Internship Report: Uncertainty in Computer Vision

Domantas Giržadas, s1008829
Radboud University
TNO - IVS Department

1 INTRODUCTION

Uncertainty is one of the key components of human reasoning. We modulate our behaviour in different situations, based on our confidence in the knowledge we posses. Some traits that have enabled humans to thrive in an incredibly complex environment (such as curiosity), lie on a foundation of apt estimation of confidence and uncertainty [21].

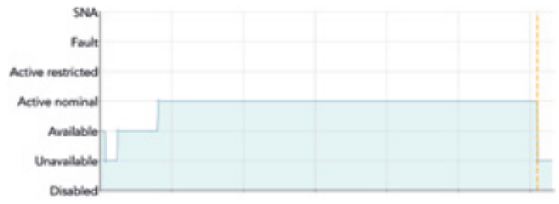
As machine learning-based systems are growing more complex and take up tasks that have a real impact on (or require interaction with) humans, it is becoming clear that in order to perform these tasks appropriately and further their capabilities in the real world, intelligent systems must also posses the ability to estimate and express uncertainty [16, 22]. An insightful case in point accident has been featured in a report by the Dutch Safety Board, where a Tesla car in autopilot mode crashed into a pole after driving straight through a roundabout [1]. The driver assistance system estimated the state of the vehicle to be "nominal" right up to the moment of the crash and showed no indication of uncertainty about this estimation (see panel (b) of Figure 1).

If the system had been capable of appropriate expression of uncertainty, the driver of the vehicle could have been alerted and asked to take over control. One might say that this is only a matter of improving the accuracy of the state estimation model. However, as such systems are meant to function in the real world, there are countless possible situations where the system might fail and it is impossible to validate them all for accurate estimation. Therefore, expressing uncertainty is a crucial requirement for the development and real-world deployment of safety-critical systems [16], such as autonomous vehicles or collaborative robots.

Despite its importance, explicit expression of uncertainty still lacks a general consensus about the best approaches for specific applications. This seems to baffle machine learning practitioners, making it difficult for uncertainty estimation to get out of the “future developments” category. An approachable and practical overview of the key concepts related to this topic shall raise awareness about the importance of uncertainty estimation, while also building a basic foundation of knowledge for anyone interested in machine learning. Providing such an overview is one of the key goals of this project.

In this report, two major sections will cover the findings, discovered via the two core procedures: theoretical literature review and practical experiments. The theoretical findings will be covered in Section 2. This section will summarise essential knowledge about the concept of uncertainty in the context of machine learning. This also includes overviews about uncertainty estimation methods and assessing the quality of these estimations. Meanwhile, Section 3 will cover implemented practical uncertainty estimation experiments. Namely, the approaches and implementational details will be discussed in Section 3.1 while the obtained results will be presented and discussed in Sections 3.2 and 3.3 respectively.

(a) The Tesla Model S after the accident



(b) Autopilot system state log (time point of the crash marked by the dashed line)

Figure 1: Tesla car in autopilot mode drives straight through a roundabout and crashes.
Source: Adapted from a report by the Dutch Safety Board [1].

2 THEORETICAL FINDINGS

The first major part of this project consists of theoretical knowledge, gathered via literature review and summarisation. The following subsections will cover the main theoretical findings and key concepts, which have provided a foundation for the practical experiments and analyses of Section 3.

2.1 Uncertainty in Machine Learning

When it comes to uncertainty in the context of computational modelling, it is commonly classified into two groups: ***Aleatoric*** and ***Epistemic*** uncertainty. The two types capture different aspects of model confidence and doubt.

2.1.1 Aleatoric uncertainty.

Firstly, *aleatoric* uncertainty expresses the amount of intrinsic randomness and unpredictability in the input data. It can stem from naturally overlapping classes (an example can be seen in Figure 2) or noise, caused by the sensors that collect the input data (such as image grain in pictures). As this type of uncertainty is intrinsically part of the input, it is impossible to reduce it by tuning or modifying the computational model (i.e., using more training data has no effect). Nonetheless, estimating this data unpredictability is very beneficial for any system that is meant to interact with the real world, which is riddled with ambiguities and complex situations. Having a capability of evaluating the intrinsic unpredictability of a situation allows a system to make more appropriate decisions in complicated scenarios.

