Bachelorarbeit

Uncertainty in Recurrent Decision Tree Classifiers

Eberhard Karls Universität Tübingen
Mathematisch-Naturwissenschaftliche Fakultät
Wilhelm-Schickard-Institut für Informatik
Explainable Machine Learning
Stefan Wezel, stefan.wezel@student.uni-tuebingen.de, 2020

Bearbeitungszeitraum: von-bis

Betreuer/Gutachter: Prof. Dr. Zeynep Akata, Universität Tübingen
Betreuer: Stephan Alaniz, Max Planck Institut für Informatik

Abstract

Recurrent Decision Tree Classifiers have proven to be capable of providing explanations for their classifications while yielding state of the art results in prediction accuracy on several image classification tasks. However, they may utilize features that they are highly uncertain about. We hypothesize that when using uncertainty information, the RDTC can provide more faithful explanations and become more applicable in real-life scenarios. Based on RDTC, we propose a model that utilizes uncertainty information. We either enhance the vocabulary of RDTC with an uncertainty token, allowing for a ternary decision tree, or, we restrict the RDTC from using uncertain attributes. We investigate how this uncertainty information can be used in generating interpretable model outputs and how it affects the model's performance on several benchmark tasks.

Contents

Chapter 1

Introduction

The recent surge in the popularity of machine learning methods has made the fields ambivalent nature obvious. Public opinion is split between hype and mistrust. Systems, using machine learning have failed at seemingly simple tasks [36], displayed racist and sexist behavior [26, 72, 83], and even caused fatalities [82]. Such cases, where artificial intelligence has made severe mistakes are funny at best and outright dangerous at worst. Most importantly, they have made it clear that, despite being deployed more frequently than ever before, machine learning has yet to prove itself [72, 17, 83]. These cases show that methods that allow introspection and behave faithfully are more important than ever. Addressing such issues, Alaniz and Akata [2] propose a model that can reliable justify its classification by stating utilized features in a decision tree. Here, we build on this method. We extend the model so that it is aware of and able to express its uncertainties. We hypothesize that such a model may be perceived by a user as more trustful than a model that does not.

We, as humans have developed an intuition on how to make decisions under uncertainty [7] and are able to operate in a domain of limited and noisy data. Moreover, uncertainty is an important measure in science that allows us to express how well something is known or can be known. Scientists from fields like medicine, physics, or meteorology rely on uncertainty information for making and communicating decisions or predictions. Outside the scientific world, uncertainty information is important in finance or law among others [47, 12].

Uncertainty information is valuable. Especially in real-world scenarios, where we might encounter only limited or noisy data to learn from. Where predictions, far into the future are required. Or, where decisions with high consequences have to be made. As machine learning becomes more prevalent in our everyday lives, in science, finance, or law, the need for methods, utilizing uncertainty becomes more evident.

Unfortunately, the output of most neural network architectures does not contain uncertainty information. For example, in classification, typical architectures use a softmax layer, to turn model outputs into a probability density function over classes[57]. Since it is a probability distribution, the softmax output is often misinterpreted as a measure of how certain a model is regarding each class [73]. However, this distribution only arises from each value relative to every other value and is a point estimate, turned into a distribution. The point estimate cannot carry any uncertainty information and the softmax certainly does not add such information. This allows a softmax class probability to be overconfident for a highly

uncertain class [21]. Our key contributions are to propose a model, that allows to (1) study the effect of (epistemic) uncertainty for attribute-based recurrent decision tree classifier and actively uses and expresses uncertainty information. We (2) test the proposed model's performance on popular datasets and compare it to different state-of-the-art methods. We investigate uncertainties in the model and data and evaluate its effect on performance.
Such a model may be applied in settings, where a high confidence classification for out-of-domain examples may have severe consequences. For example in medical settings, in law, or autonomous driving. There, in cases of high uncertainty, a human could be notified and consulted, preventing the model from making a (potentially wrong) high-risk decision. The model would still provide a decision tree as an explanation, which could be utilized by a human user to make a final decision.

Chapter 2

Related Work

Building on, often centuries-old, mathematical foundations, and on more recent psychological and neurobiological insights, machine learning has evolved throughout the last decades and a vast corpus of literature has emerged from this development. Here we will briefly go over some historic aspects, introduce important concepts, and give an overview of the current state of explainable methods and uncertainty in machine learning.

2.1 Neural Networks

Artificial neural networks (ANNs) have become an integral part in the lives of many. Inspired by the cognitive processes of the mammalian brain, several architectures, and learning algorithms have been proposed.

2.1.1 Historical Context

The idea of artificial intelligence (AI) is dating back to antiquity and was debated in various domains ranging from chemistry [58] to philosophy and mathematics [46]. A watershed moment in its development is marked by McCulloch and Pitts [50], who modeled a neuron in an approach that unified findings from neurophysiology and mathematics. Later, Hebb [29] proposed a theory of how the strengthening and weakening of connections between neurons encode gained knowledge. This idea was also applied in the perceptron [68] which could learn to distinguish linearly separable classes. Marvin and Seymour [49] showed that the perceptron was more useful when connecting many and using a middle layer in a so-called multilayer perceptron (MLP). If this hidden layer has a non-linear activation function it could learn to classify not linearly separable data. This idea of many simple but connected units remains a key principle in ANNs. However, simulating the behavior of neural networks requires many computations, and the sequential nature of hardware available at the time prohibited connectionist ideas to be applied in practice.
Over the decades, ANNs have risen and fallen in popularity among AI researchers. Different architectures and learning algorithms have been proposed but many failed to establish. Important milestones were Rumelhart et al. [69], who extended learning rules to MLPs in the backpropagation algorithm, the Long short-term memory by Hochreiter and

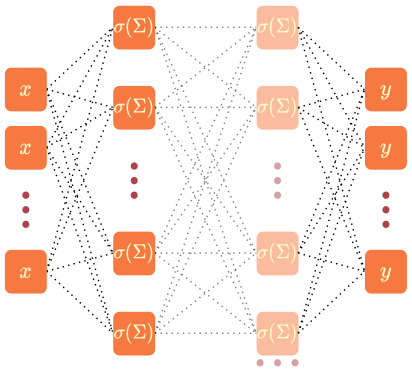

Figure 2.1: A MLP can consists of an input layer, an arbitrary number of hidden layers, and an output layer. Hidden layers typically have non-linear activation functions.

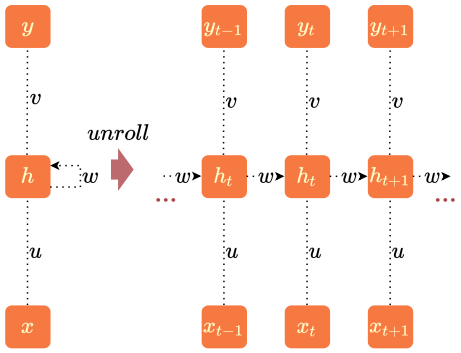

Figure 2.2: The recurrent neural network takes in an input for every time step. Additionally it takes in its own hidden state from the last time step and computes an output for each time step.

Schmidhuber [30], and the convolutional neural net (CNN), proposed by LeCun et al. [45] which allowed to classify handwritten digits.
However, these methods still failed to establish far beyond the realm of research. The development of graphics processing units (GPUs) and the availability of large-scale datasets have caused this to change in the 2010s, where for some tasks ANNs have exceeded human performance [70].

2.1.2 Multilayer Perceptron

The MLP is a rather simple architecture. It consists of an input layer, an arbitrary number of hidden layers, and an output layer. Each neuron is typically connected to each neuron in the next layer through weighted connections. Thus each neuron gets the weighted output from the last layer's neurons as input. Non-linear activation functions allow the network to learn not linearly separable problems. Their rather simple structure and effectiveness make them a widely used tool [4]. In the RDTC model, $f_{AttrMLP}$, $f_{QuestMLP}$, and $f_{ClassMLP}$ are used to lean mappings from the complex output of other models to a simpler structure, that can i.e. be interpreted by a human. For example from ResNet features to attributes.

2.1.3 Recurrent Neural Networks

Many types of data don't provide samples of fixed length. This is the case for most language or time-series data among others. An MLP, however, is incapable of processing such samples due to its fixed amount of input neurons. To alleviate this, recurrent neural networks (RNNs) use the output of their neurons as input after being multiplied with a weight matrix. Many concepts of RNNs exist [23]. Here we will have a closer look at a very simple example, depicted in Figure 2.2. The RNN takes in an input in_t which is transformed by weight matrix u and added to the hidden state of the last time step times weight matrix w. This forms the hidden state of the current time step. To obtain an output

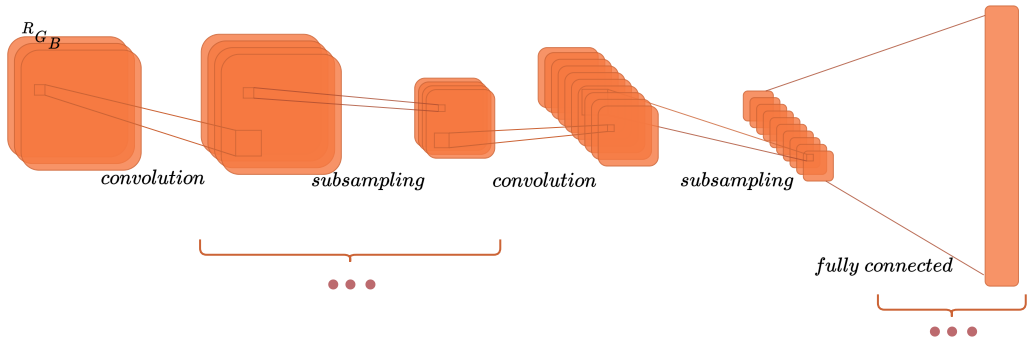

Figure 2.3: A CNN with two convolutional and subsampling layers. A fully connected layer transforms the learned features into a human interpretable output. The displayed structures can be repeated an arbitrary number of times. Modern convolutional architectures may have have a large number of layers [28, 75, 81].

at a given time step, the hidden state is multiplied with a weight matrix v. The update rules can be stated as:

$$h_t = f_{act}(u \circ in_t + w \circ h_{t-1}) \tag{2.1}$$

$$out_t = f_{act}(v \circ h_t) \tag{2.2}$$

An RNN can be trained by the Backpropagation through time algorithm (BPTT). Werbos [78] propose to unfold an RNN for each time step or element of the input. This unfolding makes it equivalent to a feed-forward ANN (such as the MLP introduced in Subsection 2.1.2) and weights can be adjusted according to the gradients.
As basic RNN structures suffer from vanishing and exploding gradient problems [5], several improvements have been proposed such as the LSTM [30] and the Gated recurrent unit (GRU)[11]. The RDTC model relies on an LSTM to classify the decision tree which updates its depth after each question is asked and answered and thus could not be classified by an MLP.

