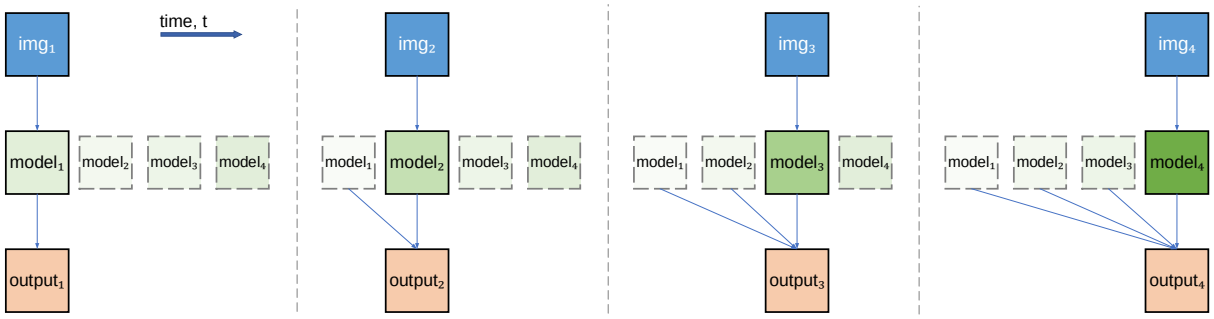
CHALMERS
UNIVERSITY OF TECHNOLOGY



Predictive Performance and Calibration of Deep Ensembles Spread Over Time

A simple way of limiting the computational load of deep ensembles when applied to sequence data

Master's thesis in Data Science and AI

ALEXANDER BODIN
ISAK MEDING

DEPARTMENT OF ELECTRICAL ENGINEERING
DIVISION OF SIGNAL PROCESSING AND BIOMEDICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023
www.chalmers.se

Master's thesis 2023

Predictive Performance and Calibration of Deep Ensembles Spread Over Time

A simple way of limiting the computational load of deep ensembles when applied to sequence data

ALEXANDER BODIN
ISAK MEDING

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Signal Processing and Biomedical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2023

Predictive Performance and Calibration of Deep Ensembles Spread Over Time
A simple way of limiting the computational load of deep ensembles when applied to sequence data
ALEXANDER BODIN
ISAK MEDING

Examiner: Lennart Svensson, Chalmers University of Technology
Industrial Supervisors at Zenseact: Joakim Johnander, Christoffer Petersson, and Adam Tonderski

Master's Thesis 2023
Department of Electrical Engineering
Division of Signal Processing and Biomedical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Schematic illustrating the proposed approach – an ensemble spread over time. Here the previous prediction from models that do not run inference on this timestep also count towards the output of the model, as described in section 1.1.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2023

Predictive Performance and Calibration of Deep Ensembles Spread Over Time
A simple way of limiting the computational load of deep ensembles when applied to sequence data

Alexander Bodin & Isak Meding
Department of Electrical Engineering
Division of Signal Processing and Biomedical Engineering
Chalmers University of Technology

Abstract

In recent years, machine learning models that can provide uncertainty estimates that match their observed accuracy have seen an increased interest in academia. Such models are called *calibrated*, a quality essential for the safe application of neural networks in high-stakes situations. However, good calibration is not enough – high predictive performance is also essential. Autonomous driving (AD) is a setting where this combination of model qualities is much-needed, with the additional requirements of real-time processing of sensor inputs such as camera video sequences. Deep ensembles (DEs) are state-of-the-art for non-Bayesian uncertainty quantification with high predictive performance. However, their deployment in AD has been limited due to their high computational load.

We propose the deep ensemble spread over time (DESOT), a simple modification to DEs that seeks to limit their computational load on image sequence data by letting a single ensemble member perform inference on each frame of the sequence. We apply this proposed system to the problem of traffic sign recognition (TSR), a subfield of AD with a distinctly long-tailed class distribution. DESOTs display predictive performance competitive with DEs for traffic sign classification, using only a fraction of the computational power. For in-distribution uncertainty performance, DESOTs outperform MC-dropout and perform on par with DEs. We conduct two out-of-distribution (OOD) experiments. First, we show that DESOTs increase calibration robustness to common augmentations compared to single models while matching DEs. Second, we test performance on a completely unseen class, for which all models increase their uncertainty in terms of output distribution entropy. Post-hoc calibration using temperature scaling is also evaluated and is shown to improve the uncertainty quantification performance of DESOTs, both in and out of distribution.

Keywords: *Machine learning, artificial intelligence, computer vision, deep ensemble, deep neural network, uncertainty quantification, calibration, traffic sign recognition.*

Acknowledgements

First of all, we would like to thank our industrial supervisors Joakim Johnander, Christoffer Petersson, and Adam Tonderski for their never-ending support and encouragement throughout our work with this thesis. Without the interesting discussions we had, working on this project would not have been nearly as fun and rewarding. We would also like to thank Zenseact for allowing us to use their facilities and computational resources, without which this thesis would have taken a lot longer to finish. Lastly, we would like to thank our examiner at Chalmers, Lennart Svensson, for facilitating this thesis project.

Alexander Bodin and Isak Meding, Gothenburg, June 2023

Acronyms

+ T	temperature scaling applied to model.
AD	autonomus driving.
CNN	convolutional neural network.
DE	deep ensemble.
DESOT	deep ensemble spread over time.
ECE	expected calibration error.
MC-dropout	Monte Carlo dropout.
MCE	maximum calibration error.
ML	machine learning.
NN	neural network.
OOD	out-of-distribution.
pp	percentage point(s).
px	pixel(s).
SM	single model.
SotA	state of the art.
TSR	traffic sign recognition.
UQ	uncertainty quantification.
ZOD	Zenseact open dataset.

Contents

List of Figures

List of Tables

1

Introduction

Recently deep machine learning has irrevocably changed the landscape for many industries, not least the automotive industry. The quest for autonomus driving (AD) vehicles is part of a big push, where machine learning (ML) techniques are key. A problem with the application of ML is that it has been shown that modern machine learning models are over-confident in their predictions, and have become more so amid the performance developments in recent decades [1].

In safety-critical applications, such as AD, the overconfidence of modern neural networks is particularly troublesome since the model's certainty in its output greatly influences what actions are sensible to take [2]. Naturally, it is then of great importance that the probability estimates that the model reports correspond to the actual predictive performance observed in the model's output over time. This is typically referred to as *calibration* and is the measure of how well the subjective output probabilities and the observed long-term prediction performance match [3]. A model that closely and precisely assesses its own uncertainty is referred to as *well calibrated* [4], [3].

There are two main paradigms of machine learning models: Bayesian and non-Bayesian models [5]. A Bayesian model treats model parameters as stochastic variables, each with a probability distribution that is updated during training based on the input data. This enables high-quality posterior distributions over the output space, with great uncertainty quantification performance. One such model is the Bayesian neural network. The disadvantage of this Bayesian approach is that the models are typically difficult to implement and slow to train compared to a normal neural network (NN) [6]. This limits the Bayesian models to smaller networks or various approaches for approximating some other Bayesian model. Therefore, non-Bayesian approaches such as ensembles are more common in practice. For an ensemble, a number of member models are trained and are all applied to each data point at the time of inference. Ensembles have been shown to produce an improved predictive performance in machine learning [7], [8], with the disadvantage of increased computational load during training- and inference time since it typically scales linearly with the number of ensemble members.

It has long been known that ensembles can quantify uncertainty in their predictions [9]. However, the ability of neural network ensembles to produce uncertainty

estimates of quality that rivals Bayesian models, while also achieving high predictive performance was first demonstrated in 2017 by Lakshminarayanan *et al.* [6]. This allowed for a practical and high-performance alternative to the Bayesian approach. This type of ensemble has since been called deep ensemble (DE) [10]–[12], and continues to be the state of the art (SotA) in the field of uncertainty quantification (UQ) for machine learning [12], [13]. As previously mentioned, uncertainty estimation performance in terms of good calibration is essential in safety-critical tasks such as AD. Thus, the benefits of DEs are of interest in the AD space, but their large computational load limits deployment.

This thesis proposes a new model that we call a deep ensemble spread over time (DESOT), an augmented version of deep ensembles. This new model limits computational load while hopefully upholding the benefits of deep ensembles concerning predictive efficacy and uncertainty quantification performance.

1.1 Background and context

Autonomous driving systems employ a wide array of sensors that allow the vehicle to perceive its surroundings and take appropriate action [14], [15]. These many sensors are then used for localization and mapping, path planning, decision making, and ultimately vehicle control [14]. One of the most important of these sensors is the camera, and AD systems use a suite of them to gain a full surround view of the environment. These many cameras produce continuous video feeds that the car has to process in real time, which requires a lot of computational power due to the additional requirements of low latency.

One subtask of AD that uses the cameras of the vehicle is traffic sign recognition (TSR), which is all about detecting and classifying the traffic signs that the vehicle encounters on the road [16]. Modern TSR systems take the image sequences produced by the vehicle's cameras and use advanced machine learning models to classify the signs [17], [18]. Just like any deep machine learning system, these systems could benefit from using ensembles to boost performance. However, their application is limited by the computational resources available in the car, which have to be shared across all the functions previously mentioned.

This project is commissioned by Zenseact, a software company developing autonomous driving solutions for major car makers. Zenseact wants to explore the use case for a system that integrates the aforementioned aspects by investigating whether ensembles can be implemented in a way that improves performance compared to individual machine learning models, but which is more resource-efficient than traditional deep ensembles. The machine learning setting that is in focus is that which works with temporal information in the form of sequences of images. Each sequence tracks an object across time, meaning the informational content is slightly augmented versions of the same object.

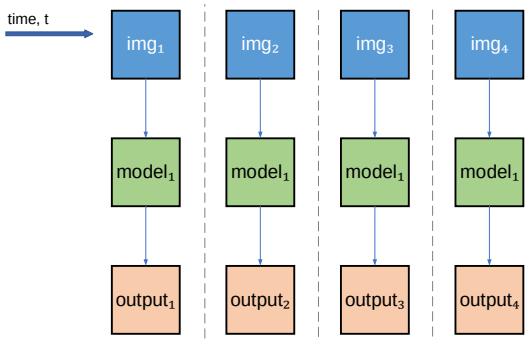


Figure 1.1: Single model architecture. One model performs inference on the input image at each time step.

