Erasmus University Rotterdam

Erasmus School of Economics

Master Thesis in Business Analytics and Quantative Marketing

Forecasting Pedestrians Movement Paths and its Uncertainties using Evidential Neural Networks

Author

S.J. (Stein) Dijkstra (467846)

Supervisor: O. (Olga) Kuryatnikova

Second assessor: P.C. (Paul) Bouman

April 30, 2024

Abstract

In high-stakes decision-making scenarios, accurately predicting pedestrian behavior is crucial for the safety and reliability of autonomous vehicles in urban environments. However, traditional machine learning models often provide deterministic predictions, lacking the ability to quantify uncertainties inherent in dynamic and complex environments. This paper investigates three methodologies—evidential neural networks, Bayesian neural networks, and deep ensembles—to extend neural networks' capabilities by predicting uncertainty as a confidence interval. In addition, we suggest an alternative regularization term for the evidential neural network. We evaluate these methods on a regression dataset with heteroskedasticity and on the UCY and ETH pedestrian forecasting datasets. The results show that the evidential neural network excels in predicting pedestrian movement uncertainties, outperforming the other methods. Furthermore, we observe that evidential networks have significant computational benefits during prediction.

The content of this thesis is the sole responsibility of the author and does not reflect the view of the supervisor, second assessor, Erasmus School of Economics or Erasmus University.

Contents

1 Introduction

In high-stakes decision-making processes, the ability to not only make accurate predictions but also quantify the associated uncertainties is of great importance, particularly in scenarios involving dynamic and complex environments where deterministic point forecasts may fall short. One such context is pedestrian behavior forecasting, a critical aspect of ensuring the safety and reliability of autonomous vehicles navigating urban environments.

Traditional machine learning models often provide point predictions lacking the nuance required for handling the unpredictability and diversity of pedestrian behaviors. Moreover, the advent of deep learning, including neural networks and their variants such as recurrent neural networks, convolutional neural networks, and encoder-decoder architectures, has further worsened these issues due to increased model complexity. These models tend to yield deterministic predictions, failing to account for the multiple potential outcomes that pedestrians' actions might exhibit, especially in scenarios where split-second decisions can have significant implications. The limitations of point predictions become clear when applied to self-driving vehicles' interactions with pedestrians, where a singular prediction is insufficient to make safe driving choices. For instance, a sudden choice of a pedestrian to either cautiously wait at a crosswalk or quickly cross the road demands a system capable of incorporating uncertainty into decision-making. Solving these issues is thus necessary to advance the capabilities of autonomous vehicles.

In this paper, we research the possibilities of extending the capabilities of neural networks and their variants to encompass not only point predictions but also the consideration of uncertainty. Here, we want to estimate the uncertainty as a confidence interval, a bound that includes the actual value with some confidence p. We utilize three distinct methodologies to achieve this: evidential neural networks, Bayesian neural networks, and deep ensembles. These approaches are implemented and evaluated to incorporate uncertainty into predictive models effectively. We test these methods using a regression and a pedestrian forecast dataset, assessing each method's performance and robustness in handling uncertainty in real-world forecasting tasks. As such, we aim to reduce the problem of point forecasting of neural networks and thus improve autonomous decision-making.

The three methodologies mentioned above are founded on distinct principles. Evidential models modify neural network outputs by changing single-point predictions to predict parameters within a predetermined distribution. Consequently, the network's output estimates a complete distribution in a single forward pass, allowing uncertainty to be determined. In the case of Bayesian neural networks, we implement the Monte Carlo (MC) dropout approximation, introducing uncertainty through the randomized removal of neural network elements during training. Lastly, the ensemble method capitalizes on the variability inherent in neural network architectures, leveraging the likelihood of different local optima. Robust uncertainty estimates can be achieved by averaging these estimations and accounting for their errors across various data areas.

In comparing these methodologies, we have two main criteria for evaluation. Firstly, we seek models with comparable or better accuracy than current methods. Given the intended application of these models in safety-critical and high-stakes scenarios, a less accurate model will not be implemented in practice. Additionally, we aim to ensure the correct uncertainty prediction. We want the confidence intervals based on the uncertainty estimate to encompass the actual values and avoid excessive width. In practical terms, achieving this balance ensures that the models provide reliable uncertainty estimates, which is crucial for informed decision-making in various contexts.

Recently, Nayak et al. (2022) tried to estimate the uncertainty for pedestrian movement paths. Although their research showed promising results in model accuracy, they showed opportunities for further research in uncertainty prediction, which we address in this research. In addition, we want to implement and test the recently introduced evidential neural network on a real-world use case, as it has shown promising results on for example 3D object detection (Paat et al., 2023).

These considerations lead us to the central research question we aim to address in this thesis:
Which of the three considered methodologies is optimal for estimating the uncertainty of predictions generated by a neural network?
Here, the concept of "optimal" is a combination of the two criteria outlined in the previous paragraph. More specifically, this question encompasses three subquestions: *What uncertainty estimation method obtains the most accurate predictions?*, *What uncertainty estimation method obtains the most reliable uncertainty estimates?*, and *How do the results of the accuracy and uncertainty estimate change between datasets and use-cases?* By addressing these questions, we provide insights that improve the understanding and application of uncertainty estimation with neural networks.

Based on these research questions, the study is structured as follows: First, we explore the methodology of evidential neural networks and suggest a new regularization term. Then, to address the previous questions, we conduct numerical experiments. Initially, we estimate a regression problem with heteroscedastic errors. Afterward, we carry out experiments using two pedestrian datasets, specifically focusing on the ETH (collected by the ETH Zurich) (Pellegrini et al., 2009) and UCY (collected by the University of Cypress) (Lerner et al., 2007) datasets. These datasets contain various scenarios where pedestrian movements have been tracked, allowing us to answer our research questions effectively.

This research is part of a domain characterized by numerous recent developments, and accordingly, this thesis can contribute to this field of research. Specifically, we expect the following contributions:

1. Introduction of a new regularization term for the evidential network based on the definition of a confidence interval.

2. Application of evidential networks to estimate predictive uncertainty.

3. Improved predictions of the confidence interval of pedestrian movement compared to previous literature.

4. A comparison of three different methods to estimate uncertainty on a real-world dataset.

5. Proposal of a metric to evaluate the confidence intervals based on a distribution-free assumption.

The scientific relevance of this thesis can be described based on these contributions. In particular, we believe that the first and second contributions add new insights to the discussion on evidential neural networks (Amini et al., 2020; Bengs et al., 2023; Malinin et al., 2020; Meinert & Lavin, 2021; Meinert et al., 2023; Oh & Shin, 2022). Furthermore, the third contribution has incorporated the conclusion and suggestions of Nayak et al. (2022). Specifically, we have adopted their recommendation to utilize other uncertainty estimation techniques and to improve the quality of the uncertainty forecast. Lastly, fourth and fifth contributions have been proposed by the survey of Gawlikowski et al. (2023) as the main avenues for further research in the area of uncertainty estimation.

The societal relevance of this research is evident, especially given the significance of uncertainty estimation in high-stakes scenarios. The findings of this study could serve as a tool to update the algorithms in scenarios such as self-driving cars, where the accurate prediction of uncertainty is critical for ensuring safety and reliability. Moreover, improving machine learning algorithms with better uncertainty estimation techniques can increase their trustworthiness.

The subsequent sections of the thesis are structured as follows: Section 2 delves into the literature review, including recent advancements in uncertainty estimation methods. In Section 3, we elaborate on the methodology, providing insights into the theoretical background and offering details on implementation. Next, in Section 4, we discuss the datasets utilized in our study. Following this, Section 5 presents the results obtained from our experiments. Lastly, in Section 6, we draw conclusions from our findings and explore potential avenues for further research.

2 Literature Review

A great introduction to the field of uncertainty forecasting is Gawlikowski et al. (2023). This paper is an extensive survey of several branches of uncertainty estimation methods. These include the methods used in this paper but also other uncertainty estimation methods appropriate for classification and reinforcement learning problems. Moreover, this paper highlights the critical areas for further research. Another paper that overviews different methods is Jospin et al. (2022). However, whereas Gawlikowski et al. (2023) focus on theoretical aspects, the main focus of Jospin et al. (2022) is on the practical implementation. In particular, they provide a clear overview of several approximations for variational inference methods, including the Monte Carlo dropout method. Thus, their paper was used during the implementation of the methods. Ulmer et al. (2021) provides an overview focused on evidential neural networks. As such, they do a deep dive into different methods and highlight the significant difference between uncertainty estimation methods for regression and categorical problems. Moreover, they provide a good discussion of the differences and similarities in the literature related to evidential neural networks.

One of the recent developments in uncertainty forecasting is evidential neural networks. These networks, initially derived for categorical data, try to model the output as a distribution with unknown parameters instead of directly (Sensoy et al., 2018). This leads to a network that considers the evidence of training data during prediction. This evidence corresponds to the observed samples in the training set. As such, it can show less confidence if the input is significantly different from the training data. The main focus has been on choosing distribution (Malinin & Gales, 2018) and loss function (Malinin & Gales, 2019) to improve this methodology. Based on the structure of the distribution, it was also hypothesized that evidential networks could observe data and model uncertainty. This distinction could lead to out-of-distribution detection (Malinin & Gales, 2019). Here, data uncertainty is inherent to the data, whereas model uncertainty is related to the shortcomings of the model, the sum of which is equal to the predictive uncertainty (Gawlikowski et al., 2023).

The subsequent development was redefining the evidential model for regression data (Malinin et al., 2020). Here, the main challenge is to redefine the loss function for a regression problem since the loss function for the categorical distribution is not well defined for regression problems (Amini et al., 2020). Instead, Amini et al. (2020) propose a loss function that includes a negative likelihood and a regularization parameter. However, this loss function was insufficient to find the optimal parameter values. This was first shown by Meinert and Lavin (2021) by noting that the regularization term had a typo. Later, in Meinert et al. (2023) and Oh and Shin (2022), it was shown that the loss function did not solve to a unique solution. In their papers, Meinert et al. (2023) and Oh and Shin (2022) provide alternative formulations for the loss function. In this thesis, we add to this discussion by proposing a new regularization term for the loss function. One of the other goals of the network was to distinguish data and model uncertainty, similar to the categorical evidential network. However, Bengs et al. (2023) proved this is impossible to achieve using the class of loss function originally introduced in Malinin et al. (2020). Therefore, in this thesis, we will not consider data and model uncertainty. Instead, we focus on the predictive uncertainty, equal to the sum of model and data uncertainty. Despite the theoretical concerns in Bengs et al. (2023), the evidential network has shown very promising results (Amini et al., 2020; Malinin et al., 2020; Meinert et al., 2023; Oh & Shin, 2022).

Another class of uncertainty estimation method is Bayesian neural networks (Gawlikowski et al., 2023). The main idea of these models is to define the weights of neural networks as random variables instead of as scalar weights. As a consequence, the output of the network will not be a singular prediction but a posterior distribution. The main issue with this methodology is that exact inference is intractable due to the high number of parameters and non-linearities. Thus, several approximations have been proposed instead of estimating a Bayesian network directly.

The first of these approximation methods is Monte Carlo dropout (Gawlikowski et al., 2023). The Monte Carlo dropout model adds an uncertainty distribution to the network by randomly setting weights to zero with a given probability. Historically, this technique was proposed as a method to prevent overfitting. However, as an uncertainty estimation technique, it can be shown that an arbitrary neural network with Monte Carlo dropout is equivalent to a Gaussian Bayesian network (Gal & Ghahramani, 2016). For this method, the main challenge is finding the correct dropout probability, as analyzed by Gal and Ghahramani (2015), and Gal et al. (2017). The second challenge is that it can be shown that dropout underestimates the true uncertainty (Gal & Ghahramani, 2016). This has been addressed in Li and Gal (2017) at the cost of model complexity. Monte Carlo dropout has gained popularity due to its ease of implementation and availability of implementation guides. For example, Jospin et al. (2022) and Hron et al. (2018) streamline the methodology and highlight common pitfalls.

Another approximation of a Bayesian neural network is variational inference. The main idea of this methodology is to estimate the actual posterior distribution by a variational distribution. Then, during estimation, we want to ensure that this variational distribution is as similar as possible to the true posterior based on the Kullback-Leibler divergence. This methodology was initially introduced by Hinton and Van Camp (1993) and Barber and Bishop (1998), where it is shown that the true posterior can be approximated by a Gaussian distribution with zero (Hinton & Van Camp, 1993) or non-zero (Barber & Bishop, 1998) covariance elements. Further analysis of this methodology led to the derivation of the Bayes by backdrop algorithm (Blundell et al., 2015), which leads to a further improvement in computation time. Nevertheless, compared to the Monte Carlo Dropout approximation, implementing it remains more complex and computationally intensive. As such, we use the Monte Carlo dropout approximation in this thesis.

Another alternative to the Monte Carlo dropout approximation are sample methods. These are based on sampling the true posterior using Markov Chain Monte Carlo (MCMC) and do not require assumptions on the output distribution. This technique was introduced by R. Neal (1992) adapting MCMC using Hamiltonian dynamics to obtain a Hybrid Monte Carlo method. This technique was further developed in R. M. Neal et al. (2011), Dubey et al. (2016), and Li and Gal (2017). For this method, the main challenges lie in the training procedure. In particular, during each iteration, the entire sample must be processed (Gawlikowski et al., 2023). Therefore, it is impractical to apply to datasets with many data points.

A third family of uncertainty estimation models are ensemble models. Ensemble models are not specific to neural networks and are applied in a broad range of applications. The ensembling of neural networks, similar to Monte Carlo dropout, was initially introduced to reduce overfitting and to improve the robustness of the predictions (Gawlikowski et al., 2023). Nevertheless, it can be adapted to estimate the predictive uncertainty. The main challenge for this methodology is to ensure enough variety between models (Lakshminarayanan et al., 2017; Renda et al., 2019). Several techniques have been proposed to achieve this, such as the random shuffling of data, data augmentation, and random initialization. Additionally, Herron et al. (2020) propose to induce variety by varying neural network architecture.

Recently, there has also been an increase in research on the application of these uncertainty estimation techniques for real-world datasets. An example of a use case is self-driving cars. Lütjens et al. (2019) incorporate uncertainty in the autonomous navigation system. In particular, they applied the ensemble and Monte Carlo dropout approaches and found that the model could recognize previously unseen scenarios. Another example is the paper of Nayak et al. (2022). Nayak et al. (2022) focus explicitly on forecasting pedestrian movement paths using the Monte Carlo dropout method. Interestingly, they find that in addition to the uncertainty estimation, the accuracy of the method also improves. A third example is the paper of Paat et al. (2023), where the evidential neural network approach is applied to model 3D object detection. Another area where uncertainty estimation is relevant is weather forecasting. Examples of this are Leutbecher and Palmer (2008), which can forecast air flow variations, and Parker (2013), which uses ensemble to predict the variance of climate change impact. Lastly, uncertainty estimation methods are increasingly used in medical analysis, as shown by the survey of Abdullah et al. (2022).