2.1.4 Convolutional Neural Networks

CNN's have proven to be a useful tool for various image-related tasks. They are a type of feed-forward network with the underlying concept of learning filters that correspond to features in the data. Typically this is done over multiple layers to learn to recognize low-level concepts such as edges in layers close to the input layer and high-level concepts in layers further away from the input. In practice, after convolutions and non-linearities are applied, the results are usually subsampled, using a pooling operation to reduce the number of necessary parameters. The AbL agent uses a convolutional neural net to learn features from images.

2.2 Explainable Machine Learning

While interpretable models exist in machine learning, they are often outperformed by modern ANNs with a large number of parameters, on a variety of tasks [24, 25, 71, 63].

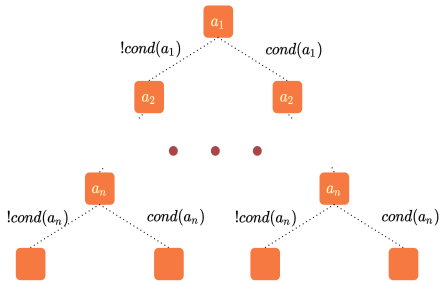

Figure 2.4: A decision tree splits data according to values of attributes.

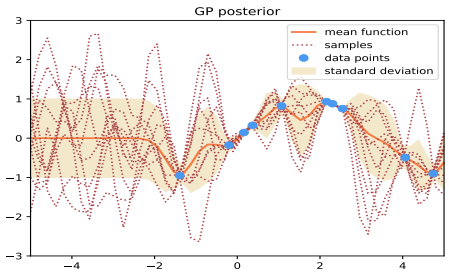

Figure 2.5: A Gaussian process is a distribution over the space of functions that describe data (blue).

Understanding the processes inside a model with a large number of parameters is not trivial. Thus, various approaches have been developed to understand a model's decision process for given data. Such methods, that are interpretable or can explain their decisions are inevitable for safety-critical applications. They allow a human user to understand how a model came to a certain output and allow for a faithful human-AI interaction to emerge.

2.2.1 Decision Trees

Decisions trees provide an inherently interpretable model. Given data with features $a_1, ..., a_n$ a decision tree iteratively splits data according to conditions for each attribute. Splits occur for each attribute a_i where samples from data are either above or below a given threshold for their values of a_i. Optimal splits can be found by learning algorithms or by expert knowledge. Nodes represent the resulting splits and edges represent the corresponding answer. Decision trees are a frequently used tool in data mining [79]. Popular algorithms to generate decision trees from data include Iterative Dichotomiser 3 and C4.5, both suggested by Quinlan [64, 65]. Another popular algorithm is Classification and Regression Trees, introduced by Breiman et al. [9]. The key concept behind these algorithms is to create splits that maximize information gain for each available attribute.
To reduce the size and increase the interpretability of generated trees, pruning can be applied [52]. Pruning can also reduce the risk of overfitting ([35]).
The RDTC builds a decision tree by asking questions about the presence of attributes in a given image. Questions by the RDT can be viewed as nodes and the AbL's answers as edges.

2.2.2 Leveraging Attributes

While decision trees are suitable for tabular data, applying them to image data is not trivial. Images are parameterized by pixels rather than attributes, so attributes have to be provided as side information. Ideally, those attributes should correspond to features that characterize classes found in the image data. Therefore, a popular way of acquiring side information is through human labeling of data according to characteristics suggested by domain experts. Wah et al. [77] let users label different kinds of bird species according to attributes and collect them in the CUB dataset. Another dataset containing attributes as

side information is proposed by Xian et al. [80].
Attributes can be used to create an interpretable output of a neural network. Akata et al. [1] propose to learn a linear mapping between feature representations learned by a neural network and learned attribute representations. In other areas, attributes are used as well. Kulkarni et al. [41] and Ordonez et al. [59] use attributes for image captioning. Lampert et al. [43] and Palatucci et al. [61] use attributes for zero-shot learning.

2.2.3 Multi-agent Communication

Communication between different agents is a popular technique in reinforcement learning [27, 44, 10, 33, 14]. Foerster et al. [19] propose to let one agent send messages containing categorical symbols that can be used by a second agent to solve a problem. Multi-agent communication has also been used to create interpretable outputs by Rodriguez et al. [67] who take into account that different agents may have different understandings of concepts. Communication between two agents is also the basis for learning in the RDTC.

2.3 Uncertainty

Uncertainty is an important factor in human decision making. Modeling uncertainty in machine learning methods is useful for ensuring responsible decision making of algorithms and allows a user to build trust in these models. Uncertainty can arise from data. This is called aleatoric uncertainty. It compromises all kinds of noise inherent to the data. If this noise is static (i.e. limited precision of a sensor) it is called homoscedastic uncertainty. If noise is not static (i.e. occlusions in an image) the uncertainty is called heteroscedastic uncertainty.
On the other hand, a model can also be uncertain about its parameters. This uncertainty is called epistemic uncertainty and is high if a model has seen few data. By giving it more data, it can be reduced. Those two kinds of uncertainty are displayed for a computer vision task in Figure 2.6.

2.3.1 Bayesian Inference

Applying Bayes' theorem to update prior beliefs as more data becomes available is called Bayesian inference [8]. Bayes' theorem was introduced by Bayes [3] and is stated as

$$p(a|b) = \frac{p(b|a) \cdot p(a)}{p(b)}.$$

Intuitively put, it computes the posterior conditional probability of a random variable a given, the likelihood of b being true, given a is true: $p(b|a)$, a prior believe $p(a)$ and a marginalization term $p(b)$. Methods of Bayesian inference are popular in different fields of science, such as neurology, psychology, medicine, epidemiology, geography and machine learning [20, 76, 38, 62, 16, 22]. They can be used to incorporate prior knowledge, multiple sources of evidence, or, in general, construct large joint probability density functions [74].

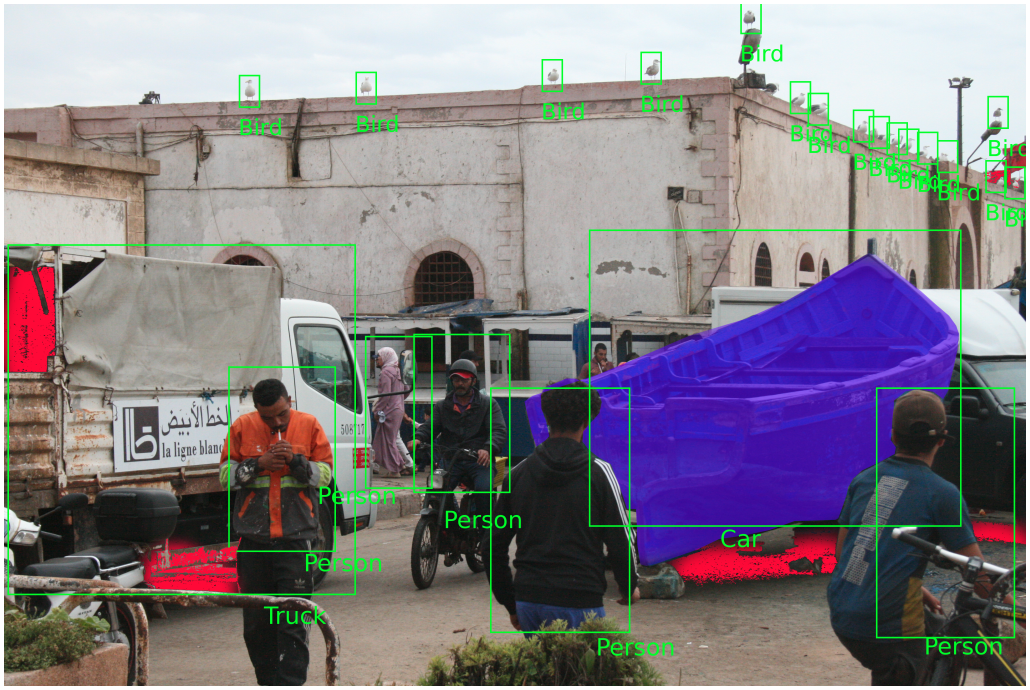

Figure 2.6: Areas with high aleatoric uncertainty are marked red. These are mostly due to occlusions in the image. The area in purple is due to high epistemic uncertainty since the model has never seen a boat before. It misclassifies it as a car but is uncertain about this classification.

2.3.2 Variational Inference

Methods of variational inference are used when intractable integrals are encountered in Bayesian inference [34]. Suppose, we want to find the predictive posterior $p(y^*|x^*, X, Y)$ for some datapoint x^*, we need to compute

$$p(y^*|x^*, X, Y) = \int p(y^*|x^*, w)p(w|X, Y)dw.$$

The true posterior over model parameters, given data $p(w|X, Y)$ is not trivial to find an is often intractable in practice. Therefore an approximate distribution $q_\theta(w)$, with optimizable parameters θ is used. Ideally, $q_\theta(w)$ should be similar to the true posterior. Thus, we minimize the KL-divergence between the approximate distribution and the true posterior.

$$KL(q_\theta(w)|p(w|X, Y)).$$

Given an approximate distribution, the predictive posterior can be written as

$$p(y^*|x^*) \approx \int p(y^*|x^*, w)q_\theta(w)dw.$$

To find the variational distribution $q^*_\theta(w)$ with the optimal set of parameters, we have the objective

$$\mathcal{L}_{VI} := \int q_\theta(w) log\ p(Y|X, w)dw - KL(q_\theta(w)|p(w))$$

that is optimized with respect to parameters θ. Here, $p(w)$ is a prior that is often assumed to be a standard Gaussian. According to Bishop [6], maximizing the log evidence lower bound is equivalent to minimizing the KL-divergence but does not require having access to the true posterior $p(w)$. This effectively leaves us with an optimization task where we need to compute derivatives instead of integrals. This, in turn, allows us to apply powerful automatic differentiation tools, such as deep learning frameworks.

2.3.3 Gaussian Processes

Gaussian processes are a powerful, generative model that can be applied to a variety of machine learning tasks [48]. They build on the notion that data can be described by (possibly infinitely) many functions. Gaussian processes (GPs) assign probabilities to functions that describe given data. Starting with a prior distribution over functions a GP returns a posterior over the space of functions, given new data. Of course, distributions over functions are somewhat hard to define. Therefore, we will view a function $f(x)$ as a vector where the x denotes the index that will retrieve the function value at this point. Now, a GP is a joint normal distribution over the points of this vector, defined by a mean function $\mu(x)$ and a covariance function $\Sigma(x)$ with $\Sigma(x) = \Sigma_{ij} = k(x_i, x_j)$. The kernel function k has to return a positive definite matrix and should return a high output for similar x_i and x_j.
GPs thus view each x_i as a random variable, that is dependent on all other $x_j \neq x_i$.
To perform inference in a GP, one simply takes the conditional probability density distribution (PDF) of a given x_i conditioned on all other $x_j \neq x_i$. Since Gaussian distributions are closed under conditioning, the resulting PDF is Gaussian as well.
To retrieve a meaningful posterior from a GP, it has to be trained in a supervised fashion. We will describe the key steps in pseudocode.

