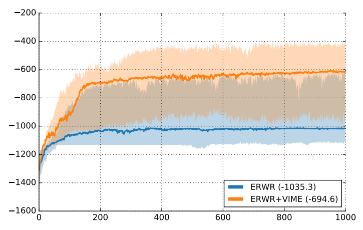
To further strengthen this point, we have evaluated VIME on the highly difficult hierarchical task SwimmerGather in Figure 4.5 whose reward signal is naturally sparse. In this task, a two-link robot needs to reach “apples” while avoiding “bombs” that are perceived through a laser scanner. However,



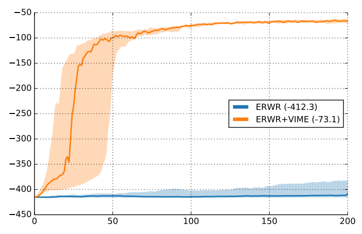
(a) CartPole



(b) CartPole Swingup



(c) DoublePendulum



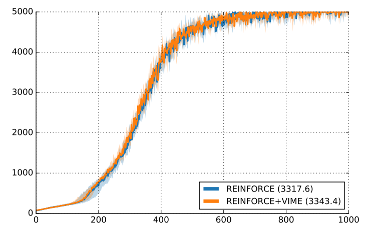
(d) MountainCar

Figure 4.3: Performance of ERWR with (+VIME) and without exploration for different continuous control tasks

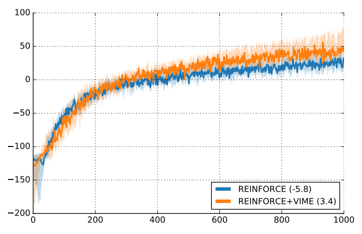
before it can make any forward progress, it has to learn complex locomotion primitives in the absence of any reward. None of the RL methods tested previously in [Duan et al., 2016] were able to make progress with naïve exploration. Remarkably, VIME leads the agent to acquire coherent motion primitives without any reward guidance, achieving promising results on this challenging task.

Next, we investigate whether VIME is widely applicable by (i) testing it on environments where the reward is well shaped, and (ii) pairing it with different RL methods. In addition to TRPO, we choose to equip REINFORCE [Williams, 1992] and ERWR [Kober and Peters, 2009] with VIME because these two algorithms usually suffer from premature convergence to suboptimal policies [Duan et al., 2016, Peters and Schaal, 2007], which can potentially be alleviated by better exploration. A description of these algorithms can be found in Appendix 4.B.2.

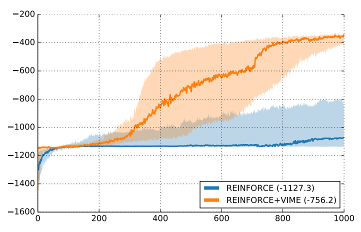
Their performance is shown in Figures 4.2, 4.3, and 4.4 on several well-established continuous control tasks. Furthermore, Figure 4.5 shows



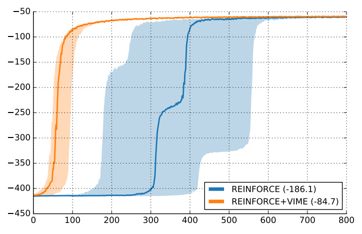
(a) CartPole



(b) CartPoleSwingup



(c) DoublePendulum

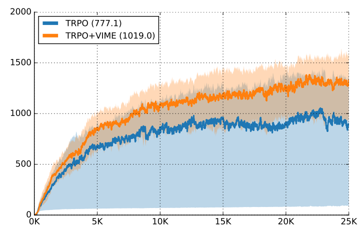


(d) MountainCar

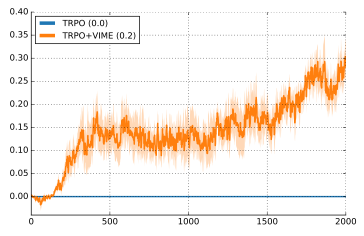
Figure 4.4: Performance of REINFORCE with (+VIME) and without exploration for different continuous control tasks

the same comparison for the Walker2D locomotion task. In the majority of cases, VIME leads to a significant performance gain over heuristic exploration. Our exploration method allows the RL algorithms to converge faster, and notably helps REINFORCE and ERWR avoid converging to a locally optimal solution on DoublePendulum and MountainCar. We note that in environments such as CartPole, a better exploration strategy is redundant as following the policy gradient direction leads to the globally optimal solution. Additionally, we tested adding Gaussian noise to the rewards as a baseline, which did not improve performance.

To give an intuitive understanding of VIME's exploration behavior, the distribution of visited states for both naïve exploration and VIME after convergence is investigated. Figure 4.6 (right) shows that using Gaussian control noise exhibits random walk behavior: the state visitation plot is more condensed and ball-shaped around the center. In comparison, VIME leads to a more diffused visitation pattern, exploring the states more efficiently, and hence reaching the goal more quickly.

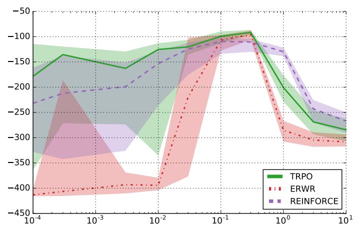


(a) Walker2D



(b) SwimmerGather

Figure 4.5: Performance of TRPO with and without VIME on the high-dimensional Walker2D locomotion task and the hierarchical task SwimmerGather



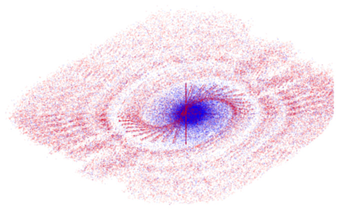


Figure 4.6: VIME: performance over the first few iterations for TRPO, REINFORCE, and ERWR in function of η on MountainCar (left); Comparison of TRPO+VIME (red) and TRPO (blue) on MountainCar: visited states until convergence (right).

Finally, we investigate how η, as used in in Eq. (4.7), trades off exploration and exploitation behavior. On the one hand, higher η values should lead to a higher curiosity drive, causing more exploration. On the other hand, very low η values should reduce VIME to traditional Gaussian control noise. Figure 4.6 (left) shows the performance on MountainCar for different η values. Setting η too high clearly results in prioritizing exploration over getting additional external reward. Too low of an η value reduces the method to the baseline algorithm, as the intrinsic reward contribution to the total reward r' becomes negligible. Most importantly, this figure highlights that there is a wide η range for which the task is best solved, across different algorithms.

4.4 Related Work

A body of theoretically oriented work demonstrates exploration strategies that are able to learn online in a previously unknown MDP and incur a polynomial amount of regret—as a result, these algorithms find a near-optimal policy in a polynomial amount of time. Some of these algorithms are based on the principle of optimism under uncertainty: E^3 [Kearns and Singh, 2002], R-Max [Brafman and Tennenholtz, 2003], UCRL [Jaksch et al., 2010]. An alternative approach is Bayesian reinforcement learning methods, which maintain a distribution over possible MDPs [Kolter and Ng, 2009, Guez et al., 2014, Sun et al., 2011, Ghavamzadeh et al., 2015]. The optimism-based exploration strategies have been extended to continuous state spaces, for example, [Pazis and Parr, 2013, Osband et al., 2016b], however these methods do not accommodate nonlinear function approximators (in contrast to VIME).

Practical RL algorithms often rely on simple exploration heuristics, such as ϵ-greedy and Boltzmann exploration [Sutton, 1998]. However, these heuristics exhibit random walk exploratory behavior, which can lead to exponential regret even in case of small MDPs [Osband et al., 2016b].

Our proposed method of utilizing information gain can be traced back to [Storck et al., 1995], and has been further explored in [Sun et al., 2011, Still and Precup, 2012, Little and Sommer, 2013]. Other metrics for curiosity have also been proposed, including prediction error [Thrun, 1992, Stadie et al., 2015], prediction error improvement [Lopes et al., 2012], and leverage [Subramanian et al., 2016]. All these methods have only been tested on small problems, and are not directly applicable to high dimensional continuous control tasks. We refer the reader to [Schmidhuber, 2010, Oudeyer and Kaplan, 2007] for an extensive review on curiosity and intrinsic rewards.

Recently, there have been various exploration strategies proposed in the context of deep RL. [Stadie et al., 2015] proposes to use the ℓ^2 prediction error as the intrinsic reward. [Oh et al., 2015] performs approximate visitation counting in a learned state embedding using Gaussian kernels. [Osband et al., 2016a] proposes a form of Thompson sampling, training multiple value functions using bootstrapping. Although these approaches can scale up to high-dimensional state spaces, they generally assume discrete action spaces. Finally, [Mohamed and Rezende, 2015] proposes a variational method for information maximization in the context of optimizing *empow-*

erment, which, as noted by [Salge et al., 2014], does not explicitly favor exploration.

4.5 Conclusions

We have proposed Variational Information Maximizing Exploration (VIME), a curiosity-driven exploration strategy for continuous control tasks. Variational inference is used to approximate the posterior distribution of a Bayesian neural network that represents the environment dynamics. Using information gain in this learned dynamics model as intrinsic rewards allows the agent to optimize for both external reward and intrinsic surprise simultaneously. Empirical results show that VIME performs significantly better than heuristic exploration methods across various continuous control tasks and algorithms. As future work, we would like to investigate measuring surprise in the value function and using the learned dynamics model for planning.

4.A Bayesian Neural Networks (BNNs)

We demonstrate the behavior of a BNN [Blundell et al., 2015] when trained on simple regression data. Figure 4.7 shows a snapshot of the behavior of the BNN during training. In this figure, the red dots represent the regression training data, which has a 1-dim input x and a 1-dim output. The input to the BNN is constructed as $x = [x, x^2, x^3, x^4]$. The green dots represent BNN predictions, each for a differently sampled θ value, according to $q(\cdot; \phi)$. The color lines represent the output for different, but fixed, θ samples. The shaded areas represent the sampled output mean plus-minus one and two standard deviations.

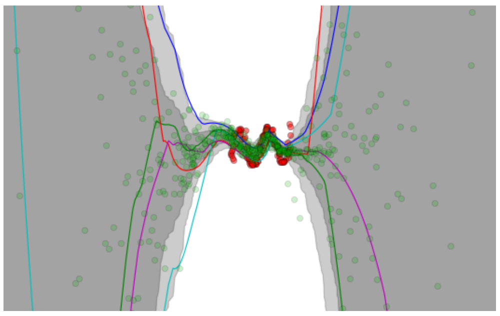


Figure 4.7: BNN output on a 1-dim regression task. Shaded areas: sampled output mean $\pm$ one/two standard deviations. Red dots: targets; green dots: prediction samples. Colored lines: neural network functions for different $\theta \sim q(\cdot; \phi)$ samples.

The figure shows that the BNN output is very certain in the training data range, while having high uncertainty otherwise. If we introduce data outside of this training range, or data that is significantly different from the training data, it will have a high impact on the parameter distribution $q(\theta; \phi)$. This is tested in Figure 4.8: previously unseen data is introduced right before training iteration 10,000. The KL divergence from posterior to prior (y-axis) is set out in function of the training iteration number (x-axis). We see a sharp spike in the KL divergence curve, which represents the BNN's surprise about this novel data. This spike diminishes over time as the BNN learns to fit this new data, becoming less surprised about it.

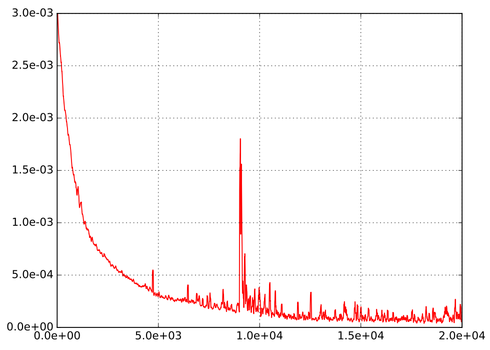


Figure 4.8: Just before iteration 10,000 we introduce data outside the training data range to the BNN. This results in a KL divergence spike, showing the model's surprise.

4.B Experimental Setup

In case of the classic tasks CartPole, CartPoleSwingup, DoublePendulum, and MountainCar, as well as in the case of the hierarchical task SwimmerGather, the dynamics BNN has one hidden layer of 32 units. For the locomotion tasks Walker2D and HalfCheetah, the dynamics BNN has two hidden layers of 64 units each. All hidden layers have rectified linear unit (ReLU) nonlinearities, while no nonlinearity is applied to the output layer. The number of samples drawn to approximate the variational lower bound expectation term is fixed to 10. The batch size for the policy gradient methods is set to 5,000 samples, except for the SwimmerGather task, where it is set to 50,000. The replay pool has a fixed size of 100,000 samples, with a minimum size of 500 samples for all but the SwimmerGather task. In this latter case, the replay pool has a size of 1,000,000 samples. The dynamics BNN is updated each epoch, using 500 iterations of Adam [Kingma and Ba, 2015], with a batch size of 10, except for the SwimmerGather task, in which 5,000 iterations are used. The Adam learning rate is set to 0.0001 while the batches are drawn randomly with replacement from the replay pool. In the second-order KL divergence update step, λ is set to 0.01. The BNN prior weight distribution is a fully factorized Gaussian with μ sampled from a different Gaussian distribution $\mathcal{N}(\mathbf{0}, I)$, while ρ is fixed to $\log(1 + e^{0.5})$.