3 Methodology

This section presents the different uncertainty methods, the neural network architecture, and the metrics. First, we will define the necessary notation. In this thesis, we considered a supervised learning problem with continuous output parameters; that is, we wish to construct function f that takes an input space $\mathcal{X}$ and generate outputs from an output space $\mathcal{Y}$ that is $f : \mathcal{X} \to \mathcal{Y}$. Since we consider a continuous regression problem, we have $\mathcal{Y} \in \mathbb{R}^{d_y}$. Here d_y represents the output dimension, where $d_y = 1$ represents a standard regression problem and $d_y > 1$ represents multiple outputs. We will define the function f as a neural network. In particular, we will define this function as dependent on a matrix of model weights $\mathbf{w}$. The model weights $\mathbf{w}$ in combination with the model's architecture can completely describe the network. Furthermore, we define $\mathbf{W}$ as a matrix of random variables of the model weights, which will be used in the Bayesian neural network.

To estimate the model f, we rely on a sample of training data. That is pairs of (x_i, y_i) for all $i = 1 \dots n$, where n is the sample size from dataset. Furthermore, to test the accuracy of the models, we have access to a set of test observations (x_i^{test}, y_i^{test}) for all $i = 1 \dots n_{test}$.

The goal of the uncertainty estimation methods is not only to predict a single point prediction $\hat{y}$ but also to predict the associated confidence interval. That is, we want to estimate $\Theta(x)$ where $\Theta : \mathcal{X} \to [\mathbb{R}]^{d_y}$ is the confidence interval estimator, where $[\mathbb{R}]$ represents the set of closed and bounded intervals. The confidence interval represents some range that contains the true value with some confidence level p. The estimator Θ will be constructed based on the prediction $\hat{y}$ and the estimated variance s^2, the details of which will be specified for each model.

The remaining part of the methodology is organized as follows. Section 3.1 presents an overview and comparison of the three different estimation methods. Then, we will explain the details of each of the three methodologies. After that, we explain the specifics of neural networks relevant to our use case. Lastly, we will describe and define the metrics to quantify the quality of predictions.

3.1 Overview of uncertainty estimation techniques

In this thesis, we apply several uncertainty estimation methods, as outlined in the literature review. Specifically, this thesis compares three techniques: an evidential neural network, a Bayesian neural network, and an ensemble approach. Table 1 provides an overview of these methods.

The first method we use in this thesis is an evidential network, also known as a prior network (Malinin et al., 2020). The fundamental concept here involves specifying a parameterized distribution over the output variables. Then, the weights of the neural network are optimized according to the log-likelihood of the data and parameterized distribution. Thereafter, based on the properties of the distribution, we can estimate the confidence region using the predicted parameters of the neural network. The main benefit of this method compared to alternatives is its computing speed. Particularly, the estimated mean and confidence interval can be predicted using only a single forward pass, yielding significant computing time advantages. However, the model is reliant on an accurate specification of the distribution.

The second method we implement is a Bayesian neural network, where we specify a distribution over the network's parameters. This approach can result in a more intricate model definition and estimation procedure. However, exact optimization is often computationally infeasible (Gawlikowski et al., 2023), necessitating the application of approximations. In this thesis, we consider the Monte Carlo dropout approximation (Gal & Ghahramani, 2016). This technique introduces a probability element to the network during training and prediction by "dropping" weights during the computation of each observation. The uncertainty is estimated by performing many forward passes of the network for each sample. Consequently, this approach enhances the robustness of the model by incorporating uncertainty into its predictions at the cost of computation time.

The final methodology we use is deep ensembles. This approach involves training multiple independent models. Then, during prediction, the output of all models is averaged, and the corresponding uncertainty is estimated. The ensemble often yields better results than any of the individual models. However, a key assumption is that the individual models are sufficiently different. In this method, we will specifically focus on constructing an ensemble of neural networks. Compared to the other models, the main drawback of the ensemble model is that multiple neural networks must be trained, leading to worse computational performance.

Table 1: Overview of three different uncertainty estimation techniques

	Evidential network	Monte Carlo dropout	Deep Ensemble
Description	Uncertainty is estimated by determining the parameters of some known distribution	Uncertainty is estimated by using a model where parameters are (explicitly) random variables	The uncertainty is estimated by combining outputs of several other model, variation among those is crucial
	$x_i \overset{\mathbf{w}}{\Rightarrow} \theta_i \Rightarrow \Theta(x_i;\theta_i)$	$x_i \overset{\mathbf{W}\sim G}{\Rightarrow} \Theta(x_i)$	$x_i \begin{array}{l} \overset{\mathbf{w}_1}{\Rightarrow} \hat{y}_{1i} \\ \overset{\mathbf{w}_2}{\Rightarrow} \hat{y}_{2i} \\ \vdots \\ \overset{\mathbf{w}_n}{\Rightarrow} \hat{y}_{ni} \end{array} \Rightarrow \hat{\Theta}(x_i)$
Number of networks trained	1	1	Many
Computational cost of training	Low	Medium	Very high
Computational cost of predictions	Low	High	Medium
Sensitivity to initialization	High	Low	Low

3.2 Evidential networks

As discussed in Section 2, there has been a recent increase in research on evidential neural networks, primarily driven by their computational and theoretical advantages. The original goal of the evidential neural network was to disentangle data and model uncertainty (Amini et al., 2020; Malinin et al., 2020). In this thesis, we will only consider predictive uncertainty, where the predictive uncertainty is the sum of data and model uncertainty (Gawlikowski et al., 2023; Malinin, 2019). We choose to consider predictive uncertainty since recently it has been shown that it is impossible to separate data and model uncertainty using evidential neural networks (Bengs et al., 2023; Meinert et al., 2023).

Next, we first present the specification of the distribution of y_i. The main goal here is to specify a distribution with sufficient flexibility, which we use to derive the log-likelihood over the parameters of the network, then we specify the relation between y_i and x_i. The results in this section will be derived for $y_i \in \mathbb{R}^d$ where $d = 1$. Although multivariate results exist (Malinin et al., 2020; Meinert & Lavin, 2021), we choose to present and derive univariate results due to ease of notation. In practice, we can model multivariate outputs by modeling multiple univariate outputs separately. However, even though we model the outputs separately, due to shared model weights $\mathbf{w}$ of the neural network the outputs are related.

Unlike standard neural networks, the first main idea of evidential neural networks is to parameterize y_i over some probability distribution. Since y_i is continuous, we choose the normal distribution (Amini et al., 2020; Malinin et al., 2020). That is

$$y_i \sim \mathcal{N}(\mu_i, \sigma_i^2), \tag{1}$$

or equivalently

$$p(y_i|\mu_i, \sigma_i) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left\{-\frac{(y_i - \mu_i)^2}{2\sigma_i^2}\right\}, \tag{2}$$

such that y_i is normal with mean μ_i and standard deviation σ_i. In this and further equations, p denotes the pdf of a distribution. Usually, an assumption-free approach is used for neural networks. As such, the main drawback of the evidential neural network is the reliance on a model assumption, with the benefit of being able to estimate uncertainty.

As shown by Meinert et al. (2023) through numerical experiments, directly estimating μ_i and σ_i is ineffective. Instead, the second key idea, as introduced by Amini et al. (2020) and Malinin et al. (2020), is to define a distribution over μ_i and σ_i to introduce more flexibility. This structure is similar to hierarchical distributions in hierarchical Bayesian methods. We can visualize this idea using Figure 1. On the right, we see that different y_i are normally distributed based on the parameters of μ_i and σ_i. On the left, we see multiple possible realizations of distributions over μ_i and σ_i, where we note that if we have more data, the variance of μ_i and σ_i will be less.

In line with the work of Malinin et al. (2020) and Amini et al. (2020), we choose a normal-inverse gamma (NIG) distribution as the higher distribution over the mean and standard deviation. The main benefit of this choice is the ease of computation due to its theoretical properties. Based on this assumption, μ_i is normally distributed as

$$\mu_i \sim \mathcal{N}(\gamma_i, \sigma_i^2 \nu_i^{-1}), \tag{3}$$

with parameters $\gamma_i \in \mathbb{R}$ and $\nu_i > 0$ and the variance is distributed according to the inverse gamma distribution

$$\sigma_i^2 \sim \Gamma^{-1}(\alpha_i, \beta_i), \tag{4}$$

with parameters $\alpha_i > 1$ and $\beta_i > 1$. Jointly, they are now distributed according to the normal inverse gamma distribution

$$p(\mu_i, \sigma_i^2|\gamma_i, \nu_i, \alpha_i, \beta_i) = p(\mu_i|\sigma_i^2, \gamma_i, \nu_i)p(\sigma_i^2|\alpha_i, \beta_i). \tag{5}$$

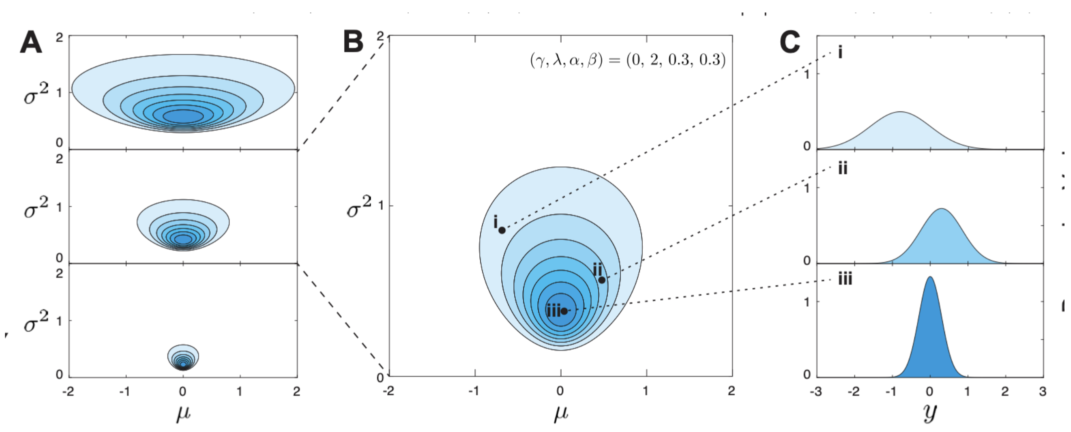

Figure 1: Adapted from Amini et al. (2020). On the left, we can see the different realization of the normal inverse gamma distribution as in Equation (5), in the middle, we see the distribution after estimation, and on the right, we see the possible distributions of y_i with specified mean and standard deviation

Next, we will derive the consequence of our choice that y_i is normally distributed and that (μ_i, σ_i) is distributed according to the normal inverse gamma distribution. This proof follows the work of Amini et al. (2020) but is reformulated and clarified using additional steps.

Theorem 1. *Let y be normal distributed with mean and variance μ, σ and let μ be normally distributed with mean γ, and variance $\frac{\sigma^2}{\nu}$ and let σ be inverse gamma distributed with parameters α, β then y given $\gamma, \nu, \alpha, \beta$ is distributed according to a univariate t-distribution with 2α degrees of freedom, location parameter γ, and scale parameter $\frac{\beta(1+\nu)}{\nu\alpha}$.*

Proof. Let us first write out the distributions of μ and σ. That is,

$$p(\mu|\sigma^2, \gamma, \nu) = \frac{1}{\sqrt{2\pi\frac{\sigma^2}{\nu}}} \exp\left\{-\frac{\nu}{2\sigma^2}(\mu-\gamma)^2\right\} \tag{6}$$

and

$$p(\sigma^2|\alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} \left(\frac{1}{\sigma^2}\right)^{\alpha+1} \exp\left\{-\frac{\beta}{\sigma^2}\right\} \tag{7}$$

respectively, where $\Gamma(.)$ is the gamma function. Now the joint normal inverse gamma distribution using Equation (5) can be written as

$$p(\mu, \sigma^2|\gamma, \nu, \alpha, \beta) = \frac{\beta^\alpha\sqrt{\nu}}{\Gamma(\alpha)\sqrt{2\pi\sigma^2}} \left(\frac{1}{\sigma^2}\right)^{\alpha+1} \exp\left\{-\frac{2\beta+\nu(\gamma-\mu)^2}{2\sigma^2}\right\}. \tag{8}$$

To find $p(y|\gamma, \nu, \alpha, \beta)$ we rewrite $p(y|\gamma, \nu, \alpha, \beta)$ as

$$p(y|\gamma, \nu, \alpha, \beta) = \int_0^\infty \int_{-\infty}^\infty p(y|\mu, \sigma^2)p(\mu, \sigma|\gamma, \nu, \alpha, \beta)d\mu d\sigma^2, \tag{9}$$

where we use the law of total probability. Now, using Equation (8) and the definition of a normal distribution, as well as the grouping of terms, we find

$$p(y|\gamma, \nu, \alpha, \beta) = \int_0^\infty \int_{-\infty}^\infty \frac{\beta^\alpha\sqrt{\nu}}{2\pi\Gamma(\alpha)\sigma^2} \left(\frac{1}{\sigma^2}\right)^{\alpha+1} \exp\left\{-\frac{2\beta}{2\sigma^2} - \frac{(y-\mu)^2}{2\sigma^2} - \frac{\nu(\gamma-\mu)^2}{2\sigma^2}\right\} d\mu d\sigma^2. \tag{10}$$

Now completing the square for μ, see Greenberg (2012), and rewriting we find

$$p(y|\gamma,\nu,\alpha,\beta) = \int_0^\infty \frac{\beta^\alpha \sqrt{v}}{\sqrt{2\pi\sigma^2}\Gamma(\alpha)(1+\sqrt{v})} \left(\frac{1}{\sigma^2}\right)^{\alpha+1} \exp\left\{-\frac{2\beta}{2\sigma^2} - \frac{\nu(y-\gamma)^2}{(1+\nu)2\sigma^2}\right\} d\sigma^2$$
$$\int_{-\infty}^\infty \frac{1}{\sqrt{2\pi}} \frac{1}{\sqrt{\frac{\sigma^2}{1+v}}} \exp\left\{\frac{(\mu - \frac{1}{1+\nu}(\nu\gamma + y))^2}{\frac{\sigma^2}{(1+\nu)}}\right\} d\mu. \quad (11)$$

Next, we recognize the normal distribution in the second integral that is integrated over its whole domain and thus is equal to 1. Thus,

$$p(y|\gamma,\nu,\alpha,\beta) = \int_0^\infty \frac{\beta^\alpha \sqrt{v}}{\sqrt{2\pi\sigma^2}\Gamma(\alpha)(1+\sqrt{v})} \left(\frac{1}{\sigma^2}\right)^{\alpha+1} \exp\left\{-\frac{2\beta}{2\sigma^2} - \frac{\nu(y-\gamma)^2}{(1+\nu)2\sigma^2}\right\} d\sigma^2. \quad (12)$$

Now similarly rewrite for σ^2 and the inverse gamma distribution

$$p(y|\gamma,\nu,\alpha,\beta) = \frac{\Gamma(\alpha+1/2)}{\Gamma(\alpha)} \sqrt{\frac{\nu}{\pi}} (2\beta(1+\nu))^{-\alpha} (\nu(y-\gamma)^2 + 2\beta(1+\nu))^{-(\frac{1}{2}+\alpha)}$$
$$\int_{-\infty}^\infty \frac{1}{\Gamma(\alpha+1/2)} (\nu(y-\gamma)^2 + 2\beta(1+\nu))^{\frac{1}{2}+\alpha} \left(\frac{1}{\sigma^2}\right)^{\alpha+3/2} \exp\left\{(\nu(y-\gamma)^2 + 2\beta(1+\nu))\frac{1}{2\sigma^2}\right\} d\sigma, \quad (13)$$

and we find that

$$p(y|\gamma,\nu,\alpha,\beta) = \frac{\Gamma((2\alpha+1)/2)}{\Gamma(2\alpha/2)} \sqrt{\frac{\nu}{\pi}} (2\beta(1+\nu))^{-\alpha} (\nu(y-\gamma)^2 + 2\beta(1+\nu))^{-(\frac{1}{2}+\alpha)}. \quad (14)$$

Finally in Equation (14) we can recognize a student t-distribution with parameters γ, $\frac{\beta(1+\nu)}{\nu\alpha}$ and 2α degrees of freedom, or equivalently

$$y|\gamma,\nu,\alpha,\beta \sim T\left(\gamma, \frac{\beta(1+\nu)}{\nu\alpha}, 2\alpha\right), \quad (15)$$

with T defined as a location-scale t-distribution. This concludes our proof. □

To summarize, based on the assumption that y_i is normal and the distribution over (μ_i, σ_i) is normal inverse gamma, we find that we can estimate y_i based on the t-distribution. This result is used in the next section to determine the loss function, expected value, and variance as a function of the parameters $(\gamma_i, \nu_i, \alpha_i, \beta_i)$. Additionally, we see that the degrees of freedom of the student t-distribution can either give the distribution a so-called "heavy tail" if $2\alpha_i$ is close to one, which means that there is more probability mass in the extreme values or be similar to a normal distribution if $2\alpha_i$ is increasingly large.