-create a prior-
x_{test}= $\{x_n\}$
K_{prior} = kernel(x_{test}, x_{test}, w)
L_{prior} = cholesky_decomposition(K_{prior})
$f_{prior} = L_{prior} \cdot \mathcal{N}(dim = (n, num\_samples); 0, 1)$
-take new data into account and compute new posterior-
x_{train}, y_{train} = training data
K = kernel(x, x, w)
L = cholesky_decomposition(K)
K_s = kernel(x_{train}, x_{test}, w)
L_k = solve(L, K_s)
$\mu = L_k^T \cdot$ solve(L, y_{train})
-draw samples from posterior-
L = cholesky_decomposition(K_{ss} - $L_k^T \cdot L_k$)
$f_{posterior} = \mu + L \cdot \mathcal{N}(dim = (n, num\_samples); 0, 1)$
- prediction at test point x_i is now the sample's value at position i

2.3.4 Dropout Variational Inference

Deep neural networks (DNNs) are applied in safety-critical processes where information about a model's uncertainty is of crucial importance [54, 40]. Standard DNN's however do not yield uncertainty information. Bayesian neural networks (BNNs) are an architecture that can model uncertainty reliably [53]. However, they require modeling a probability density function over each parameter which causes high computational costs. A method using variance arising from dropout has been proposed by Gal and Ghahramani [21] who proofed that dropout in neural networks is equal to Monte-Carlo integration in GPs and thus can be used to model epistemic uncertainty.
With a GP, we want to find a posterior distribution over function that explains given data. We can view a finite model like a neural network as an approximation to a GP [13]. Thus, by optimizing a model, we minimize the Kullbach-Leibler (KL) divergence between our finite model and the corresponding GP [21]. In practice, however, the GP's posterior often requires computing intractable integrals. To alleviate this, methods of variational inference, such as Monte-Carlo integration are applied. Monte-Carlo integration in a GP corresponds to averaging forward passes in a dropout neural net. According to Gal and Ghahramani [21], the resulting variance can be viewed as the model's uncertainty.
Additionally, Kendall and Gal [37] propose to combine aleatoric and epistemic uncertainty in a unified approach. For classification tasks, this has been simplified by Kwon et al. [42].

Chapter 3

Uncertain RDTC

The Recurrent decision tree model, proposed by Alaniz and Akata [2], consists of two communication agents. A Recurrent Decision Tree agent (RDT) and an Attribute-based Learner agent (AbL). Their common goal is to solve a classification task through communication. The RDT has no direct access to data and can only ask questions regarding attributes. Given answers by the AbL, it can then store the answers in a memory and thus incrementally build a decision tree that is used for the final classification. The learned decision three allows introspection.
Based on their work, we propose a model, where the AbL is aware and able to express its uncertainty for given attributes (See Figure 3.1). We use this uncertainty information by either preventing the RDT from using uncertain attributes or by allowing the AbL to answer with 'I don't know' additionally to the options 'Yes' or 'No'[1].

3.1 RDTC

Before we introduce our modifications to the RDTC model, we will give a detailed explanation of its architecture, learning algorithm, and how the two agents communicate.

3.1.1 Communication

The two agents communicate with each other through discrete messages. For a given image x, the AbL returns a an answer $\hat{a}$ that is a tensor with binary values with size number of attributes $\times$ decision size. The intention behind using only binary values in the AbL's answer is increased interpretability. Each row in this tensor indicates the presence/absence of an attribute and the columns correspond to the decision options 'Yes' or 'No'. Strictly speaking, the AbL thus answers all possible questions the RDT can ask, in advance to any questions.
The RDT is then able to ask questions by posing an index c_t. By accessing the AbL's answer at this index it receives answer $d_t = \hat{a}[d_t]$. This information is then used to build an internal

[1] A Pytorch implementation of our model can be found at `github.com/wastedsummer/urdtc`

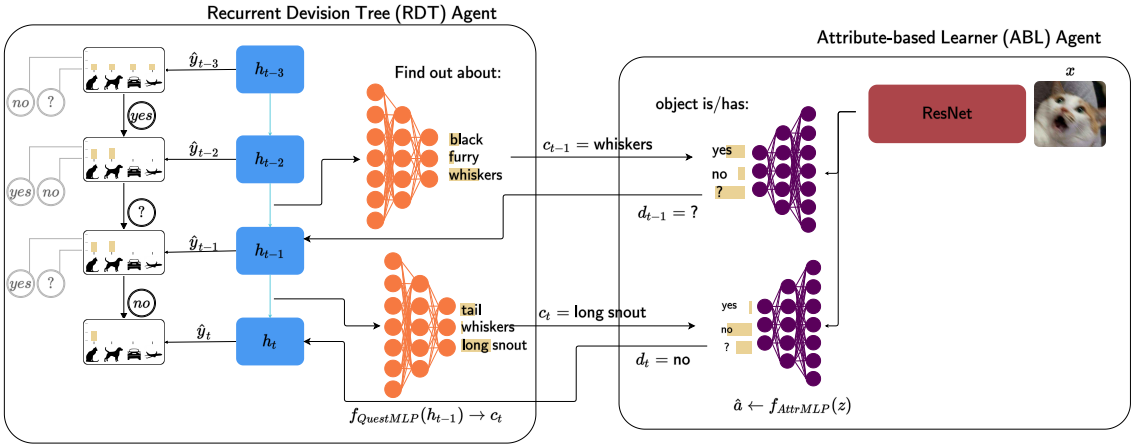

Figure 3.1: The RDT asks questions about presence, absence or uncertainty of attributes. The answers, given by the AbL are used by the RDT agent to make a classification each iteration.

decision tree, used for a classification output and to come up with a new question. In the following, we will go over each agent in more detail and explain how this model is trained.

3.1.2 AbL

The AbL agent is a vision model, consisting of a CNN and an MLP. The CNN learns image features and the MLP learns a mapping from features to human-annotated attributes (if available). For all experiments, we use a ResNet [28] to extract features. For a given image x, the CNN extracts feature vector z. This is passed to the $f_{AttributeMLP}$ and the resulting vector of logits is put through a softmax with temperature (TempSoftmax, See Eq. 3.1). The output is a tensor $p(\hat{a}|z)$ with shape $number\_attributes \times decision\_size$. This corresponds to a probability distribution over the decision option for each attribute. To get an interpretable, binary response, in the forward pass, the result of the TempSoftmax is put through an arg max function. This is, however, not differentiable. In the backward pass, it is therefore replaced by the identity function. Thus, the final response of the AbL agent for a given image x is $\hat{a} = argmax(p(\hat{a}|z))$. The RDT agent can then access answers for specific attributes by indexing.

$$TempSoftmax(log\ \pi) = \frac{exp((log\ \pi_i)/\tau)}{\sum_{j=1}^{K} exp((log\ \pi_j)/\tau)} = d_t, \tag{3.1}$$

where $log\ \pi$ are the logit values for either 'yes', or 'no' per attribute. (3.2)

3.1.3 RDT

The RDT agent consists of several models. At its core lies a Long-short Term Memory (LSTM). This recurrent model allows the RDT to come up with new questions, based on it's hidden state h_{t-1} from the last (or initial) communication step.

The $f_{QuestMLP}$ is responsible for thinking of new questions by returning an index that can be used to access attribute information in the response from $f_{AttributeMLP}$. The output of $f_{QuestMLP}$ are logit values $log\ p(c_t|h_{t-1})$ over possible indices $c_t\ in\{1,2,...,|A|\}$. To get an actual index, the final output needs to be discrete. Thus we need to turn the logit values into a categorical distribution and sample from it. This is done via the Gumbel softmax function [32]:

$$GumbelSoftmax(log\ \pi) = \frac{exp((log\ \pi_i + g_i)/\tau)}{\sum_{j=1}^{K} exp((log\ \pi_j + g_j)/\tau)}\ \text{, where} \tag{3.3}$$

$$log\ \pi = log\ p(c_t|h_{t-1})\ \text{, and thus:} \tag{3.4}$$

$$c_t = GumbelSoftmax(log\ p(c_t|h_{t-1})). \tag{3.5}$$

where for $\tau \to 0$, the distribution approaches a one-hot vector with a 1 at the maximum of the categorical distribution and 0 everywhere else, turning it into a sample, effectively.
The resulting tensor c_t then can be used to index $\hat{a}$. The response is d_t which indicates absence, presence (or other possible answers) of the attribute at index c_t. Question c_t and answer d_t then update the explicit memory $\mathcal{M}^{(t)} = \mathcal{M}^{(t-1)} \oplus (c_t, d_t)$. Note that during test time, rather than sampling c_t, the index with the highest probability is chosen through an argmax because it does not need to be differentiable.
The explicit memory $\mathcal{M}^{(t)}$, c_t, d_t and the LSTM's last hidden state h_{t-1} are then used to compute the current hidden state h_t, which is then, in turn, used by the $f_{QuestMLP}$ to pose a new question.
The explicit memory $\mathcal{M}^{(t)}$ is also the basis for the class prediction. An additional MLP, the $f_{ClassMLP}$ returns class probabilities $\hat{y}_t$ based on $\mathcal{M}^{(t)}$ for every communication step:

$$\hat{y}_t = f_{ClassMLP(\mathcal{M}^{(t)})}. \tag{3.6}$$

The classification is the final output of the model and is used to calculate a loss that can be optimized.

3.1.4 Learning

Ideally, the model should come to a classification in as few communication steps as possible. A tree resulting from few communication steps is shallow and easier to interpret. To keep the model from using deep trees, the classification loss is divided by the number of time steps (communication steps):

$$\mathcal{L} = \frac{1}{T}\sum_{t=1}^{T} \mathcal{L}_{CE}(y, \hat{y}_t) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{i} y_i \cdot log\ \hat{y}_{t,i}. \tag{3.7}$$

This ensures, that the lower the number of time steps T, the lower the whole term, and thus, the lower the loss.
Besides the classification loss, an additional attribute loss can be leveraged to encourage learning attributes that correspond to provided human-annotated side information.

$$\mathcal{L} = \frac{1}{T}\sum_{t=1}^{T}\Big[(1-\lambda)\mathcal{L}_{CE}(y, \hat{y}_t) + \lambda\mathcal{L}_{CE}(\alpha_{y,c_t}, \hat{\alpha}_{c_t})\Big] \tag{3.8}$$

How much each loss contributes to the total loss can be determined by hyperparameter λ.