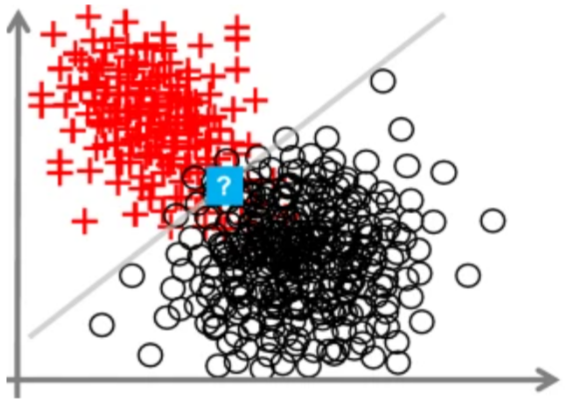
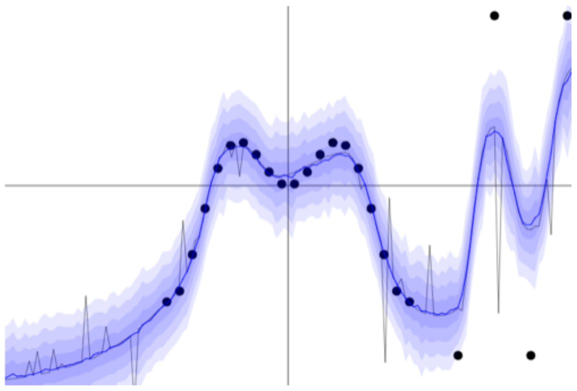


Figure 2: Example of overlapping classes
Source: Adapted from [12].

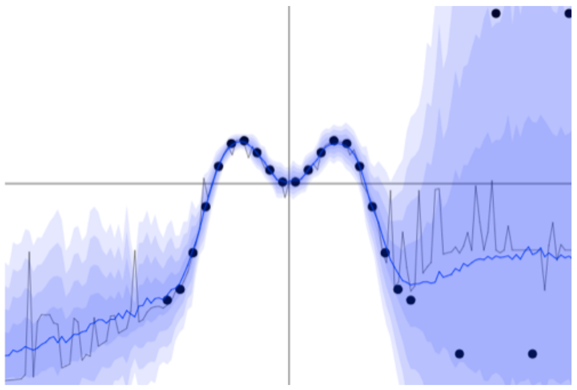
There are two types of approaches to modelling aleatoric uncertainty - *homoscedastic* and *heteroscedastic*. Homoscedastic modelling is input-independent, all of the inputs are assumed to contain the same amount of randomness (unpredictability) within the task domain (see panel (a) of Figure 3). While easy to implement and reasonably effective for capturing static sensor noise, such a strong assumption causes problems when tackling more complicated tasks, where different input values (or regions) express different amounts of randomness and ambiguity. In such cases, capturing the relationship between the input and its respective amount of uncertainty is necessary. This input-dependent modelling approach is referred to as heteroscedastic uncertainty modelling (see panel (b) of Figure 3).

2.1.2 *Epistemic Uncertainty.*

Secondly, there is *epistemic* uncertainty, which covers the other source of uncertainty - the computational model itself. Epistemic uncertainty describes the amount of possible variability in the parameters of the model. In other words, it evaluates how uncertain the model is about the correctness of its own parameters. This type of uncertainty stems from the lack of input space coverage during training. It can be caused by sparse data or underfitting during the training phase. Theoretically, this type of uncertainty can be reduced to zero if the model perfectly captures the real behaviour of the problem domain. However, perfectly capturing the behaviour of any real-world system is usually extremely computationally costly or even practically impossible. Therefore, expressing epistemic uncertainty can provide the model with very useful insight about its own lack of knowledge.



(a) Homoscedastic approach



(b) Heteroscedastic approach

Figure 3: Visualisation of different uncertainty modelling approaches.
Source: Adapted from https://github.com/yaringal/HeteroscedasticDropoutUncertainty.

2.1.3 *Expression.*

Modelling either type of uncertainty requires insight about the conceptual meaning of the underlying predictive model's parameters. There are various methods that express the two types differently well. However, generally, in order to estimate uncertainty, a distribution of predictions is required. This distribution can be represented in a wide range of ways, as a continuous distribution via parameter estimation or a discrete distribution via sample collection. Nonetheless, the following rule of thumb about expression of different types of uncertainty applies generally:

Aleatoric uncertainty is the variability within predictions, while epistemic uncertainty is the variability between predictions.

For instance, if multiple predictions of the model have low spread by themselves (estimating a high score for one class and low for the others in classification or having a low standard deviation estimate in regression), but do not agree between each other (predicting different classes in classification or different mean values in regression), then the model is expressing low aleatoric and high epistemic uncertainty. Meanwhile, if the predictions are all similar, but each one estimates a high variability (similar scores for different classes or high standard deviation estimations), then we have high aleatoric and low epistemic uncertainty.

2.2 Uncertainty Calibration

Once an estimation of uncertainty is made, how do we know it is an appropriate estimation? This is the main challenge of uncertainty estimation, especially in the context of machine learning-based systems. Taking a classification task as an example, most commonly-used models rely on clear-cut decision boundaries that divide the input space into areas that belong to specific classes (For example - see Figure 4). However, such input space division often fails with out-of-distribution (OOD) sample classification, as these samples usually lie in a location of the input space that is far from any decision boundary (for example, the location marked with "?" in Figure 4), resulting in an inappropriately confident classification. Therefore, uncertainty estimation methods must be evaluated with regard to calibration. This kind of evaluation is usually made with one core assumption - **the estimated confidence of the model has to match the accuracy of the model**[1]. This means that, for example, half of the model's predictions that have an estimated confidence of 50% should be correct, which is very intuitive.