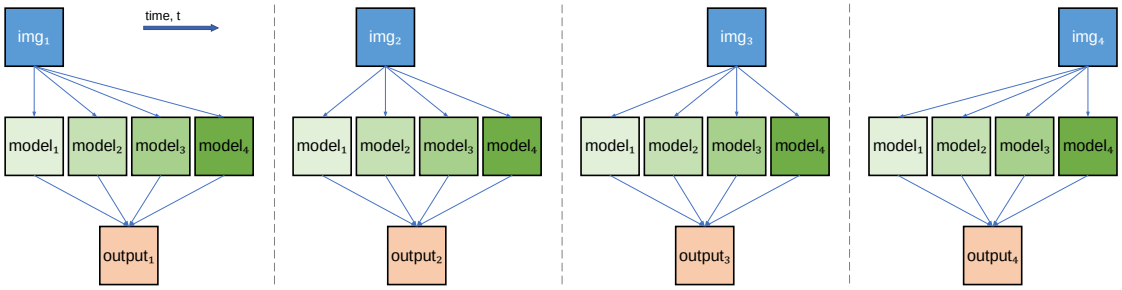


Figure 1.2: Traditional ensemble architecture. Every ensemble member performs inference on the input image in each time step.

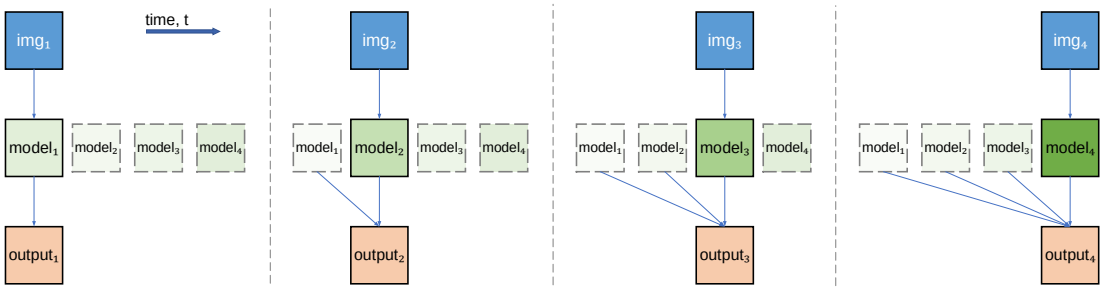


Figure 1.3: Schematic illustrating the proposed approach – an ensemble spread over time. Note that the number of forward passes is the same as for the single model in Figure 1.1. The previous prediction from models that do not run inference on this timestep also count towards the model output.

All machine learning models considered in this thesis are trained and produce their outputs in the single-frame setting, but their predictions are then aggregated across time steps in the sequence. In this manner, the typical way to apply a single model is to let it produce outputs for the single frame in each time step in the sequence, and then aggregate the predictions across time (see Figure 1.1). In a similar manner, a deep ensemble would be applied by letting every member produce their prediction for each time step. The outputs across members are then aggregated for the current

frame before it is combined across time steps (see Figure 1.2). The proposed system combines the outputs of each member, just as for the traditional ensemble (see Figure 1.2), but only one member runs inference in each time step (see Figure 1.3). This means that the same number of forward passes are made as in the case of the single model approach, a fact that limits the computational load of the system.

1.2 Aim

The aim of this thesis is to implement an ensemble spread over time and evaluate its performance against some high-performing models in uncertainty quantification. Baseline models are established and the performance of ensembles spread over time is compared to the performance of these. The system's uncertainty quantification performance in and out of distribution is also compared to the baselines. Additionally, temperature scaling [1], a post-training calibration method, and its effects on UQ performance are evaluated.

1.3 Research questions

Here follow the research questions that we aim to answer.

Is it possible to achieve the benefits of a traditional deep ensemble in terms of predictive- and uncertainty quantification performance using a deep ensemble spread over time? (A)

This is the main research question of the thesis, but it is quite broad. To answer this question we need to also answer the following questions.

How does the predictive performance for deep ensembles spread over time compare to that of the chosen baselines in terms of F1-score and accuracy? (B)

How does the UQ performance of deep ensembles spread over time compare to the chosen baselines in terms of ECE, MCE, and Brier reliability? (C)

Do deep ensembles spread over time successfully adjust their confidence on OOD data by increased entropy, and how does their performance compare to the chosen baselines? (D)

1.4 Delimitations

Since the goal of this thesis is to investigate whether the benefits of using ensembles can be achieved using the proposed ensemble spread over time, only relatively standard machine learning architectures are used. This increases the generalizability of the results, and in extension also means that obtaining SotA predictive performance was not a priority of this thesis.

One of the main motivations of DESOT is to limit the computational load at the time of inference in order to enable the use of deep ensembles in settings with limited computational budgets. However, directly measuring the computational load of different ML models is highly non-trivial since many factors influence performance. Such factors include but are not limited to the speed of data transfer, memory capacity, as well as what other processes are running simultaneously. Because of this, it was decided that the number of model forward passes was to be used as a proxy for the computational load. This proxy is relatively useful since all models tested have the same basic architecture. Before actually employing DESOT or any other ML model on a real system, a thorough performance test would have to be conducted on the machine in question.

While spatiotemporal fusion models such as those proposed by Ji *et al.* [19] and Tran *et al.* [20] are promising, they are challenging to design, obtain annotated data for, and train. Due to this, it is common to design detection, segmentation, and classification models for the single frame setting. Then, the single-frame predictions are combined in some way. This is how the system proposed in this thesis works, and such systems have proved to work fairly well. Furthermore, the spatiotemporal fusion approach is in some ways orthogonal to the ensembling scheme proposed for this thesis, where the two could conceivably be combined. For example, a spatiotemporal convolution similar to that proposed by Tran *et al.* [20] might be used to create an optimal combination rule between the outputs from individual ensemble members, instead of using simple averaging. For these reasons, we chose to limit the scope of the thesis to models that do not explicitly model the additional information that comes from the temporal aspects of a sequence of frames.

1.5 Contributions

The main contribution of this thesis is to propose the idea of ensembles spread over time, an idea that to the best of our knowledge is novel. The method is compared to some of the most common non-Bayesian alternatives. DESOTs are shown to be competitive with traditional deep ensembles, both on pure predictive performance and uncertainty quantification performance, while using a fraction of the computations. Experiments for predictive performance on rare classes are conducted, which seems to be a strong point of DESOTs. While more work is needed to investigate whether the results generalize to other tasks, datasets, and model architectures, we believe that this project serves as a strong baseline for further research into ensembles spread over time.

1.6 Discussion of ethical and sustainability aspects

The advent of autonomous vehicles and more intelligent artificial intelligence will radically change our society in a multitude of ways. This project is a small part of the journey to fully autonomous driving cars and although it might not have radical effects in isolation, the combined work of many related projects will. Therefore, potential issues related to ethics and sustainability must be taken into account.

The safety aspects of AD are an important reason why it is such an important innovation. Around the world, there are 1.35 million fatal accidents every year [21], and more than 90 % of these are due to human error [22]. Vehicle safety is identified as a key factor in lowering these losses [21]. With AD software the driving style of the world's safest human can hopefully be replicated and even exceeded, reducing the accident count greatly and in the best case lowering it to zero. The technology is not yet there but this project might be a step on the way there.

Having millions of cameras on the road, and recording every second being necessary for a working AD system comes with the risk of these cameras being used for malevolent purposes, making it essential to have a significant level of security around the AD software and the data generated [22]. Personal integrity is also a key concern in this regard, as well as the impact on the greater society. This means that companies working in AD must have strict ethical guidelines in order to ensure that the data collected by their systems is handled in a correct manner.

Autonomous vehicles will most likely change the way we travel, providing convenient, fast, and cheap transport [23]. This will most likely come to the detriment of having more vehicles on the roads creating more pollution in the form of air particulates and noise pollution. Part of this issue will be reduced by converting the fleet of cars to electric, but there will still be pollution from the road surface and rubber tires.

Vehicle utilization with AD is likely to increase [24] as services for sharing vehicles or utilizing fleets like public transport are created, which will potentially increase the practicality of not owning a private vehicle. The result of this is a better utilization of Earth's limited resources which will help reduce the environmental strain from the production of new vehicles, ultimately helping the world reach the UN's climate goals.

Equity and equality will be affected greatly around the world, especially in poorer communities. Transportation can be a big financial strain on families with low income, but in the long term with AD, the cost of transportation can come down to be as low as ten times less than owning a private vehicle [24], close to the cost of public transport. This allows for more equity and equality as opportunities for work, education, and leisure can be more accessible to everyone.

2

Theory

In this section, some of the most relevant previous work and the theory behind the concepts used will be presented with the aim of placing the thesis into a context in literature.

2.1 Convolutional neural networks

Convolutional neural networks (CNNs) are used for increasingly complex tasks, which in turn require an increasing number of parameters to be added. The added parameters come with a cost both in terms of memory and in terms of computational footprint for training and inference. However, it has been shown [25] that when the number of parameters increases, using ensembles will speed up training and inference, as well as increase prediction accuracy compared to individual models with the same total number of parameters.

A CNN model consists of convolutional layers and many fully connected neural layers. The outputs from the final fully connected layer in a neural network are called *logits*. These are real-valued numbers that communicate the network's prediction.

2.1.1 Residual networks

Residual networks (ResNets) [26] is a type of deep convolutional neural network containing residual connections over blocks. A standard deep neural network can be hard to train since vanishing gradients start to become a problem for deeper neural networks [26]. Vanishing gradients is a problem that is caused by the products of gradients of the loss function getting increasingly small. In a deep network, the gradient may approach zero, leaving the next layers unable to update the weights. This can be mitigated by adding residual connections which allow the gradient to be passed through allowing for deeper CNNs. There are many different sizes of ResNets to choose from, for example, ResNet18 or ResNet50, which contain 18 layers and 50 layers respectively.