The classic tasks make use of a neural network policy with one layer of 32 tanh units, while the locomotion tasks make use of a two-layer neural network of 64 and 32 tanh units. The outputs are modeled by a fully factorized Gaussian distribution $\mathcal{N}(\mu, \sigma^2 I)$, in which μ is modeled as the network output, while σ is a parameter. The classic tasks make use of a neural network baseline with one layer of 32 ReLU units, while the locomotion tasks make use linear baseline function.

The tasks have the following state and action dimensions: CartPole, $\mathcal{S} \subseteq \mathbb{R}^4$, $\mathcal{A} \subseteq \mathbb{R}^1$; CartPoleSwingup, $\mathcal{S} \subseteq \mathbb{R}^4$, $\mathcal{A} \subseteq \mathbb{R}^1$; DoublePendulum, $\mathcal{S} \subseteq \mathbb{R}^6$, $\mathcal{A} \subseteq \mathbb{R}^1$; MountainCar $\mathcal{S} \subseteq \mathbb{R}^3$, $\mathcal{A} \subseteq \mathbb{R}^1$; locomotion tasks HalfCheetah, $\mathcal{S} \subseteq \mathbb{R}^{20}$, $\mathcal{A} \subseteq \mathbb{R}^6$; and Walker2D, $\mathcal{S} \subseteq \mathbb{R}^{20}$, $\mathcal{A} \subseteq \mathbb{R}^6$; and hierarchical task SwimmerGather, $\mathcal{S} \subseteq \mathbb{R}^{33}$, $\mathcal{A} \subseteq \mathbb{R}^2$. The time horizon is set to $T = 500$ for all tasks.

4.B.1 Environments

This section describes the environments used in the experiments in detail; taken from [Duan et al., 2016]. They are shown visually in Figure 4.9.

CartPole This classic task in dynamics and control theory has been originally described by [Stephenson, 1908], and first studied in a learning context by [Donaldson, 1960], [Widrow, 1964], and [Michie and Chambers, 1968]. An inverted pendulum is mounted on a pivot point on a cart. The cart itself is restricted to linear movement, achieved by applying horizontal forces. Due to the system's inherent instability, continuous cart movement is needed to keep the pendulum upright. The observation consists of the cart position x, pole angle θ, the cart velocity $\dot{x}$, and the pole velocity $\dot{\theta}$. The 1-dim action consists of the horizontal force applied to the cart body. The reward function is given by $r(s, a) = 10 - (1 - \cos(\theta)) - 10^{-5}\|a\|_2^2$. The episode terminates when $|x| > 2.4$ or $|\theta| > 0.2$.

CartPoleSwingup A slightly more complex version of the previous task has been proposed by [Kimura and Kobayashi, 1999] in which the system should not only be able to balance the pole, but first succeed in swinging it up into an upright position. This tasks extends the working range of the inverted pendulum to 360°. This is a nonlinear extension of the previous task [Doya, 2000]. The same observations and actions as in CartPole are used. The reward function is given by $r(s, a) = \cos(\theta)$. The episode terminates when $|x| > 3$, with a penalty of -100.

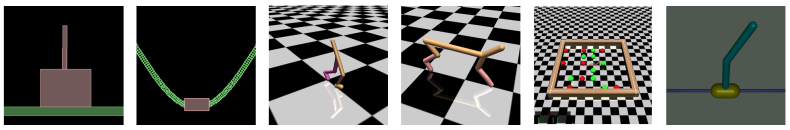


Figure 4.9: Illustrations of the rllab tasks used in the experiments; from left to right: CartPole, MountainCar, Walker2D, HalfCheetah, Swimmer-Gather, DoublePendulum

MountainCar We implement a continuous version of the classic task described by [Moore, 1990]. A car has to escape a valley by repetitive application of tangential forces. Because the maximal tangential force is limited, the car has to alternately drive up along the two slopes of the valley in order to build up enough inertia to overcome gravity. This brings a challenge of exploration, since before first reaching the goal among all trials, a locally optimal solution exists, which is to drive to the point closest to the target and stay there for the rest of the episode. The observation is given by the horizontal position x and the horizontal velocity $\dot{x}$ of the car. The reward is given by $r(s, a) = -1 + x_{\text{height}}$, with x_{height} the car's vertical offset. The episode terminates when the car reaches a target height of 0.6. Hence the goal is to reach the target as soon as possible.

DoublePendulum This task extends the CartPole task by replacing the single-link pole by a two-link rigid structure. Similar to this former task, the goal is to stabilize the two-link pole near the upright position. This task is more difficult than single-pole balancing, since the system is even more unstable and requires the controller to actively maintain balance [Furuta et al., 1978]. The observation includes the cart position x, joint angles (θ_1 and θ_2), and joint velocities ($\dot{\theta}_1$ and $\dot{\theta}_2$). We encode each joint angle as its sine and cosine values. The action is the same as in cart-pole tasks. The reward is given by $r(s, a) = 10 - 0.01x_{\text{tip}}^2 - (y_{\text{tip}} - 2)^2 - 10^{-3} \cdot \dot{\theta}_1^2 - 5 \cdot 10^{-3} \cdot \dot{\theta}_2^2$, where $x_{\text{tip}}, y_{\text{tip}}$ are the coordinates of the tip of the pole. The episode is terminated when $y_{\text{tip}} \leq 1$.

The following two tasks are more challenging than the basic tasks due to high degrees of freedom. In addition, a great amount of exploration is needed to learn to move forward without getting stuck at local optima. Since we put penalty on excessive controls as well as falling over, during the initial stage of learning, when the robot is not yet able to move forward for a sufficient distance without falling, apparent local optima exist including staying at the origin or diving forward cautiously.

Walker2D [Levine and Koltun, 2013, Schulman et al., 2015] The walker is a planar biped robot consisting of 7 links, corresponding to two legs and a torso, along with 6 actuated joints. The 21-dim observation includes joint angles, joint velocities, and the coordinates of center of mass. The reward is given by $r(s, a) = v_x - 0.005 \cdot \|a\|_2^2$. The episode is terminated when $z_{\text{body}} < 0.8$, $z_{\text{body}} > 2.0$, or when $|\theta_y| > 1.0$.

HalfCheetah [Wawrzyński, 2007, Heess et al., 2015] The half-cheetah is a planar biped robot with 9 rigid links, including two legs and a torso, along with 6 actuated joints. The 20-dim observation includes joint angles, joint velocities, and the coordinates of the center of mass. The reward is given by $r(s, a) = v_x - 0.05 \cdot \|a\|_2^2$. No termination condition is applied.

Many real-world tasks exhibit hierarchical structure, where higher level decisions can reuse lower level skills. For instance, robots can reuse locomotion skills when exploring the environment [Dietterich, 2000]. The following task has hierarchical structure.

SwimmerGather For this task, the agent needs to learn to control the swimmer robot to collect food and avoid bombs in a finite region. The agent receives range sensor readings about nearby food and bomb units. It is given a positive reward when it reaches a food unit, or a negative reward when it reaches a bomb.

Sparse reward modifications In MountainCar, the agent receives a reward of +1 when the goal state is reached, namely escaping the valley from the right side. In CartPoleSwingup, the agent receives a reward of +1 when $\cos(\beta) > 0.8$, with β the pole angle. Therefore, the agent has to figure out how to swing up the pole in the absence of any initial external rewards. In HalfCheetah, the agent receives a reward of +1 when $x_{\text{body}} > 5$. As such, it has to figure out how to move forward without any initial external reward.

4.B.2 Reinforcement Learning Algorithms

This section describes the different reinforcement learning algorithms used in the experiments; taken from [Duan et al., 2016].

REINFORCE This algorithm [Williams, 1992] estimates the gradient of expected return $\nabla_\alpha \mu(\pi_\alpha)$ using the likelihood ratio trick:

$$\widehat{\nabla_\alpha \mu(\pi_\alpha)} = \frac{1}{NT} \sum_{i=1}^{N} \sum_{t=0}^{T} \nabla_\alpha \log \pi_\alpha \left(a_t^i | s_t^i\right) \left(R_t^i - b_t^i\right), \tag{4.27}$$

where $R_t^i = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}^i$ and b_t^i is a baseline that only depends on the state s_t^i to reduce variance. Hereafter, an ascent step is taken in the direction of the estimated gradient. This process continues until α_k converges.

Episodic reward-weighted regression (ERWR) This algorithm [Kober and Peters, 2009] formulates the policy optimization as an Expectation-Maximization problem to avoid the need to manually choose learning rate, and the method is guaranteed to converge to a locally optimal solution. At each iteration, this algorithm optimizes a lower bound of the log-expected return: $\alpha = \arg\max_{\alpha'} \mathcal{L}(\alpha')$, where

$$\mathcal{L}(\alpha) = \frac{1}{NT} \sum_{i=1}^{N} \sum_{t=0}^{T} \log \pi_\alpha \left(a_t^i | s_t^i\right) \rho\left(R_t^i - b_t^i\right). \tag{4.28}$$

Here, $\rho : \mathbb{R} \rightarrow \mathbb{R}_{\geq 0}$ is a function that transforms raw returns to nonnegative values. Following [Deisenroth et al., 2013], we choose ρ to be $\rho(R) = R - R_{\min}$, where $R_{\min}$ is the minimum return among all trajectories collected in the current iteration.

Trust Region Policy Optimization (TRPO) This algorithm [Schulman et al., 2015] allows more precise control on the expected policy improvement than using natural policy gradients through the introduction of a surrogate loss. At each iteration, we solve the following constrained optimization problem (replacing expectations with samples):

$$\begin{aligned} \text{maximize}_\alpha \quad & \mathbb{E}_{s \sim \rho_{\alpha_k}, a \sim \pi_{\alpha_k}} \left[\frac{\pi_\alpha(a|s)}{\pi_{\alpha_k}(a|s)} A_{\alpha_k}(s, a) \right] \\ \text{s.t.} \quad & \mathbb{E}_{s \sim \rho_{\alpha_k}} \left[D_{\mathrm{KL}}(\pi_{\alpha_k}(\cdot|s) \| \pi_\alpha(\cdot|s)) \right] \leq \delta_{\mathrm{KL}}, \end{aligned} \tag{4.29}$$

where $\rho_\alpha = \rho_{\pi_\alpha}$ is the discounted state-visitation frequencies induced by π_α, $A_{\alpha_k}(s, a)$ is estimated by the empirical return minus the baseline, and δ_{KL} is a step size parameter which controls how much the policy is allowed to change per iteration.

References

[Auer, 2003] Auer, P. (2003). Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3:397–422.

[Blundell et al., 2015] Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 1613–1622.

[Brafman and Tennenholtz, 2003] Brafman, R. I. and Tennenholtz, M. (2003). R-Max - a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3:213–231.

[Deisenroth et al., 2013] Deisenroth, M. P., Neumann, G., and Peters, J. (2013). A survey on policy search for robotics, foundations and trends in robotics. *Foundations and Trends in Robotics*, 2(1-2):1–142.

[Dietterich, 2000] Dietterich, T. G. (2000). Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303.

[Donaldson, 1960] Donaldson, P. E. K. (1960). Error decorrelation: a technique for matching a class of functions. In *Proceedings of the 3th International Conference on Medical Electronics*, pages 173–178.

[Doya, 2000] Doya, K. (2000). Reinforcement learning in continuous time and space. *Neural Computation*, 12(1):219–245.

[Duan et al., 2016] Duan, Y., Chen, X., Houthooft, R., Schulman, J., and Abbeel, P. (2016). Benchmarking deep reinforcement learning for continous control. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 1329–1338.

[Furuta et al., 1978] Furuta, K., Okutani, T., and Sone, H. (1978). Computer control of a double inverted pendulum. *Computers & Electrical Engineering*, 5(1):67–84.

[Ghavamzadeh et al., 2015] Ghavamzadeh, M., Mannor, S., Pineau, J., and Tamar, A. (2015). Bayesian reinforcement learning: A survey. *Foundations and Trends in Machine Learning*, 8(5-6):359–483.

[Graves, 2011] Graves, A. (2011). Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2348–2356.

[Guez et al., 2014] Guez, A., Heess, N., Silver, D., and Dayan, P. (2014). Bayes-adaptive simulation-based search with value function approximation. In *Advances in Neural Information Processing Systems 27 (NIPS)*, pages 451–459.

[Heess et al., 2015] Heess, N., Wayne, G., Silver, D., Lillicrap, T., Erez, T., and Tassa, T. (2015). Learning continuous control policies by stochastic value gradients. In *Advances in Neural Information Processing Systems 28 (NIPS)*, pages 2926–2934.

[Hester and Stone, 2015] Hester, T. and Stone, P. (2015). Intrinsically motivated model learning for developing curious robots. *Artificial Intelligence*.

[Hinton and Van Camp, 1993] Hinton, G. E. and Van Camp, D. (1993). Keeping the neural networks simple by minimizing the description length of the weights. In *COLT*, pages 5–13.

[Itti and Baldi, 2005] Itti, L. and Baldi, P. F. (2005). Bayesian surprise attracts human attention. In *Advances in Neural Information Processing Systems (NIPS)*, pages 547–554.

[Jaksch et al., 2010] Jaksch, T., Ortner, R., and Auer, P. (2010). Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11:1563–1600.

[Kakade et al., 2003] Kakade, S. M., Kearns, M. J., and Langford, J. (2003). Exploration in Metric State Spaces. *Proceedings of the 20th International Conference on Machine Learning (ICML)*, pages 306–312.

[Kearns and Singh, 2002] Kearns, M. and Singh, S. (2002). Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2):209–232.

[Kimura and Kobayashi, 1999] Kimura, H. and Kobayashi, S. (1999). Stochastic real-valued reinforcement learning to solve a nonlinear control problem. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 510–515.