Next, we relate x_i to the parameters $\{\gamma_i, \nu_i, \alpha_i\beta_i,\}$ through a neural network, and thus the parameters are a function of x_i and $\mathbf{w}$

$$\{\gamma_i, \nu_i, \alpha_i\beta_i,\} = f(x_i, \mathbf{w}) \quad (16)$$

where $f()$ represents the neural network, and $\mathbf{w}$ are the model weights that we optimize, which are equal for all observations.

3.2.1 Details of network optimization

In this section, we use Theorem 1 to define the loss function and the necessary quantities to estimate and make predictions with the evidential neural network. First, we determine the expectation and variance of y_i as a function of the parameters $(\gamma_i, \nu_i, \alpha_i, \beta_i)$. Using Equations (1) and (15) and we find

$$\mathbb{E}[y_i] = \mathbb{E}[\mu_i] = \gamma_i, \tag{17}$$

and using Equation (15) and the properties of the t-distribution we find

$$\text{Var}[y_i] = \frac{2\alpha_i}{2\alpha_i - 2}\frac{\beta_i(1+\nu_i)}{\nu_i\alpha_i} = \frac{\beta_i(1+\nu_i)}{\nu_i(\alpha_i - 1)}. \tag{18}$$

In addition, we also define the heuristic estimate for the variance as defined in Meinert et al. (2023) by

$$\text{Var}_{\text{heuristic}}[y_i] = \frac{\beta_i(1+\nu_i)}{\nu_i\alpha_i}. \tag{19}$$

The rationale for this heuristic is that the variance of a t-distribution can be approximated by the variance of a normal distribution if the degrees of freedom, that is, $2\alpha_i$, is sufficiently large.

Next, we will derive the virtual observations based on the NIG distribution; for further details, see Jordan (2009). The virtual observations are derived from the Bayesian interpretation of priors. Particularly, this quantity serves as the number of hypothetical data points to have the same effect as the NIG distribution. In the case of the normal inverse gamma distribution, we have two sources of virtual observations, namely for μ_i and σ_i. Based on this, we note that μ_i has ν_i virtual observation, and σ_i^2 has $2\alpha_i$ virtual observation. Thus, now we define total evidence

$$\Phi_i = \nu_i + 2\alpha_i. \tag{20}$$

Thus, Φ_i can be interpreted as a measure of the certainty of the model. Alternatively, when considering Figure 1, increasing Φ_i has the effect of decreasing the uncertainty of (μ_i, σ_i^2) or equivalently going down on the leftmost part of the figure. Here, we follow the definition of Meinert and Lavin (2021) for Φ_i that is corrected from the original definition in Amini et al. (2020).

We need to derive the loss function that maximizes model fit according to our result in Theorem 1. As is common, given a known distribution with unknown parameters, we use the negative log-likelihood (NLL) to maximize the model fit. Thus, here using the fact that $y_i|\gamma_i, \nu_i, \alpha_i, \beta_i$ follow as t-distribution and using the log-likelihood of the t-distribution, we derive

$$\begin{aligned}\mathcal{L}_i^{NLL}(\mathbf{w}) = \frac{1}{2}\log(\frac{\pi}{\nu_i(\mathbf{w})}) - \alpha_i(\mathbf{w})\log(\Omega_i(\mathbf{w})) + (\alpha_i(\mathbf{w}) + \frac{1}{2})\log((y_i - \gamma_i(\mathbf{w}))^2\nu_i(\mathbf{w}) + \\ \Omega_i(\mathbf{w})) + \log\left(\frac{\Gamma(\alpha_i(\mathbf{w}))}{\Gamma(\alpha_i(\mathbf{w}) + 1/2)}\right),\end{aligned} \tag{21}$$

where $\mathbf{w}$ are the weights of the neural network, which we optimize. These model weights determine the parameters $\gamma_i, \nu_i, \alpha_i, \beta_i$ for a given x_i and $\Omega_i = 2\beta_i(1+\nu_i)$. Thus, we have derived the loss function that determines model fit. However, this loss function is insufficient to find the correct parameters (Meinert &

Lavin, 2021). The large amount of weights in $\mathbf{w}$ means that the estimated network can overfit the data and predict a variance that is too small, which leads to an incorrect uncertainty prediction. Thus, in the next subsection, we discuss the regularization term we need to add to the loss function.

3.2.2 Regularization

In this subsection, we will discuss three regularization terms and how they affect estimation. The goal of these regularization terms is to improve the estimation of predictive uncertainty. In this research, we consider the regularization term by Amini et al. (2020), by Meinert et al. (2023), and a newly proposed term. As discussed in the previous section, the log-likelihood loss is insufficient to estimate the parameters. This can be observed by considering the fact that the loss function fits a t-distribution with three unknown quantities γ_i, $\frac{\beta_i(1+\nu_i)}{\nu_i+\alpha_i}$, $2\alpha_i$ with four unknown parameters. Particularly, since two different values $\frac{\beta_i(1+\nu_i)}{\nu_i+\alpha_i}$ can have the same negative log likelihood by scaling β_i and ν_i.

To solve this, Amini et al. (2020) introduced the regularization term

$$\mathcal{L}_i^{R_{wrong}}(\mathbf{w}) = |y_i - \mathbb{E}[y_i]|\Phi_i = |y_i - \gamma_i(\mathbf{w})|(2\alpha_i(\mathbf{w}) + \nu_i(\mathbf{w})), \tag{22}$$

which includes the virtual observations Φ_i. The term here differs from the original in Amini et al. (2020) since we use the corrected virtual observation term derived in Subsection 3.2.1. This term contains two parts, the absolute error of the mean and model evidence, and is supposed to contain a trade-off between them. This means that if we have a good prediction (a small absolute error), the model evidence can be bigger (uncertainty smaller). However, as shown by Meinert et al. (2023) and Oh and Shin (2022), this term is insufficient to regularize the parameters, particularly due to a missing restriction on the parameter β_i. To show this, we can take the derivative of the loss function in Equation (21) including the regularization term, and we find that for all $\beta_i(\nu_i)$ proportional to $\frac{1}{1+\nu_i^-1}$ the negative log-likelihood can be minimized irrespective of ν_i (Meinert et al., 2023).

As such, Meinert et al. (2023) propose to utilize the regularization term

$$\mathcal{L}_i^{R}(\mathbf{w}) = \frac{|y_i - \mathbb{E}[y_i]|}{\sigma_{y_i}}\Phi_i = |(y_i - \gamma_i(\mathbf{w}))|\sqrt{\frac{\alpha_i(\mathbf{w})\nu_i(\mathbf{w})}{\beta_i(\mathbf{w})(1+\nu_i(\mathbf{w}))}}(2\nu_i(\mathbf{w}) + \alpha_i(\mathbf{w})), \tag{23}$$

which now includes β_i by scaling the absolute errors by the scale of the t-distribution of y_i. The main reason for scaling by the absolute errors is that this reduces the effects of large but insignificant errors.

In this thesis, we introduce a new regularization term. The goal of this regularization term is to improve the estimation of the predictive uncertainty. Notably, the regularization term is derived from the definition of the confidence interval, see Equation (25). If the models are correctly estimated, this definition should hold for any sample (y_i, x_i). However, as we have previously discussed, due to overfitting, this constraint often does not hold in practice. As such, the proposed regularization term tries to resolve this.

Definition 1. *We define the adapted regularization term* $\mathcal{L}_i^{R_{new}}$ *as*

$$\frac{|y_i - \mathbb{E}[y_i]|}{\sigma_{\hat{y}_i}} = |y_i - \gamma_i(\mathbf{w})| \sqrt{\frac{\alpha_i(\mathbf{w})\nu_i(\mathbf{w})}{\beta_i(\mathbf{w})(1+\nu_i(\mathbf{w}))}}, \tag{24}$$

which is derived as an approximate bound based on the definition of a confidence interval.

As mentioned, we want that the probability that the actual value y is within the confidence interval $[\hat{\gamma} - t^{2\hat{\alpha}}_{(1-p)/2}\hat{\sigma}, \hat{\gamma} + t^{2\hat{\alpha}}_{(1-p)/2}\hat{\sigma}]$ is equal to p, based on the parameters. This is equivalent to

$$\mathbb{P}(|y - \hat{\gamma}| \leq t^{2\hat{\alpha}}_{(1-p)/2}\hat{\sigma}) \geq p, \tag{25}$$

where $\hat{\gamma}$ and $\hat{\sigma} = \frac{\beta_i(\mathbf{w})(1+\nu_i(\mathbf{w}))}{\nu_i(\mathbf{w})\alpha_i(\mathbf{w})}$ are the parameters estimated, $t^{2\alpha}_{(1-p)/2}$ is the $(1-p)/2$ critical value of a Student t distribution with $2\alpha_i$ degrees of freedom and the probability P depends on possible samples (y, x)

Since we can not guarantee Equation (25) for the whole population (x, y), we instead require this constraint to hold for our training sample (x_i, y_i), which leads to the constraint

$$|y_i - \gamma_i(\mathbf{w})| \leq t^{2\alpha_i(\mathbf{w})}_{(1-p)/2}\sigma_i(\mathbf{w}). \tag{26}$$

This is similar to one of the classical approaches to solve problems with chance constraints, see theorem 1 of Calafiore and Campi (2005). Thus, we have replaced the probabilistic constraint with a constraint over a sample, where the sample is the same as our training sample. The next step is to rewrite Equation 26 using a Lagrangian relaxation as part of the loss function. That is

$$\min_{\mathbf{w}} \sum_i \mathcal{L}_i^{NLL}(\mathbf{w}) + \lambda \frac{|y_i - \gamma_i(\mathbf{w})|}{\sigma_i(\mathbf{w})}, \tag{27}$$

where $\mathcal{L}_i^{NLL}(\mathbf{w})$ is the negative log-likelihood, λ represents the regularization parameter, and where we recognize $\frac{|y_i - \gamma_i(\mathbf{w})|}{\sigma_i(\mathbf{w})}$ as our regularization term $\mathcal{L}_i^{R_{new}}$. Again, we note that γ_i and σ_i are dependent on the model weights $\mathbf{w}$.

Based on the theorem presented in Calafiore and Campi (2005), we can derive some properties on the sample approximation used in the derivation. Particularly, if we have a convex problem, we know that if

$$N \geq \frac{n}{p\zeta} - 1, \tag{28}$$

where N is the sample size and n is the number of observations, then with probability not smaller than $1 - \zeta$ the optimal solution to the minimization problem of the negative log-likelihood with constraint (25) is a feasible solution to the minimization problem of the negative log-likelihood with constraint (26) (Calafiore & Campi, 2005). Thus, for our regularization term to hold, we need a sufficient number of samples, depending on the number of parameters we optimize. We optimize the weights matrix $\mathbf{w}$ which, unfortunately, contains many parameters. Thus, it is likely that we need a large amount of data in order for this regularizer to work well.

3.2.3 Summary

We define two models based on the theory of the previous sections. These are the evidential network and adapted evidential network. They differ in their loss function; the evidential network uses the regularization term (Equation (23)) introduced by Meinert et al. (2023), and the adapted evidential uses the regularization term proposed in this thesis (Equation (24)) . Additionally, they differ in how they estimate the variance and construction of the confidence interval. Particularly, the evidential network uses the heuristic variance and critical values of a normal distribution based on the assumption that the estimated degrees of freedom are sufficiently large, whereas the adapted evidential network uses the variance and critical values of the t-distribution. Appendix A shows how the confidence interval and critical values of the predictions can be constructed. Here, we must note that the used critical values are likely not sampled from the correct distribution. However, it is unclear what alternative distribution could be used.

For both methods, the model weights are estimated based on a neural network architecture, which will be explained further in Section 3.5. A particularity of an evidential neural network is that we need four parameters for each output, representing $\gamma_i, \nu_i, \alpha_i, \beta_i$. Based on the assumed distribution, we restrict the parameters of the model as $\nu_i \geq 0, \alpha_i \geq 1$, and $\beta_i \geq 0$, which is achieved through an activation function.

To summarize, the evidential network uses the loss function

$$\sum_i \mathcal{L}_i^{NLL}(\mathbf{w}) + \lambda_{\text{evidential}} \mathcal{L}_i^R(\mathbf{w}). \tag{29}$$

we estimate the predicted value of y_i as

$$\hat{y_i} = \hat{\gamma}_i \tag{30}$$

and based on the heuristic variance in Equation (19) we estimate

$$s^2_{i,\text{evidential}} = \frac{\hat{\beta}_i(1+\hat{\nu}_i)}{\hat{\nu}_i\hat{\alpha}_i}. \tag{31}$$

Then, we construct the confidence intervals as

$$\Theta(x_i)^{(1-p)/2}_{\text{evidential}} = [\hat{y_i} - z_{(1-p)/2}s_{i,\text{evidential}}, \hat{y_i} + z_{(1-p)/2}s_{i,\text{evidential}}] \tag{32}$$

where $z_{(1-p)/2}$ is the two-sided critical value of the normal distribution. We use the critical value of the normal distribution, assuming that the degrees of freedom are sufficiently large. For multivariate data, we construct the confidence region as an ellipsoid with zero covariance according to Johnson and Wichern (2014), but with the critical values of $z_{(1-p)/2}$.