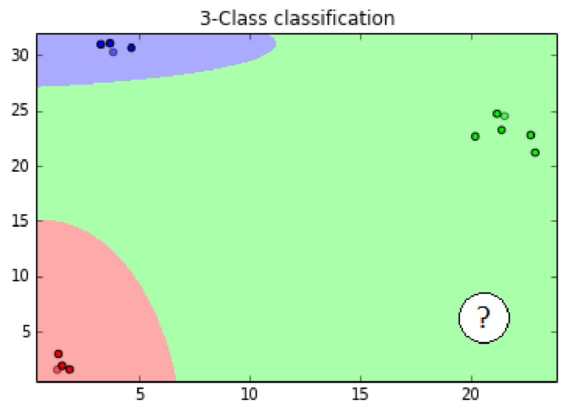


Figure 4: Example of classification decision boundary problem
Source: Adapted from https://tinystruggles.com/2014/03/24/classification-with-scikit-learn.html.

The "Confidence = Accuracy" target also gives us means of evaluating uncertainty calibration both qualitatively and quantitatively. Firstly, a very direct visual analysis can be made by plotting the estimated confidence against the accuracy of the predictions - such a graph is called a ***reliability diagram*** (see Figure 5). In this graph, the target is a diagonal line, which represents a model that perfectly matches its confidence estimations with the accuracy of its

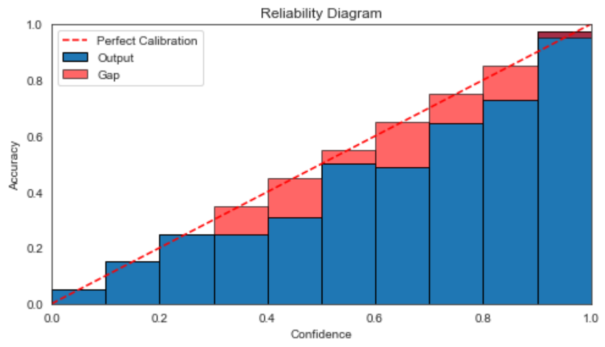


Figure 5: Example of a reliability diagram
Source: Adapted from https://medium.com/@danyal.wainstein1/managing-uncertainty-part-1-diagnosing-miscalibration-ffb21db583f0.

predictions. Analysing a model via this plot gives us insight about the model's over(/under)confidence in specific certainty ranges. In addition, a few quantitative metrics stem from the use of a reliability diagram. Models can be directly compared with respect to the average or maximum deviation from the diagonal line (***Expected Calibration Error (ECE)*** and ***Maximum Calibration Error (MCE)*** respectively) [7]. One can also evaluate the absolute area (sum of deviations) between the diagonal line and the model's resulting graph (known as the ***Area Under the Calibration Error Curve (AUCE)*** metric [9]).

2.3 Uncertainty Estimation Methods [2].

As mentioned before, there is a wide variety of different approaches for uncertainty estimation. The choice of a specific approach depends on the task at hand and the architecture of the predictive model. Even looking at the tasks in the autonomous driving domain, we find numerous different sub-tasks and problem sub-domains, each demanding specific model architectures and input/output formats [2]. Nevertheless, generally, most of uncertainty estimation methods belong to one of four main categories [6]:

- **Bayesian Methods**
- **Ensemble Methods**
- **Test-Time Augmentation Methods**
- **Single Network Deterministic Methods**

Each of the categories have their advantages, while also posing unique challenges, making different approaches more applicable in different scenarios.

2.3.1 Bayesian Methods.

The most direct and explicit approach of extracting uncertainties from a predictive model is converting its deterministic parameters into probability distributions. Bayesian Neural Networks estimate the posterior distribution of model's parameters $p(\theta|x,y)$, based on the provided inputs x, target outputs y and an assumed prior

[1]In this report, $confidence$ is equal to $1-uncertainty$, or $100\%-uncertainty(\%)$

[2]The contents of this section are inspired by and based on a survey by J. Gawlikowski et al. [6]

parameter distribution $p(\theta)$, by iteratively applying Bayes theorem:

$$p(\theta|x,y) = \frac{p(y|x,\theta)p(\theta)}{p(y|x)} \propto p(y|x,\theta)p(\theta) \quad (1)$$

However, performing inference in a Bayesian framework involves marginalisation of the likelihood $p(y|x,\theta)$:

$$p(y^*|x^*,x,y) = \int p(y^*|x^*,\theta)p(\theta|x,y)d\theta \quad (2)$$

This integral is computationally intractable in most realistic applications. Therefore, approximation techniques are applied (e.g. variational inference, Monte Carlo-based sampling methods - for details see [6]), which make the computation possible. Consequently, despite the strong theoretical foundation and high potential of Bayesian methods, implementing them requires a strong grasp of Bayesian statistics and take more programming effort due to a rather complicated inference procedure. In addition, Bayesian networks require sampling during inference, which makes these approaches computationally costly.