[Kingma and Ba, 2015] Kingma, D. and Ba, J. (2015). Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

[Kingma et al., 2015] Kingma, D. P., Salimans, T., and Welling, M. (2015). Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2575–2583.

[Kober and Peters, 2009] Kober, J. and Peters, J. R. (2009). Policy search for motor primitives in robotics. In *Advances in Neural Information Processing Systems 22 (NIPS)*, pages 849–856.

[Kolter and Ng, 2009] Kolter, J. Z. and Ng, A. Y. (2009). Near-Bayesian exploration in polynomial time. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, pages 513–520.

[Levine and Koltun, 2013] Levine, S. and Koltun, V. (2013). Guided policy search. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 1–9.

[Little and Sommer, 2013] Little, D. Y. and Sommer, F. T. (2013). Learning and exploration in action-perception loops. *Frontiers in Neural Circuits*, 7.

[Lopes et al., 2012] Lopes, M., Lang, T., Toussaint, M., and Oudeyer, P.-Y. (2012). Exploration in model-based reinforcement learning by empirically estimating learning progress. In *Advances in Neural Information Processing Systems 25 (NIPS)*, pages 206–214.

[Michie and Chambers, 1968] Michie, D. and Chambers, R. A. (1968). BOXES: An experiment in adaptive control. *Machine Intelligence*, 2:137–152.

[Mnih et al., 2015] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland,

A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

[Mohamed and Rezende, 2015] Mohamed, S. and Rezende, D. J. (2015). Variational information maximisation for intrinsically motivated reinforcement learning. In *Advances in Neural Information Processing Systems 28 (NIPS)*, pages 2116–2124.

[Moore, 1990] Moore, A. (1990). Efficient memory-based learning for robot control. Technical report, University of Cambridge, Computer Laboratory.

[Oh et al., 2015] Oh, J., Guo, X., Lee, H., Lewis, R. L., and Singh, S. (2015). Action-conditional video prediction using deep networks in Atari games. In *Advances in Neural Information Processing Systems 28 (NIPS)*, pages 2845–2853.

[Osband et al., 2016a] Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. (2016a). Deep exploration via bootstrapped DQN. In *Advances in Neural Information Processing Systems 29 (NIPS)*, pages 4026–4034.

[Osband et al., 2016b] Osband, I., Van Roy, B., and Wen, Z. (2016b). Generalization and exploration via randomized value functions. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 2377–2386.

[Oudeyer and Kaplan, 2007] Oudeyer, P.-Y. and Kaplan, F. (2007). What is intrinsic motivation? A typology of computational approaches. *Frontiers in Neurorobotics*, 1:6.

[Pazis and Parr, 2013] Pazis, J. and Parr, R. (2013). PAC optimal exploration in continuous space Markov decision processes. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI)*.

[Peters and Schaal, 2007] Peters, J. and Schaal, S. (2007). Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, pages 745–750.

[Salge et al., 2014] Salge, C., Glackin, C., and Polani, D. (2014). *Empowerment–An Introduction*, pages 67–114.

[Schmidhuber, 1991] Schmidhuber, J. (1991). Curious model-building control systems. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pages 1458–1463.

[Schmidhuber, 2007] Schmidhuber, J. (2007). Simple algorithmic principles of discovery, subjective beauty, selective attention, curiosity & creativity. In *Proceedings of the 10th International Conference on Discovery Science (DS)*, pages 26–38.

[Schmidhuber, 2010] Schmidhuber, J. (2010). Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247.

[Schulman et al., 2015] Schulman, J., Levine, S., Moritz, P., Jordan, M. I., and Abbeel, P. (2015). Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 1889–1897.

[Stadie et al., 2015] Stadie, B. C., Levine, S., and Abbeel, P. (2015). Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*.

[Stephenson, 1908] Stephenson, A. (1908). On induced stability. *Philosophical Magazine*, 15(86):233–236.

[Still and Precup, 2012] Still, S. and Precup, D. (2012). An information-theoretic approach to curiosity-driven reinforcement learning. *Theory in Biosciences*, 131(3):139–148.

[Storck et al., 1995] Storck, J., Hochreiter, S., and Schmidhuber, J. (1995). Reinforcement driven information acquisition in non-deterministic environments. In *Proceedings of the 5th International Conference on Artificial Neural Networks (ICANN)*, pages 159–164.

[Subramanian et al., 2016] Subramanian, K., Isbell Jr, C. L., and Thomaz, A. L. (2016). Exploration from demonstration for interactive reinforcement learning. In *Proceedings of the International Conference on Autonomous Agents & Multiagent Systems (AAMAS)*, pages 447–456.

[Sun et al., 2011] Sun, Y., Gomez, F., and Schmidhuber, J. (2011). Planning to be surprised: Optimal Bayesian exploration in dynamic environments. In *Proceedings of the 4th International Conference on Artificial General Intelligence (AGI)*, pages 41–51.

[Sutton, 1998] Sutton, R. S. (1998). *Introduction to reinforcement learning*. The MIT Press.

[Thrun, 1992] Thrun, S. B. (1992). Efficient exploration in reinforcement learning. Technical report.

[Wawrzyński, 2007] Wawrzyński, P. (2007). Learning to control a 6-degree-of-freedom walking robot. In *Proceedings of the International Conference on Computer as a Tool (EUROCON)*, pages 698–705.

[Widrow, 1964] Widrow, B. (1964). Pattern recognition and adaptive control. *IEEE Transactions on Industry Applications*, 83(74):269–277.

[Williams, 1992] Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256.

Chapter 5

A Study of Count-Based Exploration for Deep Reinforcement Learning

Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip De Turck & Pieter Abbeel, *#Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning*, Deep Reinforcement Learning Workshop at Neural Information Processing Systems (NIPS), 2016

UC Berkeley, OpenAI, Ghent University – imec

⋆ ⋆ ⋆

Count-based exploration algorithms are known to perform near-optimally when used in conjunction with tabular reinforcement learning (RL) methods for solving small discrete Markov decision processes (MDPs). It is generally thought that count-based methods cannot be applied in high-dimensional state spaces, since most states will only occur once. Recent deep RL exploration strategies are able to deal with high-dimensional continuous state spaces through complex heuristics, often relying on optimism in the face *of uncertainty or intrinsic motivation. In this work, we describe a surprising finding: a simple generalization of the classic count-based approach can reach near state-of-the-art performance on various high-dimensional and/or continuous deep RL benchmarks. States are mapped to hash codes, which allows to count their occurrences with a hash table. These counts are then used as a reward bonus, according to the classic count-based exploration theory. We find that simple hash functions can achieve surprisingly good results on many challenging tasks. Furthermore, we show that a domain-dependent learned hash code may further improve these results. Detailed analysis reveals important aspects of a good hash function: i) Having appropriate granularity; ii) Encoding information relevant to solving the MDP. This exploration strategy achieves near state-of-the-art performance on both continuous control tasks and Atari 2600 games, while providing a simple yet powerful baseline for solving MDPs that require considerable exploration.*

5.1 Introduction

Reinforcement learning (RL) studies an agent acting in an initially unknown environment, learning through trial and error to maximize rewards. It is impossible for the agent to act near-optimally until it has sufficiently explored the environment and identified all of the opportunities for high reward, in all scenarios. A core challenge in RL is how to balance exploration—actively seeking out novel states and actions that might yield high rewards and lead to long-term gains; and exploitation—maximizing short-term rewards using the agent's current knowledge. While there are exploration techniques for finite MDPs that enjoy theoretical guarantees, there are no fully satisfying techniques for high-dimensional state spaces; therefore, developing more general and robust exploration techniques is an active area of research.

Most of the recent state-of-the-art RL results have been obtained using simple exploration strategies such as uniform sampling [Mnih et al., 2015] and i.i.d./correlated Gaussian noise [Schulman et al., 2015, Lillicrap et al., 2015]. Although these heuristics are sufficient in tasks with well-shaped rewards, the sample complexity can grow exponentially (with state space size) in tasks with sparse rewards [Osband et al., 2016b]. Recently developed exploration strategies for deep RL have led to significantly improved performance on environments with sparse rewards. Bootstrapped DQN [Osband et al., 2016a] led to faster learning in a range of Atari 2600 games by training an ensemble of Q-functions. Intrinsic motivation methods using pseudo-counts achieve state-of-the-art performance on Montezuma's Revenge, an extremely challenging Atari 2600 game [Bellemare et al., 2016]. Variational Information Maximizing Exploration (VIME, [Houthooft et al., 2016]) encourages the agent to explore by acquiring information about environment dynamics, and performs well on various robotic locomotion problems with sparse rewards. However, we have not seen a very simple and fast method that can work across different domains.

Some of the classic, theoretically-justified exploration methods are based on counting state-action visitations, and turning this count into a bonus reward. In the bandit setting, the well-known UCB algorithm of [Lai and Robbins, 1985] chooses the action a_t at time t that maximizes $\hat{r}(a_t) + \sqrt{\frac{2\log t}{n(a_t)}}$ where $\hat{r}(a_t)$ is the estimated reward, and $n(a_t)$ is the number of times action a_t was previously chosen. In the MDP setting, some of the algorithms have similar structure, for example, Model Based Interval Estimation–Exploration Bonus (MBIE-EB) of [Strehl and Littman, 2008] counts state-action pairs with a table $n(s,a)$ and adding a bonus reward of the form $\frac{\beta}{\sqrt{n(s,a)}}$ to encourage exploring less visited pairs. [Kolter and Ng, 2009] show that the inverse-square-root dependence is optimal. MBIE and related algorithms assume that the augmented MDP is solved analytically at each timestep, which is only practical for small finite state spaces.

This chapter presents a simple approach for exploration, which extends classic counting-based methods to high-dimensional, continuous state spaces. We discretize the state space with a hash function and apply a bonus based on the state-visitation count. The hash function can be chosen to appropriately balance generalization across states, and distinguishing between states. We select problems from rllab [Duan et al., 2016] and Atari 2600 [Bellemare et al., 2013] featuring sparse rewards, and demonstrate near state-of-the-art performance on several games known to be hard for naïve exploration strategies. The main strength of the presented approach is that it is fast, flexible and complementary to most existing RL algorithms.

In summary, this chapter proposes a generalization of classic count-based exploration to high-dimensional spaces through hashing (Section 5.2); demonstrates its effectiveness on challenging deep RL benchmark problems and analyzes key components of well-designed hash functions (Section 5.3).

5.2 Methodology

First the notation used throughout this chapter is described. Next, we introduce count-based exploration through static hashing, after which count-based exploration through learned hashing is set forth.

5.2.1 Notation

This chapter assumes a finite-horizon discounted Markov decision process (MDP), defined by $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \rho_0, \gamma, T)$, in which $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, $\mathcal{P}$ a transition probability distribution, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ a reward function, ρ_0 an initial state distribution, $\gamma \in (0, 1]$ a discount factor, and T the horizon. The goal of RL is to maximize the total expected discounted reward $\mathbb{E}_{\pi,\mathcal{P}} \left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t)\right]$ over a policy π, which outputs a distribution over actions given a state.

5.2.2 Count-Based Exploration via Static Hashing

Our approach discretizes the state space with a hash function $\phi : \mathcal{S} \to \mathbb{Z}$. An exploration bonus is added to the reward function, defined as

$$r^+(s, a) = \frac{\beta}{\sqrt{n(\phi(s))}}, \tag{5.1}$$

where $\beta \in \mathbb{R}_{\geq 0}$ is the bonus coefficient. Initially the counts $n(\cdot)$ are set to zero for the whole range of ϕ. For every state s_t encountered at time step t, $n(\phi(s_t))$ is increased by one. The agent is trained with rewards $(r + r^+)$, while performance is evaluated as the sum of rewards without bonuses.

Note that our approach is a departure from count-based exploration methods such as MBIE-EB since we use a state-space count $n(s)$ rather than a state-action count $n(s, a)$. State-action counts $n(s, a)$ are investigated in Appendix 5.D, but no significant performance gains over state counting could be witnessed.

Clearly the performance of this method will strongly depend on the choice of hash function ϕ. One important choice we can make regards the *granularity* of the discretization: we would like for "distant" states to be be counted separately while "similar" states are merged. If desired, we can incorporate prior knowledge into the choice of ϕ, if there would be a set of salient state features which are known to be relevant.

Algorithm 5.1: Count-based exploration through static hashing

```
1 Define state preprocessor $g : \mathcal{S} \to \mathbb{R}^D$
2 (In case of SimHash) Initialize $A \in \mathbb{R}^{k \times D}$ with entries drawn i.i.d.
    from the standard Gaussian distribution $\mathcal{N}(0, 1)$
3 Initialize a hash table with values $n(\cdot) \equiv 0$
4 for each iteration $j$ do
5     Collect a set of state-action samples $\{(s_m, a_m)\}_{m=0}^{M}$ with policy $\pi$
6     Compute hash codes through any LSH method, e.g., for
        SimHash, $\phi(s_m) = \operatorname{sgn}(Ag(s_m))$
7     Update the hash table counts $\forall m : 0 \le m \le M$ as
        $n(\phi(s_m)) \leftarrow n(\phi(s_m)) + 1$
8     Update the policy $\pi$ using rewards $\left\{ r(s_m, a_m) + \frac{\beta}{\sqrt{n(\phi(s_m))}} \right\}_{m=0}^{M}$
        with any RL algorithm
9 end
```

Algorithm 5.1 summarizes our method. The main idea is to use locality-sensitive hashing (LSH) to convert continuous, high-dimensional data to discrete hash codes. LSH is a popular class of hash functions for querying nearest neighbors based on certain similarity metrics [Andoni and Indyk, 2006]. A computationally efficient type of LSH is SimHash [Charikar, 2002], which measures similarity by angular distance. SimHash retrieves a binary code of state $s \in \mathcal{S}$ as

$$\phi(s) = \operatorname{sgn}(Ag(s)) \in \{-1, 1\}^k, \tag{5.2}$$

where $g : \mathcal{S} \to \mathbb{R}^D$ is an optional preprocessing function and A is a $k \times D$ matrix with i.i.d. entries drawn from a standard Gaussian distribution $\mathcal{N}(0, 1)$. The value for k controls the granularity: higher values lead to fewer collisions and are thus more likely to distinguish states.