2.1.2 Evaluating the predictive performance of neural networks

There are many different ways of evaluating the predictive performance of a machine learning model in a classification task. Given a binary classification problem, a model prediction can be characterized as either a true positive (TP), a false negative (FN), a false positive (FP), or a true negative (TN) (see Table 2.1).

Table 2.1: Confusion matrix across a set of predictions for a binary classification problem.

		Prediction	
		True	**False**
Ground Truth	**True**	True positive	False negative
	False	False positive	True negative

Over a data set for which the model produces predictions, the counts for each of these four positions of the confusion matrix can be computed. Then, a number of metrics can be computed. The first of which (and most commonly used [27]) is *accuracy*, which is defined as

$$accuracy = \frac{TP + TN}{TP + FP + FN + TN}. \tag{2.1}$$

One well-known problem with accuracy is that it is insensitive to class imbalances in the dataset, which for imbalanced datasets can lead to deceptively high perceived performance. To combat this problem, two additional metrics and a derivative of them might be employed. These are *precision* and *recall*, which are defined as [27]

$$precision = \frac{TP}{TP + FP} \tag{2.2}$$

and

$$recall = \frac{TP}{TP + FN}. \tag{2.3}$$

Intuitively, precision is the share of positive predictions that were actually positive, and recall is the share of true positives that were captured in the prediction. From these two metrics, the F_β-score can be computed. It is defined as

$$F_\beta = \frac{(\beta^2 + 1) \cdot precision \cdot recall}{\beta^2 \cdot precision + recall} \tag{2.4}$$

where $\beta \in \mathbb{R}$, $\beta \geq 0$ [27]. The F_β-score ranges from 0 to 1, where higher is better. If $\beta = 1$, the score equally weights precision and recall and is then called the *F1-score*, often written without a subscript. This is the most common version of the F_β-score.

There are some extensions of the binary F1-score to the multiclass classification domain. For these, we use the naming employed in their implementation in scikit learn. One extension is the F1_{micro}-score, which computes the different fields in the confusion matrix sample-wise one versus all, such that all samples are equally weighted in the resulting score. This is equivalent to accuracy. Another is F1_{macro}, which uses this same operation class-wise and takes the unweighted average across classes. The former approach will allow the majority classes to dominate the score, while the latter will relatively over-weight the importance of prediction performance on samples belonging to rare classes. F1_{macro} is useful in cases when performance on each class is equally important, and is usually what is meant when referring to F1-score in a multiclass setting. This is also the naming convention we use.

2.1.3 Interpreting neural network outputs as probabilities for classification tasks

A classification neural network is a function that maps an input example i to an output vector of dimension C, which is the number of classes. In order to interpret neural network outputs as probabilities, it is common to use the function that in machine learning is colloquially called the *Softmax* function. It is applied to the final neural network output logits vector $\mathbf{z} \in \mathbb{R}^C$ and transforms each element z_j into the range $[0, 1]$ $s.t.$ $\sum_{j=1}^{C} \text{softmax}(z_j) = 1$. Its mathematical expression was first introduced by Boltzmann [28] in 1868 as the Boltzmann distribution, and later used by Bridle [29] for interpreting neural net outputs as probabilities. Given an output logit vector $\mathbf{z}$, the softmax value for the element $z_j \in \mathbf{z}$ is computed as

$$\text{softmax}(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{C} e^{z_k}}. \tag{2.5}$$

Softmax applied to a vector is defined element-wise. The prediction $\hat{y}$ for a neural network given an input example x is typically said to be

$$\hat{y} = \arg\max_j \{\text{softmax}(z_1), \text{softmax}(z_2), ..., \text{softmax}(z_j), ..., \text{softmax}(z_C)\} \tag{2.6}$$

where j is an index in the logit vector. The model's confidence in that prediction is then simply $conf(\hat{y}) = \text{softmax}(z_{\hat{y}})$. One notable property of the softmax function is that it is scale-variant due to the nature of exponentials, meaning that $\text{softmax}(\mathbf{z}) \neq \text{softmax}(\mathbf{z}/r)$ $\forall$ $r \in \mathbb{R}, r \neq 1$.

2.2 Uncertainty quantification and calibration

The theory of uncertainty quantification has long been of interest in academic circles but it came to be a more closely studied subject during the development of weather forecasting when meteorological forecasters give subjective estimates of the probability of various weather phenomena. As we shall later see, these concepts quite naturally generalize to the machine learning context.

One key aspect of the quality of a forecaster's predictions is their *calibration*, a concept which is commonly defined as the deviation of the forecaster's subjective probability assigned to an event from the long-term empirical probability of that event [3], [4], [30]. Calibration is also commonly referred to as *reliability* [31]. For the sake of clarity and mathematical correctness, consider an example adapted from DeGroot & Fienburg (1983) [3]. Assume that there are two possible events for a given day: rain, and no rain. For a given day i, a forecaster gives a subjective probability p_i of the chance of rain from a set of K possible values in the set $\mathcal{P}, s.t.\ p_i \in \{P_1, ..., P_K\}$. Why limiting the number of possible values is necessary will become apparent later. At the end of the day, he observes the outcome y_i of the event, where y_i is an indicator variable that is 1 in the case of rain and 0 otherwise. Let $\nu(p)$ be the long-term distribution of the forecaster's predictions of rain. Next, let $\rho(y_i = 1|p_i)$ be the long-term empirical probability of rain given the forecaster's prediction of rain p_i. That is, given that the forecaster predicts a probability p_i of rain on day i, what is the actual probability that it will rain based on historical predictions. With this background, a forecaster is said to be *well calibrated* if $\rho(y = 1|p) = p$ for all values of $p \in \mathcal{P}$ such that $\nu(p) > 0$ [4]. For example, this means that out of 10,000 days that a well calibrated forecaster claimed that the probability of rain was 0.4, it actually rained 4,000 of them. This also gives insight into why the number of allowed values that the predictions can take on has to be limited — it is necessary to allow for statistically sound empirical estimates of $\nu(p)$ and $\rho(y = 1|p)$.

Interestingly and importantly, a forecaster can be well calibrated but produce completely uninformative predictions. The forecaster might simply observe the climatological base rate (long-term probability) for rain on any given day, and give that as his prediction of the chance of rain every day. Using the definition of calibration from earlier, this will lead to well calibrated but useless predictions. Thus, the notion of *refinement* is given as a complement to calibration [3]. Assume that forecasters A and B are both well calibrated. Let forecaster A give the climatological base rate as his prediction on any given day, and let forecaster B give either $p = 0$ or $p = 1$ on any day and always be correct. Then A is least refined, since any well calibrated forecaster is at least as refined as him, and B is most refined. See DeGroot and Fienberg [3] for a more mathematically expressive definition of refinement. Using this intuition regarding forecasters A and B, refinement can be seen as a measure of the usability of the forecasts given by the forecaster.

One important point of contention in meteorological circles early in the field's life cycle was how to evaluate the quality of a forecast. This created a need for metrics that are not possible for the forecaster to artificially game by adjusting their forecast

to optimize for a higher score [30], such as by simply mimicking the known base rate of the events [3]. This prompted developments in the theory surrounding *proper scoring rules*, which are reward functions that encourage forecasters to give their actual subjective forecast probabilities rather than trying to game the system. More formally, suppose a forecaster believes the probability of rain to be p_a. A scoring rule is one which rewards the forecaster $S(p_a)$ if it rains, and $S(1-p_a)$ if it doesn't. Assume that the forecaster wishes to maximize his reward. If the forecaster instead gives p_b as his prediction of rain, contrary to his actual subjective prediction p_a, his expected reward is $p_a S(p_b) + (1-p_a)S(1-p_b)$. If the scoring rule is proper, the choice to give $p_b = p_a$ maximizes the forecaster's reward. If the scoring rule is *strictly proper*, then $p_b = p_a$ is the only prediction that maximizes the reward [3], [32]. Here, we have assumed that the forecaster wishes to maximize his reward, but the same of course holds for the negation of a score that the forecaster wishes to minimize.

Many classification problems, such as weather forecasting, are not binary but multi-class, where an event can fall into any class $c \in \{1, ..., C\}$. A prediction of subjective probabilities for an event x_i, $i \in \{1, ..., N\}$, where N is the number of events, are to be given by the forecaster. Following the notation from the previous paragraph but expanded to a multi-class problem, we denote the subjective probabilities given by the forecaster as a categorical distribution $\mathbf{p}(x_i) = (p_{i1}, p_{i2}, p_{i3}, ..., p_{iC})$ across the classes, such that for any instance x_i, $\sum_{j=1}^{C} p_{ij} = 1$. With this background, a score was developed by Brier [30], that has later become known as the Brier score and is defined as

$$BS = \frac{1}{N} \sum_{j=1}^{C} \sum_{i=1}^{N} (p_{ij} - y_{ij})^2 \tag{2.7}$$

where y_{ij} is an indicator variable that is 1 if instance i falls into class j, and 0 otherwise. The Brier score of a perfect forecaster (forecaster B previously mentioned) will be 0. Forecasts of lower quality receive a higher Brier score.

An important feature of the Brier score is that it is a strictly proper scoring rule [3]. More correctly, its negative is a proper scoring rule. However, as previously discussed, this makes no difference in practice since the same argument can be made for the negative of a score that the forecaster wishes to minimize. Murphy [31] showed that the Brier score can be decomposed into three distinct parts

$$BS = Reliability - \underbrace{Resolution + Uncertainty}_{\text{Refinement}} \tag{2.8}$$

and Bröcker [33] later showed more generally that any strictly proper scoring rule can be decomposed in such a way.