3.2 Uncertainty

Uncertainty information is a crucial part of robust vision systems. In safety-critical application, it could be used to prevent the model from using high-risk strategies by leveraging attributes it is highly uncertain about for its classification. Or, if all attributes are deemed highly uncertain (for example on completely ood data), the model could decide to consult a human instead of making an overconfident classification.
Uncertainty comes either from variance in data (aleatoric uncertainty) or from variance in model parameters (epistemic uncertainty). Thus, there is an inherent relationship between variance and uncertainty. The variance of a random variable, which in our case is the variational predictive distribution $p(\hat{y}|x)$ with $y \sim p(\hat{y}|x)$, is defined as:

$$Var_q(p(\hat{y}|x)) = \mathbb{E}_q\left[(y - \mathbb{E}[y])^2\right] \tag{3.9}$$

$$= \mathbb{E}_q\left[yy^T\right] - \mathbb{E}_q[y]\,\mathbb{E}_q[y]^T. \tag{3.10}$$

To decompose the term into variance arising from data and variance from the model q itself, according to Kwon et al. [42], we can write it as:

$$Var_q(p(\hat{y}|x)) = \mathbb{E}_q\left[yy^T\right] - \mathbb{E}_q[y]\,\mathbb{E}_q[y]^T \tag{3.11}$$

$$= \underbrace{\int_\Omega \left[diag(\mathbb{E}_{p(\hat{y}|x,w)}[\hat{y}]) - \mathbb{E}_{p(\hat{y}|x,w)}[\hat{y}] \cdot \mathbb{E}_{p(\hat{y}|x,w)}[\hat{y}]\right] q_\theta(w)dw}_{aleatoric} \tag{3.12}$$

$$+ \underbrace{\int_\Omega \left[\mathbb{E}_{p(\hat{y}|x,w)}[\hat{y}] - \mathbb{E}_{q_\theta(\hat{y}|x,w)}[\hat{y}]\right]\left[\mathbb{E}_{p(\hat{y}|x,w)}[\hat{y}] - \mathbb{E}_{q_\theta(\hat{y}|x,w)}[\hat{y}]\right]^T}_{epistemic}. \tag{3.13}$$

3.2.1 Estimating Epistemic Uncertainty with Dropout Neural Networks

Epistemic uncertainty corresponds to the variance of the model's parameters. In a traditional dropout neural network, there is no random dropping out of neurons in the test stage, and weights are fixed. This would result in no variance of model outputs for multiple forward passes given an input. However, dropping out neurons in a random manner results in an inherent variance of model parameters. Gal and Ghahramani [21] proof that this variance corresponds to epistemic (model) uncertainty. Here, we will go through their proof.
The key idea is that a dropout neural net is equivalent to an approximation of a Gaussian process. To randomly dropout neurons, we sample binary values that are either 0 or 1 for every neuron. The sampled value is then multiplied with the neuron's value. In the following, the dropout neural network's layers are denoted by L, the respective weight matrices are denoted by $W_{l \in L}$, the bias vectors by b_l, the network output, given datapoint $x_i \in X$ with size N by $\hat{y}_i$, and the true value by $y_i \in Y$. The objective of the network is to minimize a given loss function $E(y_i, \hat{y}_i)$. Additionally, a weighted regularization term can be added to the loss function. A popular technique is to use L_2 regularization, weighted by

a weight decay term λ. The final resulting loss term is:

$$\mathcal{L}_{dropout} := \frac{1}{N}\sum_{i=1}^{N} E(y_i, \hat{y}_i) + \lambda \sum_{l=1}^{L}(||W_l||_2^2 + ||b_l||_2^2).$$

For the special case of a single hidden layer dropout network (which we will consider for simplicity), we can rephrase this objective as

$$\mathcal{L}_{dropout} := \frac{1}{N}\sum_{i=1}^{N} E(y_i, \hat{y}_i) + \lambda(||W_1||_2^2 + ||W_2||_2^2 + ||b||_2^2) \tag{3.14}$$

, where W_1 is the weight matrix connecting the input to the hidden layer, and W_2 connecting the hidden to the output layer. The bias b_2 Let us take a step back to a very general point of view. A problem, we typically encounter in machine learning is how to predict a function value y^* from a datapoint x^*. In Bayesian terms, we want to find the predictive posterior

$$p(y^*|x^*, X, Y) = \int p(y^*|x^*, w)p(w|X, Y)dw \tag{3.15}$$

where w is a set of adjustable parameters. To model our function of interest well, we need to find optimal values for w. If we break down Equation 3.15 further, we get

$$p(y^*|x^*, X, Y) = \int \underbrace{p(y^*|x^*, w)}_{\mathcal{N}(\underbrace{y; \hat{y}(x,w)}_{\hat{y}(x,w=\{W_1,W_2,...,W_n\})}, \tau^{-1}\mathbf{I}_D)} \cdot \underbrace{p(w|X, Y)}_{intractable} dw \tag{3.16}$$

, where we see that the posterior over w, given data is intractable. Therefore we need to approximate it with variational distribution $q_\theta(w)$. We want this distribution to mimic the true posterior closely, so we want to find

$$argmin_\theta KL(q_\theta(w)|p(w|X, Y)).$$

As we have seen in Section 2.3.2, this would require access to the true posterior, which we cannot compute. Therefore we maximize the log evidence lower bound

$$\mathcal{L}_{VI} := \int q_\theta(w) log p(Y|X, w)dw - KL(q_\theta(w)||p(w))$$

and assume $p(w)$ to be a standard Gaussian. As discussed also in Section 2.3.2, we effectively turn computing integrals into an optimization problem, where we need to compute derivatives. Using the optimized variational distribution $q_\theta^*(w)$, we can compute an approximate predictive posterior

$$q_\theta(y^*|x^*) = \int p(y^*|x^*, w)q_\theta^*(w)dw$$

for a given datapoint x^*. This in turn can be approximated with

$$q_\theta(y^*|x^*) = \sum_{t=1}^{T} p(y^*|x^*, w_t)$$

where each w_t is a sample from our variational distribution $p^*_\theta(w)$.
Now, in Gaussian processes, as discussed in Section 2.3.3, we need a covariance function of the form

$$K(x,y) = \int \mathcal{N}(w;0,l^{-2}I_Q)p(b)\sigma(w^T x+b)]sigma(w^T y+b)dwdb$$

to retrieve a covariance matrix. To avoid computing integrals, this covariance function can be approximated with

$$\hat{K}(x,y) = \frac{1}{K}\sum_{K}^{k=1}\sigma(w_k^T x+b_k)\sigma(w_k^T y+b_k)$$

where w_k is sampled from $\mathcal{N}(0,l^{-2}I_Q)$ and b_k from $p(b)$. This is called Monte-Carlo integration [51].
We get the following predictive posterior for our GP:

$$w_k \sim \mathcal{N}(0,l^{-2}I_Q), w_d \sim \mathcal{N}(0,l^{-2}I_K), b_k \sim p(b)$$
$$W_1 = [w_k]_{k=1}^K, W_2 = [w_d]_{d=1}^D, b = [b_k]_{k=1}^K, w = \{W_1, W_2, b\}$$
$$p(y^*|x^*,w) = \mathcal{N}\left(y^*; \sqrt{\frac{1}{K}}\sigma(x^* W_1+b)W_2, \tau^{-1}I_N\right)$$

, and finally:

$$p(y^*|x^*,X,Y) = \int p(y^*|x^*,w)p(w|X,Y)dw.$$

We use $q_\theta(w) = q_\theta(W_1)q_\theta(W_2)q_\theta(b)$ as an approximation to $p(w|X,Y)$, where

$$q_\theta(W_1) = \Pi q_\theta(w_q), \text{ and } q_\theta(w_q) = p_1\mathcal{N}(m_q, s^2, I_K) + (1-p_1)\mathcal{N}(0,s^2,I_K), \tag{3.17}$$
$$\text{with } p_1 \text{ as a probability } \in [0,1], s>0, \text{ and } M_1 = \left[m_q\right]_{q=1}^Q \in \mathbb{R}^{K\times D} \tag{3.18}$$

We eventually want to recover a dropout neural network (as specified in Eq. 3.14) from our GP. In our dropout network, we rely on sampling from a Bernoulli random variable. In our GP, we, so far, sample from a mixture of Gaussian distributions. We cannot change this, because the KL-divergence between a discrete distribution (i.e. Bernoulli) and a continuous distribution (i.e. Gaussian) is not defined. Therefore we will make two assumptions about the Gaussian mixture in Eq. 3.17.
We set p to a fixed value. Also, we want the second term of the sum to be zero. To sample zero from a zero-centered Gaussian distribution with a very high probability, we simply set the standard deviation to a very small value. Small enough, so that a device with limited precision will view it as zero and always return zero. These two assumptions leave us with a Gaussian distribution that resembles a Bernoulli distribution. Thus, we are effectively sampling from a Bernoulli random variable in our GP, with a still defined KL-divergence. We get

$$\mathcal{N}_{GP-MC} \approx logp(Y|X,\hat{w}) - \frac{p_1 l^2}{2}||M_1||_2^2 - \frac{p_2 l^2}{2}||M_2||_2^2 - \frac{l^2}{2}||m||_2^2, \text{ with } \hat{w} \sim q_\theta(w)$$

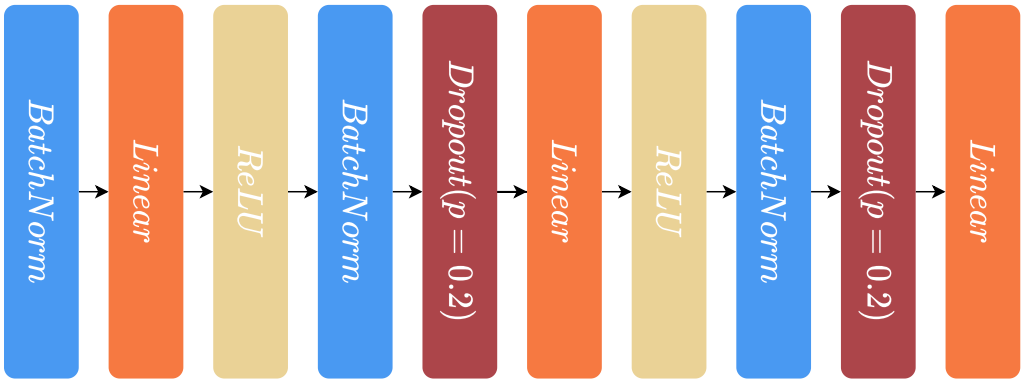

Figure 3.2: We used both BatchNorm and dropout layers for $f_{attrMLP}$.

as approximated log evidence lower bound. To recover the dropout objective from Eq. 3.14, we can scale it by a constant $\frac{1}{N\tau}$.

$$\mathcal{L}_{GP_MC} \propto -\frac{1}{2N}\sum_{n=1}^{N}||y_n - \hat{y}_n||_2^2 - \frac{p_1 l^2}{2N\tau}||M_1||_2^2 - \frac{p_2 l^2}{2N\tau}||M_2||_2^2 - \frac{l^2}{2N\tau}||m||_2^2$$

and set hyperparameters τ and l to appropriate values. According to Gal and Ghahramani [21], this generalizes to deep GPs and Dropout networks with more than one hidden layer, as well.