5.2.3 Count-Based Exploration via Learned Hashing

When the MDP states have a complex structure, as is the case with image observations, measuring their similarity directly in pixel space fails to provide the semantic similarity measure one would desire. Previous work in computer vision [Lowe, 1999, Dalal and Triggs, 2005, Tola et al., 2010]

Algorithm 5.2: Count-based exploration using learned hash codes

```
1  Define state preprocessor $g : \mathcal{S} \to \{0,1\}^D$ as the binary code
     resulting from the autoencoder (AE)
2  Initialize $A \in \mathbb{R}^{k \times D}$ with entries drawn i.i.d. from the standard
     Gaussian distribution $\mathcal{N}(0,1)$
3  Initialize a hash table with values $n(\cdot) \equiv 0$
4  for each iteration $j$ do
5      Collect a set of state-action samples $\{(s_m, a_m)\}_{m=0}^{M}$ with
         policy $\pi$
6      Add the state samples $\{s_m\}_{m=0}^{M}$ to a FIFO replay pool $\mathcal{R}$
7      if $j$ mod $j_{\text{update}} = 0$ then
8          Update the AE loss function in Eq. (5.3) using samples
             drawn from the replay pool $\{s_n\}_{n=1}^{N} \sim \mathcal{R}$, for example
             using stochastic gradient descent
9      end
10     Compute $g(s_m) = \lfloor b(s_m) \rceil$, the $D$-dim rounded hash code for
         $s_m$ learned by the AE
11     Project $g(s_m)$ to a lower dimension $k$ via SimHash as
         $\phi(s_m) = \operatorname{sgn}(Ag(s_m))$
12     Update the hash table counts $\forall m : 0 \le m \le M$ as
         $n(\phi(s_m)) \leftarrow n(\phi(s_m)) + 1$
13     Update the policy $\pi$ using rewards
         $\left\{ r(s_m, a_m) + \frac{\beta}{\sqrt{n(\phi(s_m))}} \right\}_{m=0}^{M}$ with any RL algorithm
14 end
```

introduce manually designed feature representations of images that are suitable for semantic tasks including detection and classification. More recent methods learn complex features directly from data by training convolutional neural networks [Krizhevsky et al., 2012, Simonyan and Zisserman, 2014, He et al., 2015]. Considering these results, it may be difficult for SimHash to cluster states appropriately using only raw pixels.

When the MDP states have a complex structure, as is the case with image observations, measuring their similarity directly in pixel space fails to provide the semantic similarity measure one would desire. Previous work in computer vision [Lowe, 1999, Dalal and Triggs, 2005, Tola et al., 2010] introduce manually designed feature representations of images that are suit-

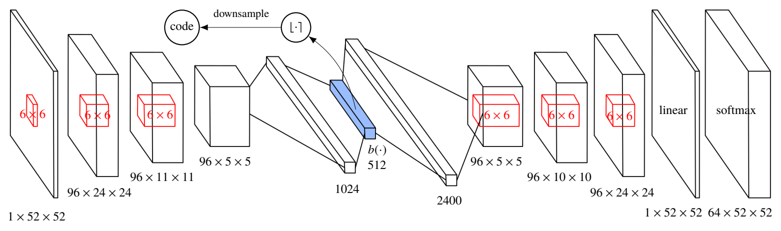


Figure 5.1: The autoencoder (AE) architecture; the solid block represents the dense sigmoidal binary code layer, after which noise $U(-a, a)$ is injected.

able for semantic tasks including detection and classification. More recent methods learn complex features directly from data by training convolutional neural networks [Krizhevsky et al., 2012, Simonyan and Zisserman, 2014, He et al., 2015]. Considering these results, it may be difficult for a method such as SimHash to cluster states appropriately using only raw pixels.

Therefore, rather than using SimHash, we propose to use an autoencoder (AE) to learn meaningful hash codes in one of its hidden layers as a more advanced LSH method. This AE takes as input states s and contains one special dense layer comprised of D sigmoid functions. By rounding the sigmoid activations $b(s)$ of this layer to their closest binary number $\lfloor b(s) \rceil \in \{0, 1\}^D$, any state s can be binarized. This is illustrated in Figure 5.1 for a convolutional AE.

As such, the loss function over a set of collected states $\{s_i\}_{i=1}^N$ is defined as

$$L\left(\{s_n\}_{n=1}^N\right) = -\frac{1}{N} \sum_{n=1}^{N} \Big[\log p(s_n) \\ - \frac{\lambda}{K} \sum_{i=1}^{D} \min \left\{ (1 - b_i(s_n))^2, b_i(s_n)^2 \right\} \Big]. \quad (5.3)$$

This objective function consists of a negative log-likelihood term and a term that pressures the binary code layer to take on binary values, scaled by $\lambda \in \mathbb{R}_{\geq 0}$. The reasoning behind this latter term is that it might happen that for particular states, a certain sigmoid unit is never used. Therefore, its value might fluctuate around $\frac{1}{2}$, causing the corresponding bit in binary code $\lfloor b(s) \rceil$ to flip over the agent lifetime. Adding this second loss term ensures that an unused bit takes on an arbitrary binary value.

For Atari 2600 image inputs, since the pixel intensities are discrete values in the range $[0, 255]$, we make use of a pixel-wise softmax output layer [van den Oord et al., 2016] that shares weights between all pixels. The architectural details are described in the Supplementary Material and are depicted in Figure 5.1. Because the code dimension often needs to be large in order to correctly reconstruct the input, we apply a downsampling procedure to the resulting binary code $\lfloor b(s) \rceil$, which can be done through random projection to a lower-dimensional space via SimHash as in Eq. (5.2).

On the one hand, it is important that the mapping from state to code needs to remain relatively consistent over time, which is nontrivial as the AE is constantly updated according to the latest data (Algorithm 5.2 line 8). A solution is to downsample the binary code to a very low dimension, or by slowing down the training process. On the other hand, the code has to remain relatively unique for states that are both distinct and close together on the image manifold. This is tackled both by the second term in Eq. (5.3) and by the saturating behavior of the sigmoid units. States already well represented by the AE tend to saturate the sigmoid activations, causing the resulting loss gradients to be close to zero, making the code less prone to change.

5.3 Experiments

Experiments were designed to investigate and answer the following research questions:

1. Can count-based exploration through hashing improve performance significantly across different domains? How does the proposed method compare to the current state of the art in exploration for deep RL?

2. What is the impact of learned or static state preprocessing on the overall performance when image observations are used?

To answer question 1, we run the proposed method on deep RL benchmarks (rllab and ALE) that feature sparse rewards, and compare it to other state-of-the-art algorithms. Question 2 is answered by trying out different image

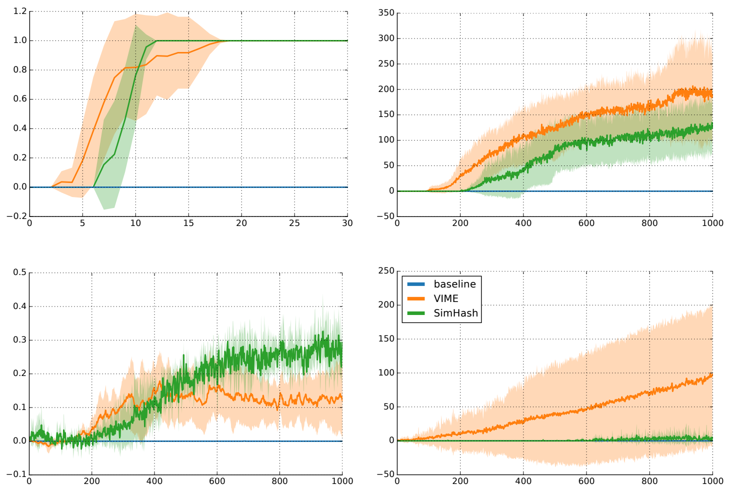


Figure 5.2: Mean average return of different algorithms on rllab tasks with sparse rewards; the solid line represents the mean average return, while the shaded area represents one standard deviation, over 5 seeds for the baseline and SimHash.

preprocessors on Atari 2600 games. Trust Region Policy Optimization (TRPO, [Schulman et al., 2015]; see also Appendix 4.B.2) is chosen as the RL algorithm for all experiments, because it can handle both discrete and continuous action spaces, it can conveniently ensure stable improvement in the policy performance, and is relatively insensitive to hyperparameter changes. The hyperparameters settings are reported in Appendix 5.A.

5.3.1 Continuous Control

The rllab benchmark [Duan et al., 2016] consists of various control tasks to test deep RL algorithms. We selected several variants of the basic and locomotion tasks that use sparse rewards, as shown previously in Figure 4.9, and adopt the experimental setup as defined in [Houthooft et al., 2016]—a description can be found in Appendix 4.B.1. These tasks are all highly difficult to solve with naïve exploration strategies, such as adding Gaussian noise to the actions.

Figure 5.2 shows the results of TRPO (baseline), TRPO-SimHash, and

VIME [Houthooft et al., 2016] on the classic tasks MountainCar and CartPoleSwingup, the locomotion task HalfCheetah, and the hierarchical task SwimmerGather. Using count-based exploration with hashing is capable of reaching the goal in all environments (which corresponds to a nonzero return), while baseline TRPO with Gaussian control noise fails completely. Although TRPO-SimHash picks up the sparse reward on HalfCheetah, it does not perform as well as VIME. In contrast, the performance of SimHash is comparable with VIME on MountainCar, while it outperforms VIME on SwimmerGather.

5.3.2 Arcade Learning Environment

The Arcade Learning Environment (ALE, [Bellemare et al., 2013]), which consists of Atari 2600 video games, is an important benchmark for deep RL due to its high-dimensional state space and wide variety of games. In order to demonstrate the effectiveness of the proposed exploration strategy, six games are selected featuring long horizons while requiring significant exploration: Freeway, Frostbite, Gravitar, Montezuma's Revenge, Solaris, and Venture. The agent is trained for 500 iterations in all experiments, with each iteration consisting of 0.1 M steps (the TRPO batch size, corresponds to 0.4 M frames). Policies and value functions are neural networks with identical architectures to [Mnih et al., 2016]. Although the policy and baseline take into account the previous four frames, the counting algorithm only looks at the latest frame.

BASS To compare with the autoencoder-based learned hash code, we propose using Basic Abstraction of the ScreenShots (BASS, also called Basic; see [Bellemare et al., 2013]) as a static preprocessing function g. BASS is a hand-designed feature transformation for images in Atari 2600 games. BASS builds on the following observations specific to Atari: i) the game screen has a low resolution, ii) most objects are large and monochrome, and iii) winning depends mostly on knowing object locations and motions. We designed an adapted version of BASS[1], that divides the RGB screen into square cells, computes the average intensity of each color channel inside a cell, and assigns the resulting values to bins that uniformly partition the

[1]The original BASS exploits the fact that at most 128 colors can appear on the screen. Our adapted version does not make this assumption.

Table 5.1: Atari 2600: average total reward after training for 50 M time steps. Boldface numbers indicate best results. Italic numbers are the best among our methods.

	Freeway	Frostbite[1]	Gravitar	Montezuma	Solaris	Venture
TRPO (baseline)	16.5	2869	486	0	2758	121
TRPO-pixel-SimHash	31.6	4683	468	0	2897	263
TRPO-BASS-SimHash	28.4	3150	*604*	*238*	1201	*616*
TRPO-AE-SimHash	***33.5***	***5214***	482	75	*4467*	445
Double-DQN	33.3	1683	412	0	3068	98.0
Dueling network	0.0	4672	588	0	2251	497
Gorila	11.7	605	**1054**	4	N/A	**1245**
DQN Pop-Art	33.4	3469	483	0	**4544**	1172
A3C+	27.3	507	246	142	2175	0
pseudo-count[2]	29.2	1450	–	**3439**	–	369

[1] While [Vezhnevets et al., 2016] reported best score 8108, their evaluation was based on top 5 agents trained with 500M time steps, hence not comparable.
[2] Results reported only for 25 M time steps (100 M frames).