The reliability part of the Brier score decomposition is often referred to as *Brier reliability* and is in itself a measure of calibration and can be used as such. In the binary case, it is defined by DeGroot and Fienberg [3] as

$$Brier\ reliability = \sum_{p \in P} \nu(p)(p - \rho(p))^2. \tag{2.9}$$

Remember from before that P is the set of allowed values that the subjective forecast probabilities p can take on, $\rho(p)$ is the actual probability of rain given the forecaster's prediction, and $\nu(p)$ is the long-term distribution of the forecaster's prediction of rain. The Brier reliability score is non-negative, and a perfectly reliable forecaster gets a score of zero.

The Brier reliability can also be expanded to the multi-class setting, where Brier [30] first defined his score

$$Brier\ reliability = \sum_{j=1}^{C} \sum_{k=1}^{K} \frac{n_{jk}}{N_j} \left(p_{jk} - \frac{o_{jk}}{n_{jk}} \right)^2 \tag{2.10}$$

where K is the number of different values that the subjective probabilities can take on, n_{jk} is the number of times that the k^{th} forecast was issued for class j, N_j is the number of occurrences in class j, and o_{jk} is the total number of events that occurred before the jk^{th} forecast was issued [34]. Note that Equation 2.10 is the *empirical* decomposition of the Brier score, which is why the term $\frac{o_{jk}}{n_{jk}}$ in Equation 2.10 is used as an estimate for $\rho(x)$. By the same reasoning, the term $\frac{n_{jk}}{N_j}$ is used as an estimate for $\nu(p)$. This is also why the number of different values that the subjective probabilities are allowed to take on must be limited to only K. If the probabilities p_i were allowed to take on any real number in the range $[0, 1]$, it would not be possible to calculate the empirical long-term distributions $\nu(p)$ and $\rho(p)$ since the number of observations is finite and the possible output space is continuous and therefore infinite. Thus, in practice, binning of the forecast probabilities into K bins is used for such scenarios.

2.2.1 Expanding the concepts of predictive uncertainty and calibration to neural networks

We introduced the concept of calibration in the meteorological setting in section 2.2, and it is now time to expand it to the neural network setting. When introducing the concept, we used the definition that calibration is "the deviation of the forecaster's subjective probability assigned to an event from the long-term empirical probability of that event". Thus, we need to expand the meaning of subjective probability and long-term empirical probability. In the neural network setting of this thesis, we lean on the theory from subsection 2.1.3 where the softmax output distribution is interpreted as class-wise output probabilities. This is the subjective probability in

the definition above. With "the long-term empirical probability of that event", we refer to the actual probability, across the whole dataset, that a given class was the ground-truth class given the probability that was reported in the categorical output distribution.

One interpretation of the uncertainty of the neural network in its output is the *entropy* of the predictive distribution. Here, entropy is meant in the information theoretical sense, which has intuitive connections to the physical interpretation. First introduced by Shannon [35] for digital communications, the idea of entropy in information theory is closely related to the average information content of the message to be relayed. Assume a message of length N, with C distinct characters, or values. If the fractions of the different characters' prevalence in the message are denoted $p_j = \frac{n_j}{N}, j \in \{1, ..., C\}$, where n_j is the count of the j^{th} character, the informational entropy of the message can be expressed by

$$entropy = -\sum_{j=1}^{C} p_j \log\ p_j \tag{2.11}$$

which has later become known as Shannon's entropy. If the logarithm used is of base 2, then the unit is bits. If the natural logarithm is used, the unit is nats [35]. In information theory, Shannon entropy acts as a lower bound on the number of bits, or nats, needed to relay the message in its theoretically most compressed state.

Applying Shannon's entropy to the probability vector produced by the softmax function allows for an interpretation where the entropy of the categorical output distribution $\mathbf{p}(x_i)$, for an input example x_i, is a measure of model uncertainty. The intuition is that if the network is uncertain, we expect a flat output distribution across classes, which yields a high entropy. Conversely, if the network is very certain in its prediction, we expect a very sharp distribution, which results in a low entropy. The output distribution is a frequentist point estimate of the distribution over classes, and the uncertainty is therefore not the uncertainty across model parameters in the Bayesian sense.

2.2.2 Additional ways to measure calibration

One common way to visualize the degree of calibration for any model is a reliability diagram [3], [36] (see Figure 2.1 for an example). It plots the observed accuracy of the model against the self-reported confidence that the model had in the prediction. For a perfectly calibrated model, the reliability diagram would be the identity function [1]. Typically, the reliability diagram is drawn as a histogram with B equal-width bins, in which the height of any bin $b \in \mathcal{B} = \{1, ..., B\}$, is the expected (average) accuracy in that bin. The interpretation of model prediction and confidence from subsection 2.1.3 can be used to place each sample into the bin representing the confidence for the top prediction of the sample. Let $\mathcal{L}_b$ be the set of indices of the predictions in the b^{th} bin, the accuracy and the confidence for that bin [1], [10] can

be defined as

$$\text{acc}_{avg}(\mathcal{L}_b) = \frac{1}{|\mathcal{L}_b|} \sum_{i \in \mathcal{L}_b} \mathbf{1}(\hat{y}_i = y_i) \tag{2.12}$$

and

$$\text{conf}_{avg}(\mathcal{L}_b) = \frac{1}{|\mathcal{L}_b|} \sum_{i \in \mathcal{L}_b} \text{conf}(\hat{y}_i). \tag{2.13}$$

Remember from before that we define model confidence as the probability assigned to the top prediction. Then, two numerical metrics can be computed based on the reliability diagram. These are expected calibration error (ECE) and maximum calibration error (MCE) [37]. ECE is computed as

$$\text{ECE} = \sum_{b \in \mathcal{B}} \frac{|\mathcal{L}_b|}{N} |\text{acc}_{avg}(\mathcal{L}_b) - \text{conf}_{avg}(\mathcal{L}_b)| \tag{2.14}$$

where N is the total number of predictions, and it is the weighted average of the expected calibration error across all B bins. Intuitively, this is the average difference between the observed expected calibration error and the identity function across bins in the reliability diagram. MCE is computed as

$$\text{MCE} = \max_{b \in \mathcal{B}} |\text{acc}_{avg}(\mathcal{L}_b) - \text{conf}_{avg}(\mathcal{L}_b)| \tag{2.15}$$

and is the maximum expected calibration error across all B bins or the largest gap between the average bin accuracy and average bin confidence.

Reliability diagrams and the metrics derived from them have in recent years come to be criticized [38], [39] for some quite serious flaws. This is despite their relatively wide use in deep learning literature. The criticisms include the fact that ECE, as it was first introduced in [37], is a metric intended for binary classification. It has since often been used for multi-class problems, and in that case, it only considers the most probable class for calibration evaluation. Vaicenavicius *et al.* [39] calls this induced binary classification since it relies on an implied true-class-versus-the-rest classification problem. This means that a large part of the information in the output distribution is quietly discarded, leading to a possibly uninformative score.

Many practical applications of neural networks also require all probabilities to be calibrated, not just the top prediction [39]. Additionally, the B bins are all equally wide which leads to an unequal distribution of predictions in the bins since most modern NNs are over-confident [1], often leading to a heavy right-skew in the bin count distribution in the reliability diagram. This means that despite large numbers of validation samples the bins of lower confidence might be close to empty, leading to the performance on a few samples largely dominating the score since the absolute difference in all bins is weighted equally. Furthermore, the choice of the number of bins is a hyperparameter to which the reliability diagram is highly sensitive [40],

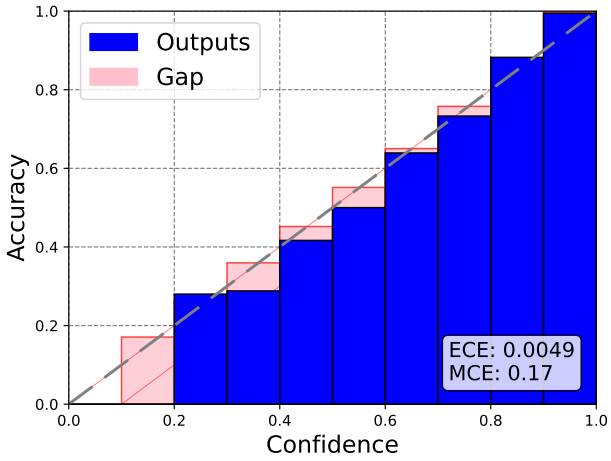


Figure 2.1: Example of a reliability diagram. The height of the blue bars is the average accuracy in each bin, and the height of the pink bars is the average confidence in each bin. The dotted gray line is the identity function that a reliable model's predictions follow.

making comparisons difficult and leading to a bias-variance tradeoff in the number of bins. Lastly, the predictions within a bin may have a high variance but close to zero mean in confidence, making the reliability diagram deceptive. For example, a bin may have a distribution of samples resembling a U shape or a uniform distribution, giving the bin the same mean but a different variance. For all these reasons, some modifications to ECE are proposed in [38]. These include adaptive bin widths to make the bin distribution uniform and other measures to increase how well the metrics reflect the calibration of the model. With these shortcomings in mind, reliability diagrams, and their derivative metrics remain widely used in literature and practice, perhaps due to their intuitive nature.

2.2.3 Temperature scaling

Because of the properties of proper scoring rules outlined earlier in this section, it is to be expected that any neural network that is trained for a classification task with the use of a proper scoring rule as the loss function will be well calibrated by default since this minimizes loss. However, this is not what is observed empirically. Guo *et al.* [1] show that modern neural networks are over-confident when evaluated on an unseen validation data set. To combat this problem, they propose *temperature scaling*, a simple method where a single scalar $T \in \mathbb{R}, T > 0$ is used to scale the output logits before softmax is applied. This affects the entropy of the output distribution, and if used in conjunction with the prediction interpretation in Equation 2.6 adjusts the network's confidence. Importantly, the class order in the predicted distribution is not altered by this augmentation. In this framework, the temperature T that optimizes calibration is found by minimizing a proper scoring rule with respect to T on a separate validation data set.