The adapted evidential network uses the loss function

$$\sum_i \mathcal{L}_i^{NLL}(\mathbf{w}) + \lambda_{\text{adptevidential}} \mathcal{L}_i^{R_{new}}(\mathbf{w}). \tag{33}$$

we estimate the predicted value of y_i as

$$\hat{y_i} = \hat{\gamma}_i \tag{34}$$

and based on the variance in Equation (18) we estimate

$$s^2_{i,\text{adptevidential}} = \frac{\hat{\beta}_i(1+\hat{\nu}_i)}{\hat{\nu}_i(\hat{\alpha}_i - 1)}. \tag{35}$$

Then, we construct the confidence intervals as

$$\Theta(x_i)^{(1-p)/2}_{\text{adptedevidential}} = [\hat{y}_i - t^{2\hat{\alpha}_i}_{(1-p)/2} s_{i,\text{adptevidential}}, \hat{y}_i + t^{2\hat{\alpha}_i}_{(1-p)/2} s_{i,\text{adptevidential}}], \tag{36}$$

where $t^{2\hat{\alpha}_i}_{(1-p)/2}$ is the two-sided critical value of the student t distribution with $2\hat{\alpha}_i$ degrees of freedom. For multivariate data, we construct the confidence region as an ellipsoid with zero covariance according to Johnson and Wichern (2014), but with the critical values of $t^{2\hat{\alpha}_i}_{(1-p)/2}$. These models are compared in the result section.

3.3 Monte Carlo dropout

As mentioned in Section 3.1, the second method to estimate that we will use is Monte Carlo dropout. Monte Carlo dropout was initially introduced to reduce overfitting, but it has been shown to be an effective way of estimating uncertainty (Hron et al., 2018). The main idea is that the dropout layers add minor disturbances in the network that highlight the output of similar but slightly different inputs. Thus, the main requirement for a Monte Carlo dropout model is that this randomness is sufficient to capture the uncertainty in the model.

To define an MC-dropout network with Bernoulli dropout, we introduce the variable $z_{l,j} \in \{0, 1\}$ such that $z_{l,j} = 1$ if node j in layer $l-1$ is active, and $z_{l,j} = 0$ if the node is dropped out and equivalently

$$z_{l,j} \sim \text{Bernoulli}(q), \tag{37}$$

where q is the dropout probability. Furthermore, we define $\mathbf{z}_k$ as the matrix with parameters $z_{l,j}$ for draw k. Now, using this definition, we can define the weight matrix of the network $\mathbf{W}_k$ as the result of performing the dropout according to $\mathbf{z}_k$ on the weights matrix $\mathbf{w}$, where the index k denotes the draw. Further explanation of nodes, layers, and weights can be found in Section 3.5.

Using this definition, we find that the output of the dropout model is highly multi-modal. This can be seen due to the bi-modal value of each node in the dropout, based on $\mathbf{z_k}$. Thus, due to the dropout, the output of each node can be zero or its associated value, which leads to many possible outcomes.

Using this model, we can estimate the predictions as

$$\hat{y}_i = \frac{1}{K}\sum_{k=1}^{K} y^*(x_i, z_k), \tag{38}$$

where x_i are the input variables, y^* is the output of the network for draw k, z_k is the dropout for draw k, and K represents the number of sample predictions . This is similar to performing K stochastic forward passes through the network and taking the average as a result. We determine the variance using

$$s^2_{\text{dropout}} = \frac{1}{K-1}\sum_{k=1}^{K}(y^*(x_i, z_k) - \hat{y}_i). \tag{39}$$

This is equivalent to calculating the sample variance; here, we use the same draws $y_i^*(x_i, z_k)$ as during the calculation of $\hat{y}_i$ for ease of calculation. One drawback of the MC dropout approach is that due to the multi-modal nature of the output layer, the predictions of Equations (38) and (39) only approximate the true values. Furthermore, since z_k is random, we have little control over how much variance is introduced.

To calculate the confidence intervals, we use that the different runs are independent. Then, based on the central limit theorem, we derive

$$\Theta(x_i)^{1-p}_{\text{dropout}} = [\hat{y}_i + z_{(1-p)/2} s_{i,\text{dropout}}, \hat{y}_i - z_{(1-p)/2} s_{i,\text{dropout}}], \tag{40}$$

where $\hat{y}_i$, s_{dropout} are predicted using Equation (38) and (39), and $z_{(1-p)/2}$ represents the critical value of the two-sided confidence interval of a normal distribution. The critical values might not be accurate since the assumption that different runs are independent is questionable since there is an overlap in dropped-out weights between runs; nevertheless, it should give a good approximation. For multivariate data, we construct the confidence region as an ellipsoid with zero covariance according to Johnson and Wichern (2014), but with the critical values of $z_{(1-p)/2}$.

When we use dropout as an approximation tool to predict the uncertainty, we need to take care that it can no longer be interpreted as a regularization tool (Hron et al., 2018). Thus, as recommended by Hron et al. (2018), we will add an additional l^2 weight regularization to each layer. The benefit of this is twofold: on the one hand, we reduce overfitting, and thus, our results should be more generalizable, and on the other hand, it leads to good theoretical properties. Specifically, as shown by Gal and Ghahramani (2016), a Monte Carlo dropout model with l^2 regularisation is equivalent to variational inference with a normal prior. Thus, the loss function can be written as

$$L_{dropout} = \frac{1}{N} \sum_i^n (y_i - y_i^*)^2 + \lambda \sum_{i,j} \mathbf{w}_{i,j}^2, \tag{41}$$

where $\frac{1}{N} \sum (y_i - y_i^*)^2$ represents the MSE loss and $\lambda \sum_{i,j} \mathbf{w}_{i,j}^2$ represents the l^2 loss function over the network weights, and λ is defined as in Gal and Ghahramani (2016).

Now, implementing the model is relatively straightforward. We construct a neural network (see Section 3.5) with dropout and l^2 regularization on every layer. Next, the training remains unchanged compared to deterministic neural networks. To calculate the predictions, we will use the following implementation:

Algorithm 1 Estimate mean and standard deviation based on Monte Carlo dropout

```
function PREDICTDROPOUT(model, $x_{pred}$,$K$)    ▷ Where model - the dropout model,$x_{pred}$ - array of x to be predicted, $K$ - number of draws
    means=[]
    stds=[]
    for $x_i$ in $x_{pred}$ do
        $x_i^{repeated} = repeat(x_i, K)$
        $y^*$ = model.predict($x_i^{repeated}$)    ▷ During prediction we also use MC Dropout
        means.append( $y^*$.mean())
        stds.append( $y^*$.std(ddof=1))
    end for
end function
```

Thus, in this pseudocode, we repeat x_i, K times, and for each value, we will generate the outputs y^*, where y^* is a list of length K. In practice, the K times repetition of x_i leads to computational overhead. Lastly, we determine the mean and standard deviation according to Equation (38) and (39).

3.4 Ensemble

The main idea of determining the uncertainty using an ensemble of neural networks is that different models will produce different predictions due to varying hyper-parameters and the highly non-linear landscape of the loss function. It has been shown by Herron et al. (2020) that the ability to predict the uncertainty of an ensemble is dependent on the variability of the base networks. Thus, ideally, we need to construct different networks that make errors based on different elements of the input data.

To introduce this uncertainty, we vary several elements in each network. Firstly, as mentioned by Lakshminarayanan et al. (2017), we randomize the order of the data for each model; this difference in order will, in combination with stochastic gradient descent, lead to models converging to different local optima. Secondly, as advised by Herron et al. (2020), we rely on networks with different topologies. In particular, for the architecture of the model, we vary the number of layers in the network, the number of nodes per layer, and the activation function of each layer. Furthermore, regarding the optimization process, we vary the batch size and optimizer. Finally, to obtain the prediction and uncertainty, we perform model averaging and calculate the variance (Renda et al., 2019). This is equivalent to Equation (38) and (39); however, now the K stochastic forward passes are replaced by the result of the R different models. We implement it as follows.

Algorithm 2 Estimate mean and standard deviation based on ensemble

```
function PREDICTENSEMBLE(list(models), $x_{pred}$,)    ▷ Where model - ensemble,$x_{pred}$ - array of x to be predicted,
    predictions=[]
    for model in list(models) do
        $prediction_i$ = model.predict($x_{pred}$)
        predictions.append($prediction_i$)
    end for
    means = predictions.mean(axis=0)
    std = predictions.std(axis=0,ddof=1)
end function
```

Mathematically we calculate

$$\hat{y}_i = \frac{1}{R}\sum_{r=1}^{R} y_r^*(x_i) \tag{42}$$

and

$$s_{i,\text{ensemble}}^2 = \frac{1}{R-1}\sum_{r=1}^{R}(y_r^*(x_i) - \hat{y}_i)^2, \tag{43}$$

where R is the number of models used in the ensemble, and y_r^* is the output of model r. We determine the confidence interval based on the central limit theorem, given that we have a sufficient number of models as

$$\Theta(x_i)_{\text{ensemble}}^{1-p} = [\hat{y}_i + z_{(1-p)/2}s_{i,\text{ensemble}}, \hat{y}_i - z_{(1-p)/2}s_{i,\text{ensemble}}], \tag{44}$$

where $\hat{y_i}$, $s_{i,\text{ensemble}}$ are predicted using the ensemble, and $z_{(1-p)/2}$ represents the critical value of the two-sided confidence interval of a normal distribution. For multivariate data, we construct the confidence region as an ellipsoid with zero covariance according to Johnson and Wichern (2014), but with the critical values of $z_{(1-p)/2}$.

3.5 Neural networks

In this section, we give an overview of neural networks with an explicit interest in the models used for our use cases. First, we briefly introduce neural networks. Then, we go into detail on two types of time series layers. Next, we explain many-to-many forecasting, and lastly, we discuss the implemented model architectures.

3.5.1 Introduction of neural networks

A neural network represents a computational model inspired by the biological structure and function of the human brain. The network consists of interconnected nodes known as neurons, organized into layers. The typical architecture includes an input layer receiving initial data, hidden layers for processing information through weighted connections and non-linear activation functions, and an output layer generating final predictions. During training, the network adjusts connection weights $\mathbf{w}$ to minimize some loss function between predictions and actual outcomes, employing optimization techniques like gradient descent and backpropagation. Neural networks have been applied to various tasks, such as image and speech recognition, natural language processing, and pattern recognition, rendering them pivotal in modern machine learning and artificial intelligence systems. We recommend the book of Goodfellow et al. (2016) for further details about neural networks.

3.5.2 RNN-layers

In this thesis, we use a one-dimensional convolutional layer. This convolutional layer looks at successive values and modifies them using a filter with certain weights; see Figure 2. These filters can extract local patterns and features and return a feature map with a higher dimensionality than the original data. This feature map is then used in successive layers of the neural network, which enables us to find patterns and trends in the data.

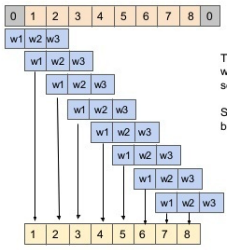

Figure 2: Representation of a one dimensional convolutional filter adapted from Ghosh (2018)

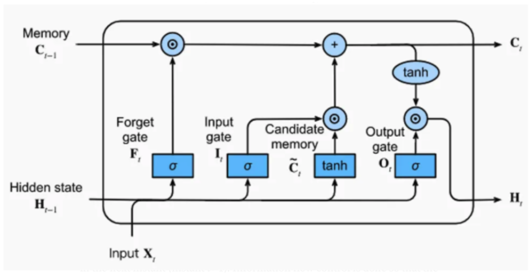

Figure 3: Overview of LSTM module adapted from Calzone (2022)

In addition to the convolutional layer, we also use a Long Short-Term Memory (LSTM) layer. This LSTM is a recurrent neural network architecture designed to address the challenges of learning long-range dependencies in sequential data. Traditional RNNs struggle with capturing information from distant time steps due to the vanishing gradient problem. LSTMs overcome this limitation by introducing a more sophisticated memory cell structure. LSTMs have a set of gates, including an input gate, a forget gate, and an output gate, which control the flow of information through the cell, see Figure 3. This allows LSTMs to selectively remember or forget information over long sequences, making them well-suited for time series tasks. They are particularly suited for forecasting pedestrian paths since they can predict trends.

3.5.3 Many-to-many forecasting

Using neural networks, we are not only able to predict a singular output, but we are also able to create a many-to-many forecast. Here, many-to-many forecasting refers to a predictive modeling technique that aims to predict multiple future outcomes simultaneously based on historical data. This contrasts with many-to-one forecasting, where only a single future value is predicted. Many-to-many forecasting is particularly interesting here since it has computation benefits.

In many-to-many forecasting with neural networks, the output layer is adjusted to accommodate the prediction of multiple future time steps. This adjustment involves setting the output layer to produce multiple outputs, each corresponding to a specific future time step. For example, if the task is to predict the following three values in a time series sequence, the output layer of the neural network would have three output nodes, each representing the predicted value for a particular time step. Therefore, the dimensionality of the output layer increases to match the number of future time steps to be forecasted.

3.5.4 Architecture

For the model of the regression task, we use a standard dense neural network. Particular to our use-case is that we only have a single input node. Furthermore, we utilize three hidden layers with 64, 64, and 32 nodes, respectively. Lastly, the output layers are dependent on the uncertainty estimation model. As previously discussed, we need four output nodes for the evidential networks, representing the values γ, ν, α, and β, or a single output node representing y^* for the Monte Carlo dropout and ensemble methods. Furthermore, we use the rectified linear unit (relu) activation function in each layer of the network. This activation function takes the maximum output of a node and zero. We optimize the network using the Adam optimizer with a batch size of 16.

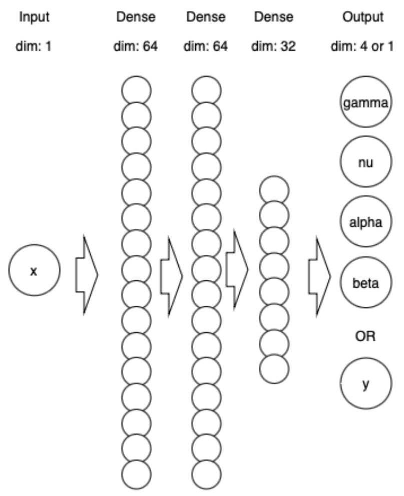

Figure 4: Network architecture for the regression problem

The neural network architecture for the pedestrian dataset is more sophisticated since we take a time series as the input and as an output. According to the results of Nayak et al. (2022), we use an encoder-decoder structure with one-dimensional convolutional layers as the encoder and LSTM layers as the decoder. The encoder-decoder is practical in the time series application because the encoder can find patterns in the dataset. Then, the decoder can utilize these patterns to create the output series. For this model, we utilize two encoding layers and one decoding layer, all with the relu activation function. This is in accordance with the results by Nayak et al. (2022). The dimension of the output layer is equal to twelve by two for the ensemble and dropout models and twelve by eight for the evidential models. The twelve represents the number of time steps, and the two resembles the x, and y-axis; for the evidential model, we have twelve times eight output parameters since we have an γ, ν, α and β for both the x as y direction at each time step. This is visualized in Figure 5.