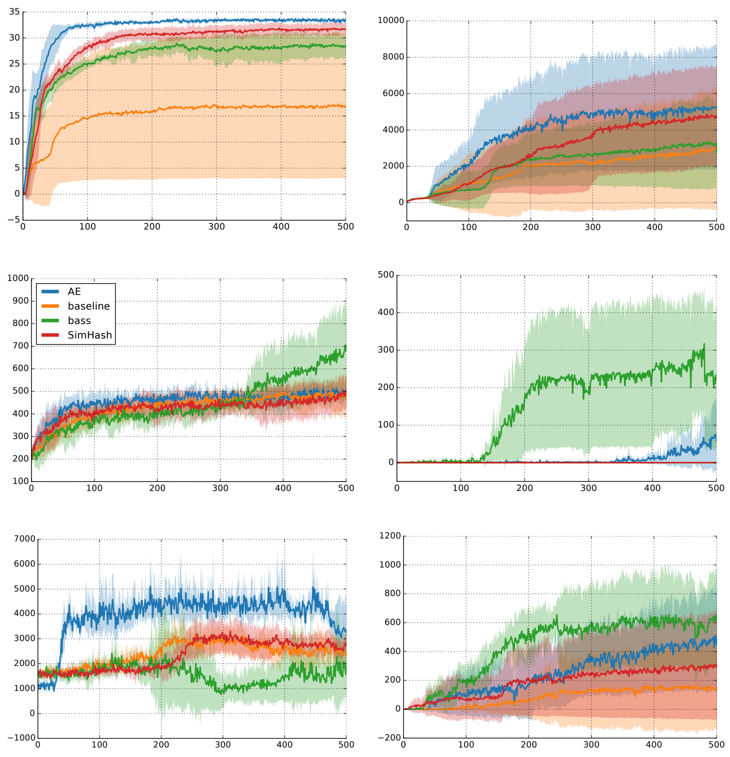


Figure 5.3: Atari 2600 games: the solid line is the mean average undiscounted return per iteration, while the shaded areas represent the one standard deviation, over 5 seeds for the baseline, TRPO-pixel-SimHash, and TRPO-BASS-SimHash, while over 3 seeds for TRPO-AE-SimHash.

intensity range $[0, 255]$. Mathematically, let C be the cell size (width and height), B the number of bins, (i, j) cell location, (x, y) pixel location, and z the channel, then

$$\text{feature}(i, j, z) = \left\lfloor \frac{B}{255C^2} \sum_{(x,y)\in\, \text{cell}(i,j)} I(x, y, z) \right\rfloor . \tag{5.4}$$

Afterwards, the resulting integer-valued feature tensor is converted to an integer hash code ($\phi(s_t)$ in Line 6 of Algorithm 5.1). A BASS feature can be regarded as a miniature that efficiently encodes object locations, but remains invariant to negligible object motions. It is easy to implement and introduces little computation overhead. However, it is designed for generic Atari game images and may not capture the structure of each specific game very well.

We compare our results to double DQN [van Hasselt et al., 2016b], dueling network [Wang et al., 2016], A3C+ [Bellemare et al., 2016], double DQN with pseudo-counts [Bellemare et al., 2016], Gorila [Nair et al., 2015], and DQN Pop-Art [van Hasselt et al., 2016a] on the "null op" metric[2]. We show training curves in Figure 5.3 and summarize all results in Table 1. Surprisingly, TRPO-pixel-SimHash already outperforms the baseline by a large margin and beats the previous best result on Frostbite. TRPO-BASS-SimHash achieves significant improvement over TRPO-pixel-SimHash on Montezuma's Revenge and Venture, where it captures object locations better than other methods.[3] TRPO-AE-SimHash achieves near state-of-the-art performance on Freeway, Frostbite and Solaris.[4]

As observed in Table 1, preprocessing images with BASS or using a learned hash code through the AE leads to much better performance on Gravitar, Montezuma's Revenge and Venture. Therefore, a static or adaptive preprocessing step can be important for a good hash function.

In conclusion, our count-based exploration method is able to achieve remarkable performance gains even with simple hash functions like SimHash on the raw pixel space. If coupled with domain-dependent state preprocessing techniques, it can sometimes achieve far better results.

5.4 Related Work

Classic count-based methods such as MBIE [Strehl and Littman, 2005], MBIE-EB and [Kolter and Ng, 2009] solve an approximate Bellman equation as an inner loop before the agent takes an action [Strehl and Littman, 2008]. As such, bonus rewards are propagated immediately throughout the state-action space. In contrast, contemporary deep RL algorithms prop-

[2]The agent takes no action for a random number (within 30) of frames at the beginning of each episode.

[3]We provide videos of example game play and visualizations of the difference bewteen Pixel-SimHash and BASS-SimHash at https://www.youtube.com/playlist?list=PLAd-UMX6FkBQdLNWtY8nH1-pzYJA_1T55

[4]Note that some design choices in other algorithms also impact exploration, such as ϵ-greedy and entropy regularization. Nevertheless, it is still valuable to position our results within the current literature.

agate the bonus signal based on rollouts collected from interacting with environments, with value-based [Mnih et al., 2015] or policy gradient-based [Schulman et al., 2015, Mnih et al., 2016] methods, at limited speed. In addition, our proposed method is intended to work with contemporary deep RL algorithms, it differs from classical count-based method in that our method relies on visiting unseen states first, before the bonus reward can be assigned, making uninformed exploration strategies still a necessity at the beginning. Filling the gaps between our method and classic theories is an important direction of future research.

A related line of classical exploration methods is based on the idea of *optimism in the face of uncertainty* [Brafman and Tennenholtz, 2002] but not restricted to using counting to implement "optimism", e.g., R-Max [Brafman and Tennenholtz, 2002], UCRL [Jaksch et al., 2010], and E^3 [Kearns and Singh, 2002]. These methods, similar to MBIE and MBIE-EB, have theoretical guarantees in tabular settings.

Bayesian RL methods [Kolter and Ng, 2009, Guez et al., 2014, Sun et al., 2011, Ghavamzadeh et al., 2015], which keep track of a distribution over MDPs, are an alternative to optimism-based methods. Extensions to continuous state space have been proposed by [Pazis and Parr, 2013] and [Osband et al., 2016b].

Another type of exploration is curiosity-based exploration. These methods try to capture the agent's surprise about transition dynamics. As the agent tries to optimize for surprise, it naturally discovers novel states. We refer the reader to [Schmidhuber, 2010] and [Oudeyer and Kaplan, 2007] for an extensive review on curiosity and intrinsic rewards.

Several exploration strategies for deep RL have been proposed to handle high-dimensional state space recently. [Houthooft et al., 2016] propose VIME, in which information gain is measured in Bayesian neural networks modeling the MDP dynamics, which is used an exploration bonus. [Stadie et al., 2015] propose to use the prediction error of a learned dynamics model as an exploration bonus. Thompson sampling through bootstrapping is proposed by [Osband et al., 2016a], using bootstrapped Q-functions.

The most related exploration strategy is proposed by [Bellemare et al., 2016], in which an exploration bonus is added inversely proportional to the square root of a *pseudo-count* quantity. A state pseudo-count is derived from its log-probability improvement according to a density model over the state space, which in the limit converges to the empirical count. Our method is similar to pseudo-count approach in the sense that both methods are performing approximate counting to have the necessary generalization over unseen states. The difference is that a density model has to be designed and learned to achieve good generalization for pseudo-count whereas in our case generalization is obtained by a wide range of simple hash functions (not necessarily SimHash). Another interesting connection is that our method also implies a density model $\rho(s) = \frac{n(\phi(s))}{N}$ over all visited states, where N is the total number of states visited. Another method similar to hashing is proposed by [Abel et al., 2016], which clusters states and counts cluster centers instead of the true states, but this method has yet to be tested on standard exploration benchmark problems.

5.5 Conclusions

This chapter demonstrates that a generalization of classical counting techniques through hashing is able to provide an appropriate signal for exploration, even in continuous and/or high-dimensional MDPs using function approximators, resulting in near state-of-the-art performance across benchmarks. It provides a simple yet powerful baseline for solving MDPs that require informed exploration.

5.A Hyperparameter Settings

For the rllab experiments, we used batch size 5000 for all tasks except SwimmerGather, for which we used batch size 50000. CartpoleSwingup makes use of a neural network policy with one layer of 32 tanh units. The other tasks make use of a two layer neural network policy of 32 tanh units each for MountainCar and HalfCheetah, and of 64 and 32 tanh units for SwimmerGather. The outputs are modeled by a fully factorized Gaussian distribution $\mathcal{N}(\mu, \sigma^2 I)$, in which μ is modeled as the network output, while σ is a parameter. CartPoleSwingup makes use of a neural network baseline with one layer of 32 ReLU units, while all other tasks make use of a linear baseline function. For all tasks, we used TRPO step size 0.01 and discount factor $\gamma = 0.99$. We choose SimHash parameter $k = 32$ and bonus coefficient $\beta = 0.01$, found through a coarse grid search.

For Atari experiments, a batch size of 100000 is used, while the KL divergence step size is set to 0.01. The policy and baseline both have the following architecture: 2 convolutional layers with respectively 16 and 32 filters, sizes 8×8 and 4×4, strides 4 and 2, using no padding, feeding into a single hidden layer of 256 units. The nonlinearities are rectified linear units (ReLUs). The input frames are downsampled to 52×52. The input to policy and baseline consists of the 4 previous frames, corresponding to the frame skip of 4. The discount factor was set to $\gamma = 0.995$. All inputs are rescaled to $[-1, 1]$ element-wise. All experiments used 5 different training seeds, except the experiments with the learned hash code, which uses 3 different training seeds. Batch normalization [Ioffe and Szegedy, 2015] is used at each policy and baseline layer. TRPO-pixel-SimHash uses binary codes of size $k = 256$; BASS (TRPO-BASS-SimHash) extracts features using cell size $C = 20$ and $B = 20$ bins. The autoencoder for the learned embedding (TRPO-AE-SimHash) uses a binary hidden layer of 512 bit, which are projected to 64 bit.

RAM states in Atari 2600 games are integer-valued vectors over length 128 in the range $[0, 255]$. Experiments on Montezuma's Revenge with RAM observations use a policy consisting of 2 hidden layers, each of size 32. RAM states are rescaled to a range $[-1, 1]$. Unlike images, only the current RAM is shown to the agent. Experiment results are averaged over 10 random seeds.

In addition, we apply counting Bloom filters [Fan et al., 2000] to maintain a small hash table. Details can be found in Appendix 5.C.

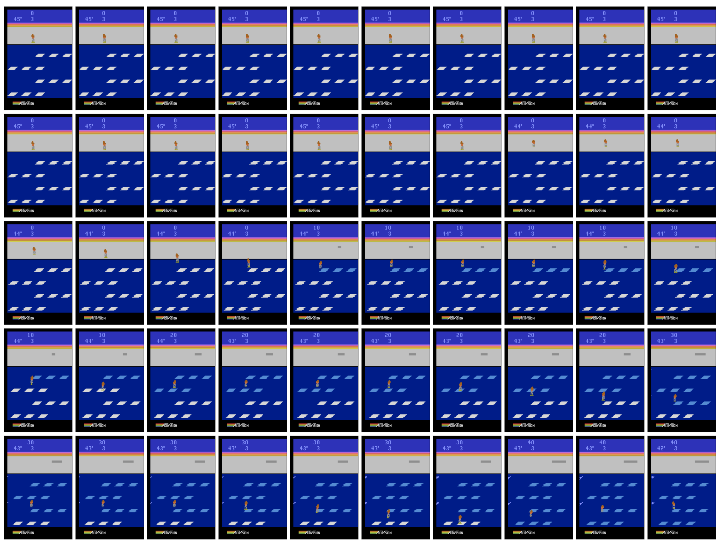
The autoencoder used for the learned hash code has a 512 bit binary code layer, using sigmoid units, to which uniform noise $U(-a, a)$ with $a = 0.3$ is added. The loss function Eq. (5.3), using $\lambda = 10$, is updated every $j_{\text{update}} = 3$ iterations. The architecture looks as follows: an input layer of size 52×52, representing the image luminance is followed by 3 consecutive 6×6 convolutional layers with stride 2 and 96 filters feed into a fully connected layer of size 1024, which connects to the binary code layer. This binary code layer feeds into a fully-connected layer of 1024 units, connecting to a fully-connected layer of 2400 units. This layer feeds into 3 consecutive 6×6 transposed convolutional layers of which the final one connects to a pixel-wise softmax layer with 64 bins, representing the pixel intensities. Moreover, label smoothing is applied to the different softmax bins, in which the log-probability of each of the bins is increased by 0.003, before normalizing. The softmax weights are shared among each pixel. All output nonlinearities are ReLUs; Adam [Kingma and Ba, 2015] is used as an optimization scheme; batch normalization [Ioffe and Szegedy, 2015] is applied to each layer. The architecture was shown in Figure 5.1 of Section 5.2.3.

5.B Analysis of Learned Binary Representation

Figure 5.7 shows the reconstructions of several subsequent images according to the autoencoder, while Figure 5.8 shows the corresponding codes. Figures 5.4, 5.5, and 5.6 shows the downsampled codes learned by the autoencoder for several Atari 2600 games (Frostbite, Freeway, and Montezuma's Revenge). Each row depicts 50 consecutive frames (from 0 to 49, going from left to right, top to bottom). The pictures in the right column depict the binary codes that correspond with each of these frames (one frame per row).