Guo *et al.* [1] claim that temperature scaling is the best-performing calibration method in most cases, while also being the fastest and simplest to implement. It has later been shown that a temperature scaled single model performs poorly for out-of-distribution data [13]. It should be noted that deep ensembles and temperature scaling are not mutually exclusive, but can be combined. Ashukha *et al.* [12] even go as far as to claim that all models, including ensembles, should be temperature scaled before they are compared on UQ performance since some models might be uncalibrated by default. This notion is corroborated by Minderer *et al.* [41], who claim that temperature scaling helps unveil the underlying differences in calibration that are otherwise obscured by simple average under- or overconfidence.

2.2.4 Aleatoric and epistemic uncertainty

The total predictive uncertainty for a model's output is often partitioned into *aleatoric uncertainty* and *epistemic uncertainty*. Epistemic uncertainty is that which is due to a lack of knowledge about the world and therefore can be mitigated by the collection of additional information. An example of epistemic uncertainty might be out-of-distribution data where the network has not trained on a given class. On the other hand, aleatoric uncertainty is uncertainty that is inherent to the world, usually because of some stochasticity in the data generation, and it is therefore impossible to incorporate and compensate for in the model [42]. An example of aleatoric uncertainty is the noise generated by a sensor or the inherent stochasticity when rolling dice. The distinction between aleatoric and epistemic uncertainty is useful since it allows for the formalization of which parts of the uncertainty can be reduced by way of model augmentation or extension of the dataset and which cannot [42]. Consequently, this thesis concerns itself with how well the chosen models perform for both the cases of epistemic uncertainty as well as aleatoric uncertainty.

2.3 Neural network ensembles

Ensembles are a collection of models which together help predict the outcome of the model. Ensembling is a technique used in all fields of machine learning that obtains a higher-performing model from a diverse set of worse-performing ones [8]. This concept has been extended to neural networks where the same effects can be seen when aggregating the results. There are a multitude of different ensembling techniques available to practitioners [12].

2.3.1 Deep ensembles

One of the many ensemble approaches that serves to promote diversity among ensemble members is random weight initialization of the neural network. This approach is used by Lakshminarayanan *et al.* [6] in combination with random shuffling of training data in their 2017 paper on uncertainty estimation using ensembles, and it is one of the most commonly used in practice due to its simplicity. This type of model is often called deep ensemble (DE).

DEs were shown to have good predictive performance on par with other techniques, as well as high UQ performance both for in-distribution and out-of-distribution (OOD) data [6]. As described in the original paper, training DEs is quite simple, with three steps involved. The first is to use a proper scoring rule as the loss function, the second is to optionally use adversarial training to increase robustness, and lastly to train the ensemble using randomized initialization of model parameters to increase variety in the ensemble [6]. Many common loss functions, such as cross-entropy loss, are strictly proper scoring rules and can therefore be used in the deep ensemble framework. In practice, adversarial training is often omitted if improved robustness is not strictly necessary. It has been shown that deep ensembles are some of the best-performing models for uncertainty estimation [6], [12]. Ashukha *et al.* [12] find that deep ensembles are superior to any other model tested for uncertainty quantification given a fixed test-time budget. The intuition behind why this might be is that since each model is trained independently, each model will find different local minima in the high dimensional loss space, which makes the model's ability to quantify its uncertainty more robust [12].

Lakshminarayanan *et al.* [6] could also demonstrate that the model had the attractive property of decreasing its certainty of prediction in out-of-distribution examples, which was demonstrated using an ensemble trained on the MNIST dataset on examples from the NotMNIST dataset which contains letters instead of digits. It has later been verified that DEs are the SotA for UQ on OOD data [12], [13].

Now, let's more formally define deep ensembles. Assume we have a deep ensemble of M members. Following the notation used earlier, let $\mathbf{p}_m(x_i)$ be the output distribution of the m^{th} member of the ensemble, $m \in \{1, ..., M\}$, on an input example x_i that is to be classified into one of C distinct classes. This distribution is represented as a C-dimensional vector. Then, the output distribution $\mathbf{p}_{DE}(x_i)$ of the deep ensemble on input example x_i is the element-wise average over the individual members' output distribution $\mathbf{p}_m$, such that

$$\mathbf{p}_{DE}(x_i) = \frac{1}{M} \sum_{m=1}^{M} \mathbf{p}_m(x_i), \tag{2.16}$$

which is the element-wise average across the categorical output distributions for all of the M member models for input example x_i.

2.3.2 Monte Carlo dropout

Dropout was first introduced by Srivastava *et al.* [43] as a regularization measure during training to limit overfitting and increase the generalizability of the learned representation. With dropout, each neuron is turned off at random during training according to a pre-specified probability, or dropout rate, p. This helps the network not to overfit, and therefore generalize better, as it has to create a more robust representation when any neuron can be dropped at any time. Recognizing that using an ensemble of a set of models is usually beneficial for model performance, Srivastava

et al. [43] show that using dropout during inference is equivalent to sampling from an exponential set of possible smaller models which yields higher overall performance. Gal and Ghahramani [44] later showed that performing a number of forward passes through a model with dropout enabled and averaging the results can be seen as a Bayesian approximation. They chose to call this Monte Carlo dropout (MC-dropout) and claimed that it enables superior uncertainty estimation performance in both regression and classification tasks compared to vanilla models. Of note is that since the introduction of MC-dropout, Lakshminarayanan *et al.* [6], Ashukha *et al.* [12], and Ovadia *et al.* [13], have all claimed that deep ensembles are superior in uncertainty quantification. However, MC-dropout remains widely used due to its simple implementation and general improvement of performance compared to vanilla single models.

3

Methodology

In this chapter the methodology chosen for the study is outlined, beginning with a specification and motivation for the choice of machine learning problem as well as a more formal definition of the proposed model. Then, the software libraries used are mentioned, along with short motivations. The data used for the training and evaluation of the models is highlighted, along with the data pipeline used for the project.

3.1 The machine learning task at hand

One key consideration for the project was what kind of problem to apply the proposed ensemble approach to. Through discussions with Zenseact, we settled on using the model on a classification problem where the model should predict labels for cropped patch sequences of traffic signs. The reasoning behind choosing this problem instead of some more complex problem was that the focus of the thesis is to evaluate the potential of ensembles spread over time. Implementing a working system for this purpose and investigating a variety of aspects of the proposed system was chosen over solving more complex problems that in principle are not any more novel than classification, such as object detection or semantic segmentation. Though the problem domain itself is not crucially important for the thesis, it framed the project and influenced some of the choices made during the design of the investigation. It also affects how the results should be interpreted. Therefore, a brief overview of the problem and its domain is given for the rest of the section.

The chosen problem falls into the domain of traffic sign recognition (TSR), a field with a decades-long history of development [45]. The first systems available for private end users were introduced in higher-end vehicles in the late-2000s or early 2010s as an aid for drivers. These systems were often limited to a few different classes of traffic signs [46]. Lately, TSR has become an important part of AD systems, and high and reliable performance is important for safety.

There are two main subproblems of TSR, traffic sign detection and traffic sign classification [16]. This project concerns itself with the latter and assumes that regions of interest have already been identified by a system (in this specific case, human annotators) earlier in the ML pipeline. The domain is characterized by a large

number of classes with an imbalanced class distribution. Additionally, variations in illumination, perspective, and occlusions are common [45], making the problem distinctly long-tailed. Furthermore, many of the classes are very similar in shape and color, but with important differences in meaning, such as speed limit signs. Deep learning has recently started revolutionizing this domain, with many models achieving accuracies of over 95% in research settings [47]. This means that any differences in predictive performance between the models are likely to be small in absolute terms, and performance benefits might instead lie in the performance of the models on difficult examples such as short sequences or obscured scenes.

3.2 Software libraries

The implementation of the data handling, the models, and the auxiliary code for the thesis was all coded in Python. The code for handling the data, training, evaluation, and storage of trained models was built using the PyTorch software library [48], an open-source project for deep learning in Python. This allowed for easily and quickly building modular code for purposes of deep learning. The base deep neural networks used are from the torchvision library, which is also part of the PyTorch project. This means that the networks could swiftly be implemented as part of the machine learning system. The models were trained on a computational cluster by using a container environment and experiments were tracked using the Tensorboard library for Python. The use of the container environment ensured compatibility across computational devices. Additionally, Pandas and Numpy are used for handling tabular data such as reading annotations files in CSV or JSON format or doing matrix computations.

3.3 Data

The data used in this thesis comes from a subset of the publicly available Zenseact open dataset (ZOD) [49], for which a dedicated development kit is available as a pip install. The data was collected, consolidated, and released by Zenseact in order to further developments and research within AD. It was collected over two years by the company's fleet of cars in 14 countries in Europe. The dataset consists of three separate parts – *Frames*, *Sequences*, and *Drives*. In this thesis, only the Frames were used, and in these, only the images from the 120° field of view front-facing 3848×2168 pixel camera were used. An example of a frame from ZOD can be seen in Figure 3.1, where annotation boxes for the traffic signs have been overlaid. The Frames subset contains 100,000 still images and was employed as training data for the models, as well as for validation of single-frame classification performance.

Figure 3.1: An example of a frame from the ZOD frames dataset with 2D annotation boxes for traffic signs drawn on the image.

3.3.1 Annotations

Each image in Frames contains annotations for a range of different dynamic and static objects, which includes 2D- and 3D bounding boxes for pedestrians, traffic signs, poles, lane markings, and other relevant objects (see Figure 3.1 for an example). In the case of the Sequences, only the middle frame is annotated, and the rest of the frames, both before and after, are not. For the purposes of this thesis, the 2D bounding boxes for objects of the static objects subclass *Traffic signs* were of interest. In total, there are around 446,000 distinct, annotated traffic signs in the 100,000 images in Frames. The annotations for the traffic signs have been created by a team of professional annotators, and should not contain any annotations for signs that are not relevant to traffic, such as advertisements or billboards. The data contains 156 distinct classes of traffic signs, including two specialty classes – *NotListed* and *unclear*. The NotListed class is used by the annotators when the traffic sign in question does not fall into any of the other 155 classes. Such signs might for example include destination signs. The unclear class is used when the class for some reason is difficult to determine. This might for example include cases when the sign is heavily occluded or the sign is otherwise difficult for a human annotator to classify. The class distribution of the single-frame traffic sign dataset is distinctly long-tailed, where some classes have in excess of 10,000 unique instances, while some classes have fewer than ten examples.