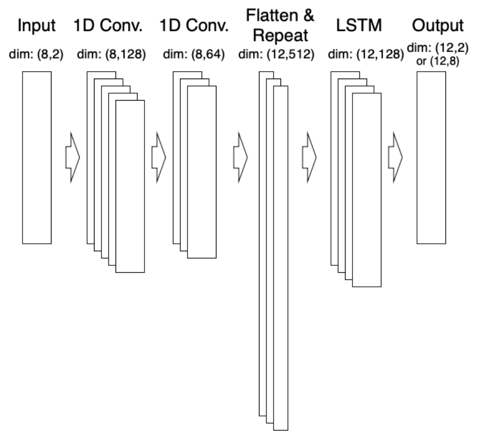

Figure 5: neural network structure for the pedestrian dataset

3.6 Metrics

This section discusses the metrics we use to compare the models. First, we discuss the metrics we use to measure the prediction's accuracy. Then, we discuss how to measure the quality of the uncertainty forecast.

3.6.1 Accuracy metric

Even though one of the goals of the methodology is to forecast the uncertainty of the model, it remains important to obtain accurate predictions. Some authors, such as Nayak et al. (2022), have seen that incorporating uncertainty prediction during the forecast step can improve the accuracy of the prediction. Moreover, from a practical point of view, a model with good uncertainty estimation but worse accuracy will likely not be implemented.

For the regression problem, we measure accuracy using the mean squared error (MSE) metric. This can be calculated using

$$\text{MSE} = \frac{1}{N_{test}} \sum_{i}^{N_{test}} (y_i^{test} - \hat{y}_i)^2, \tag{45}$$

where we use the observations that we have saved for the test set.

For our pedestrian dataset, we predict a time series as output, and thus, we need to modify the MSE. As such, we use two different metrics; firstly, we use the average displacement error (ADE), which can be seen as the average MSE over all timesteps. This metric indicates how well the model follows the true data over all timesteps and is calculated using

$$\text{ADE} = \frac{1}{N_{test}T} \sum_{i}^{N_{test}} \sum_{t}^{T} (y_{i,t}^{test} - \hat{y}_{i,t})^2, \tag{46}$$

where $\hat{y}_{i,t}$ represent the prediction of observation i at time t.

Additionally, to see how well the model predicts in the long term, we calculate the final displacement error (FDE). This metric indicates how well the model can predict at the end of the forecast horizon and, as such, how well it can be used to make decisions further in advance. The FDE is as

$$\text{FDE} = \frac{1}{N_{test}} \sum_{i}^{N_{test}} (y_{i,t_f}^{test} - \hat{y}_{i,t_f})^2, \tag{47}$$

where t_f is the last timestep and $\hat{y}_{i,t_f}$ is the prediction of test observation i at this timestep.

3.6.2 Uncertainty metric

Similar to the article of Dewolf et al. (2023), we want to test the coverage probability of the confidence interval. That is, we want to test if the $p\%$ predicted confidence interval contains $p\%$ of actual values of the test set. Mathematically, we write this as

$$\text{CIS} = \frac{1}{N_{test}} \sum_{i}^{N_{test}} \mathbb{1}(y_i^{test} \in \Theta(x_i^{test})), \tag{48}$$

where $\mathbb{1}$ denotes the indicator function. The Confidence Interval Score (CIS) is an approximate measure of the quality of uncertainty estimation. This approximation counts the number of data points where the true value y_i^{test} is contained within the Confidence interval $[\hat{y}_i - ks_i, \hat{y}_i + ks_i]$ where $\hat{y}_i$ is the predicted value, s_i is the predicted standard deviation and k is some constant, related to the confidence level of the interval.

For two-dimensional data, we define a point to be inside the confidence interval if it is inside the ellipse with center $\hat{y}_i$ and axes ks_i. Mathematically, we can test this using

$$\frac{d_1}{r_1} + \frac{d_2}{r_2} \leq 1, \tag{49}$$

where d_j is the euclidean distance between $\hat{y}_i$ and y_i^{test} in dimension j and r_j is the confidence length in dimension j. Now we define the CIS at time t as

$$\text{CIS(t)} = \frac{1}{N_{test}} \sum_{i}^{N_{test}} \mathbb{1}(y_{i,t}^{test} \in \Theta(x_{i,t}^{test})), \tag{50}$$

where $\mathbb{1}$ denotes the indicator function and on average over all timesteps as

$$\text{CIS} = \frac{1}{N_{test}T} \sum_{i}^{N_{test}} \sum_{t}^{T} \mathbb{1}(y_{i,t}^{test} \in \Theta(x_{i,t}^{test})), \tag{51}$$

where $\mathbb{1}$ denotes the indicator function. The Confidence Interval Score can not be compared strictly numerically since it needs to be accurate and small enough. Instead, based on the confidence level p, we want the CIS score to contain approximately the p percentage of observations.

In addition to the confidence intervals constructed using the critical value, we consider the confidence interval based on Chebyshev inequality, which is not based on distributional assumptions. In particular, we can say that the confidence interval $k = 4.5$, equivalently

$$\Theta(x_i) = [\hat{y}_i - 4.5s_i, \hat{y}_i + 4.5s_i], \tag{52}$$

should contain at least 95% of the observations, irrespective of the output distribution. For multivariate data, we use the confidence region as an ellipsoid with lengths according to $4.5s_i$.

4 Data

In this research, we test the methods on two use cases. Firstly, we test the uncertainty estimation methods on a non-linear regression problem with heteroskedastic variance. This data is relatively simple compared to the pedestrian dataset, and as such, we can use it as a starting point for our analysis. Secondly, we test the methods on pedestrian forecasting datasets. In these datasets, camera footage of pedestrians' walks is transformed into a time series of locations, for which we try to predict what locations they walk towards next. For this application, we use four different datasets based on three locations to generalize the results for different scenarios.

4.1 Regression data

The first dataset that we use is a regression dataset. This means we have a continuous input variable $x_i \in \mathbb{R}$ for which we try to predict a continuous output variable $y_i \in \mathbb{R}$. This dataset is similar to the experiments done by Amini et al. (2020) and Malinin et al. (2020).

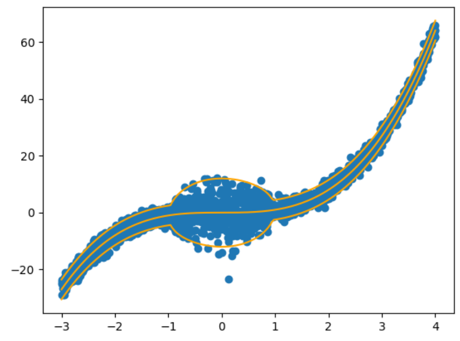

Figure 6: Regression data

We define the relation between y_i and x_i using a third-degree polynomial. Then, we construct x as 2000 uniformly distributed points between negative three and four. Based on this, we construct the errors as a function of x_i where the variance is three with a spike of 36 when x_i is close to zero. Mathematically, we represent this as

$$\begin{aligned} y_i &= x_i^3 + \epsilon_i, \\ \epsilon_i &\sim \mathcal{N}(0, \sigma_i^2), \\ \sigma_i^2 &= \max(3, 36(1 - x_i^2)). \end{aligned} \tag{53}$$

The constructed dataset is visualized in Figure 6; the lines represent the mean and the 95% confidence bound.

4.2 Pedestrian data

The data that we use to test the methods in a real-world application is pedestrian movement forecasting. This use-case was previously considered in Nayak et al. (2022) in combination with a Monte Carlo dropout method to forecast uncertainty. In particular, we have four datasets based on three locations. We have the ETH and HOTEL scenarios of the ETH dataset, initially presented in Pellegrini et al. (2009), and we have the ZARA01 and ZARA02 scenarios from the UCY dataset (Lerner et al., 2007). These datasets are the most commonly used in pedestrian forecasting (Amirian et al., 2020). Furthermore, the data is retrieved from Amirian et al. (2020).

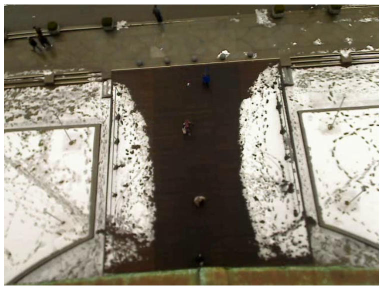
Figure 7: Image of ETH scenario

Figure 8: Image of HOTEL scenario

Figure 9: Image of ZARA01 scenario

Figure 10: Image of ZARA02 scenario

We thus have four data sets based on two papers. These datasets differ in the location where they were captured. Particularly, the ETH data was captured on the ETH campus in Zurich. The HOTEL data was captured from the perspective of a hotel. Lastly, ZARA01 and ZARA02 were captured from a Zara store's second floor. This data was originally captured on video, and later, each pedestrian was tracked and converted to a path of coordinates. Figures 7-10 show a screenshot of each scenario.

In the dataset provided by Amirian et al. (2020), these videos are processed to contain a time series of different pedestrian movement paths. In particular, they are represented in a file with the columns pedestrianid, framenumber, posx, posy, representing the pedestrian identifier, timestep and position. Based on these features, we construct our training set. Specifically, to use them in training, we need every sample to be the same length. We follow the common choice within pedestrian forecast prediction to forecast twelve timesteps based on eight timesteps of input (Nayak et al., 2022). Mathematically that is $x_i \in \mathbb{R}^{8,2}$ and $y_i \in \mathbb{R}^{12,2}$.

As such, we need to process the input data further before training. These steps are visualized in Figure 11. In particular, we first filter the data on all IDs with at least 20 timesteps. Then, we create the train and test split; this ensures that there is no overlap of pedestrians in the test set due to data augmentation. After that, we augment the data using a moving window, and lastly, we save the data in the correct format.

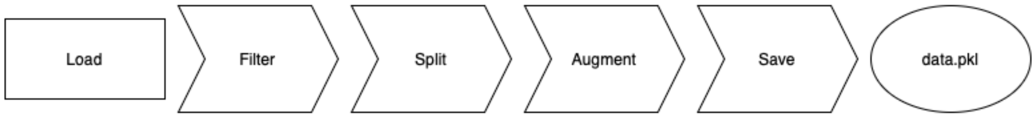

Figure 11: Data pipeline

We must augment the data to increase the number of samples (Nayak et al., 2022). In particular, from every sequence longer than 20, we create subsequences of length 20 with one observation difference using a moving window. For example,

$$[t_1, \ldots, t_{30}] \Rightarrow \begin{array}{c} [t_1, \ldots, t_8], [t_9, \ldots, t_{20}] \\ \vdots \\ [t_{11}, \ldots, t_{18}], [t_{19}, \ldots, t_{30}]. \end{array} \tag{54}$$

Thus, the sample size will be larger since a single pedestrian can be part of multiple data samples. Although this augmentation leads to many similar samples, it is unlikely to heavily affect the results. Since we first create a train and test split, different pedestrians will only be part of either the train or test set, and no overlap is possible.

In Table 2, we show some descriptive statistics of our dataset. We immediately notice that the number of series increases significantly for all datasets due to the data augmentation. This ensures that we have sufficient training data. Moreover, we notice that for all datasets, we sometimes observe that some pedestrians do not travel far. Figure 14 shows an example of this, where a pedestrian is completely standing still. Lastly, we notice that even though there are differences in distance traveled within the train and test set, they seem, in general, to be similar in behavior.

Table 2: Descriptive statistics pedestrian datasets

Dataset	Num. Series Unprocessed	Avg. number of timesteps	Num. Series *		Avg. distance (m)		Min. distance (m)	
			Train	Test	Train	Test	Train	Test
ETH	360	24.744	1809	805	9.257	8.310	0.053	0.000
HOTEL	390	16.779	789	408	3.412	2.936	0.000	0.000
ZARA01	148	33.945	1413	821	8.161	5.996	0.600	0.029
ZARA02	204	46.750	3945	1796	4.056	3.866	0.002	0.009

Note: * the number of series increased due to data augmentation.

To further introduce the data in Figures 12 to 14, we show some sample pedestrian walking paths. Here, the dark blue dots show the input data, and the light blue triangles show the data we need to predict. Within the dataset, there can be significant differences between time series. Figure 12 shows a relatively straightforward path where a pedestrian walks from the bottom-left to the top-right at a relatively constant speed. Figure 13 is already more difficult to predict since the pedestrian walked in a U-shaped pattern. This is particularly difficult since only the blue dots will be given as input data to the methods, and as such, it will need to learn that, in some cases, pedestrians might move in a U-shaped pattern. Lastly, Figure 14 shows a pedestrian standing still during the entire interval. As such, this sample is not interesting from a trajectory prediction perspective. Nevertheless, from an uncertainty quantification perspective, it is interesting how the methods might incorporate the possibility that the pedestrian could start walking.

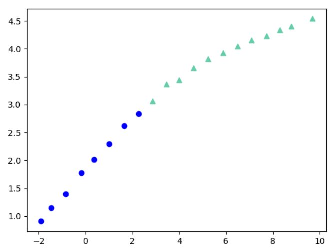

Figure 12: First example of a pedestrian movement path

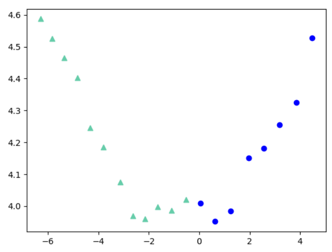

Figure 13: Second example of a pedestrian movement path

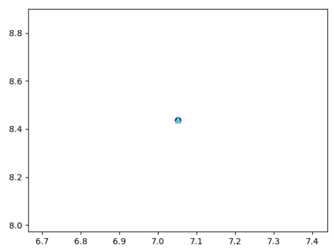

Figure 14: Third example of a pedestrian movement path

5 Results

The result section is divided based on the different use cases and datasets. First, we present the results of the regression data using figures and metrics. Then, we present figures for the uncertainty forecast of the ETH dataset. Next, we present the metrics for this dataset, and lastly, we present the metrics of the HOTEL, ZARA01, and ZARA02 datasets.

5.1 Regression data

The results of this section are based on the dataset set as described in Section 4.1. Furthermore, the evidential model, adapted evidential, and Monte Carlo dropout model have the neural network structure described in Section 3.5. We use a regularization parameter for the evidential and adapted evidential model equal to one. The Monte Carlo dropout model uses a dropout rate of 0.2, and we use 100 samples to estimate the mean and variance of each prediction. The ensemble method is based on ten networks with varying random architectures.

5.1.1 Figures

The results of the regression are presented in Figures 15 to 18. These figures show the data points in the test set, where the x-axis shows the scalar input variable and the y-axis shows the scalar target variable. In addition, each figure shows the predicted mean and upper and lower 95% confidence intervals. The construction of these intervals is described in the methodology section. The training data with target trend and confidence interval can be found in Figure 6 of the data section.

Considering the figures, we notice that Figures 15 and 16 look similar and 17 and 18 look similar. Next, we notice that all networks predict the correct trend line through the data. Notably, the methods can predict a non-linear function. This is likely due to the neural network architecture, which can model non-linearities. Now, looking at the prediction of the confidence bounds, we see that the evidential and adapted evidential models can model the heteroskedasticity of the model. In contrast, the dropout and Monte Carlo dropout models can not. Based on the literature, we know that they do not give optimal intervals for data sets that do not follow a normal distribution (Dewolf et al., 2023).