5.C Counting Bloom Filter/Count-Min Sketch

We experimented with directly building a hashing dictionary with keys $\phi(s)$ and values the state counts, but observed an unnecessary increase in computation time. Our implementation converts the integer hash codes into



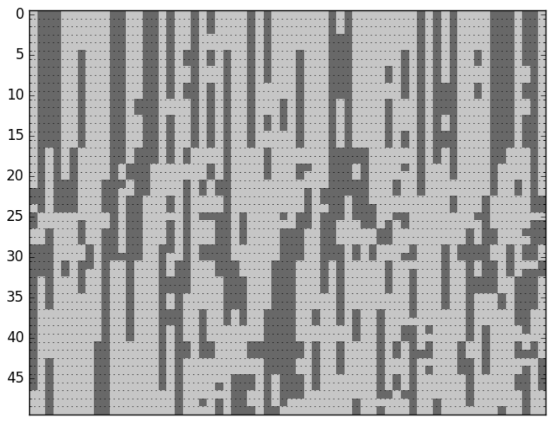
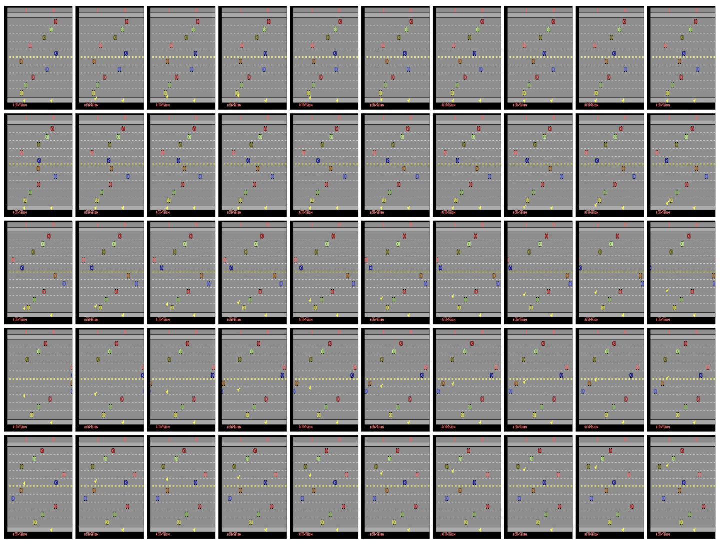


Figure 5.4: Frostbite, Freeway, and Montezuma's Revenge: subsequent frames (left) and corresponding code (right); the frames are ordered from left (starting with frame number 0) to right, top to bottom; the vertical axis in the right images correpond to the frame number.



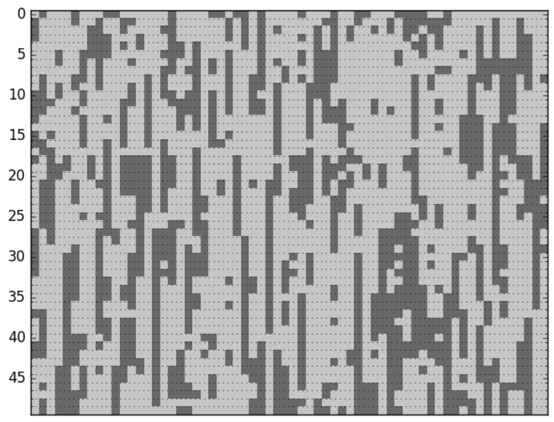
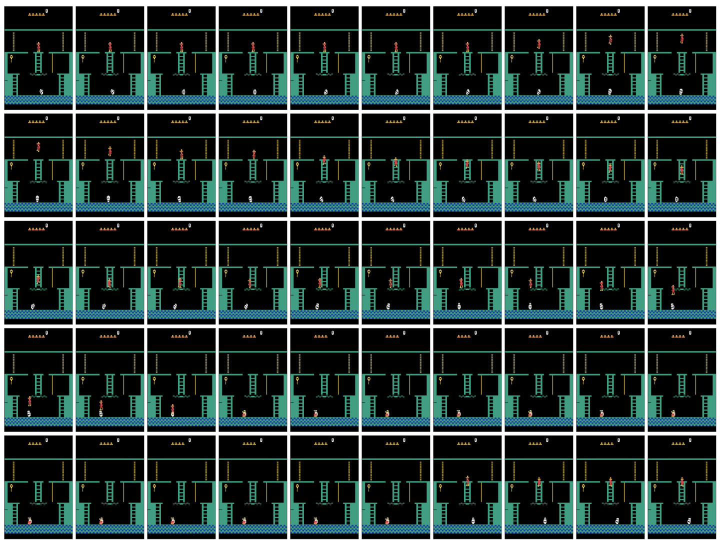


Figure 5.5: Frostbite, Freeway, and Montezuma's Revenge: subsequent frames (left) and corresponding code (right); the frames are ordered from left (starting with frame number 0) to right, top to bottom; the vertical axis in the right images correpond to the frame number.



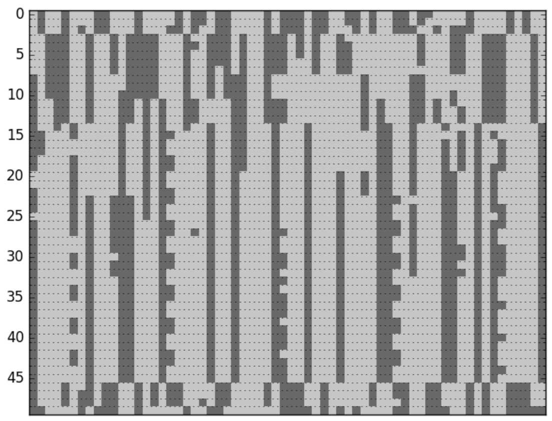


Figure 5.6: Frostbite, Freeway, and Montezuma's Revenge: subsequent frames (left) and corresponding code (right); the frames are ordered from left (starting with frame number 0) to right, top to bottom; the vertical axis in the right images correpond to the frame number.

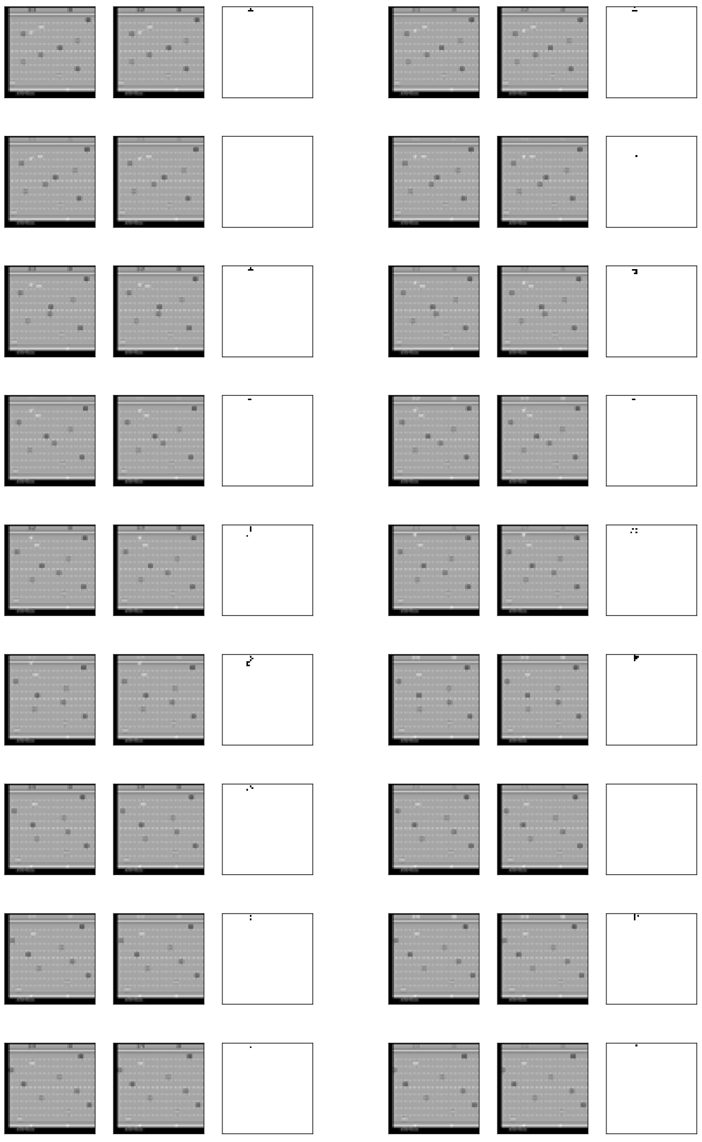


Figure 5.7: Freeway: subsequent frames; frames are ordered from left (starting with frame number 0) to right, top to bottom; the vertical axis in the right images correspond to the frame number. Within each image, the left picture is the input frame, the middle picture the reconstruction, and the right picture, the reconstruction error.

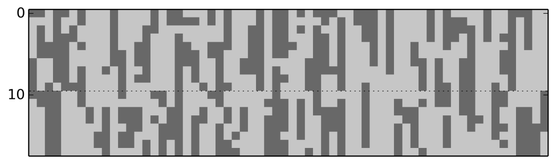


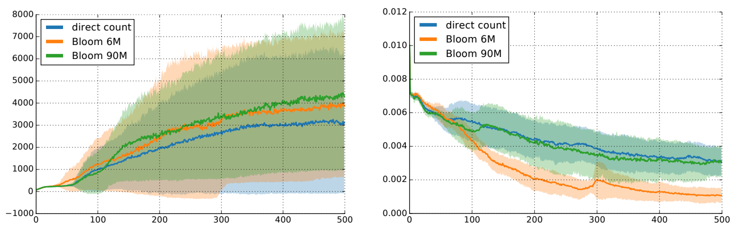
Figure 5.8: Freeway: the learned hash codes corresponding to the frames in Figure 5.7

binary numbers and then into the "bytes" type in Python. The hash table is a dictionary using those bytes as keys.

However, an alternative technique called Count-Min Sketch [Cormode and Muthukrishnan, 2005], with a data structure identical to counting Bloom filters [Fan et al., 2000], can count with a fixed integer array and thus reduce computation time. Specifically, let $m^1, \dots, m^l$ be distinct large prime numbers and define $\phi^j(s) = \phi(s) \mod m^j$. The count of state s is returned as $\min_{1 \le j \le l} n^j\left(\phi^j(s)\right)$. To increase the count of s, we increment $n^j\left(\phi^j(s)\right)$ by 1 for all j. Intuitively, the method replaces ϕ by weaker hash functions, while it reduces the probability of over-counting by reporting counts agreed by all such weaker hash functions. The final hash code is represented as $\left(\phi^1(s), \dots, \phi^l(s)\right)$.

Throughout all experiments above, the prime numbers for the counting Bloom filter are 999931, 999953, 999959, 999961, 999979, and 999983, which we abbreviate as "6 M". In addition, we experimented with 6 other prime numbers, each approximately 15 M, which we abbreviate as "90 M". As we can see in Figure 5.9, counting states with a dictionary or with Bloom filters lead to similar performance, but the computation time of latter is lower. Moreover, there is little difference between direct counting and using a very larger table for Bloom filters, as the average bonus rewards are almost the same, indicating the same degree of exploration-exploitation trade-off. On the other hand, Bloom filters require a fixed table size, which may not be known beforehand.

Theory of Bloom Filters Bloom filters [Bloom, 1970] are popular for determining whether a data sample s' belongs to a dataset $\mathcal{D}$. Suppose we have l functions ϕ^j that independently assign each data sample to an integer between 1 and m uniformly at random. Initially $1, 2, \dots, p$ are marked as



(a) Mean average undiscounted return (b) Average bonus reward

Figure 5.9: Statistics of TRPO-pixel-SimHash ($k = 256$) on Frostbite. Solid lines are the average, while the shaded areas represent the one standard deviation. Results are derived from 10 random seeds. Direct counting with a dictionary uses 2.7 times more computation than counting Bloom filters (6 M or 90 M).

0. Then every $s \in \mathcal{D}$ is "inserted" through marking $\phi^j(s)$ as 1 for all j. A new sample s' is reported as a member of $\mathcal{D}$ only if $\phi^j(s)$ are marked as 1 for all j. A bloom filter has zero false negative rate (any $s \in \mathcal{D}$ is reported a member), while the false positive rate (probability of reporting a nonmember as a member) decays exponentially in l.

Though Bloom filters support data insertion, it does not allow data deletion. Counting Bloom filters [Fan et al., 2000] maintain a counter $n(\cdot)$ for each number between 1 and p. Inserting/deleting s corresponds to incrementing/decrementing $n\left(\phi^j(s)\right)$ by 1 for all j. Similarly, s is considered a member if $\forall j : n\left(\phi^j(s)\right) = 0$.

Count-Min sketch is designed to support memory-efficient counting without introducing too many over-counts. It maintains a separate count n^j for each hash function ϕ^j defined as $\phi^j(s) = \phi(s) \mod m^j$, where m^j is a large prime number. For simplicity, we may assume that $m^j \approx m \; \forall j$ and ϕ^j assigns s to any of $1, \ldots, m$ with uniform probability.

We now derive the probability of over-counting. Let s be a fixed data sample (not necessarily inserted yet) and suppose a dataset $\mathcal{D}$ of N samples are inserted. We assume that $m^l \gg N$. Let $n = \min_{1 \le j \le l} n^j\left(\phi^j(s)\right)$ be the count returned by the Bloom filter. We are interested in computing $p(n > 0 | s \notin \mathcal{D})$. Due to assumptions about ϕ^j, we know $n^j(\phi(s)) \sim$

Binomial $\left(N, \frac{1}{m}\right)$. Therefore,

$$\begin{aligned} p(n > 0 | s \notin \mathcal{D}) &= \frac{p(n > 0, s \notin \mathcal{D})}{p(s \notin \mathcal{D})} \\ &= \frac{p(n > 0) - p(s \in \mathcal{D})}{p(s \notin \mathcal{D})} \\ &\approx \frac{p(n > 0)}{p(s \notin \mathcal{D})} \\ &= \frac{\prod_{j=1}^{l} p(n^j(\phi^j(s)) > 0)}{(1 - 1/m^l)^N} \\ &= \frac{\left(1 - (1 - 1/m)^N\right)^l}{((1 - 1/m^l)^N} \\ &\approx \frac{\left(1 - e^{-N/m}\right)^l}{e^{-N/m^l}} \\ &\approx \left(1 - e^{-N/m}\right)^l . \end{aligned} \tag{5.5}$$

In particular, the probability of over-counting decays exponentially in l. We refer the readers to [Cormode and Muthukrishnan, 2005] for other properties of the Count-Min sketch.