3.2.2 Getting Uncertainty Information

The RDTC model does not use dropout layers in the vision model. As we want to keep the ResNet as a feature extractor and be able to estimate uncertainty, we introduce dropout layers to $f_{AttributeMLP}$. We tested different configurations for $f_{AttributeMLP}$ and found that using a combination of dropout and batchnorm layers between, fully connected layers and non-linearities worked best. The setup is displayed in Figure 3.2. To retrieve uncertainty information, given the image features z from the ResNet, we do n forward passes and compute the standard deviation of the results. As proofed by Gal and Ghahramani [21] and discussed in Section 3.2.1, this is equal to approximated epistemic uncertainty. So theoretically, for examples, that are far from the distribution, the model was trained on, it should yield high epistemic uncertainty.
After uncertainty estimation, we use two different strategies of making the model aware of its uncertainties. In the first strategy, we replace selection logits from $f_{QuestMLP}$ that are on indices with high uncertainty by $-inf$ so they can not be picked by the Gumbel softmax. We call this strategy 'remove uncertain attributes' (remRDTC). In another strategy, uncertain attributes are denoted with a 1 in an extended decision tensor. We refer to this strategy as 'extended vocabulary' (extRDTC).

3.2.3 Removing Uncertain Attributes

After computing uncertainty for each attribute, in this strategy, we simply manipulate the logits tensor from $f_{QuestMLP}$ so that at each index, where an uncertain attribute is, the

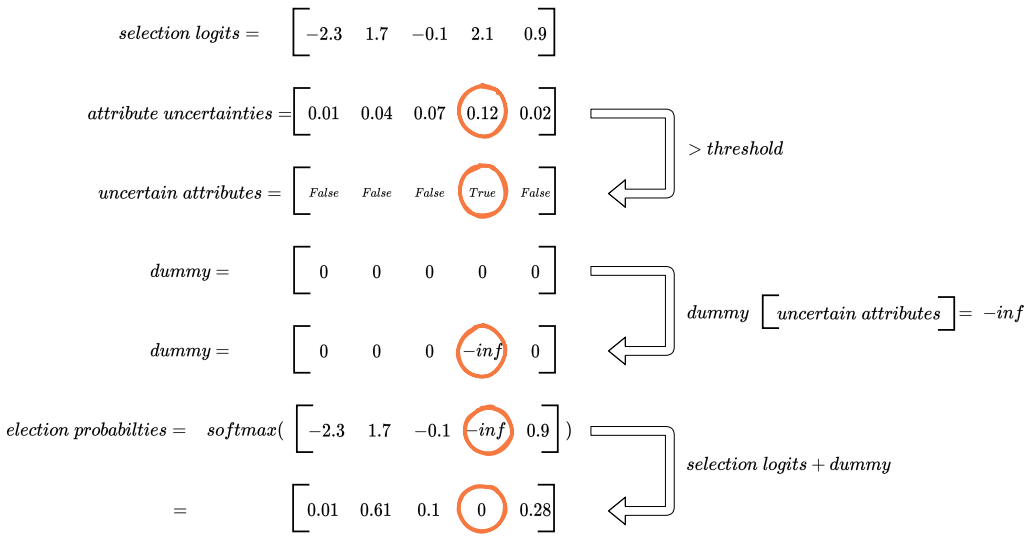

Figure 3.3: In selection logits, the output from $f_{QuestMLP}$, we replace values at indices where uncertain attributes (orange) are, with a $-inf$ value. This assures that this attribute can never be picked as c_t by the Gumbel softmax.

selection logit tensor has a $-inf$ value. This ensures that this attribute cannot be picked by the Gumbel softmax. In turn, no uncertain attribute can be picked as an index and therefore cannot be posed as a question to the AbL.
To prevent the model from misusing uncertainty information, we have to ensure that the model does not get any gradients where uncertain attributes are. This is done by detaching all attributes, affiliated with uncertainty from the computational graph. The n forward passes to compute variance are therefore done without adding to the computational graph. Tensors created based on the uncertainty tensor, also do not carry gradient information. Thus, in the masked selection tensor (Figure 3.3) indices of uncertain attributes have a gradient of 0.

3.2.4 Extending the Vocabulary

Here, we resort to an extended vocabulary of answers for $f_{AttrMLP}$. Instead of just allowing *Yes* and *No*, we also allow the AbL to answer with ?. Thus, $f_{AttrMLP}(z) \in \{0,1\}^{num\_attributes \times 3}$. However, since, $f_{AttrMLP}(z) \in \{0,1\}^{num\_attributes \times 2}$ we need to append the uncertainty information to d while avoiding conflicting answers. This process is shown, for a two attribute example in Figure 3.4. After receiving the output from $f_{AttrMLP} = d_{init}$ we initiate the uncertain column ? with a vector of 0s and append this to the d_{inint} to create d_{temp}. After computing uncertainty vector ?, we also create a negated version ! of it. This indicates all attributes where the model is certain. Subsequently, those are now used to clear rows. Vector ! has 0s where the model is uncertain. We use this to replace values in rows of uncertain attributes with 0s and get $d_{cleared}$. We also initiate a dummy tensor, that is of the same dimension as the $d_cleared$ and is all 0s but in the column to the very right, where it is all 1s. Rows of this tensor are now cleared, using ?, wherever ? has zeros, thus only rows with uncertain attributes remain and have a 1 in the column responsible for uncertainty.

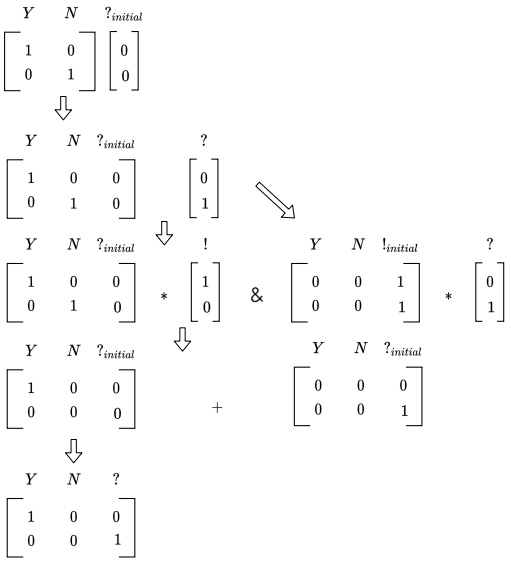

Figure 3.4: The original answer by the AbL is merged with uncertainty information so that the answer for each attribute is either 'yes', 'no' or '?'. Here we show a small example where attribute size is only 2.

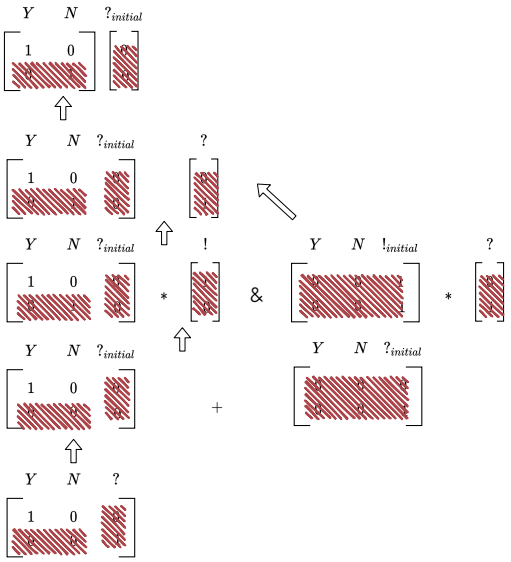

Figure 3.5: Gradients coming from attributes that are deemed uncertain are detached from the computational graph. Other elements in the tensor are either 0 (so are their respective gradients), or stem from leaf nodes.

Finally, this tensor and $d_cleared$ are added up to create our final decision d which can now be indexed by the RDT.
When using this strategy in training, we need to make sure, that no gradients come from uncertainty, so it is not misused and maximized in order for the model to profit from the additional column. We need to ensure that gradients only come from attributes, where the model is certain. For the operation, as described above, this is the case. We detach ? and ! from the computational graph. The dummy tensor is a leaf node, so no gradient is computed. Also, where values in d are 0, their activation's derivative is 0, so gradients are 0. This is the case in rows of uncertain attributes. Which gradients are used and which are blocked in the previously used example is displayed in Figure 3.5.

3.2.5 Random Attribute Removal

As a random baseline, to compare our remRDTC, and extRDTC to, we introduce 'random attribute removal' (randRDTC). Similar to the remRDTC strategy, described in Subsection 3.2.3, we prevent the RDT from asking questions. However, instead of blocking indices where uncertain attributes are, we randomly generate a tensor and block indices based on it. The probability of an index being blocked is similar to the probability of an attribute being deemed uncertain to allow fair comparison.

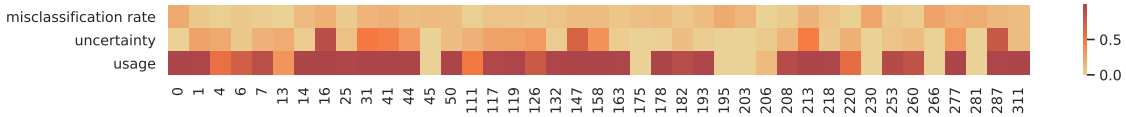

Figure 4.1: Here, we only show normalized attributes misclassification frequency, uncertainty, and usage for attributes that are used by the model.

Chapter 4

Experiments

In this section, we briefly introduce the datasets used in our experiments. We present a quantitative evaluation of the model's uncertainty and its relation to other variables, such as attribute misclassification and attribute usage. Moreover, we investigate, whether the model reliably yields high uncertainty for ood examples. Finally, we evaluate our proposed model on benchmark image classification tasks. We compare the results to those of other explainable models and the current state of the art methods.

4.1 Datesets

We test our model on image datasets that provide attributes as side information. Popular datasets for this setting are Animals with Attributes 2 (AwA2) [80], aPY [18] and CUB [77]. Datasets with attributes are frequently used for zero- or few-shot learning tasks. In the explainable machine learning community, they are popular since attributes provide a human interpretable, natural language parameterization of image features.
AwA2 is a medium scale coarse-grained dataset of 37,322 images that can be subdivided into 50 different classes of animals. Human annotators were tasked to label each image, choosing from 85 available attributes that describe each animal class. The aPY dataset is a small scale coarse-grained dataset that provides 15,339 images, annotated with 64 different attributes that describe each of the 32 classes. Finally, the most challenging dataset is a large scale fine-grained CUB. Here, the task is to classify 11,788 different images into 200 different classes. For this task, each image is annotated with 312 descriptive attributes.