2.3.2 *Ensemble and Test Time Augmentation Methods.*

As mentioned before, estimating uncertainty requires a distribution of predictions. Both ensemble methods and test time augmentation-based methods tackle this task quite directly - by generating a set of different predictions on a single input.

Ensemble methods rely on using a set of unique prediction models, which generate a discrete distribution of predictions. The uniqueness of the models is the main source of power of ensemble methods. It allows the ensemble to evaluate multiple modes (local optima of the loss landscape) and allows for an impressive coverage of the loss landscape in general [4]. Therefore, there is a number of ways of making sure the models are indeed unique:

- **Random Initialization** - Initialising the parameters of the predictive model randomly. The non-linearity of the loss landscape causes convergence to different optima, when starting from different locations in the landscape.
- **Data Shuffling** - Providing the data samples in a different order throughout training gives different results.
- **Bagging and Boosting** - Modifying / resampling the training set for different models affects the training procedure and results in finding different parameter optima.
- **Data Augmentation** - Randomly augmenting the input data during training modifies the training data points and results in a variety of different models.
- **Different Architectures** - Different model architectures have different loss landscapes. Having an ensemble of models with different architectures guarantees the coverage of different optima.

While very powerful at covering the loss landscape, ensemble methods are computationally expensive. The system has to store all the elements of the ensemble and all their individual predictions in memory, as well as perform multiple inference steps before being able to form a prediction. Consequently, there have been numerous advances in making these models less computationally demanding, such as ensemble pruning [8] (removing redundant parts of the ensemble), knowledge distillation [10] (capturing the behaviour of the ensemble in a single network), snapshot ensembling [11] (saving model parameters at several local minima instead of saving a whole separate model) or weight averaging [13] (combining the weights of different models into one).

Meanwhile, test time augmentation methods also generate a range of different predictions. However, instead of having different prediction models, this is done by augmenting the input data during inference. Slightly different inputs result in different model predictions, forming a distribution of predictions, which allows for uncertainty estimation. Augmenting the input does not guarantee good coverage of the loss landscape, like an ensemble of different models does, but this approach does cut down on the computational demands of inference, given only one model needs to be stored in memory.

2.3.3 *Single Network Deterministic Methods.*

Most of the methods above suffer from high computational demands, each requiring a number of inference iterations before forming an uncertainty estimate. This problem is tackled directly by the use of single network-based deterministic methods. These approaches aim to estimate the uncertainty of a prediction with a single inference step. There are two main ways of performing such estimation - **externally** or **internally**.

External methods estimate uncertainty by using a system, separate from the prediction model. For example, by training a separate network that performs the estimation [18] or relating the uncertainty to the distance between the test input and data points that were seen during the training phase [20].

Internal methods mostly aim to deterministically predict the parameters of the prediction distribution, practically estimating the underlying continuous distribution that generates samples, equivalent to the predictions of sample-based methods. Internal methods are often trained based on distribution divergence metrics, which means they require an implementation of a custom loss function and can not be applied on existing pre-trained architectures.

While having a strong advantage with low computational costs, single network methods also pose some challenges. Mainly, the reliability of such approaches is rather questionable, as the estimation of uncertainty heavily depends on the parameters of the model, which technically come with their own epistemic uncertainties, making rigid validation of such systems quite difficult.

3 PRACTICAL ANALYSIS

The second part of this project was focussed on a practical analysis of a selected set of uncertainty estimation methods. The aim of this analysis was to evaluate different types of models in terms of uncertainty calibration. The details, regarding the design, implementation and results of the practical experiments will be discussed in the following sections.

3.1 Methods

In order to evaluate uncertainty calibration of specific machine learning-based models, they were trained to perform an image classification task on the CIFAR-10 dataset [14]. This dataset consists of 60 000 small (32x32) colour images of objects from 10 classes: “airplane”, “automobile”, “bird”, “cat”, “deer”, “dog”, “frog”, “horse”, “ship” and “truck”. A basic convolutional neural network (LeNet [15]) has been used as a foundation of all prediction models.

For this uncertainty calibration analysis, three types of models have been chosen: **Deep Ensemble**, **Evidential Neural Network** and **Equidistant Hyperspherical Prototype Network**. The reasons for choosing these specific methods and their detailed descriptions are provided in the following sections.

3.1.1 *Deep Ensemble.*

An ensemble uncertainty estimation method, as described in Section 2.3.2, has been chosen on a basis of its strong loss landscape coverage and simplicity of implementation. For this approach, the number of ensemble members has been chosen to be 10. This decision has an effect on the overall performance of the model, its computational demands, and, likely, the calibration of the model. However, an appropriate value for this parameter is dependent on the complexity of the task and computational resource limitations. Therefore, a value of 10 has been chosen, based on the scope of this project and resource limitations.