Figure 3.2: Random sample of still images from the training data, taken from the Frames subset of the ZOD.

Figure 3.3: Random sample of eleven sequences taken from the sequences dataset that is used for comparing the tested models.

3.3.2 Data preprocessing and datasets

In order to make the data from ZOD usable for the purpose of the thesis, the 2D annotation boxes were used for cropping out the traffic signs from the single-frame dataset. A padding of ten percent was added to each side of the 2D annotation box, and the crops were then saved in their raw format. In addition to the raw files, a separate JSON file with some information on the annotations was created for each data set derived from the ZOD, which each included the image and annotation ID, the traffic sign class, and the height and width of the crop. This is useful further down the data preprocessing pipeline since it allows for filtering of the data based on some different criteria.

The standard train-test split of 90-10 from the ZOD development kit was used to divide the Frames dataset into a training and a validation set (see Figure 3.2), with a share of the frames being reserved for a sequence set (see Figure 3.3). The sequences dataset used for the thesis (note the distinction from the ZOD Sequences) was created by taking a subset of the frames from ZOD Frames and extending them using the raw video feed from internal, unpublished, and unannotated data, to create a sequence of consecutive images. The annotations from ZOD single frames were then used with an implementation of ByteTrack [50] to create 2D annotation boxes for each frame in the sequences. These tracked annotations were then processed in the same way as the training and validation sets by cropping with padding and saving each crop as an image. Care was taken to ensure no overlap between the single frames used for training, validation, and the single frames used to extract sequences from additional internal data. Including the final frame that is part of the ZOD, the tracked sequences are 11 frames long. In total, there are four distinct (sub)sets of data used for the thesis (see Figure 3.4). The tracking used for creating the sequences dataset is not perfect, and on average the tracking quality degrades as the distance from the annotated frame, *i.e.*, the last one in each sequence. A more thorough discussion of data quality is included in section A.1 in the appendix.

There are a few reasons for using the additional internal data instead of the ZOD Sequences. First of all, the ZOD Sequences were not available at the start of the work on the thesis. The use of additional data also increases the available sequences for model evaluation from the 1429 independent sequences in the ZOD Sequences to 29,359 independent sequences, meaning more statistically sound estimates of model performance can be made. As previously mentioned, TSR is distinctly long-tailed, which means that more data efficiently allows for higher coverage of the space of possible input data.

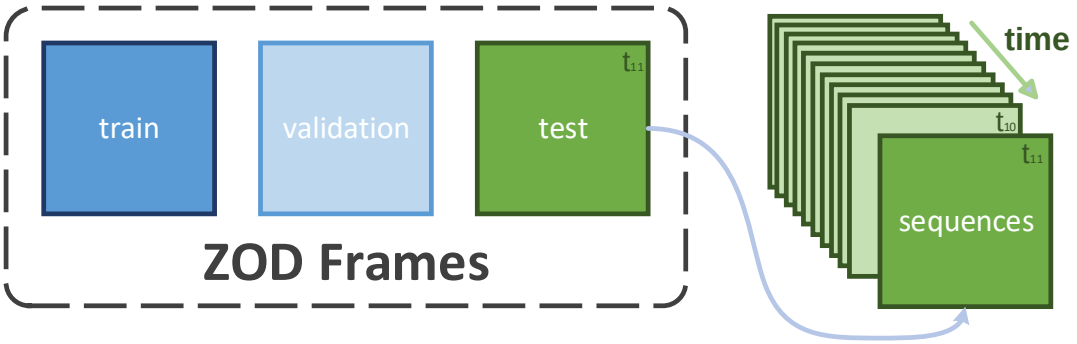


Figure 3.4: Visualisation of the datasets used for the project. The three datasets *train*, *validation*, and *test* are all mutually exclusive subsets of the ZOD Frames dataset. The datasets *sequences* is an extension from the frames in the *test* dataset where the frames prior have been tracked and cropped. Note that t_i denotes the i^{th} frame of a sequence, $i \in \{1, ..., 11\}$, each frame in such a sequence originating from the same video as the corresponding t_{11}-frame.

3.3.3 Dataset implementation details

The datasets and data loaders were implemented in PyTorch. As previously mentioned, the implementation allows for data filtering. This is done in Pandas during dataset initialization. The filtering is based on explicitly excluded classes, which in practice was mainly used to filter out the data points labeled as unclear, but also classes with too few occurrences. This helped with training since the unclear class is uninformative in its nature, and classes with too few occurrences are uninformative due to a lack of variety. The filtering functionality also allows for sorting away crops based on size (in pixels). This was useful when training the models, but also for evaluation when running experiments and diagnostics. When the dataset is queried for a new item by the data loader, a transform is first applied to the crop. For training, all crops were resized to 64×64 pixels and then a random crop was made which reduced the size to 56×56 pixels. Then, normalization was applied, which maps the RGB values from their range $[0, 255]$ to the range $[-1, 1]$, which in practice speeds up training. The transform used for testing and validation was the same as for training but without random cropping. This was to ensure that the results are deterministic.

3.4 Experimental approach

The core approach of the thesis is comparing a single model and a standard ensemble consisting of M members with an ensemble model of the same number of members, but where the inference is spread among the members and images over time. Thus, the thesis extends previous literature conducted on still-image single frames to sequences of single frames and proposes a simple and novel approach to

applying ensembles on such sequence data. The single model serves as an expected lower bound on performance and a deep ensemble [6] serves as the performance target, referred to as the upper bound. Do note that the upper bound is not actually tractable to be used in car due to a limited computational budget. It is simply included as a theoretical maximum of the performance of DESOT that we might hope to observe. We have chosen to refer to these two bounds as the baselines.

The thesis investigates where between the two performance bounds the proposed model falls. In practice, this was done by comparing the DESOT to the two baseline models. MC-dropout [44] was used as an extra model of comparison, chosen because of its simple implementation and common use by practitioners.

3.4.1 Formal model definition

Define a sequence $\mathbf{x} \in \mathbb{R}^{T \times H \times W \times 3}$ as a vector of T distinct still images, each with a height of H pixels, a width of W pixels, and three separate color channels. Now, we have a classification problem where a model must produce a categorical output distribution across C classes for such a sequence $\mathbf{x}$. Assume that there is a set of M different deep NN models that can each conduct this classification. A single model $m \in \{1, ..., M\}$ produces a categorical output distribution $\mathbf{p}_m(x^t)$ for each single image x^t at timestep $t \in \{1, ..., T\}$ in the sequence $\mathbf{x}$. Then, the final output distribution for model m is defined as

$$\mathbf{p}_m(\mathbf{x}) = \frac{1}{T} \sum_{t=1}^{T} \mathbf{p}_m(x^t), \tag{3.1}$$

which is the element-wise (class-wise) average across the output distributions for each image in the sequence at different time steps. This setup is the lower baseline we use for this thesis – a single model that produces a prediction at each time step (see Figure 1.1 for a visualization). In practice, one would use a window size such that Equation 3.1 constitutes a moving average across some images. Due to the shortness of our sequences, only 11 frames, we have chosen to use the average across the entire sequence for a model.

Now imagine that all M models are used for classifying each image in the sequence, such that the final output distribution for the sequence is

$$\mathbf{p}_{DE}(\mathbf{x}) = \frac{1}{M} \sum_{m=1}^{M} \mathbf{p}_m(\mathbf{x}) = \frac{1}{MT} \sum_{m=1}^{M} \sum_{t=1}^{T} \mathbf{p}_m(x^t), \tag{3.2}$$

which is how we choose to apply deep ensembles [6] to sequences – averaged across the images of the sequence $\mathbf{x}$. See Figure 1.2 for a visualization of a DE applied to a sequence of images. DE_M will be used to denote an M-member deep ensemble.

The proposed model, which we call deep ensemble spread over time (DESOT), instead only uses one model $m \in \{1, ..., M\}$ for any given image x^t, $t \in \{1, ..., T\}$ to produce a categorical output distribution, but the models are alternated such that any given model m is used on average T/M times for a certain sequence of T images. Analogously to the notation used for DEs, DESOT$_M$ will denote an M-member deep ensemble spread over time. Assume that the order that the models are used on the sequence is defined by an ordered list O, $|O| = T$. For any sequence, the final output distribution of the DESOT across the C classes is

$$\mathbf{p}_{DESOT}(\mathbf{x}) = \frac{1}{|O|} \sum_{t=1}^{|O|} \mathbf{p}_t(x^t), \tag{3.3}$$

where t is some time step $t \in \{1, ..., T\}$, and $\mathbf{p}_t(x^t)$ is the posterior distribution of the ensemble indicated at the t^{th} position in O. This definition of DESOT perhaps trivially, but also importantly, means that they can only be applied to sequences because they fundamentally rely on the alternation of the members on neighboring frames. If the DESOT model were to be used on a sequence length of one, the model would be equivalent to a standard single model. See Figure 1.3 for a visualization of a DESOT.

3.4.2 Computational footprint

One practical aspect that is key to if an ML model can at all be employed for a certain problem and situation is the size of the computational footprint of the model. If a model's computational footprint is larger than what the computational capacity of the system running the model can handle within the latency requirements of the specified task, then it simply cannot be used. This is one of the main motivations of DESOT, *i.e.*, that it limits the computational resources needed to run the model compared to a traditional deep ensemble.

Assume that running inference on an input x requires T computations when run on a single model. Then, performing inference on that same input will require MT computations for a DE with M members, meaning that the computational footprint scales linearly with the number of ensemble members for a DE. This is a problem in the AD space since all computations have to be performed in the car in real-time with minimal latency. Furthermore, there are many tasks that have to be performed by the system at each timestep other than just traffic sign recognition, which means that each subsystem has even stricter limits for computational footprint. This motivates the investigation of if DESOTs that only perform inference on one model each timestep can perform well while limiting the computational footprint of the ensemble.