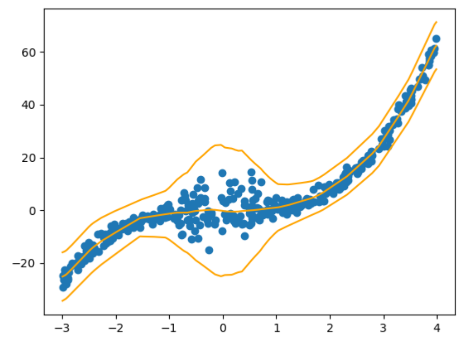

Figure 15: Result of evidential model with 95% confidence bound

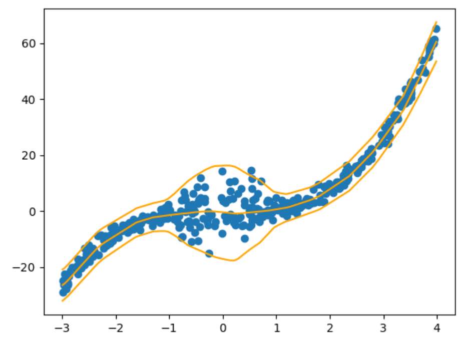

Figure 16: Result of adapted evidential model with 95% confidence bound

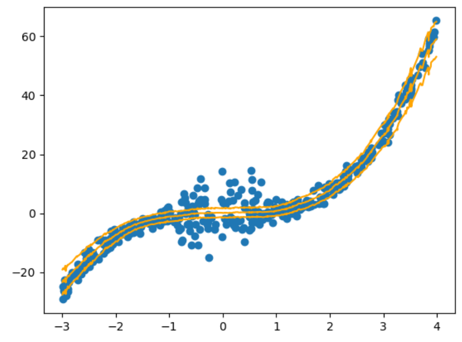

Figure 17: Result of MC Dropout model with 95% confidence bound

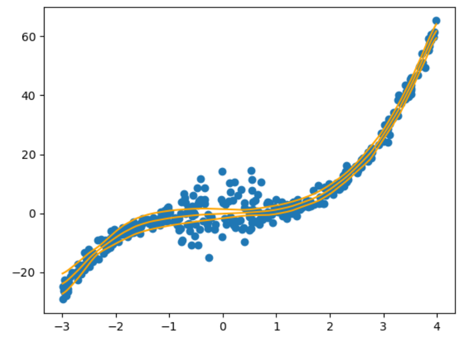

Figure 18: Result of ensemble model with 95% confidence bound

If we now compare Figures 15 and 16, we see that the confidence bounds of Figure 15 are wider than 16. Although a wider bound will have more data points within the confidence interval, this is not desired since it is excessively broad. As such, we prefer the uncertainty estimate for the adapted evidential model for this dataset based on the figures. Looking at Figure 17, we notice that the bound of the confidence interval is not smooth for x larger than 2.5; we hypothesize that this is due to the high multi-modality of the output distribution and, thus, our reliance on the sampled data points. We also notice that the predicted intervals around -3 to -1 and around 2 to 4 are more accurate compared to Figure 18. Figure 18 predicts a confidence interval that is too thin almost everywhere in the figure. We suspect that the relative simplicity of the data causes this and that all models in the ensemble converge to the same optimum; thus, potentially, more variety needs to be introduced in the models of the ensemble for this dataset.

5.1.2 Metrics

Using the visualization of Figures 15 to 18, we have seen that all models seem to predict the correct trend but that there are significant differences in predicted uncertainty. In this section, we quantify these insights in Table 3 using the metrics defined in the methodology section.

First, considering the MSE, we observe that all models are relatively similar, with the ensemble being slightly more accurate than the other models. Next, we look at the confidence interval score (CIS) for the 50% and 95% confidence interval and the 95% confidence bound based on Chebyshev. First, looking at the evidential model, we see that for both p is 0.5 and p is 0.95, more points lie inside the confidence interval than expected; we have previously seen this in the figure where the confidence interval had too much width. Next, the CIS of the adapted evidential model is still larger than 0.5 and 0.95 but to a lesser extent than the evidential model. For both the dropout and ensemble, the CIS is significantly too low, likely caused by the fact that the heteroskedasticity around 0 is not modeled correctly. Even based on the Chebyshev bound, it is significantly lower than 95%, indicating a misspecification. Thus, for this dataset, we prefer the adapted evidential model, which potentially works well due to the high number of data points required for the regularization term.

Table 3: Results of metrics for the regression dataset

Model	MSE	CIS (p)			Training time (s)	Prediction time (s)
		0.5*	0.95*	>0.95**		
Evidential Network	3.256	0.672	1.000	1.000	3.773	0.066
Adapted Evidential Network	3.217	0.575	0.962	1.000	3.731	0.052
MC Dropout	3.420	0.252	0.637	0.755	3.470	9.251
ensemble	3.197	0.207	0.500	0.792	37.520	0.477

Note: * based on the confidence interval, ** based on Chebyshev bound.

Comparing training times, we notice that the evidential, adapted evidential, and MC dropout models are relatively similar in training time, whereas the ensemble takes about ten times as long. This is logical since we need to train ten models to use as an ensemble. Comparing prediction times, we see that the evidential and adapted evidential are the fastest and only take around 0.05 seconds to predict the test data points. The ensemble method takes around ten times longer at 0.5 seconds to estimate the predictions since we must perform ten forward passes for each data point. The dropout method creates predictions that are the slowest at 9.5 seconds or around 185 times slower than the evidential model. This can be partly explained by the fact that we perform 100 forward passes for each prediction and partly due to the overhead of putting the data in the correct format when predicting.

5.2 Pedestrian data

In this section, we present the results for the pedestrian datasets; details on the datasets can be found in Section 4.2, and the evidential, adapted evidential, and Monte Carlo dropout models have the neural network structure described in Section 3.5. Again, we use a regularization parameter equal to one for the evidential and adapted evidential model. The Monte Carlo dropout model uses a dropout rate equal to 0.2, and we use 100 samples to estimate the mean and variance of each prediction. The ensemble method is based on ten networks with varying architectures. Next, we look at several examples of the uncertainty predictions for the ETH dataset, and then we calculate and discuss the metrics for the ETH dataset in detail. Lastly, we give an overview of the results of the HOTEL, ZARA01, and ZARA02 datasets.

5.2.1 Figures

We show the first examples in Figures 19 to 22. This example shows a relatively simple movement path where someone walks from the top left to the bottom right at a relatively constant speed. In the figure, the dark blue dots are given as input, the light blue triangles are the target points, the red pluses represent the predicted mean, and the red ellipsoids show the uncertainty interval.

In Figure 19, we see the prediction of the evidential model, where we notice that the trend is slightly off, but the predicted distance traveled is relatively accurate. The predicted confidence interval gets larger over time, which is exactly what we would hope. Although we notice that not all points are within the confidence interval, this makes sense since we plot the 50% confidence interval. In Figure 20, we plot the results of the adapted evidential model. Here, we see a slightly more curved predicted movement path compared to Figure 19, which leads to a surprisingly accurate prediction for the last time steps. The confidence interval of the adapted evidential model also gets larger over time but is smaller than the predicted interval of the evidential model. Potentially, this means that the size of the confidence interval is too small, or that the confidence interval of the evidential is too large. Next, we consider the results of the Monte Carlo dropout model in Figure 21. The predicted movement path is peculiar, mainly because we notice that the second predicted timestep and speed of the predicted path are incorrect. Furthermore, we see that the size of the confidence interval is almost constant over time. Potentially, this is caused by insufficient variance introduced by the dropout. Figure 22 shows the output of the ensemble model. Here, we notice that the trend is in the right direction. However, the predicted traveled distance is too low. Unfortunately, similar to the Monte Carlo dropout model, we do not see an increasing size of the confidence interval over time. We suspect this is due to insufficient variance in the models.

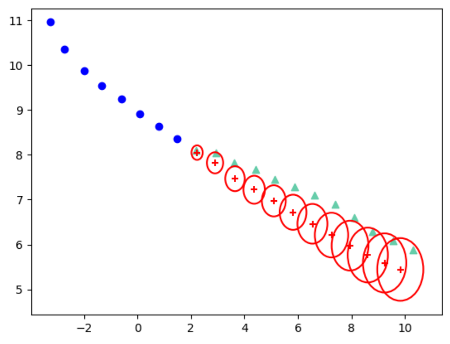

Figure 19: Result of Evidential model with 50% confidence interval

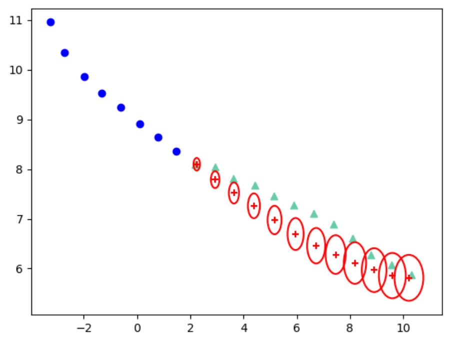

Figure 20: Result of the adapted evidential model with 50% confidence interval

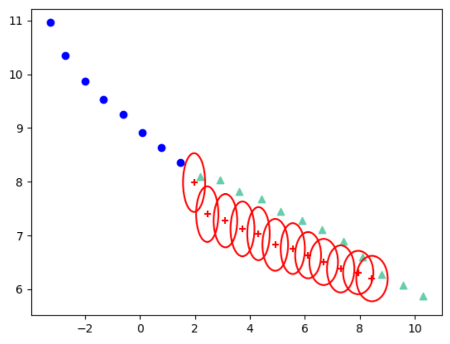

Figure 21: Result of the MC dropout model with 50% confidence interval

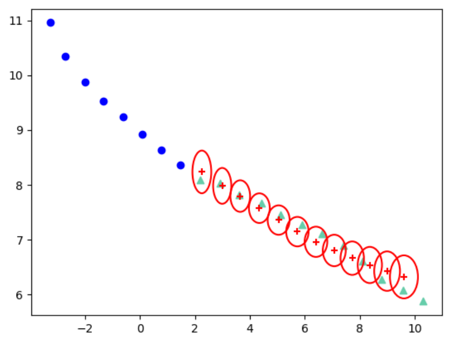

Figure 22: Result of the ensemble model with 50% confidence interval

Next, in Figures 23 to 26, we see a slightly more complicated situation. Here, the input is a pedestrian moving from right to left, and then they decide to take a left turn during the prediction horizon. Again, the evidential network performs well, as shown in Figure 23. Although the predicted trend is slightly off, this is compensated by a larger confidence interval. A downside is that the confidence interval could be larger than necessary. The adapted evidential network, shown in Figure 24, predicts a similar network as the evidential model, only the confidence interval is smaller. This highlights the trade-off between the probability that a point is inside the confidence interval and the excessive width of the confidence interval. In Figure 29, we see the result for the MC dropout model. Except for the first predicted step, the predicted trend looks promising. Again, similar to the previous example, the size of the confidence interval does not seem to increase sufficiently over time. The results of the ensemble model, see Figure 26 are not as good. Notably, the trend is not going in the right direction, and the confidence interval is too small. Potentially, in this example, all neural networks of the ensemble converged to a similar "wrong" path. This example shows the benefit of using the evidential family of models, particularly if we consider the unobserved possibility that the pedestrian would not have made the turn.

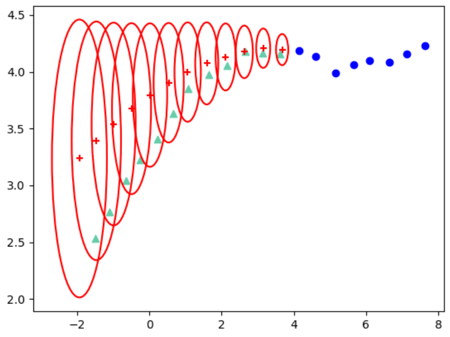

Figure 23: Result of Evidential model with 50% confidence interval

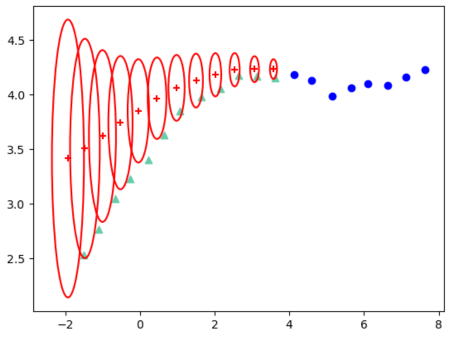

Figure 24: Result of the adapted evidential model with 50% confidence interval

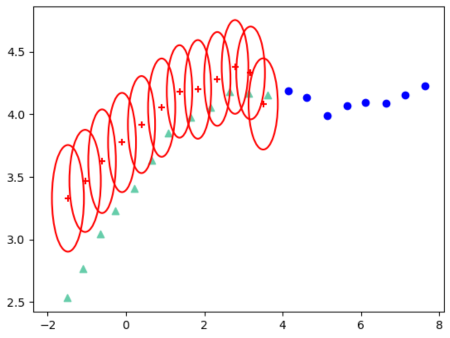

Figure 25: Result of the MC dropout model with 50% confidence interval

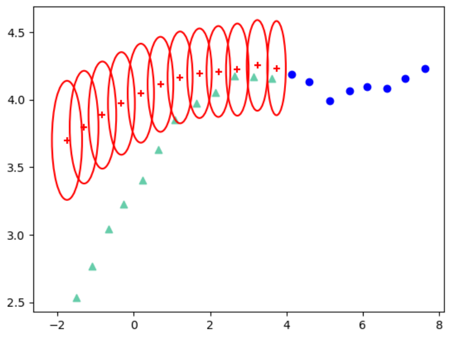

Figure 26: Result of the ensemble model with 50% confidence interval

In the third example, shown in Figures 27 to 30, we show someone walking from the bottom right to the top left, who then decides to make a big left turn. Unfortunately, we see that none of the models can accurately predict the last steps of the output. The evidential model in Figure 27 can predict most points, except for the second and last time step. Again, the confidence interval gets larger over time. The adapted evidential model, again, predicts confidence intervals that are too small. In figure 29, we see some strange behavior; the predicted path is unevenly spaced, and the confidence intervals seem larger than in the previous examples. We suspect this is caused by conflicting realization of the highly multi-modal output of the MC dropout model. The ensemble model in Figure 30 shows an incorrect path with an almost constant confidence interval. Again, the evidential network seems to perform best.