Each member of the ensemble has been trained by using cross-entropy loss. A prediction of a single member is considered to be the softmax-scaled output of a network. While there are numerous ways of combining the predictions of different ensemble members into one [5], due to the scope of this project, unweighted averaging approach (the final prediction is the mean of all predictions) has been used for this task.

3.1.2 *Evidential Neural Network.*

The second approach is an evidential deep learning-based uncertainty estimation model [19]. The core concept of this approach is predicting the parameters of a distribution of predictions, making this model an internal single network deterministic method, as described in Section 2.3.3. As a classification task has been chosen for this project, the α (alpha) parameter of a Dirichlet distribution was to be predicted.

The implementation of the Evidential Neural Network (ENN) has been based on the author's implementation [3] and converted to use the PyTorch library instead of TensorFlow. For this method, the basic LeNet architecture has been slightly modified to match the original model. Namely, a ReLu activation function has been applied on the final layer of the network and the output values of the network have been increased by 1.0, to facilitate the "$\alpha = evidence + 1$" computation step.

A custom loss function, provided in the original implementation as "`loss_EDL`" function, has been used during the training of this model. Then, equivalently to the original implementation, the probability (confidence) estimations were computed by dividing the estimated α vector by its sum.

3.1.3 *Equidistant Hyperspherical Prototype Network.* [4]

The third approach is a hypersperical prototype-based model [3, 17]. Here, a neural network learns to place data points on a hypersphere in a way that makes them end up as close as possible to their respective class prototypes, that are placed equidistantly on the hyperspere and are fixed throughout the whole process (see Figure 6 for an example). After training, the model is capable of placing new inputs on the hypersphere, which allows for uncertainty estimation via cosine similarity to the fixed class prototypes. This method belongs between internal and external single network deterministic approaches. Technically, this approach does not directly estimate prediction uncertainty, but rather the similarity between a sample and a class prototype. This estimation of similarity makes out-of-distribution (OOD) sample detection rather straight-forward - it can be done by applying a minimum similarity threshold (based on training data) to determine how dissimilar to the class prototypes samples can be, before being considered as OOD. However, this model only estimates the aleatoric uncertainty component (data uncertainty) and contains no expression of epistemic uncertainty.

The implementation of the Equidistant Hyperspherical Prototype Network has been based on an implementation provided by the internship supervisor Jeroen Manders.

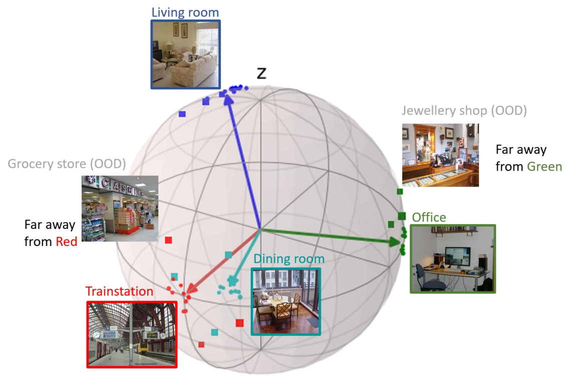


Figure 6: An example of a hypersphere and equidistant class prototypes
Source: https://gertjanburghouts.github.io/model/uncertainty/2021/10/30/uncertainty.html, [3]

The results of these three methods were compared to a baseline approach - using **softmax-scaled outputs of a neural network** trained with cross-entropy loss, as an estimation of prediction confidence. 10 copies of each model have been trained with random weight initialisation, in order to capture a 95% confidence range of the obtained results.

The same optimisation parameters (see Appendix A) have been used for all models in order to reduce the amount of latent variables.

Implementation examples (with more basic training procedures) of all models can be found in respective demonstration notebooks at https://github.com/dgirzadas/Uncertainty-in-CV.

After the models have been trained, a comparative analysis has been performed. The models were compared qualitatively by examining the prediction distribution results and reliability diagrams of each model, as well as quantitatively, via the metrics discussed in Section 2.2 - ECE, MCE and AUCE.

3.2 Results

After training each model for 200 epochs, the models have reached training accuracies noted down in Table 1 and visualised in panel (a) of Figure 7.

It is observable that all of the models have reached high accuracy levels for the training set. None of the model copies deviate strongly

[3]https://muratsensoy.github.io/uncertainty.html
[4]The Equidistant Hyperspherical Prototype Network is refered to as "The Prototype Network" later in the report.