3.4.3 Evaluating predictive performance

In connection to Research questions A and B, the main evaluation criteria for predictive performance is to evaluate the different models on the sequences and compare

them in terms of F1-score. As previously mentioned, TSR is a domain that typically has many classes and unequal class distribution. Since all classes are important, evaluating the performance of DESOT explicitly on rare classes is of value since the model performance on these is otherwise made ambiguous by the majority classes. Additionally, a key aspect of answering research question B is the effect of ensemble size and the length of the sequences on the observed performance of the models.

3.4.4 Uncertainty quantification and the difficulties of measuring calibration

Regarding Research question C, a key point of interest is to investigate if the high UQ performance of deep ensembles that have been noted by a number of authors, [6], [12], [13], extends to DESOT. In connection with the same research question, the effects of temperature scaling on the calibration of single models and ensembles are of interest.

A discussion regarding the difficulty of measuring and quantifying calibration is in order. As is accounted for in section 2.2, finding the calibration of a model requires knowing the long-term distribution $\nu(p)$ of prediction p for each class c as well the long-term distribution $\rho(y = c|p)$ of the probability of the class being true given the model prediction. In theory, these distributions should be stationary, and an infinite number of observations should have been made. In practice, of course, these assumptions never hold, and so we have to make do with empirical estimates. These estimates are in practice made by binning the predictions into K bins, which is necessary since the number of observations is limited.

Metrics based on reliability diagrams such as ECE or MCE only take the probability assigned to the top class into account, what we have chosen to refer to as confidence. Good calibration by these metrics only means that the confidence is calibrated, not the probabilities assigned to the other $C - 1$ classes. The uneven class distribution in TSR means that confidences for minority classes are sparse, meaning the quality of the empirical estimate of the aforementioned distributions is poor, which only compounds the issue. The equal weighting of confidence buckets despite varying population sizes also worsens the problems. For a review of issues with ECE, see subsection 2.2.2. For real-life applications such as TSR, higher requirements are usually placed on the outputs from AI models. This includes that not only the top class confidence should be calibrated, but also the probabilities assigned to the other $C - 1$ classes [39]. Brier reliability, as implemented for this thesis, takes these other $C - 1$ class probabilities into account. This means that for some classes, the number of samples that can be used for estimating calibration increases from less than ten to the full size of the dataset compared to if only confidence is used for measuring calibration. This raises the validity of the calibration estimates included in this thesis. However, due to the widespread use of ECE, results in ECE are also included but should be interpreted warily.

3.4.5 Evaluating performance on OOD data

An essential aspect of any ML task is to have a generalized model that can handle OOD data. The Zenseact open dataset (ZOD) is used for the project, and more information about it can be found in section 3.3. In the ZOD, there are many examples of traffic signs labeled as NotListed, as well as unclear images that are hard to classify even for a human. These have their own class labels in the ZOD. This allowed us to qualitatively test how well the model(s) perform on OOD data in the sense that we would like the models to exhibit high uncertainty, measured using entropy, for OOD examples. It has previously been shown that deep ensembles perform well for this kind of qualitative OOD evaluation [6], [12]. However, their performance has not been tested when applied to sequences of single frames, and has not been compared to ensembles spread over time. This means that there is a gap in research that is interesting to explore.

Another way of testing the OOD performance of ML models is to use augmentations of varying intensity to see how the accuracy, confidence, and uncertainty displayed by the models change with the augmentation intensity. This approach is employed by Ovadia *et al.* [13], who use a set of augmentations including rotations and blur to test just this. We have chosen to refer to this kind of OOD generation as *progressive* OODness due to the increasing intensity of augmentation, but Ovadia *et al.* [13] refer to it as shifted OOD data. They show that DEs are SotA on this sort of progressive OOD [13]. This kind of comparison of the models on OOD data is interesting since it mimics some of the edge cases that TSR systems might encounter in the real world, such as rotated signs. It also allows for other comparisons to be made, since one of the conclusions of Ovadia *et al.* [13] is that in-distribution UQ performance is not a good indicator of OOD UQ performance. They found this to be especially true for MC-dropout and temperature scaling of single models. What they did not test was the performance of temperature scaling applied to ensembles or MC-dropout. We employ this same approach to test the behavior of the models on progressive OOD in order to characterize their behavior, as a complement to their performance on complete OOD. We also extend the literature by extending the discussion to sequence data and DESOTs.

4

Empirical Findings

To answer the research questions, a set of experiments have been performed. In this chapter, the experimental setup for these experiments, as well as the empirical findings, are presented. After the results of the experiments for one aspect of model performance have been shown, these results are discussed.

4.1 Experimental setup

In this section, the experimental setup that was used to obtain the results is introduced. The training and evaluation procedures are also accounted for.

4.1.1 Choice of model architecture and size

All models used for this thesis are based on the ResNet [26] CNN architecture. There were a few reasons for this choice. First of all, most literature testing the performance of deep ensembles include variants of this model architecture, *e.g.* [12], [13], [38]. The influential paper by Guo *et al.* [1] that introduced temperature scaling as a means to combat the overconfidence of modern neural networks also used a variant of this model for empirically proving their claims. Thus, its use in the closest related literature makes it suitable for our investigation. From a practical point of view, the ResNet models have been shown to provide great performance for image classification on many datasets, including ImageNet and CIFAR-10 [26]. They are also known to work well for TSR [17], [18], [51].

A small study was conducted in order to decide what size of ResNet model to use (see section A.2 in the appendix), the conclusion of which was that the performance difference between ResNets of different sizes is negligible for the chosen problem. Therefore, the smaller ResNet18 version was chosen as the final base model for all experiments. MC-dropout was implemented on the same ResNet18 model architecture as the vanilla models, but with an additional dropout layer after each non-linearity that was active during testing. A dropout rate of 0.2 was used. Due to time constraints, the dropout rate was not tuned to achieve optimal performance.

4.1.2 Training and evaluation

All models are trained from scratch on the single-frame training dataset described in subsection 3.3.2 using a base learning rate of 0.0005 on the AdamW optimizer [52]. The PyTorch implementation of cosine annealing, first introduced by Loshchilov and Hutter [53], is used to schedule the learning rate to progressively decrease during training for faster and more stable convergence. A batch size of 256 is used. The NotListed and unclear classes are excluded from the training set, along with classes with fewer than 10 occurrences in total between the training and validation datasets. Crops that are smaller than 16 pixels along the smallest dimension are also excluded from the data. In line with the definition of deep ensembles by Lakshminarayanan *et al.* [6], all members were trained separately on the same data with random weight initialization. The optional adversarial training was not employed. The cross-entropy loss function was used since it's a proper scoring rule.

In total, 25 independent vanilla ResNet18 models were trained, along with five MC-dropout versions of the same. In order to ensure statistically sound performance estimates, all 25 models were run to obtain single model performance, and five independent 5-member ensembles were created, as well as five 10-member ensembles. For the evaluation of MC-dropout, all 5 models were separately evaluated.

Here follows a glossary of the notation used for the different models throughout the rest of the report. When evaluated on the sequences dataset, all models use a combination rule of simple averaging for combining predictions across frames.

- **$\mathbf{DE_M}$**: This denotes an M-member deep ensemble operating on an image. When evaluated on the sequences dataset, DE_M means that a full M-member deep ensemble is run on each frame.

- **$\mathbf{DESOT_M}$**: This denotes an M-member deep ensemble spread over time, where each member operates on a different frame in the sequence. For a more formal definition of DESOTs, see subsection 3.4.1.

- **Single model (SM)**: This denotes a standard single ResNet18-model. Note that this is conceptually the same as a one-member ensemble, and is therefore equivalent to both a DE_1 and a DESOT_1.

- **MC-dropout**: A single ResNet18-model trained and evaluated with an extra dropout layer after each non-linearity. The implementation used is similar to a single model in the sense that one forward pass is conducted for each time step in the sequence. The outputs are then averaged across time.

- **+ T**: A suffix added to any of the previous model names that is used to denote that temperature scaling has been applied to that model.

Table 4.1: Predictive performance for each model tested on the sequence dataset in terms of accuracy and F1-score. The results include plus and minus one standard deviation of performance between runs.

Model	Accuracy	F1-score
SM	0.9734 ± 0.0006	0.8112 ± 0.0175
$DESOT_5$	0.9760 ± 0.0003	**0.8326 ± 0.0093**
DE_5	**0.9764 ± 0.0001**	0.8273 ± 0.0101
MC-dropout	0.9710 ± 0.0009	0.7679 ± 0.0271

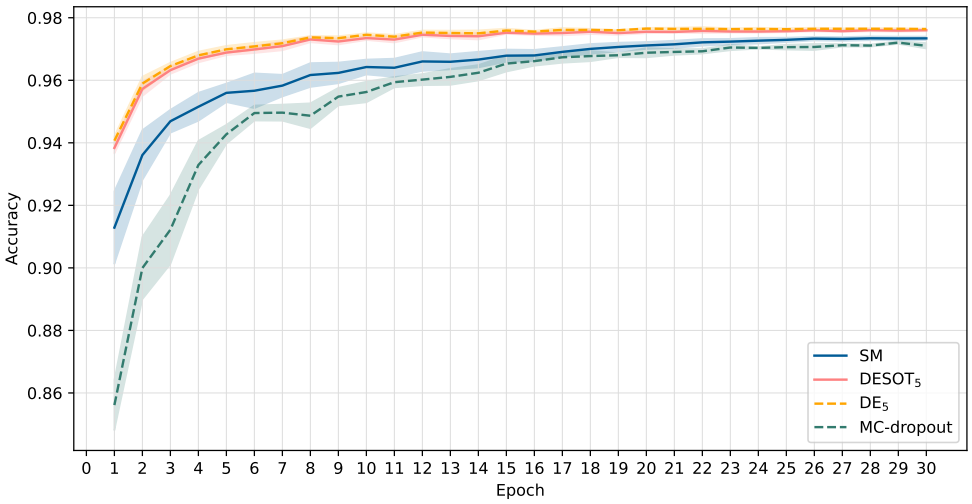


Figure 4.1: Graph comparing the predicting performance on the sequences dataset in terms of accuracy of a 5-member DESOT with a 5-member DE, a single model, as well as a MC-dropout model. The error bars are drawn for ±1 std. The ensemble spread over time ($DESOT_5$) performs on par with the deep ensemble (DE_5) despite requiring only 20% as much computation, while outperforming the single model.