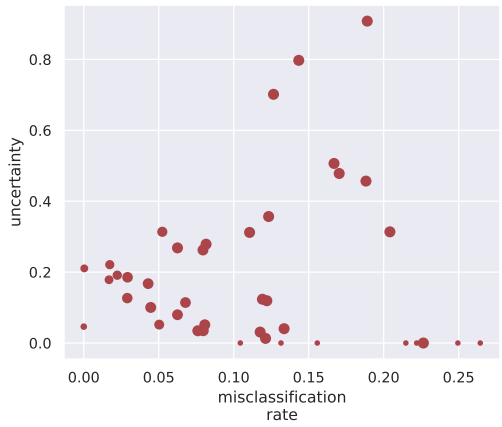

Figure 4.2: Misclassification rates of attributes and their respective uncertainties. The size of the points represents how often they are used. Despite having a low (linear) correlation, the uncertainty seems to be higher for attributes that are often misclassified.

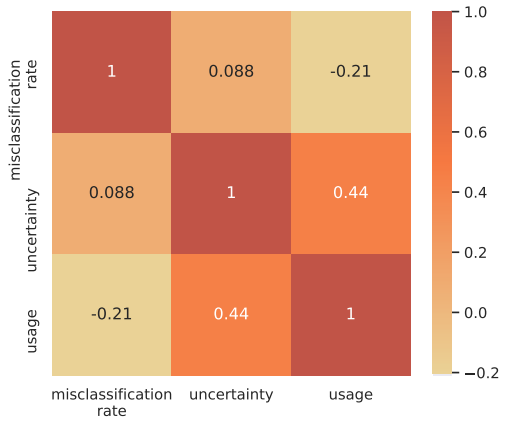

Figure 4.3: Correlations between misclassification rate, uncertainty, and usage of attributes.

4.2 Statistics on Uncertainty

The purpose of the proposed model is not only to evaluate it on benchmark datasets but also to quantitatively investigate uncertainties in the model. We examined correlations between variables such as attribute uncertainty, usage, and misclassification on the CUB dataset. From Figure 4.1 it becomes obvious that only a small subset of the 312 attributes is actually used to make decisions. We can already see, that most of the attributes that are used, however, are rarely misclassified. As apparent in the top row, all attributes here are classified correctly quite reliably. However, there seems to be no obvious correlation between uncertainty and misclassification as among the frequently used attributes (and thus reliably classified correctly) there are some attributes that have high uncertainty values.
In Figure 4.2 we can see that this is true and uncertain attributes appear among all levels of attribute accuracy. Figure 4.3 proves further proofs this as the correlation between the two is close to 0. It also shows a high negative correlation between attribute misclassification frequency and usage. We can view misclassification frequency as inverse accuracy and thus have a positive correlation of 0.5 between usage and accuracy, suggesting that the model actively avoids attributes that it is likely to classify.

4.3 Testing on Out-of-domain Data

An important aspect of our model is that it is not only self-aware regarding uncertainty but also allows for human introspection. Here, we test whether examples, far from the training data, the model has seen cause high uncertainty.

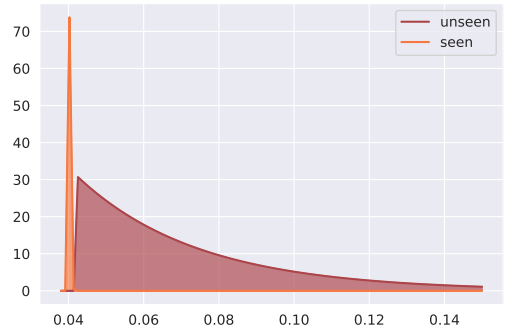

Figure 4.4: Maximum likelihood model for the exponential distribution of uncertainty per example. Examples from classes that have unseen attributes have higher attribute uncertainties.

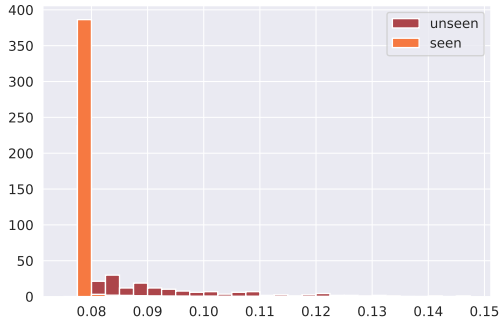

Figure 4.5: Distribution of uncertainty values of seen and unseen attributes in zero-shot-attribute setting (See Section 4.3). The model is more uncertain about attributes it has not encountered in the training set.

4.3.1 Attribute Zero Shot Setting

As a sanity check, we first investigate attribute uncertainty in a zero-attribute setting. We exclude certain classes from training data, so that some attributes are never seen by the model in training. For our setting, we use the CUB dataset, as it is the most challenging out of the introduced three. We construct a setting, where certain attributes do not occur in training data, and thus have to find classes, that have exclusive attributes. In CUB, we find a set of 24 classes that posses 5 exclusive attributes that do not occur in any of the other classes[1].
After training our model on the proposed split, we investigate (1) uncertainties of classes that contain unseen attributes (Figure 4.4) and (2) uncertainties of the unseen attributes themselves (Figure 4.5). For (1), we computed the mean uncertainty per example. For (2), we consider all uncertainty values of each attribute. We observe that the 24 classes, possessing unseen attributes have an overall higher uncertainty. Figure 4.4 displays the modeled distribution of mean uncertainty per example, divided into classes that only posses attributes, occurring in the training data, and classes that possess 5 attributes, never encountered during training. We observe, that the latter have an overall higher uncertainty. Displayed in Figure 4.5, is the distribution of uncertainty values of attributes available during training, and the uncertainty values of attributes, unseen by the model, before the test stage. Here, we also observe an overall higher uncertainty for the latter.

4.3.2 Zero Shot Setting

Further, we investigate the uncertainty of our model in a classical zero-shot setting. We use a split of 150+50, where 150 classes are seen during training and validation. The remaining 50 classes are only seen in the test stage. We compare uncertainties for examples from classes seen during training to uncertainties of examples from unseen classes. Figure 4.6 shows that unseen classes generally have higher uncertainty.

[1] This setting can be replicated by using the *zero_attr_train_test_split.txt* file in the provided Github repository.

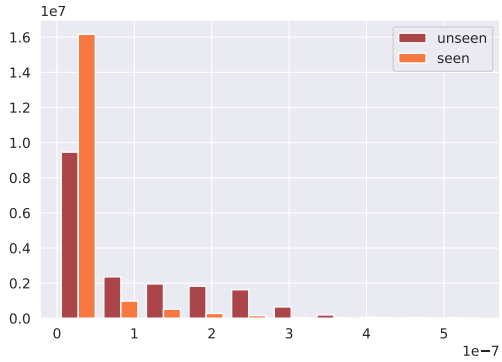

Figure 4.6: Uncertainty values of attributes from seen and unseen classes. Examples from unseen classes have a higher attribute uncertainty.

Figure 4.7: Distribution of uncertainty values of seen and unseen attributes in zero-shot-attribute setting (See Section 4.3). The model is more uncertain about attributes it has not encountered in the training set.

However, the overall most frequent uncertainty value is 0 for both classes. This is due to features, not present in the image, the corresponding attribute, typically the corresponding attribute(s) will yield low uncertainty.

4.4 Results

We compare our model to different other approaches, that either exemplify the current state of the art or are explainable.
ResNet-152 [28] is not explainable. Here, the ResNet was pre-trained on ImageNet data [15] data and then fine-tuned for the specific tasks. We also use a ResNet trained in this manner in our proposed uRDTC model to extract features. When we train the uRDTC model, we keep the weights in the ResNet fixed and only adjust all other parameters in the model.
XDT and DT are both traditional decision tree models that serve as a baseline. For DT Alaniz and Akata [2] propose the following method: DT is given features extracted by the previously introduced pre-trained ResNet. Splits are created according to values in the extracted feature vector. For each split, one feature is chosen and a split is created. This is repeated until either each leaf node corresponds to exactly one class or an early stopping-criterion is met. Moreover, Alaniz and Akata [2] introduce XDT, where each split is given a semantic meaning, for the splits to correspond to interpretable attributes and not only features. The XDT is therefore not given features, directly stemming from the ResNet, but attributes, learned by an MLP that was trained to predict attributes given ResNet features (similar to $f_{AttrMLP}$). They train both DT and XDT using the CART algorithm [9]. As a splitting criterion, they chose the Gini impurity index over entropy-based methods due to its computational advantage [66].
Another explainable model is Deep Neural Decision Forest (dNDF), proposed by Kontschieder et al. [39]. They learn an ensemble of trees by optimizing routing probabilities in each tree's internal nodes. Every path through a tree eventually leads to class distribution, represented

in leaf nodes. All internal nodes are sigmoid-activated, differentiable stochastic functions, parameterized by an optimizable θ. The final prediction is an average over the resulting decision tree forest. One downside of the resulting averaged stochastic routing decisions is the reduced interpretability as they allow for multiple possible routes. The RDTC models only yield on possible route for a given example. Therefore, we consider only an dNDF using a single tree instead of an ensemble mean.

4.4.1 Comparing RDTC to Other Models

	AWA2	aPY	CUB
ResNet [28]	98.2 ± 0.0	85.1 ± 0.6	79.0 ± 0.2
DT	78.0 ± 0.4	64.3 ± 0.6	19.3 ± 0.3
dNDF[39]	97.6 ± 0.2	85.0 ± 0.6	73.8 ± 0.3
RDTC[2]	98.0 ± 0.1	85.7 ± 0.7	78.1 ± 0.2
XDT	73.9 ± 0.9	59.9 ± 1.5	4.9 ± 1.3
aRDTC[2]	98.1 ± 0.0	85.3 ± 0.3	77.9 ± 0.6
remRDTC(ours)	0.0 ± 0.0	0.0 ± 0.0	77.7
extRDTC(ours)	0.0 ± 0.0	0.0 ± 0.0	77.8

4.4.2 Comparing Different RDTC Models

We evaluate different configurations of our model on the three datasets. We compare the original aRDTC model to models using our two proposed strategies of remRDTC and extRDTC. We also include randRDTC as a baseline, that randomly removes attributes, based on the average probability of an attribute having an uncertainty higher than the specified threshold. Additionally to class accuracy, we also consider the model's attribute accuracy.

AWA2

	aRDTC [2]	Random Baseline	remRDTC	extRDTC
Class				
Attribute				

aPY

	aRDTC [2]	Random Baseline	remRDTC	extRDTC
Class				
Attribute				

CUB

	aRDTC [2]	Random Baseline	remRDTC	extRDTC
Class			77.7	77.8
Attribute			77.4	82.6

Chapter 5

Discussion

Uncertainty estimation in deep learning is not trivial. It is an active topic of research and therefore subject of ongoing change and discussion. We will critically reflect on our work and discuss whether epistemic uncertainty (that is only considering attributes) is useful in our setting and how we could alleviate remaining issues. We also take a look at a hypothetical example, on how uncertainty information could be used.