5.D Robustness Analysis

This section investigates the robustness of the proposed approach.

5.D.1 Granularity

While our proposed method is able to achieve remarkable results without requiring much tuning, the granularity of the hash function should be chosen wisely. Granularity plays a critical role in count-based exploration, where the hash function should cluster states without under-generalizing or over-generalizing. Table 5.2 summarizes granularity parameters for our hash functions. In Table 5.3 we summarize the performance of TRPO-pixel-SimHash under different granularities. We choose Frostbite and Venture on which TRPO-pixel-SimHash outperforms the baseline, and choose as reward bonus coefficient $\beta = 0.01 \times \frac{256}{k}$ to keep average bonus rewards at approximately the same scale. $k = 16$ only corresponds to 65536 distinct hash codes, which is insufficient to distinguish between semantically distinct states and hence leads to worse performance. We observed that $k = 512$ tends to capture trivial image details in Frostbite, leading the agent to believe that every state is new and equally worth exploring. Similar results are observed while tuning the granularity parameters for TRPO-BASS-SimHash and TRPO-AE-SimHash.

Table 5.2: Granularity parameters of various hash functions

SimHash	k: size of the binary code
BASS	C: cell size; B: number of bins for each color channel
AE	k: downstream SimHash parameter; binary code size λ: binarization parameter
SmartHash	s: grid size for the agent's (x, y) coordinates

Table 5.3: Average score at 50 M time steps achieved by TRPO-pixel-SimHash

k	16	64	128	256	512
Frostbite	3326	4029	3932	**4683**	1117
Venture	0	218	142	263	**306**

Table 5.4: Average score at 50 M time steps achieved by TRPO-SmartHash on Montezuma's Revenge (RAM observations)

s	1	5	10	20	40	60
score	2598	2500	**3533**	3025	2500	1921

The best granularity depends on both the hash function and the MDP. While adjusting granularity parameter, we observed that it is important to lower the bonus coefficient as granularity is increased. This is because a higher granularity is likely to cause lower state counts, leading to higher bonus rewards that may overwhelm the true rewards.

5.D.2 A Case Study of Montezuma's Revenge

Montezuma's Revenge is widely known for its extremely sparse rewards and difficult exploration [Bellemare et al., 2016]. While our method does not outperform [Bellemare et al., 2016] on this game, we investigate the reasons behind this through various experiments. The experiment process below again demonstrates the importance of a hash function having the correct granularity and encoding relevant information for solving the MDP.

Table 5.5: Interpretation of particular RAM entries in Montezuma's Revenge

RAM index	Group	Meaning
3	room	room number
42	agent	x coordinate
43	agent	y coordinate
52	agent	orientation (left/right)
27	beam walls	on/off
83	beam walls	beam wall countdown (on: 0, off: $36 \to 0$)
0	counter	counts from 0 to 255 and repeats
55	counter	death scene countdown
67	objects	objects (doors, skull and key) in 1st room
47	skull	x coordinate (both 1st and 2nd rooms)

Our first attempt is to use game RAM states instead of image observations as inputs to the policy (details in Appendix 5.A), which leads to a game score of 2500 with TRPO-BASS-SimHash. Our second attempt is to manually design a hash function that incorporates domain knowledge, called *SmartHash*, which uses an integer-valued vector consisting of the agent's (x, y) location, room number and other useful RAM information as the hash code. The best SmartHash agent is able to obtain a score of 3500. Still the performance is not optimal. We observe that a slight change in the agent's coordinates does not always result in a semantically distinct state, and thus the hash code may remain unchanged. Therefore we choose grid size s and replace the x coordinate by $\lfloor (x - x_{\min})/s \rfloor$ (similarly for y). The bonus coefficient is chosen as $\beta = 0.01\sqrt{s}$ to maintain the scale relative

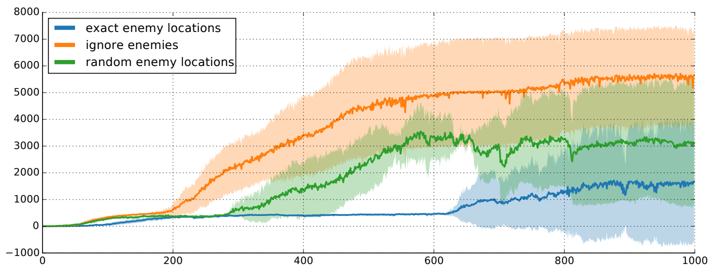


Figure 5.10: SmartHash results on Montezuma's Revenge (RAM observations): the solid line is the mean average undiscounted return per iteration, while the shaded areas represent the one standard deviation, over 5 seeds.

to the true reward[5] (see Table 5.4). Finally, the best agent is able to obtain 6600 total rewards after training for 1000 iterations (1000 M time steps), with a grid size $s = 10$.

Table 5.5 lists the semantic interpretation of certain RAM entries in Montezuma's Revenge. SmartHash, as described in Section 5.D.2, makes use of RAM indices 3, 42, 43, 27, and 67. "Beam walls" are deadly barriers that occur periodically in some rooms.

During our pursuit, we had another interesting discovery that the ideal hash function should not simply cluster states by their visual similarity, but instead by their relevance to solving the MDP. We experimented with including enemy locations in the first two rooms into SmartHash ($s = 10$), and observed that average score dropped to 1672 (at iteration 1000). Though it is important for the agent to dodge enemies, the agent also erroneously "enjoys" watching enemy motions at distance (since new states are constantly observed) and "forgets" that his main objective is to enter other rooms. An alternative hash function keeps the same entry "enemy locations", but instead only puts randomly sampled values in it, which surprisingly achieves better performance (3112). However, by ignoring enemy locations altogether, the agent achieves a much higher score (5661) (see Figure 5.10). In retrospect, we examine the hash codes generated by BASS-SimHash and find that codes clearly distinguish between visually different states (including various enemy locations), but fails to emphasize that the agent needs to

[5]The bonus scaling is chosen by assuming all states are visited uniformly and the average bonus reward should remain the same for any grid size.

explore different rooms. Again this example showcases the importance of encoding relevant information in designing hash functions.

Apart from the experimental results shown in Table 5.1 and Table 5.3, additional experiments have been performed to study several properties of our algorithm.

Hyperparameter sensitivity To study the performance sensitivity to hyperparameter changes, we focus on evaluating TRPO-RAM-SimHash on the Atari 2600 game Frostbite, where the method has a clear advantage over the baseline. Because the final scores can vary between different random seeds, we evaluated each set of hyperparameters with 30 seeds. To reduce computation time and cost, RAM states are used instead of image observations.

Table 5.6: TRPO-RAM-SimHash performance robustness to hyperparameter changes on Frostbite

	β							
k	0	0.01	0.05	0.1	0.2	0.4	0.8	1.6
–	397	–	–	–	–	–	–	–
64	–	879	2464	2243	2489	1587	1107	441
128	–	1475	4248	2801	3239	3621	1543	395
256	–	2583	4497	4437	7849	3516	2260	374

The results are summarized in Table 5.6. Herein, k refers to the length of the binary code for hashing while β is the multiplicative coefficient for the reward bonus, as defined in Section 5.2.2. This table demonstrates that most hyperparameter settings outperform the baseline ($\beta = 0$) significantly. Moreover, the final scores show a clear pattern in response to changing hyperparameters. Small β-values lead to insufficient exploration, while large β-values cause the bonus rewards to overwhelm the true rewards. With a fixed k, the scores are roughly concave in β, peaking at around 0.2. Higher granularity k leads to better performance. Therefore, it can be concluded that the proposed exploration method is robust to hyperparameter changes in comparison to the baseline, and that the best parameter settings can obtained from a relatively coarse-grained grid search.

Table 5.7: Performance comparison between state counting (left of the slash) and state-action counting (right of the slash) using TRPO-RAM-SimHash on Frostbite

	β						
k	0.01	0.05	0.1	0.2	0.4	0.8	1.6
64	879 / 976	2464 / 1491	2243 / 3954	2489 / 5523	1587 / 5985	1107 / 2052	441 / 742
128	1475 / 808	4248 / 4302	2801 / 4802	3239 / 7291	3621 / 4243	1543 / 1941	395 / 362
256	2583 / 1584	4497 / 5402	4437 / 5431	7849 / 4872	3516 / 3175	2260 / 1238	374 / 96

State and state-action counting Continuing the results in Table 5.6, the performance of state-action counting is studied using the same experimental setup, summarized in Table 5.7. In particular, a bonus reward $r^+ = \frac{\beta}{\sqrt{n(s,a)}}$ instead of $r^+ = \frac{\beta}{\sqrt{n(s)}}$ is assigned. These results show that the relative performance of state counting compared to state-action counting depends highly on the selected hyperparameter settings. However, we notice that the best performance is achieved using state counting with $k = 256$ and $\beta = 0.2$.

References

[Abel et al., 2016] Abel, D., Agarwal, A., Diaz, F., Krishnamurthy, A., and Schapire, R. E. (2016). Exploratory gradient boosting for reinforcement learning in complex domains. *arXiv preprint arXiv:1603.04119*.

[Andoni and Indyk, 2006] Andoni, A. and Indyk, P. (2006). Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 459–468.

[Bellemare et al., 2013] Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. (2013). The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279.

[Bellemare et al., 2016] Bellemare, M. G., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. (2016). Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems 29 (NIPS)*, pages 1471–1479.

[Bloom, 1970] Bloom, B. H. (1970). Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426.

[Brafman and Tennenholtz, 2002] Brafman, R. I. and Tennenholtz, M. (2002). R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3:213–231.

[Charikar, 2002] Charikar, M. S. (2002). Similarity estimation techniques from rounding algorithms. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC)*, pages 380–388.

[Cormode and Muthukrishnan, 2005] Cormode, G. and Muthukrishnan, S. (2005). An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75.

[Dalal and Triggs, 2005] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 886–893.

[Duan et al., 2016] Duan, Y., Chen, X., Houthooft, R., Schulman, J., and Abbeel, P. (2016). Benchmarking deep reinforcement learning for continous control. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 1329–1338.

[Fan et al., 2000] Fan, L., Cao, P., Almeida, J., and Broder, A. Z. (2000). Summary cache: A scalable wide-area web cache sharing protocol. *IEEE/ACM Transactions on Networking*, 8(3):281–293.

[Ghavamzadeh et al., 2015] Ghavamzadeh, M., Mannor, S., Pineau, J., and Tamar, A. (2015). Bayesian reinforcement learning: A survey. *Foundations and Trends in Machine Learning*, 8(5-6):359–483.

[Guez et al., 2014] Guez, A., Heess, N., Silver, D., and Dayan, P. (2014). Bayes-adaptive simulation-based search with value function approximation. In *Advances in Neural Information Processing Systems 27 (NIPS)*, pages 451–459.

[He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition.

[Houthooft et al., 2016] Houthooft, R., Chen, X., Duan, Y., Schulman, J., De Turck, F., and Abbeel, P. (2016). VIME: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems 29 (NIPS)*, pages 1109–1117.

[Ioffe and Szegedy, 2015] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 448–456.

[Jaksch et al., 2010] Jaksch, T., Ortner, R., and Auer, P. (2010). Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11:1563–1600.

[Kearns and Singh, 2002] Kearns, M. and Singh, S. (2002). Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2):209–232.

[Kingma and Ba, 2015] Kingma, D. and Ba, J. (2015). Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

[Kolter and Ng, 2009] Kolter, J. Z. and Ng, A. Y. (2009). Near-bayesian exploration in polynomial time. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, pages 513–520.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25 (NIPS)*, pages 1097–1105.

[Lai and Robbins, 1985] Lai, T. L. and Robbins, H. (1985). Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4–22.

[Lillicrap et al., 2015] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

[Lowe, 1999] Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Proceedings of the 7th IEEE International Conference on Computer Vision (ICCV)*, pages 1150–1157.

[Mnih et al., 2016] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. *arXiv preprint arXiv:1602.01783*.

[Mnih et al., 2015] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

[Nair et al., 2015] Nair, A., Srinivasan, P., Blackwell, S., Alcicek, C., Fearon, R., De Maria, A., Panneershelvam, V., Suleyman, M., Beattie, C., Petersen, S., et al. (2015). Massively parallel methods for deep reinforcement learning. *arXiv preprint arXiv:1507.04296*.

[Osband et al., 2016a] Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. (2016a). Deep exploration via bootstrapped DQN. In *Advances in Neural Information Processing Systems 29 (NIPS)*, pages 4026–4034.

[Osband et al., 2016b] Osband, I., Van Roy, B., and Wen, Z. (2016b). Generalization and exploration via randomized value functions. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 2377–2386.

[Oudeyer and Kaplan, 2007] Oudeyer, P.-Y. and Kaplan, F. (2007). What is intrinsic motivation? A typology of computational approaches. *Frontiers in Neurorobotics*, 1:6.

[Pazis and Parr, 2013] Pazis, J. and Parr, R. (2013). PAC optimal exploration in continuous space Markov decision processes. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI)*.

[Schmidhuber, 2010] Schmidhuber, J. (2010). Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247.