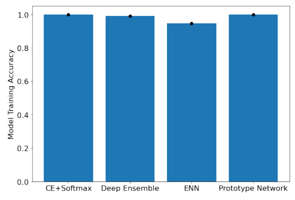
Model	Training accuracy
CE+Softmax Baseline	**0.999 ± 0.0003**
Deep Ensemble	0.992 ± 0.0066
ENN	0.947 ± 0.0064
Prototype Network	0.998 ± 0.0025

Table 1: Model training accuracies

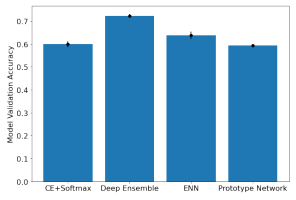
Model	Validation accuracy
CE+Softmax Baseline	0.598 ± 0.0138
Deep Ensemble	**0.722 ± 0.0073**
ENN	0.637 ± 0.0160
Prototype Network	0.592 ± 0.0061

Table 2: Model validation accuracies

from the mean result - 95% of all results lie less than 0.007 accuracy points (0.7%) away from the mean. Nonetheless, the baseline ("CE+Softmax") model seems to have the best fit on the training data, at 0.999 ± 0.0003 (99.9% ± 0.03%) accuracy. The Evidential Neural Network model seems to have the poorest performance on the training set, at 0.947 ± 0.0064 (94.7% ± 0.64%) accuracy.



(a) Model training accuracies



(b) Model validation accuracies

Figure 7: Accuracy Results of analysed models

Meanwhile, the validation results are shown in Table 2 and panel (b) of Figure 7. Here, a clearer distinction between different models, as well as a bit larger deviations between model copies can be observed. The Deep Ensemble model shows the best validation performance, with 0.722±0.0073 (72.2%±0.73%) accuracy. The Equidistant Hyperspherical Prototype Network performed the worst on the validation set, classifying only 0.592 ± 0.0061 (59.2% ± 0.61%) of the images correctly, which is slightly worse than the simple baseline method.

A reliability diagram comparison is shown in Figure 8. In this figure, it is apparent that the accuracy of the Deep Ensemble model predictions tends to be higher than the estimated confidence value, making this model underconfident. Meanwhile, all other models, including the baseline, tend to give overconfident predictions. The coverage of different prediction confidence ranges is shown in Figure 9. It is observable that the vast majority of both the baseline and the Prototype Network model predictions had very high (between 0.9 and 1.0) estimated confidence values. Predictions from the ENN model show a similar pattern, where higher confidence estimations are made more frequently. However, compared to the previously mentioned models, the ENN covers a wider range of confidence values. The Deep Ensemble model seems to show the best confidence range coverage out of all tested models. Here, most confidence range bins contain a very similar amount of samples.

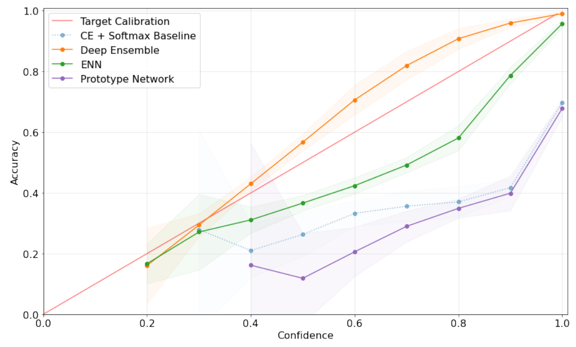


Figure 8: Reliability diagrams of analysed methods

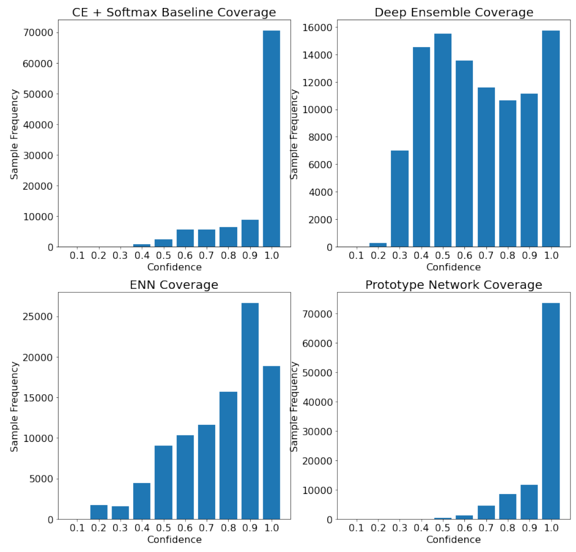


Figure 9: Confidence range coverage of analysed models

The quantitative metric results, listed in Table 3 and visualised in Figure 10, explicate the difference between the tested models. The Deep Ensemble network shows the best results with respect to all three metrics. Meanwhile, the Prototype Network gave the poorest quantitative metric results, closely overlapping with the performance of the baseline model.