5.1 Increased Convergence Time

Both of our proposed models have longer convergence time as the original RDTC, proposed by Alaniz and Akata [2]. We hypothesize why this might be the case for our model.
In both strategies we prevent the model from using gradients, coming from uncertain attributes. The model thus has less gradient information available. As a direct result, the model can update fewer parameters in each step, or make smaller changes. We hypothesize that this causes increased convergence time.

5.2 The Right Kind of Uncertainty?

In our approach, we only consider epistemic uncertainty, which is the uncertainty arising from variance in the model's parameters. However, in closed datasets, as the ones considered, epistemic uncertainty will not change very much after the model has seen every training example. Not considering data augmentation techniques, this happens within one epoch.

5.2.1 Estimating Heteroscedastic Uncertainty

Heteroscedastic is a function of data and thus can be learned. Kendall and Gal [37] propose to learn an additional output σ^2 and a loss function:

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\frac{\|y_i - f(x_i)\|^2}{2\sigma(x_i)^2} + \frac{1}{2}log\ \sigma(x_i)^2. \tag{5.1}$$

If a prediction is wrong, it encourages a high value for σ^2, thus lowering its impact. In a variational inference setting (which our dropout model is corresponding to) for classification with sample size T, Kendall and Gal [37] propose to model aleatoric uncertainty as:

$$\frac{1}{T}\sum_{t=1}^{T} diag(\hat{\sigma}_t^2). \tag{5.2}$$

Alternatively, a different approach was proposed by Kwon et al. [42] that does not require learning σ^2.

$$\frac{1}{T}\sum_{t=1}^{T} diag(\hat{p}_t) - \hat{p}_t\hat{p}_t^T \tag{5.3}$$

where $\hat{p}_t = softmax(\pi)$ and π are the unscaled logits. This is an approximation to the variational posterior:

$$\int_{\Omega}\Big[diag(\mathbb{E}_{p(y^*|x^*,w)}[y^*]) - \mathbb{E}_{p(y^*|x^*,w)}[y^*]\cdot\mathbb{E}_{p(y^*|x^*,w)}[y^*]\Big]q_\theta(w)dw, \tag{5.4}$$

which is the diagonal of the expected outputs, subtracted by the product of the expected outputs with themselves, and finally multiplied with the variational posterior over the weights $w \in \Omega$. For $T \to \infty$, equation 5.3 will converge to equation 5.4.

5.2.2 Risk versus Uncertainty

Osband [60] suggests to distinct the variance arising from dropout neural networks, from actual uncertainty. He criticizes that the approximated posterior, which the mean and variance from multiple forward passes in a dropout network are equal to, according to Gal and Ghahramani [21], do not concentrate with more data available. According to him, this variance is rather associated with what is called risk in decision theory. There, the term risk refers to the inherent stochasticity of a model. However, this is what we previously introduced as epistemic uncertainty. This issue is thus rather about semanticity. We, therefore, acknowledge it but stick to the convention of differentiating between epistemic and aleatoric uncertainty.

5.2.3 Beyond Attribute Uncertainty

As the only point, where uncertainty information is retrieved in our model is in the $f_{AttrMLP}$ which maps ResNet features to attributes, our model can only consider attribute uncertainty.

Thus, as soon as an attribute is encountered in training, its uncertainty should converge. This may be unhelpful in realistic scenarios, where an unknown class is encountered that can be described by a set of known attributes as such an example would not yield high uncertainty. Replacing $f_{ClassMLP}$ with a Dropout MLP and computing uncertainty there as well may potentially alleviate this issue and could allow further insights.
Note that this is considering that outputs learned by the $f_{AttrMLP}$ would directly correspond to attributes. This is likely not the case. As for one, we use attribute coefficient $\lambda = 0.2$ in all our settings, and second, learning such a direct mapping between feature and attribute space is not a trivial problem.

5.3 A Practical Example

In the previous sections, we introduced a model that reliably can express its uncertainties to a user in an explainable decision process. In a realistic setting, this could provide the user with valuable insights. Instead of only yielding the final classification and an explanation tree, the model can also tell the user how uncertain it is about each part of the tree. If the user sees that the model was uncertain about several attributes it used, the classification may be done manually.
Let's consider an example (displayed in Figure 5.1) where an ornithologist is tasked to do a comprehensive survey of bird species and their respective numbers. Since our ornithologist is not given any helper, she decides to use a drone and a computer vision software that takes a picture of every bird it encounters (1). After letting the drone fly around in the survey area for some days, the ornithologist inspects the data and finds that the drone collected so much data that it hardly can be examined manually. Luckily, she has heard of our UncertainRDTC and decides to use it on the data(2). For each image, the model yields a classification and an explanation. However, due to climate change, some bird species that typically do not occur in the survey area have made their way into the data collected. The model has never seen such birds as they typically only appear much further south. It is highly uncertain and notifies the ornithologist (3). The ornithologist sees that this species is unknown to the model and thus rendered it uncertain, unable to make a reliable classification. However, since she knows of this species she can quickly classify it manually.

5.4 Carbon Footprint Estimation

To produce our results, we relied on GPUs provided by the University of Tübingen ML Cluster of Excellence and Google. We are aware of the profound impact, of the CO_2 emissions caused by producing the required electricity, on the world's climate. The GPU's provided by Google were Nvidia Tesla K80 which operate up to the thermal design power of 300 W [55]. Devices of the University of Tübingen GPU cluster are Tesla V100-SXM2 which, at full capacity, operate up to a thermal design power of 300 W [56]. These specifications are without considering cooling.
While no single model was trained for more than 100 epochs, if we consider all models and all iterations of models, we accumulate circa 2000 epochs of training. Training time for one epoch was approximately 6 minutes, or in 1 hour, a model can be trained for 10 epochs.

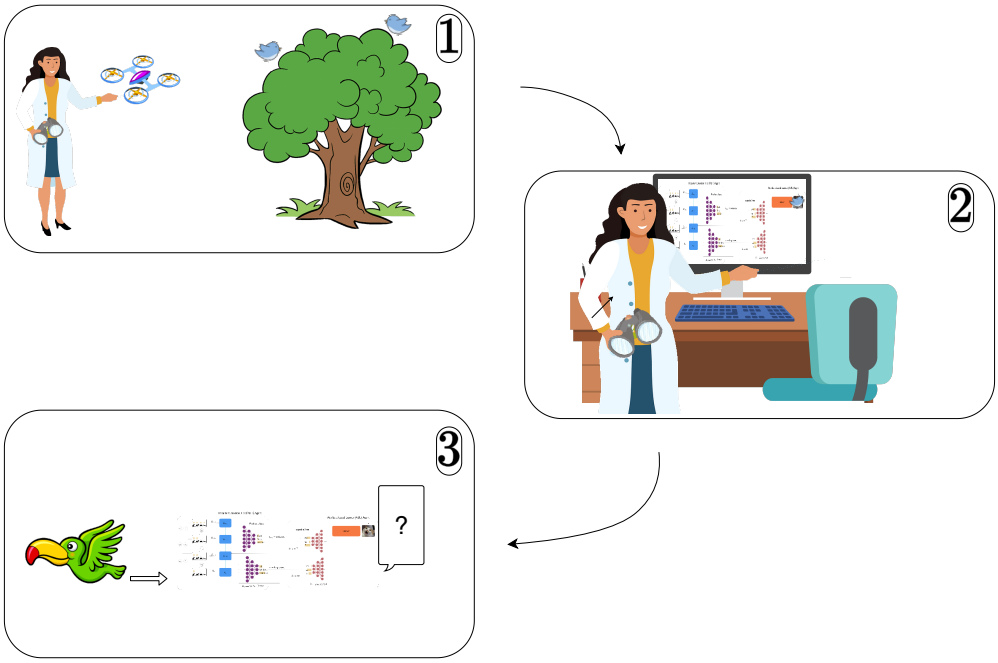

Figure 5.1: After collecting images of birds the ornithologist lets our proposed model classify the vast amount of data. Only in cases of high uncertainty, she is consulted and can classify the image manually.

Thus for 2000 epochs, we get 200 hours of training time. Considering the specification of our GPUs and assuming they are running at full capacity (which is unlikely when training a single model), we get an upper bound of $200h * 300\,kW = 60000\,kWh$ of energy consumption.
The German federal office for environmental concerns (Umweltbundesamt) estimates the amount of CO_2 necessary to produce one kilowatt-hour to be 401 g [31]. Considering, we required 60000 kWh, this would result in the CO_2 equivalent of $401\,g \cdot 60000 = 24060\,kg$.

Chapter 6

Conclusion

Extending the work of Alaniz and Akata [2], we propose an explainable model that is aware of and can express its uncertainties. The original model delivers a binary decision tree as an explanation, leveraging human-interpretable attributes provided as side information. The nodes of the resulting tree correspond to attributes and the edges to either 'Yes' or 'No' answers regarding the presence of these attributes. In our work, we give the model the ability to either avoid uncertain attributes to be used in the decision tree or extend it to a ternary decision tree, where '?' is an additional option.
While our proposed extension does not outperform the original aRDTC model, our remRDTC and extRDTC still outperform other explainable methods such as decision trees or dNDF. Moreover, the performance is comparable to that of uninterpretable state of the art methods, such as ResNet.
Our model does not only utilize uncertainty information but also allows introspection. We use this to further investigate the relationship between attribute usage, misclassification, and uncertainty. We show that our model reliably yields high uncertainty for unseen attributes and classes. This would allow the model to consult a human user in cases of high uncertainty, thus making it more applicable in real-world scenarios than comparable methods. Finally, we critically reflect on the weaknesses of the model and propose possible improvements for future work.

Bibliography

[1] Akata, Z., Perronnin, F., Harchaoui, Z., and Schmid, C. (2013). Label-embedding for attribute-based classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 819–826.

[2] Alaniz, S. and Akata, Z. (2019). Explainable observer-classifier for explainable binary decisions. *arXiv preprint arXiv:1902.01780*.

[3] Bayes, T. (1763). Lii. an essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, frs communicated by mr. price, in a letter to john canton, amfr s. *Philosophical transactions of the Royal Society of London*, (53):370–418.

[4] Beale, R. and Jackson, T. (1990). *Neural Computing-an introduction*. CRC Press.

[5] Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

[6] Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer.

[7] Bland, A. R. et al. (2012). Different varieties of uncertainty in human decision-making. *Frontiers in neuroscience*, 6:85.

[8] Box, G. E. and Tiao, G. C. (2011). *Bayesian inference in statistical analysis*, volume 40. John Wiley & Sons.

[9] Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984). Classification and regression trees. statistics/probability series.