[Schulman et al., 2015] Schulman, J., Levine, S., Moritz, P., Jordan, M. I., and Abbeel, P. (2015). Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 1889–1897.

[Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[Stadie et al., 2015] Stadie, B. C., Levine, S., and Abbeel, P. (2015). Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*.

[Strehl and Littman, 2005] Strehl, A. L. and Littman, M. L. (2005). A theoretical analysis of model-based interval estimation. In *Proceedings of the 21st International Conference on Machine Learning (ICML)*, pages 856–863.

[Strehl and Littman, 2008] Strehl, A. L. and Littman, M. L. (2008). An analysis of model-based interval estimation for Markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331.

[Sun et al., 2011] Sun, Y., Gomez, F., and Schmidhuber, J. (2011). Planning to be surprised: Optimal Bayesian exploration in dynamic environments. In *Proceedings of the 4th International Conference on Artificial General Intelligence (AGI)*, pages 41–51.

[Tola et al., 2010] Tola, E., Lepetit, V., and Fua, P. (2010). DAISY: An efficient dense descriptor applied to wide-baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(5):815–830.

[van den Oord et al., 2016] van den Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. (2016). Pixel recurrent neural networks. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 1747–1756.

[van Hasselt et al., 2016a] van Hasselt, H., Guez, A., Hessel, M., and Silver, D. (2016a). Learning functions across many orders of magnitudes. *arXiv preprint arXiv:1602.07714*.

[van Hasselt et al., 2016b] van Hasselt, H., Guez, A., and Silver, D. (2016b). Deep reinforcement learning with double Q-learning. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI)*.

[Vezhnevets et al., 2016] Vezhnevets, A., Mnih, V., Agapiou, J., Osindero, S., Graves, A., Vinyals, O., and Kavukcuoglu, K. (2016). Strategic attentive writer for learning macro-actions. In *Advances in Neural Information Processing Systems 29 (NIPS)*.

[Wang et al., 2016] Wang, Z., de Freitas, N., and Lanctot, M. (2016). Dueling network architectures for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 1995–2003.

Chapter 6

Conclusions and Future Directions

Computers are no longer programmed directly for solving complex tasks such as identifying objects in images, since these are simply too difficult to be understood explicitly by a programmer. Instead, a paradigm shift is occurring in which computers are instructed via a metaprogram, which allows them to learn tasks based on data-driven pattern extraction. This leads us to the domain of machine learning, which studies the creation of artificial learning systems.

This thesis investigated several topics under the overarching concept of *structured decision making* based on machine learning. This entails that an artificial system outputs several predictions, while taking into account their interrelations. Both spatially- and temporally-structured decision making was researched, through use of structured prediction and reinforcement learning respectively. Several novel methods have been proposed and applied to use cases such as autonomous vehicles, robot locomotion, and autonomous video game playing.

6.1 Conclusions

In Chapters 2 and 3, structured decision making in the spatial domain is investigated. We focused on advanced perception for autonomous vehicles, which is essential for allowing such systems to understand their environment.

Classification models were used to assign semantic classes to image pixels originating from vehicle camera input. Because pixels are highly correlated, a structured-output machine learning model was proposed in Chapter 2, which classifies pixels simultaneously.

This model takes into account visual contextual cues between over-segmentation regions within a second-order adjacency neighborhood. It is formulated as an energy-based function using feature-dependent unary potentials, and pairwise potentials that differentiate between adjacent neighbors in a first- and second-degree neighborhood. After optimization by means of max-margin learning, most-probable label assignments are obtained through graph cuts inference using α-expansion. Quantitative and qualitative results both indicate that using second-degree contextual information allows for highly accurate segmentations by better filtering out spurious labels.

Furthermore, these methods were applied to image segmentation in autonomous agricultural vehicles. For this task, the use of deep convolutional neural networks (CNNs) as unary classification models was proposed, as well as a CNN end-to-end segmentation method that predicts all pixel labels directly. Both models were enhanced through transfer learning using an independently trained classification model.

Chapter 3 digs deeper into the fabric of structured models. Herein, a structured prediction model that integrates back-propagation and loss-augmented inference into subgradient descent training of structural support vector machines (SSVMs) is proposed. This model departs from the traditional bifurcated approach in which a unary classifier is trained independently from the structured predictor. Furthermore, the SSVM factors are extended to neural factors, which allows both unary and interaction factors to be highly nonlinear functions of input features. Results on a complex image segmentation task show that end-to-end SSVM training, and/or using neural factors, leads to more accurate predictions than conventional subgradient descent and N-slack cutting plane training. Moreover, a deep convolutional implementation of the proposed method is applied to end-to-end training and segmentation of autonomous vehicle visual data. The results demonstrate that our model serves as a foundation for more advanced structured models, e.g., by using latent variables, learned feature representations, or complexer connectivity structures.

Temporally-structured decision making is investigated in Chapters 4 and 5. Herein, models were proposed for generating decision sequences that are not correlated in space, but in time. Specifically, we focused on reinforcement learning (RL), in which an agent learns to accomplish a goal through rewards, in an initially unknown environment. A problem in these RL decision making methods is balancing exploitation and exploration.

Therefore, Chapter 4 introduces Variational Information Maximizing Exploration (VIME), a curiosity-driven exploration strategy for high-dimensional continuous control tasks. Variational inference is used to approximate the posterior distribution of a Bayesian neural network that represents the environment dynamics. Using information gain in this learned dynamics model as intrinsic rewards allows the agent to optimize for both external reward and intrinsic surprise simultaneously. Empirical results show that VIME performs significantly better than heuristic exploration methods across various continuous control tasks and RL algorithms.

Finally, in Chapter 5 an alternative exploration strategy to VIME is studied. More specifically, we investigate the use of counting-based exploration strategies in high-dimensional and/or continuous state-action spaces. It is demonstrated that a generalization of classical counting techniques through hashing is able to provide an appropriate signal for exploration, even in continuous and/or high-dimensional Markov decision processes (MDPs) using function approximation, resulting in near state-of-the-art performance across benchmarks. These benchmarks consist of robot locomotion tasks, basic continuous control tasks, and Atari 2600 video games, which the agent has to learn through trial-and-error. Due to its simplicity, the proposed method provides an easy yet powerful baseline for solving MDPs that require informed exploration.

6.2 Future Directions

The research presented in the different chapters of this dissertation leads to several future directions. This section introduces initial progress made by the scientific community that should be further expanded, as well as ideas and problems for which solutions still have to be conceived.

Structured prediction Although the research of Chapters 2 and 3 is applied specifically to semantic segmentation with autonomous vehicles in mind, it would be interesting to explore the proposed methods in more exotic use cases that require the prediction of multiple related output variables. For example, predicting more complicated structures like graphs would be highly interesting, which could be used to learn how to solve problems that are generally not tackled through machine learning such as the traveling salesman problem.

Since the studied structured prediction and semantic segmentation models are trained through supervised learning, a major limitation is their need for large amounts of labeled data. However, it can be very costly to construct a fully-labeled dataset. For example in the autonomous agricultural vehicle segmentation dataset, labeling one image could take as long as 25 minutes by an experienced operator. A big leap forward could be made by the ability to train such models using a combination of labeled and weakly-labeled data [Zhang et al., 2015]. Examples of weakly-labeled data in image segmentation could be images with partial ground truth labels and labels in the form of bounding boxes. One could go even as far as attempting to improve segmentation methods through information extracted from textual image descriptions. Another way forward would be the creation of massive datasets by means of highly realistic virtual environments to generate synthetic images together with their corresponding labelings.

Because the need to alleviate the burden of constructing a fully-labeled dataset, transfer learning methods could prove to be highly useful. For example the ability to transfer features learned through unsupervised learning methods that require no ground truth labels, such as generative models, would be highly valuable. These independently learned features could help to bootstrap the structured prediction learning process, significantly lowering its need for labeled data.

Furthermore, the methods proposed in Chapters 2 and 3 could be extended to the temporal domain, allowing the segmentation of video data rather than still images, which could lead to more accurate predictions. This could be done by enforcing consistency between the segmentations of consecutive images via temporal superpixels [Chang et al., 2013], making use of neural factors that span between multiple video frames, using recurrent neural networks as unary classification models, or using 3-dim temporal convolutions operations [Pigou et al., 2015] in the neural factors.

Furthermore, the combination of both reinforcement learning and structured prediction is highly interesting. Current research efforsts focus on the training of a structured predictor through actor-critic RL methods, such as [Bahdanau et al., 2016]. Also, it would be fascinating to investigate whether it is possible apply RL to autonomous vehicles [Sallab et al., 2016], with the goal of learning robust driving behavior directly, avoiding the need for explicit visual understanding of the vehicle surroundings.

Reinforcement learning One way to extend the research in Chapter 4 is by using Bayesian neural networks for more than the transition dynamics model alone. Representing the state or state-action value function by such a network could allow for the measurement of surprise originating from a change in belief about how states and policies relate to cumulative rewards. This could prove to be helpful in environments such as board games, whose very simple dynamics model causes the curiosity signal to die out rather quickly. Another solution could be build RL algorithms that gracefully transition to planning, as the dynamics model becomes more and more accurate. Even though curiosity signal would still converge to zero quickly, the agent could sample transitions in simulation. Moreover, mixing model samples with real environment transitions could significantly reduce the RL algorithm's sample complexity.

Currently, the learned dynamics model tends to focus on features that do not necessarily have a strong relation to the actual solving of the MDP. This causes the agent to be highly surprised by environment dynamics that do not align with its internal belief, but are irrelevant to accomplishing its goal. It would be worthwhile to investigate alternative dynamics model objectives to cross-entropy that focus particularly on salient environment elements [Larsen et al., 2015]. One way could be to use an inverse dynamics model combined with a forward model [Agrawal et al., 2016], which has been shown to suppress irrelevant details. The study of general generative models, such as PixelCNNs [van den Oord et al., 2016] or InfoGANs [Chen et al., 2016], as a drive for curiosity-based exploration remains an ongoing effort.

The combination of the methods in Chapters 4 and 5 with methods that allow the agent to plan for surprising situations might be interesting. One possibility is the combination of curiosity-driven exploration with empowerment [Gregor et al., 2016] in high-dimensional spaces. Furthermore, the proposed methods have been generally applied in fully-observable MDP settings. It is interesting to figure out what happens when they are used in partially-observed MDPs (POMDPs), since the state observations will collide [Oh et al., 2016].

The assignment of different reward bonuses based on an underlying hierarchy is another interesting future direction. This could be done through assignment of different bonuses for different hashing granularity levels in Chapter 5, or measuring surprise not as the total parameter distribution change in Chapter 4, but measuring it separately for each individual neural network layer. The question is whether this would lead to exploratory signals that are correlated with higher-level environment features, such as the room number in the Atari 2600 game Montezuma's Revenge. Related is the study about whether or not exploration naturally leads to hierarchical policies. One possibility would be to combine the proposed exploration strategies with a recurrent neural network policy, which might allow for more accurate credit assignment, and investigate whether the hidden states capture any form of hierarchical information.

RL models such as the policy or value function first have to learn state representations through their learning algorithm, e.g., policy gradient. However, this does not make use of all information present in the environment, which makes the learning of representations a slow process. Learning a dynamics model such as in VIME makes use of environment information differently. As such, sharing parameters between the learned dynamics model and other RL models might significantly speed up learning.

References

[Agrawal et al., 2016] Agrawal, P., Nair, A., Abbeel, P., Malik, J., and Levine, S. (2016). Learning to poke by poking: Experiential learning of intuitive physics. In *Advances In Neural Information Processing Systems 29 (NIPS)*.

[Bahdanau et al., 2016] Bahdanau, D., Brakel, P., Xu, K., Goyal, A., Lowe, R., Pineau, J., Courville, A., and Bengio, Y. (2016). An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086*.

[Chang et al., 2013] Chang, J., Wei, D., and Fisher, J. W. (2013). A video representation using temporal superpixels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2051–2058.

[Chen et al., 2016] Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. (2016). InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances In Neural Information Processing Systems 29 (NIPS)*, pages 2172–2180.

[Gregor et al., 2016] Gregor, K., Besse, F., Jimenez Rezende, D., Danihelka, I., and Wierstra, D. (2016). Towards conceptual compression. In *Advances in Neural Information Processing Systems 29 (NIPS)*, pages 3549–3557.

[Larsen et al., 2015] Larsen, A. B. L., Sønderby, S. K., and Winther, O. (2015). Autoencoding beyond pixels using a learned similarity metric. *arXiv preprint arXiv:1512.09300*.

[Oh et al., 2016] Oh, J., Chockalingam, V., Singh, S., and Lee, H. (2016). Control of memory, active perception, and action in minecraft. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 2790–2799.

[Pigou et al., 2015] Pigou, L., Oord, A. v. d., Dieleman, S., Van Herreweghe, M., and Dambre, J. (2015). Beyond temporal pooling: Recurrence and temporal convolutions for gesture recognition in video. *arXiv preprint arXiv:1506.01911*.

[Sallab et al., 2016] Sallab, A. E., Abdou, M., Perot, E., and Yogamani, S. (2016). End-to-end deep reinforcement learning for lane keeping assist. *arXiv preprint arXiv:1612.04340*.

[van den Oord et al., 2016] van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., Graves, A., et al. (2016). Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems 29 (NIPS)*, pages 4790–4798.

[Zhang et al., 2015] Zhang, Y., Chen, X., Li, J., Wang, C., and Xia, C. (2015). Semantic object segmentation via detection in weakly labeled video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3641–3649.