Model \ Metric	ECE	MCE	AUCE
CE+Softmax Baseline	0.296 ± 0.034	0.483 ± 0.028	2.369 ± 0.271
Deep Ensemble	**0.064 ± 0.019**	**0.122 ± 0.045**	**0.575 ± 0.169**
ENN	0.119 ± 0.014	0.224 ± 0.030	1.069 ± 0.129
Prototype Network	0.388 ± 0.065	0.501 ± 0.057	2.718 ± 0.454

Table 3: Quantitative results of analysed methods

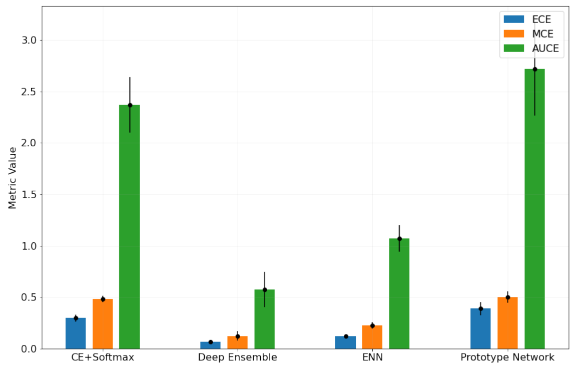


Figure 10: Quantitative results of analysed methods

3.3 Discussion

Most of the results discussed in the previous section were quite expectable. The overconfidence of softmax-scaled network outputs (used as a baseline in this project) is a well-known phenomenon in the field, while the lack of calibration in the Prototype Network predictions might be explained by the fact that this method estimates a distance between a sample and a class prototype, rather than prediction uncertainty. Also, while too high, the confidence estimations of this model follow a linear-like trend in the reliability plot (see Figure 8), which suggests that an additional post hoc calibration step could improve the calibration of this model significantly.

It is important to reiterate that all of the models were trained by using the same optimisation parameters (same optimiser type, same learning rate schedule, number of training epochs, etc.). This means that it is possible that while a complex model did not get a chance to fully converge within the restricted number of epochs, a simpler one, such as the baseline, could have gone past the optimal point and started to overfit on the training data. The difference between model training and validation accuracies (Table 1 and 2) suggests that all of the models were actually overfit on the training data, which is likely to be due to a quite simple CIFAR-10 classification task and a rather long training period (200 epochs). Follow-up research related to this project could investigate the optimal training routines for such models, re-evaluate their uncertainty calibration when they are trained optimally, as well as evaluate them on a more complicated task.

All quantitative findings of this project point towards the Deep Ensemble method being the best-calibrated uncertainty estimation method out of the evaluated ones. Such a finding is not surprising, considering the great coverage of the loss landscape by ensemble methods (as mentioned in Section 2.3.2). However, these well-calibrated uncertainty estimates come with rather high computational costs, inherent in ensemble methods. These costs hinder the possibilities of applying this method in resource and time-restricted environments, such as driver assistance systems in vehicles. However, as computational cost evaluation was outside of the scope for this project, it would prove beneficial to perform this analysis explicitly in future research, while also looking into and evaluating possible ensemble optimisation techniques (also mentioned in Section 2.3.2) that would make ensemble approaches more viable in such situations.

4 CONCLUSION

In summary, this internship project consisted of two core parts - literature review and a practical implementation experiment. The knowledge obtained from the literature review is distilled and summarised in Section 2. Here, the core concepts and ideas, related to uncertainty estimation in deep learning, are discussed and useful related literature is cited.

The second part of this project (Section 3) involves a practical implementation experiment, where three different uncertainty estimation methods (Deep Ensemble, Evidential Neural Network and Equidistant Hyperspherical Prototype Network) have been implemented and evaluated in terms of uncertainty calibration. The results have shown that the predictions of the Deep Ensemble model have the best calibration between the analysed methods, albeit at a quite high computational cost.

The topic of uncertainty estimation is a very broad one. Therefore, given a very limited amount of time spent for this project, many detailed descriptions and implementation considerations have been omitted. Some of these considerations are discussed and possible future developments of related research are suggested in Section 3.3.

5 REFLECTION

While working on this project as an intern at the IVS department of TNO, I got the opportunity to make a deep dive into this specific sub-domain of machine learning. This process was strongly aided by the kind, dedicated and competent supervision from external supervisors Jan-Pieter Paardekooper and Jeroen Manders.

Besides having this great opportunity of expanding my knowledge about uncertainty in deep learning, I have also learned about myself. As presented, this project consisted of two parts - theoretical and practical. During course assignments for courses in the AI BSc and MSc programmes, I have always felt a preference for the practical / implementational parts of these assignments. After this first-time experience of working on a project full-time, I can say that my preferences also translate directly into full-time work. Even though I enjoyed learning about important concepts in a specific domain of AI and reading literature / watching presentations about impressive findings and interesting new methods in the field, I felt more at home while implementing different models and running practical tests on them. The days went by quicker and felt more productive during the second half of this project, when I was working on the practical analysis.

I suppose it goes without saying that I find it unfortunate that I have spent majority of the time of this internship working from home, due to the strong advice to do so, because of the COVID-19 situation. However, I am thankful to the IVS department for facilitating interns working on-site. This has allowed us to get to know each other in person and feel at least a bit more as a part of the team. Besides these few intern meet-ups, I have tried to stick to the company guidelines and work from home as much as possible. This was quite challenging in general, but I have learned ways of adapting my work day to stay as productive as possible and improved my self-discipline to avoid easily reachable distractions.