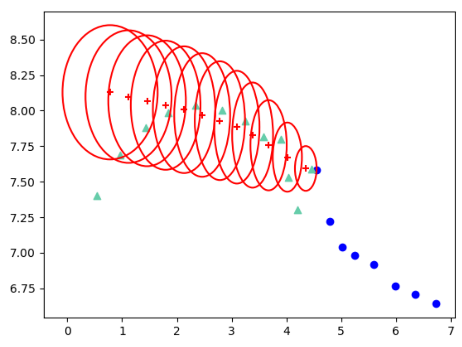

Figure 27: Result of Evidential model with 50% confidence interval

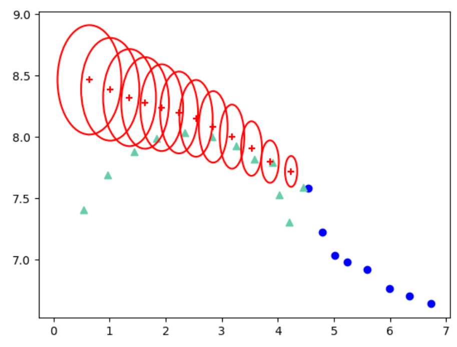

Figure 28: Result of the adapted evidential model with 50% confidence interval

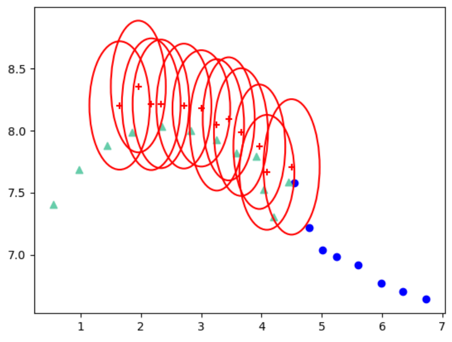

Figure 29: Result of the MC dropout model with 50% confidence interval

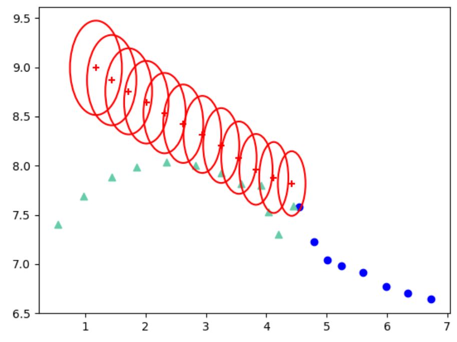

Figure 30: Result of the ensemble model with 50% confidence interval

The last example is in Figures 31 to 34. Here, we see a similar example as Figure 14, where someone stood still during the input series and started moving during the prediction interval. This example is interesting from an uncertainty prediction perspective, mainly due to its similarity to someone standing still at a crosswalk and deciding to start moving. The evidential model, Figure 31, clearly shows the benefit of an uncertainty prediction model over a standard neural network; although the predicted mean is almost stationary, the possibility of movement was incorporated in the confidence interval. Critically, this behavior was not directly caused by our assumptions when constructing the model. Instead, this is likely inferred due to the RNN layers of our model architecture. The adapted evidential model shows a similar result in Figure 32. The main difference we observe is that the width of the confidence interval changes from larger in the vertical direction to larger in the horizontal direction. Both the ensemble and MC dropout models, again, perform worse due to a constant width of the prediction interval over time.

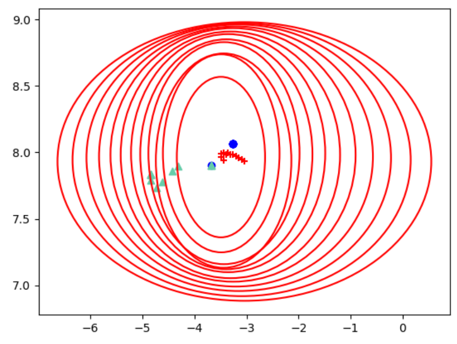

Figure 31: Result of Evidential model with 95% confidence interval

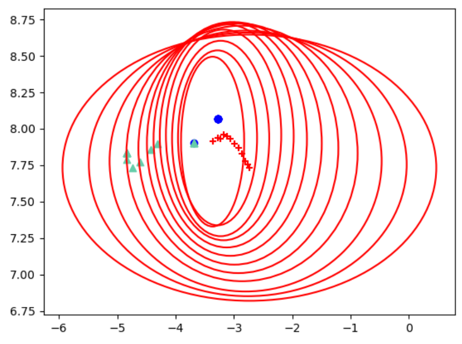

Figure 32: Result of the adapted evidential model with 95% confidence interval

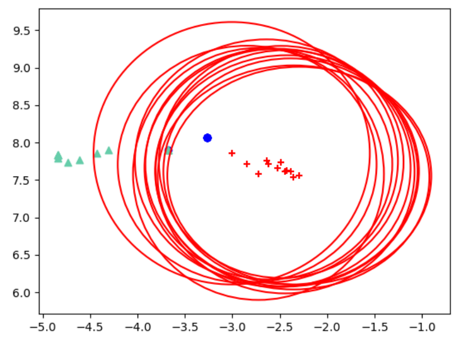

Figure 33: Result of the MC dropout model with 95% confidence interval

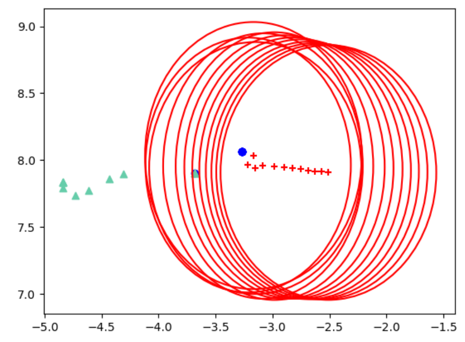

Figure 34: Result of the ensemble model with 95% confidence interval

5.2.2 Metrics

We now quantify the result using the metrics defined in the methodology section. First, we consider accuracy and computing time. Here, we look at the average displacement error (ADE) and final displacement error (FDE). Thus, we determine the average error over all time steps and the error for the final time step for the ADE and FDE, respectively.

Based on the accuracy, the evidential model is the most accurate for all metrics except for minimal FDE. We also observed that the max FDE is still around 4.4 to 4.6, which is quite large. Thus, in practice, further improvement of model accuracy is desired. Nevertheless, for uncertainty estimation models, this could be compensated with wider confidence estimates.

We now look at the training and prediction times. Here, we are primarily interested in the time required for predictions. Particularly, this thesis aims to eventually improve self-driving car algorithms, where fast computation times are required. The training and prediction time show behavior similar to that of the regression dataset. Particularly, the training time of the evidential and adapted evidential and MC dropout models are similar, whereas the ensemble takes ten times longer since ten models are trained. The prediction time is the fastest for the evidential and adapted evidential models, which predict 805 samples in around 0.2 seconds. The ensemble takes about ten times longer, and the dropout model takes around 170 times longer. This corresponds to the 10 and 100 forward passes required for the ensemble and MC dropout, respectively. Additionally, the MC dropout has some additional computational overhead to duplicate the data to create the predictions.

Table 4: Accuracy measures and computing time for ETH dataset

	ADE			FDE			Training time (s)	Prediction time (s)
	Mean	Min	Max	Mean	Min	Max		
Evidential Network	**0.522**	**0.057**	**1.841**	**0.971**	0.021	**4.399**	79.842	0.215
Adapted Evidential Network	0.628	0.068	1.987	1.120	**0.013**	4.688	78.908	0.190
MC Dropout	0.673	0.103	3.197	1.111	0.021	4.682	91.396	34.391
Deep ensemble	0.566	0.097	2.555	1.012	0.017	4.648	784.191	2.052

Note: Model with best accuracy measure in bold

Next, we examine the confidence interval score at different time steps in Table 4. First, looking at the 50% confidence interval, $p = 0.5$, we see that the evidential model is closest to the correct confidence interval. Nevertheless, we see that at the first time step more data points lie within the confidence interval than in later time steps. The adapted evidential model predicts a confidence interval that is not large enough. The MC dropout model performs well on average for $p = 0.5$, although a big drop-off occurs between the first and last timestep; we suspect the main reason for this is the almost constant confidence interval over time. For the ensemble model, we see something similar, which we also suspect is due to an almost constant confidence interval and not due to inaccurate predictions. For the 95% confidence interval, we see similar results as the 50% interval. Nevertheless, based on the Chebyshev bound, we see that for all models, more than 95% of the points lie within the interval.

Table 5: Confidence Interval Score at different time steps for the ETH dataset

p	Model	Mean	Time step						
			1	2	4	6	7	10	12
0.5*	Evidential Network	0.568	0.753	0.757	0.595	0.530	0.486	0.491	0.487
	Adapted Evidential Network	0.278	0.361	0.240	0.255	0.266	0.280	0.292	0.276
	MC Dropout	0.463	0.881	0.729	0.605	0.440	0.303	0.270	0.225
	Deep ensemble	0.436	0.840	0.794	0.573	0.374	0.282	0.226	0.185
0.95*	Evidential Network	0.957	0.998	0.994	0.988	0.975	0.970	0.902	0.884
	Adapted Evidential Network	0.860	0.954	0.929	0.862	0.829	0.827	0.830	0.826
	MC Dropout	0.889	1.000	0.964	0.958	0.925	0.892	0.810	0.696
	Deep ensemble	0.854	0.990	0.983	0.957	0.916	0.798	0.728	0.678
>0.95**	Evidential Network	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.996
	Adapted Evidential Network	0.999	1.000	1.000	1.000	1.000	1.000	0.998	0.993
	MC Dropout	0.990	1.000	1.000	0.998	0.989	0.985	0.983	0.975
	Deep ensemble	0.993	1.000	1.000	1.000	1.000	0.996	0.986	0.970

Note: * based on the confidence interval, ** based on Chebyshev bound.

We now consider the results on the HOTEL, ZARA01, and ZARA02 datasets, which we described in the data section. The main purpose of these datasets is to see how the results compare to each other. First, we present the result based on accuracy in Table 6. Now, we see that the evidential model is no longer best for all metrics. The ensemble model performs best for the HOTEL and ZARA02 datasets, whereas MC dropout and the adapted evidential method work best for ZARA01. Nevertheless, we see that the accuracy measures are relatively similar among all models. We suspect this is because all models are based on the same neural network architecture, which leads to similar predicted movement paths independent of the uncertainty estimation technique.

Table 6: Overview of accuracy measures for all datasets

	ETH		HOTEL		ZARA01		ZARA02	
	ADE	FDE	ADE	FDE	ADE	FDE	ADE	FDE
Evidential Network	**0.522**	**0.971**	0.297	0.437	0.637	1.219	0.339	0.674
Adapted Evidential Network	0.628	1.120	0.363	0.479	**0.621**	1.215	0.381	0.720
MC Dropout	0.673	1.111	0.340	0.453	0.733	**1.190**	0.555	0.873
Deep ensemble	0.566	1.012	**0.277**	**0.398**	0.870	1.401	**0.336**	**0.658**

We now present the result for the HOTEL dataset in Table 7. For the 50% confidence interval, we see that the evidential model performs best, but with too many points inside the confidence interval at the first time step. The adapted evidential is significantly too low and performs even worse than on the ETH dataset. Interestingly, this dataset also has fewer data points; see Table 2. Since we need sufficient data points for our regularization term to closely approximate the chance constraint, we suspect this could be the cause of the issue. The dropout and ensemble models contain too many data points for p=0.5. At p=0.95, the evidential model, MC dropout model, and ensemble model all seem to work well, whereas adapted evidential, again, performs the worst.

Table 7: Confidence interval score of the HOTEL dataset

Model	$p = 0.5$*			$p = 0.95$*			$p > 0.95$**		
	Mean	Time step		Mean	Time step		Mean	Time step	
		1	12		1	12		1	12
Evidential Network	0.531	0.627	0.542	0.947	0.961	0.941	0.978	1.000	0.973
Adapted Evidential Network	0.153	0.152	0.167	0.760	0.716	0.779	0.974	0.985	0.968
MC Dropout	0.718	0.814	0.657	0.958	0.998	0.936	0.986	1.000	0.971
Deep ensemble	0.739	0.826	0.657	0.955	0.983	0.926	0.983	1.000	0.968

Note: * based on the confidence interval, ** based on Chebyshev bound.

In Table 8, we show the results for the ZARA01 dataset. Based on the results, we observe that, on average, none of the models have enough points within the confidence interval for p=0.5 or p=0.95. Nevertheless, for all except for the adapted evidential model during the first time step, we do observe positive results. This leads us to believe that the main issue is in later time steps, as observed for the twelfth (and last) time step.

Table 8: Confidence interval score of the ZARA01 dataset

Model	$p = 0.5$*			$p = 0.95$*			$p > 0.95$**		
	Mean	Timestep		Mean	Timestep		Mean	Timestep	
		1	12		1	12		1	12
Evidential Network	0.447	0.674	0.452	0.872	0.990	0.790	0.989	0.999	0.971
Adapted Evidential Network	0.200	0.351	0.168	0.779	0.985	0.675	0.967	0.996	0.926
MC Dropout	0.397	0.566	0.177	0.895	0.984	0.744	0.989	1.000	0.976
Deep ensemble	0.424	0.685	0.235	0.895	1.000	0.739	0.992	1.000	0.978

Note: * based on the confidence interval, ** based on Chebyshev bound.

Lastly, we present the results for the ZARA02 dataset in Table 9. This dataset has the most number of observations. Surprisingly, the adapted evidential network works relatively well for this dataset, particularly for p =0.5. We suspect the large number of data points might have positively affected the results. For p=0.95, we observe that, on average, the CIS for all models is too low. Interestingly, the MC dropout and ensemble models at the first step seem to have a CIS of almost one but have a large drop-off at the last step, potentially again due to a relative constant uncertainty prediction interval.

Table 9: Confidence interval score of the ZARA02 datast

Model	$p = 0.5$*			$p = 0.95$*			$p > 0.95$**		
	Mean	Timestep		Mean	Timestep		Mean	Timestep	
		1	12		1	12		1	12
Evidential Network	0.611	0.756	0.557	0.884	0.986	0.810	0.967	1.000	0.926
Adapted Evidential Network	0.558	0.621	0.535	0.852	0.958	0.801	0.958	0.999	0.928
MC Dropout	0.360	0.644	0.226	0.903	0.997	0.773	0.992	1.000	0.971
Deep ensemble	0.687	0.996	0.493	0.924	1.000	0.805	0.992	1.000	0.973

Note: * based on the confidence interval, ** based on Chebyshev bound.

Generalizing the results of Tables 4 to 9, we see that based on the accuracy, there are no major differences between models. Additionally, we generally see that the evidential model obtains the best CIS. Next, for the dropout and ensemble models, we observe a significant drop-off in CIS between the first and last time steps, potentially caused by a relatively constant width of the confidence interval. Lastly, we observe a relation between the number of data points and the quality of results in CIS for the adapted evidential model.

6 Conclusion

In this thesis, we have considered four models to create uncertainty estimates based on confidence intervals. The first model we considered is an evidential neural network. The methodology of evidential neural networks is based on the assumption that the output can be modeled using a hierarchical distribution. Moreover, we minimize the negative log-likelihood during the optimization, including a regularization term. To construct the variance and confidence interval, we use a heuristic approach by assuming that our degrees of freedom are sufficiently large. The second model is also an evidential neural network where the main difference is the regularization term. In this thesis, we derive a new regularizer based on the definition of the confidence interval. Here, we note that for this regularization to hold, we need sufficient data with respect to the number of weights in the network and desired confidence level. We estimate the variance and confidence interval based on the t-distribution for this method. In addition, we also implemented the MC dropout model; this model approximates a Bayesian neural network by applying dropout during training and the predictions. Lastly, we implement an ensemble model by training ten separate neural networks and combining the predictions to construct a confidence interval.