[10] Cao, K., Lazaridou, A., Lanctot, M., Leibo, J. Z., Tuyls, K., and Clark, S. (2018). Emergent communication through negotiation. *arXiv preprint arXiv:1804.03980*.

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

[12] d'Amato, A. (1983). Legal uncertainty. *Calif. L. Rev.*, 71:1.

[13] Damianou, A. and Lawrence, N. (2013). Deep gaussian processes. In *Artificial Intelligence and Statistics*, pages 207–215.

[14] Das, A., Gervet, T., Romoff, J., Batra, D., Parikh, D., Rabbat, M., and Pineau, J. (2019). Tarmac: Targeted multi-agent communication. In *International Conference on Machine Learning*, pages 1538–1546.

[15] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

[16] Didelot, X., Gardy, J., and Colijn, C. (2014). Bayesian inference of infectious disease transmission from whole-genome sequence data. *Molecular biology and evolution*, 31(7):1869–1879.

[17] Dikmen, M. and Burns, C. M. (2016). Autonomous driving in the real world: Experiences with tesla autopilot and summon. In *Proceedings of the 8th international conference on automotive user interfaces and interactive vehicular applications*, pages 225–228.

[18] Farhadi, A., Endres, I., Hoiem, D., and Forsyth, D. (2009). Describing objects by their attributes. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1778–1785. IEEE.

[19] Foerster, J., Assael, I. A., De Freitas, N., and Whiteson, S. (2016). Learning to communicate with deep multi-agent reinforcement learning. In *Advances in neural information processing systems*, pages 2137–2145.

[20] Friston, K. J., Glaser, D. E., Henson, R. N., Kiebel, S., Phillips, C., and Ashburner, J. (2002). Classical and bayesian inference in neuroimaging: applications. *Neuroimage*, 16(2):484–512.

[21] Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059.

[22] Ghahramani, Z. (2015). Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553):452–459.

[23] Grossberg, S. (2013). Recurrent neural networks. *Scholarpedia*, 8(2):1888. revision #138057.

[24] Gunning, D., Stefik, M., Choi, J., Miller, T., Stumpf, S., and Yang, G.-Z. (2019). Xai—explainable artificial intelligence. *Science Robotics*, 4(37).

[25] Guo, W. (2020). Explainable artificial intelligence for 6g: Improving trust between human and machine. *IEEE Communications Magazine*, 58(6):39–45.

[26] Guynn, J. (2015). Google photos labeled black people 'gorillas'. *USA Today*.

[27] Havrylov, S. and Titov, I. (2017). Emergence of language with multi-agent games: Learning to communicate with sequences of symbols. In *Advances in neural information processing systems*, pages 2149–2159.

[28] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[29] Hebb, D. (1968). 0.(1949) the organization of behavior.

[30] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[31] Icha, P. and Kuhs, G. (2016). Entwicklung der spezifischen kohlendioxid-emissionen des deutschen strommix in den jahren 1990 bis 2015. *Climate Change*, 26:1–27.

[32] Jang, E., Gu, S., and Poole, B. (2016). Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

[33] Jiang, J. and Lu, Z. (2018). Learning attentional communication for multi-agent cooperation. In *Advances in neural information processing systems*, pages 7254–7264.

[34] Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1999). An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233.

[35] Kearns, M. J. and Mansour, Y. (1998). A fast, bottom-up decision tree pruning algorithm with near-optimal generalization. In *ICML*, volume 98, pages 269–277. Citeseer.

[36] Kelion, L. (2018). Ces 2018: Lg robot cloi repeatedly fails on stage at its unveil. *BBC*.

[37] Kendall, A. and Gal, Y. (2017). What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pages 5574–5584.

[38] Koch, K.-R. (2006). *Bayesian inference with geodetic applications*, volume 31. Springer.

[39] Kontschieder, P., Fiterau, M., Criminisi, A., and Rota Bulo, S. (2015). Deep neural decision forests. In *Proceedings of the IEEE international conference on computer vision*, pages 1467–1475.

[40] Krzywinski, M. and Altman, N. (2013). Importance of being uncertain.

[41] Kulkarni, G., Premraj, V., Ordonez, V., Dhar, S., Li, S., Choi, Y., Berg, A. C., and Berg, T. L. (2013). Babytalk: Understanding and generating simple image descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):2891–2903.

[42] Kwon, Y., Won, J.-H., Kim, B. J., and Paik, M. C. (2020). Uncertainty quantification using bayesian neural networks in classification: Application to biomedical image segmentation. *Computational Statistics & Data Analysis*, 142:106816.

[43] Lampert, C. H., Nickisch, H., and Harmeling, S. (2009). Learning to detect unseen object classes by between-class attribute transfer. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 951–958. IEEE.

[44] Lazaridou, A., Hermann, K. M., Tuyls, K., and Clark, S. (2018). Emergence of linguistic communication from referential games with symbolic and pixel input. *arXiv preprint arXiv:1804.03984*.

[45] LeCun, Y., Bengio, Y., et al. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995.

[46] Leibniz, G. W. (1666). *Dissertatio de arte combinatoria, in qua ex arithmeticae fundamentis complicationum ac transpositionum doctrina nouis praeceptis exstruitur... noua etiam Artis meditandis, seu Logicae inuentionis semina sparguntur. Praefixa est synopsis totius tractatus, & additamenti loco demonstratio existentiae Dei, ad mathematicam certitudinem exacta autore Gottfredo Guilielmo Leibnüzio...* apud Joh. Simon. Fickium et Joh. Polycarp. Seuboldum in Platea Nicolaea ....

[47] Liu, B. (2013). Toward uncertain finance theory. *Journal of Uncertainty Analysis and Applications*, 1(1):1.

[48] MacKay, D. J. (1998). Introduction to gaussian processes. *NATO ASI Series F Computer and Systems Sciences*, 168:133–166.

[49] Marvin, M. and Seymour, A. P. (1969). Perceptrons.

[50] McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.

[51] Metropolis, N. and Ulam, S. (1949). The monte carlo method. *Journal of the American statistical association*, 44(247):335–341.

[52] Mingers, J. (1989). An empirical comparison of pruning methods for decision tree induction. *Machine learning*, 4(2):227–243.

[53] Mullachery, V., Khera, A., and Husain, A. (2018). Bayesian neural networks. *arXiv preprint arXiv:1801.07710*.

[54] Najafabadi, M. M., Villanustre, F., Khoshgoftaar, T. M., Seliya, N., Wald, R., and Muharemagic, E. (2015). Deep learning applications and challenges in big data analytics. *Journal of Big Data*, 2(1):1.

[55] NVIDIA, V. (2015). Tesla k80 gpu accelerator board specification.

[56] NVIDIA, V. (2017). Tesla v100 pcie gpu accelerator data sheet.

[57] Nwankpa, C., Ijomah, W., Gachagan, A., and Marshall, S. (2018). Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378*.

[58] O'Connor, K. M. (1994). The alchemical creation of life (takwin) and other concepts of genesis in medieval islam.

[59] Ordonez, V., Kulkarni, G., and Berg, T. L. (2011). Im2text: Describing images using 1 million captioned photographs. In *Advances in neural information processing systems*, pages 1143–1151.

[60] Osband, I. (2016). Risk versus uncertainty in deep learning: Bayes, bootstrap and the dangers of dropout. In *NIPS Workshop on Bayesian Deep Learning*, volume 192.

[61] Palatucci, M., Pomerleau, D., Hinton, G. E., and Mitchell, T. M. (2009). Zero-shot learning with semantic output codes. In *Advances in neural information processing systems*, pages 1410–1418.

[62] Parmigiani, G. and Parmigiani, G. (2002). *Modeling in medical decision making: a Bayesian approach*. J. Wiley.

[63] Puiutta, E. and Veith, E. (2020). Explainable reinforcement learning: A survey. *arXiv preprint arXiv:2005.06247*.

[64] Quinlan, J. (1986). Induction of decision trees. mach. learn.

[65] Quinlan, J. R. (2014). *C4. 5: programs for machine learning*. Elsevier.

[66] Raileanu, L. E. and Stoffel, K. (2004). Theoretical comparison between the gini index and information gain criteria. *Annals of Mathematics and Artificial Intelligence*, 41(1):77–93.

[67] Rodriguez, R. C., Alaniz, S., and Akata, Z. (2019). Modeling conceptual understanding in image reference games. In *Advances in Neural Information Processing Systems*, pages 13155–13165.

[68] Rosenblatt, F. (1960). Perceptron simulation experiments. *Proceedings of the IRE*, 48(3):301–309.

[69] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

[70] 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 (IJCV)*, 115(3):211–252.

[71] Sarkar, S., Weyde, T., Garcez, A., Slabaugh, G. G., Dragicevic, S., and Percy, C. (2016). Accuracy and interpretability trade-offs in machine learning applied to safer gambling. In *CEUR Workshop Proceedings*, volume 1773. CEUR Workshop Proceedings.

[72] Schlesinger, A., O'Hara, K. P., and Taylor, A. S. (2018). Let's talk about race: Identity, chatbots, and ai. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–14.

[73] Sensoy, M., Kaplan, L., and Kandemir, M. (2018). Evidential deep learning to quantify classification uncertainty. In *Advances in Neural Information Processing Systems*, pages 3179–3189.

[74] Spiegelhalter, D. and Rice, K. (2009). Bayesian statistics. *Scholarpedia*, 4(8):5230. revision #185711.

[75] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.

[76] Wagenmakers, E.-J., Love, J., Marsman, M., Jamil, T., Ly, A., Verhagen, J., Selker, R., Gronau, Q. F., Dropmann, D., Boutin, B., et al. (2018). Bayesian inference for psychology. part ii: Example applications with jasp. *Psychonomic bulletin & review*, 25(1):58–76.

[77] Wah, C., Branson, S., Welinder, P., Perona, P., and Belongie, S. (2011). The Caltech-UCSD Birds-200-2011 Dataset. Technical report.

[78] Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

[79] Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Philip, S. Y., et al. (2008). Top 10 algorithms in data mining. *Knowledge and information systems*, 14(1):1–37.

[80] Xian, Y., Lampert, C. H., Schiele, B., and Akata, Z. (2019). Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(9):2251–2265.

[81] Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. (2017). Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500.

[82] Yadron, D. and Tynan, D. (2016). Tesla driver dies in first fatal crash while using autopilot mode. *The Guardian*.

[83] Zou, J. and Schiebinger, L. (2018). Ai can be sexist and racist—it's time to make it fair.

Acknowledgments

I would like to thank my pet snake, which sleeps all day, and my mother, who is the hardest working person I know. I would like to thank Zeynep and Stephan for the continuous support, critical questions, and helpful insights.

Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe und dass alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen sind, durch Angaben von Quellen als Entlehnung kenntlich gemacht worden sind. Diese Bachelorarbeit wurde in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt.

Stefan Wezel (Matrikelnummer 4080589), October 3, 2020