The external supervisors also facilitated my integration into the work routine of TNO. While joining department and specific project team meetings, I got a taste of what it is like working at a research-focussed organisation like TNO. After seeing multiple project presentations, I have also realised the actual size of the jump between academic research and practical application of artificial intelligence. Working at TNO, I had an opportunity to see work being done on both ends of the process and steps in-between. While research and academic experiments are being conducted, related to complex AI methods being applied to the sensing or control of road vehicles, very few of them have the necessary rigidity, transparency and controllability to make their way into practical tech demonstrations so far. Consequently, experts and specialists with very different backgrounds are necessary to build appropriate frameworks and push this technology forward. This realisation made me understand the importance of organisations that connect academia and industry, like TNO.

Working within a team of diverse experts who reach for a common goal seems to be a very appropriate environment for conducting a graduation project, which is why I hope to get a chance to continue my stay at TNO for conducting my MSc thesis project (and hopefully get to experience working here outside of a lockdown situation).

REFERENCES

[1] 2019. *Who is in control? Road Safety and automation in Road Traffic.* https://www.onderzoeksraad.nl/en/page/4729/who-is-in-control-road-safety-and-automation-in-road-traffic
[2] Fabio Arnez, Huascar Espinoza, Ansgar Radermacher, and François Terrier. 2020. A comparison of uncertainty estimation approaches in deep learning components for autonomous vehicle applications. *arXiv preprint arXiv:2006.15172* (2020).
[3] Gertjan J. Burghouts and Pascal Mettes. 2021. Equidistant Hyperspherical Prototypes Improve Uncertainty Quantification. *NeurIPS* (2021). Workshop on Deployable Decision Making in Embodied Systems (DDM).
[4] Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. 2019. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757* (2019).
[5] MA Ganaie, Minghui Hu, et al. 2021. Ensemble deep learning: A review. *arXiv preprint arXiv:2104.02395* (2021).
[6] Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, et al. 2021. A survey of uncertainty in deep neural networks. *arXiv preprint arXiv:2107.03342* (2021).
[7] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. In *International Conference on Machine Learning*. PMLR, 1321–1330.
[8] Huaping Guo, Hongbing Liu, Ran Li, Changan Wu, Yibo Guo, and Mingliang Xu. 2018. Margin & diversity based ordering ensemble pruning. *Neurocomputing* 275 (2018), 237–246.
[9] Fredrik K Gustafsson, Martin Danelljan, and Thomas B Schon. 2020. Evaluating scalable bayesian deep learning methods for robust computer vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 318–319.
[10] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).
[11] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. 2017. Snapshot ensembles: Train 1, get m for free. *arXiv preprint arXiv:1704.00109* (2017).
[12] Eyke Hüllermeier and Willem Waegeman. 2021. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning* 110, 3 (2021), 457–506.
[13] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. 2018. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407* (2018).
[14] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).
[15] Yann LeCun et al. 2015. LeNet-5, convolutional neural networks. *URL: http://yann. lecun. com/exdb/lenet* 20, 5 (2015), 14.
[16] Rowan McAllister, Yarin Gal, Alex Kendall, Mark Van Der Wilk, Amar Shah, Roberto Cipolla, and Adrian Weller. 2017. Concrete problems for autonomous vehicle safety: Advantages of Bayesian deep learning. International Joint Conferences on Artificial Intelligence, Inc.
[17] Pascal Mettes, Elise van der Pol, and Cees Snoek. 2019. Hyperspherical prototype networks. *Advances in neural information processing systems* 32 (2019).
[18] Maithra Raghu, Katy Blumer, Rory Sayres, Ziad Obermeyer, Bobby Kleinberg, Sendhil Mullainathan, and Jon Kleinberg. 2019. Direct uncertainty prediction for medical second opinions. In *International Conference on Machine Learning*. PMLR, 5281–5290.
[19] Murat Sensoy, Lance Kaplan, and Melih Kandemir. 2018. Evidential deep learning to quantify classification uncertainty. *Advances in Neural Information Processing Systems* 31 (2018).
[20] Joost Van Amersfoort, Lewis Smith, Yee Whye Teh, and Yarin Gal. 2020. Uncertainty estimation using a single deep deterministic neural network. In *International Conference on Machine Learning*. PMLR, 9690–9700.
[21] Lieke LF van Lieshout, Floris P de Lange, and Roshan Cools. 2019. Motives underlying human curiosity. *Nature Human Behaviour* 3, 6 (2019), 550–551.
[22] Shlomo Zilberstein. 2015. Building strong semi-autonomous systems. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

APPENDIX

A TRAINING PARAMETERS

All of the models used for the experimental part of this report were trained using the following parameters:

Optimiser	Stochastic Gradient Descent - `torch.optim.SGD()`
Learning rate scheduler	Functional scheduler - `torch.optim.lr_scheduler.LambdaLR()`
Learning rate scheduling function	$lr = (1.0 - \frac{iteration}{max\_iteration})^{0.9}$, where $max\_iteration = num\_batches * num\_epochs$
Number of training epochs (*num_epochs*)	200
Batch size	1000

Table 4: Model training parameters