These methods are tested on two types of data. First, we tested them on a regression dataset, where we needed to predict a non-linear trend with heteroskedastic variance. Then, we tested the methods on several datasets of time series data related to pedestrian movement path prediction. We use the ETH and UCY datasets, containing four scenarios, to test the output of the different methodologies.

6.1 Main findings

Summarizing the results of the regression data, we see that all models can model the non-linear trend. This is likely because of the neural network structure underlying all uncertainty estimation methods. Furthermore, the evidential and adapted evidential models work best for uncertainty estimation and heteroskedasticity modeling. The adapted evidential model is preferred for this dataset due to the smaller confidence intervals compared to the evidential model. The adapted evidential model potentially works well on this data due to a sufficient number of data points. The MC dropout and ensemble models perform worse, as they do not model the heteroskedasticity of the variance in the confidence interval.

The results of the pedestrian movement dataset show interesting results. We have seen that the evidential neural network performs well for almost all datasets. To the author's knowledge, this is the first time the evidential model was applied to pedestrian movement forecasting. As such, it seems like a significant improvement of the results based on the MC dropout model in this thesis and in the previous work of Nayak et al. (2022). Nevertheless, the evidential model still does not work perfectly for all scenarios. Namely, in some cases, insufficient data points lie within the confidence interval, whereas the interval seems excessively wide at other times. The results of the adapted evidential model are less good on three of the five datasets than the other models. The predicted confidence interval for almost all scenarios is not wide enough. Thus, this indicates that further research is necessary to improve this method. Interestingly, for this method, we do see a relation between the size of the dataset and the quality of the uncertainty prediction; this is one of our main recommendations for further research. The results of the dropout and ensemble models show similar problems. Notably, they have an almost constant width of the confidence interval over the prediction horizon. As such, the confidence interval is too wide for early time steps, whereas for later time steps, it is too narrow. We also find that the evidential and adapted evidential model outperforms the ensemble and Monte Carlo dropout models on computational time, which is essential in the application of uncertainty estimation in self-driving cars.

Based on our research, we can answer our research questions. Firstly, to answer the question: "*What uncertainty estimation method obtains the most accurate predictions?*", we find that all models produce reasonably accurate predictions, which is likely due to the characteristics of the neural network and RNN layers. Then, to answer the question: "*What uncertainty estimation method obtains the most reliable uncertainty estimates?*", we find that for the regression data, the adapted evidential model works best, and for the pedestrian forecasting, the evidential model works best most of time. Both models can model heteroskedasticity and learn uncertainty patterns for time series data without explicit modeling. Next, regarding our third subquestion: "*How do the results of the accuracy and uncertainty estimate change between datasets and use-cases?*", we find that the accuracy and uncertainty results do change between datasets. We observe that the adapted evidential model works better on datasets with more data points, and we see a difference in the uncertainty forecast quality between datasets. Finally, to answer our main research question:"*Which of the three considered methodologies is optimal for estimating the uncertainty of predictions generated by a neural network?*" we conclude that for the use-case of pedestrian movement prediction, the evidential model provides the best uncertainty prediction. This answer is based on the answers to the previous questions and on the fact that the evidential model leads to significant improvement in the prediction time.

The result of this thesis could have a positive societal impact. Firstly, the methods presented in this thesis incorporate the benefits of the neural network and improve the trustworthiness due to uncertainty estimates. Additionally, we observe that the evidential network shows improved results over the previously used MC dropout model for pedestrian movement prediction. Furthermore, the evidential models show increased computational speed during predictions. Therefore, we believe that evidential networks can be effectively applied to self-driving cars.

6.2 Limitations and further research

The results of this thesis rely on the correct construction of the confidence interval. In particular, the distribution of the critical values for the univariate and multivariate outputs need to be studied further. In addition, the evidential and adapted evidential network heavily rely on distributional assumptions. Particularly, we need to validate our choice to model the data using multiple univariate outputs with zero covariance. Furthermore, in this thesis, a constant regularization parameter equal to one was used. It is important to see how the results change for a different values. For the Monte Carlo dropout model, further analysis is needed on the value of the dropout parameter, since we only used a single value. For the ensemble model, we used only ten networks, whereas for the confidence interval are large number of networks is needed. However, increasing the number of networks could lead to longer training and prediction times.

We see several avenues for further research. Firstly, we recommend to study the evidential methodology further. In particular, we recommend to derive further insights of the regularization parameter. Specific to the adapted evidential model, it is interesting to validate our hypothesis that the model performs better with more datapoints. Additionally, we believe further theoretical results can be derived for the model if we simplify the neural network to a linear model. Secondly, we recommend to improve the construction of the confidence intervals, for this we see two approaches. The confidence intervals can be improved either through further theoretical derivations or based on approximate critical values using validation data. Thirdly, we recommend further comparisons of the evidential model against other benchmarks. Lastly, we recommend testing the methods on other applications to see to what extent the results in this research generalize.

References

Abdullah, A. A., Hassan, M. M., & Mustafa, Y. T. (2022). A review on bayesian deep learning in healthcare: Applications and challenges. *IEEE Access*, *10*, 36538–36562.

Amini, A., Schwarting, W., Soleimany, A., & Rus, D. (2020). Deep evidential regression. *Advances in Neural Information Processing Systems*, *33*, 14927–14937.

Amirian, J., Zhang, B., Castro, F. V., Baldelomar, J. J., Hayet, J.-B., & Pettre, J. (2020). Opentraj: Assessing prediction complexity in human trajectories datasets. *Asian Conference on Computer Vision (ACCV)*, (CONF).

Barber, D., & Bishop, C. M. (1998). Ensemble learning in bayesian neural networks. *Nato ASI Series F Computer and Systems Sciences*, *168*, 215–238.

Bengs, V., Hüllermeier, E., & Waegeman, W. (2023). On second-order scoring rules for epistemic uncertainty quantification. *International Conference on Machine Learning*, 2078–2091.

Blundell, C., Cornebise, J., Kavukcuoglu, K., & Wierstra, D. (2015). Weight uncertainty in neural network. *Proceedings of the 32nd International Conference on Machine Learning*, 1613–1622.

Calafiore, G., & Campi, M. C. (2005). Uncertain convex programs: Randomized solutions and confidence levels. *Mathematical Programming*, *102*, 25–46.

Calzone, O. (2022, April). An intuitive explanation of lstm. https://medium.com/@ottaviocalzone/an-intuitive-explanation-of-lstm-a035eb6ab42c

Dewolf, N., Baets, B. D., & Waegeman, W. (2023). Valid prediction intervals for regression problems. *Artificial Intelligence Review*, *56*(1), 577–613.

Dubey, K. A., J Reddi, S., Williamson, S. A., Poczos, B., Smola, A. J., & Xing, E. P. (2016). Variance reduction in stochastic gradient langevin dynamics. *Advances in Neural Information Processing Systems*, *29*.

Gal, Y., & Ghahramani, Z. (2015). Bayesian convolutional neural networks with bernoulli approximate variational inference. *arXiv preprint arXiv:1506.02158*.

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

Gal, Y., Hron, J., & Kendall, A. (2017). Concrete dropout. *Advances in Neural Information Processing Systems*, *30*.

Gawlikowski, J., Tassi, C. R. N., Ali, M., Lee, J., Humt, M., Feng, J., Kruspe, A., Triebel, R., Jung, P., Roscher, R., et al. (2023). A survey of uncertainty in deep neural networks. *Artificial Intelligence Review*, 1–77.

Ghosh, T. (2018). What does it mean by 1d convolutional neural network? - quora. https://www.quora.com/What-does-it-mean-by-1D-convolutional-neural-network

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Greenberg, E. (2012). *Introduction to bayesian econometrics*. Cambridge University Press.

Herron, E. J., Young, S. R., & Potok, T. E. (2020). Ensembles of networks produced from neural architecture search. *International Conference on High Performance Computing*, 223–234.

Hinton, G. E., & Van Camp, D. (1993). Keeping the neural networks simple by minimizing the description length of the weights. *Proceedings of the sixth annual conference on Computational learning theory*, 5–13.

Hron, J., Matthews, A., & Ghahramani, Z. (2018). Variational bayesian dropout: Pitfalls and fixes. *International Conference on Machine Learning*, 2019–2028.

Johnson, R., & Wichern, D. (2014). Chapter 5.4. In *Applied multivariate statistical analysis* (pp. 220–231). Pearson.

Jordan, M. (2009). Chapter 9. the exponential family: Conjugate priors.

Jospin, L. V., Laga, H., Boussaid, F., Buntine, W., & Bennamoun, M. (2022). Hands-on bayesian neural networks—a tutorial for deep learning users. *IEEE Computational Intelligence Magazine*, *17*(2), 29–48.

Lakshminarayanan, B., Pritzel, A., & Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in Neural Information Processing Systems*, *30*.

Lerner, A., Chrysanthou, Y., & Lischinski, D. (2007). Crowds by example. *Computer Graphics Forum*, *26*(3), 655–664.

Leutbecher, M., & Palmer, T. N. (2008). Ensemble forecasting. *Journal of Computational Physics*, *227*(7), 3515–3539.

Li, Y., & Gal, Y. (2017). Dropout inference in bayesian neural networks with alpha-divergences. *International Conference on Machine Learning*, 2052–2061.

Lütjens, B., Everett, M., & How, J. P. (2019). Safe reinforcement learning with model uncertainty estimates. *2019 International Conference on Robotics and Automation (ICRA)*, 8662–8668.

Malinin, A. (2019). *Uncertainty estimation in deep learning with application to spoken language assessment* [Doctoral dissertation].

Malinin, A., Chervontsev, S., Provilkov, I., & Gales, M. (2020). Regression prior networks. *arXiv preprint arXiv:2006.11590*.

Malinin, A., & Gales, M. (2018). Predictive uncertainty estimation via prior networks. *Advances in Neural Information Processing Systems*, *31*.

Malinin, A., & Gales, M. (2019). Reverse kl-divergence training of prior networks: Improved uncertainty and adversarial robustness. *Advances in Neural Information Processing Systems*, *32*.

Meinert, N., Gawlikowski, J., & Lavin, A. (2023). The unreasonable effectiveness of deep evidential regression. *Proceedings of the AAAI Conference on Artificial Intelligence*, *37*(8), 9134–9142.

Meinert, N., & Lavin, A. (2021). Multivariate deep evidential regression. *arXiv preprint arXiv:2104.06135*.

Nayak, A., Eskandarian, A., & Doerzaph, Z. (2022). Uncertainty estimation of pedestrian future trajectory using bayesian approximation. *IEEE Open Journal of Intelligent Transportation Systems*, *3*, 617–630.

Neal, R. (1992). Bayesian learning via stochastic dynamics. *Advances in Neural Information Processing Systems*, *5*.

Neal, R. M., et al. (2011). MCMC using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, *2*(11), 2.

Oh, D., & Shin, B. (2022). Improving evidential deep learning via multi-task learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, *36*(7), 7895–7903.

Paat, H., Lian, Q., Yao, W., & Zhang, T. (2023). Medl-u: Uncertainty-aware 3d automatic annotator based on evidential deep learning. *arXiv preprint arXiv:2309.09599*.

Parker, W. S. (2013). Ensemble modeling, uncertainty and robust predictions. *Wiley interdisciplinary reviews: Climate change*, *4*(3), 213–223.

Pellegrini, S., Ess, A., Schindler, K., & Van Gool, L. (2009). You'll never walk alone: Modeling social behavior for multi-target tracking. *2009 IEEE 12th International Conference on Computer Vision*, 261–268.

Renda, A., Barsacchi, M., Bechini, A., & Marcelloni, F. (2019). Comparing ensemble strategies for deep learning: An application to facial expression recognition. *Expert Systems with Applications*, *136*, 1–11.

Sensoy, M., Kaplan, L., & Kandemir, M. (2018). Evidential deep learning to quantify classification uncertainty. *Advances in Neural Information Processing Systems*, *31*.

Ulmer, D., Hardmeier, C., & Frellsen, J. (2021). Prior and posterior networks: A survey on evidential deep learning methods for uncertainty estimation. *arXiv preprint arXiv:2110.03051*.

A Derivation of the confidence interval

we want to find a lower bound l and upper bound u such that

$$\mathbb{P}(l < y < u) = p, \tag{55}$$

for y distributed according to a location scale t distribution with 2α degrees of freedom, location γ and scale σ^2. then we rewerite to find

$$\mathbb{P}(\frac{l-\gamma}{\sigma} < \frac{y-\gamma}{\sigma} < \frac{u-\gamma}{\sigma}) = p, \tag{56}$$

where $\frac{y-\gamma}{\sigma}$ is distributed according to a student t-distribution with 2α degrees of freedom. Using the critical values of the student t distribution we find

$$\frac{l-\gamma}{\sigma} = -t^{2\alpha}_{(1-p)/2} \text{ and } \frac{u-\gamma}{\sigma} = t^{2\alpha}_{(1-p)/2}. \tag{57}$$

thus,

$$l = \gamma - \sigma t^{2\alpha}_{(1-p)/2} \text{ and } u = \gamma + \sigma t^{2\alpha}_{(1-p)/2}. \tag{58}$$

and we find the confidence interval

$$[\gamma - \sigma t^{2\alpha}_{(1-p)/2}, \gamma + \sigma t^{2\alpha}_{(1-p)/2}]. \tag{59}$$

Unfortunately we don't know the parameters γ and σ and thus we replace them by the estimated values

$$[\hat{\gamma} - s\, t^{2\alpha}_{(1-p)/2}, \hat{\gamma} + s\, t^{2\alpha}_{(1-p)/2}]. \tag{60}$$

However, since we don't know the distribution of the estimated parameters $\hat{\gamma}$, s^2, we know that the critical value are likely no longer appropriate, but it is not clear what critical value we should use instead.

B Overview Code

The code can be found on:
https://github.com/SteinDijkstra/ForecastingPedestrianUncertainty
Structure codebase:

- Code:
  - main.py: File which executes the code programmed in the other folders
  - /data_preparation: Contains the code to generate the regression data and to preprocess the pedestrian dataset
  - /models: Contains the implementation of the evidential, adapted evidential, MC dropout and ensemble models

  - /experiment: Contains the code that organizes the experiments used in the thesis, also contains a file that helps analyze the results
  - /utility: Utility methods related to the config file

- Archive:
  - /data:
  - /ETH_data: The ETH Data retrieved from Amirian, J., Zhang, B., Castro, F. V., Baldelomar, J. J., Hayet, J.-B., & Pettre, J. (2020). Opentraj
  - /UCY_data: The UCY Data retrieved from Amirian, J., Zhang, B., Castro, F. V., Baldelomar, J. J., Hayet, J.-B., & Pettre, J. (2020). Opentraj
  - /Preprocessed: folder with processed regression, ETH and UCY data according to the data section in the thesis

- /results: folder with different runs with stored results
- /figures: folder with the figures presented in the thesis
- Other:
  - notebooks: contains jupyter notebook to create the tables and figures in the thesis
  - config: contains the config file that changes the architecture of the network, structure needs to be specified in utility/config.py