4.2 Predictive performance

We evaluate the predictive performance of the different models on the sequence dataset. Note that temperature scaling does not affect the ordering of the predicted classes, and thus not the final model prediction. Therefore, results for temperature scaling are omitted for this part of the results. As can be seen in Figure 4.1, all models reach high performance as measured in accuracy for the later epochs, $DESOT_5$ and DE_5 performing markedly better than the other models at early epochs. Notably, $DESOT_5$ performs on par with traditional DE, and these two models are slightly better than other models, even after single model convergence. Final-epoch performance is summarized in Figure 4.2. Compared using F1-score, the difference between models is greater in absolute terms, with DESOT performing about as well as DEs. For both metrics, MC-dropout performs decidedly worse than other models.

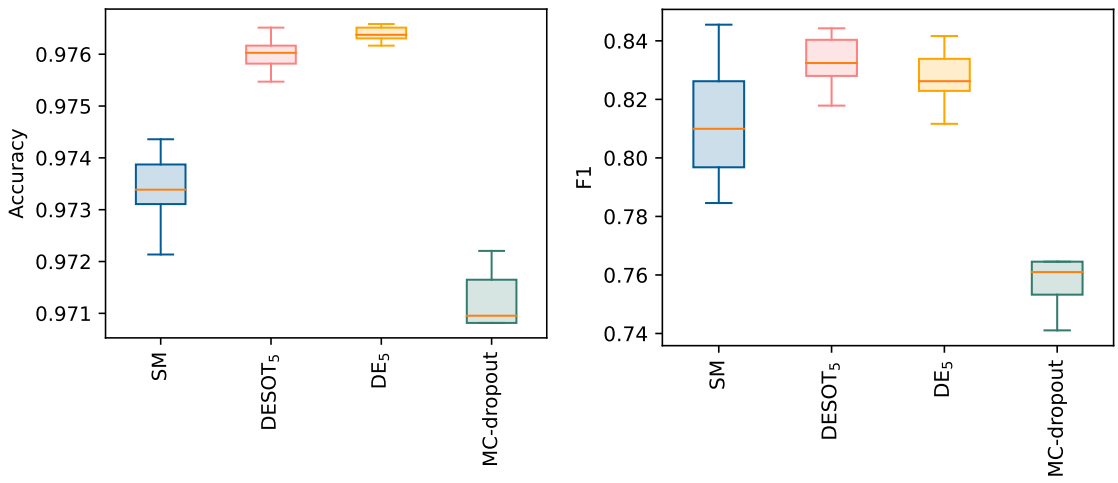


Figure 4.2: Final-epoch predictive performance on the sequences dataset, comparing DESOT_5 with a single model, DE_5 and MC-dropout. The DESOT_5 performs about as well as the DE_5 on accuracy and F1-score, and both of these perform better than the single model and MC-dropout. Again, note that DESOT_5 uses the same amount of computations as a SM or MC-dropout.

4.2.1 Evaluation on classes with few samples

Because of the high performance of DESOT when measured using F1-score, which weighs performance on all classes equally, an additional experiment on rare classes was conducted. This was done by filtering out any classes with more than 500 occurrences in the training and validation datasets, which resulted in 625 sequences. The remaining classes are what we consider *rare*. These were used to evaluate the models on rare class performance. The predictive performance results for this subset of the sequences dataset are shown in Figure 4.3. DESOT performs very favorably in this comparison, outperforming all other models, including DE. MC-dropout performs by far the worst. Just as for the predictive performance on the whole sequences dataset, the ensembles decrease the variance of the model performance compared to single models.

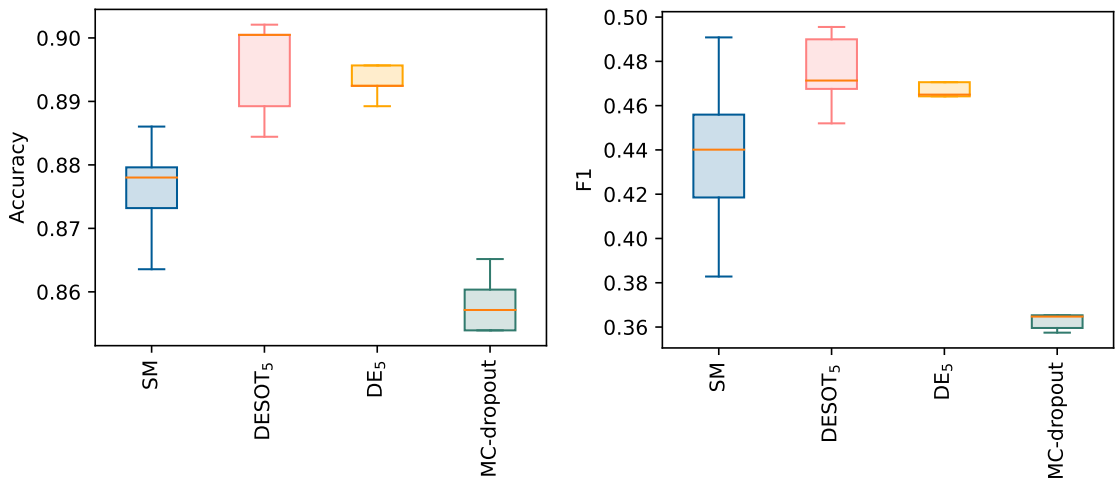


Figure 4.3: Final-epoch predictive performance on a minority class version of the sequences dataset, comparing DESOT_5 with a single model, DE_5 and MC-dropout. The DESOT_5 outperforms the DE_5 on both accuracy and F1-score. Additionally, it outperforms single models and MC-dropout by a large margin in both metrics.

4.2.2 Discussion of predictive performance

For the task of classification of traffic signs, one of the most important aspects is the predictive performance, or how well the model can classify the different signs. The results displayed in Figure 4.1 indicate that TSR might be a task that is easy to achieve high levels of predictive performance on. It can be observed that all models, including single models, achieve an accuracy exceeding 97%. However, the results also show that our model, DESOT_5, performs very well compared to the baseline models. The lower baseline, SM, achieves a slightly lower accuracy than our model. The upper baseline, DE_5, achieves a marginally higher accuracy than our model, but inside one standard deviation.

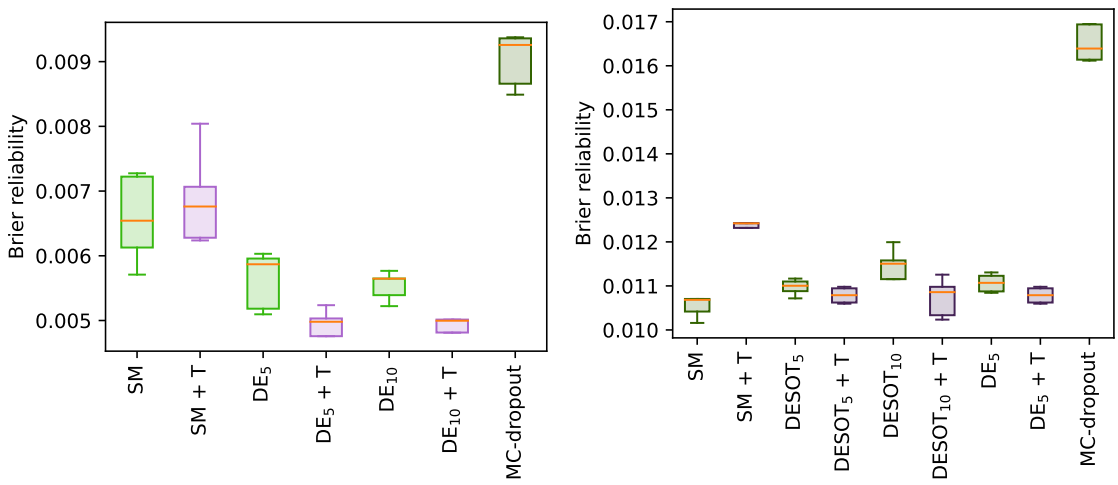
Looking at the results in terms of F1-scores, where the performance on minority classes is weighted higher somewhat changes the story, with larger absolute differences in performance between models. DESOT_5 and DE_5 both have significantly higher average scores than single models and MC-dropout. The single models display a large variance in performance, while both ensembling techniques have a smaller variance. This seems to suggest that performance in classes that are rare occurrences in the training data is where the main predictive performance benefits of DESOT over single models might lie. In Figure 4.3, the DESOT model seem to outperform the standard DE, this is unexpected as both models have the same information in the sequences. One reason for this might be due to the inherent noise of the prediction on rare classes, while the ensemble will average the prediction from more inferences, the DESOT will not be as diluted, resulting in higher probabilities for the correct rare class.

MC-dropout is an interesting method to compare against as the dropout creates a slightly different model on each forward pass, which is a sort of ensembling, as shown by Gal and Ghahramani [44]. The MC-dropout model does not perform as well as the other models. This might be due to a too-high dropout rate, a hyperparameter that was not thoroughly tuned to maximize performance.

All in all, these results show that our method performs significantly better than a single model that has a similar computational footprint and equivalently to a DE_5, which has a computational footprint five times larger than our model. The reason for the increased performance over single models might be that the members in an ensemble together have a more expressive representation of the space of possible traffic signs than a single model. The benefit of DESOTs is that they allow for using this more expressive representation while limiting computations. Still, the extra computations performed for DEs do not seem to benefit predictive performance. Perhaps this is because of the relative simplicity of the traffic sign classification task, which means that the extra computations of DE yield minimal benefits. If these results can be replicated for other tasks and datasets, this is a significant finding.

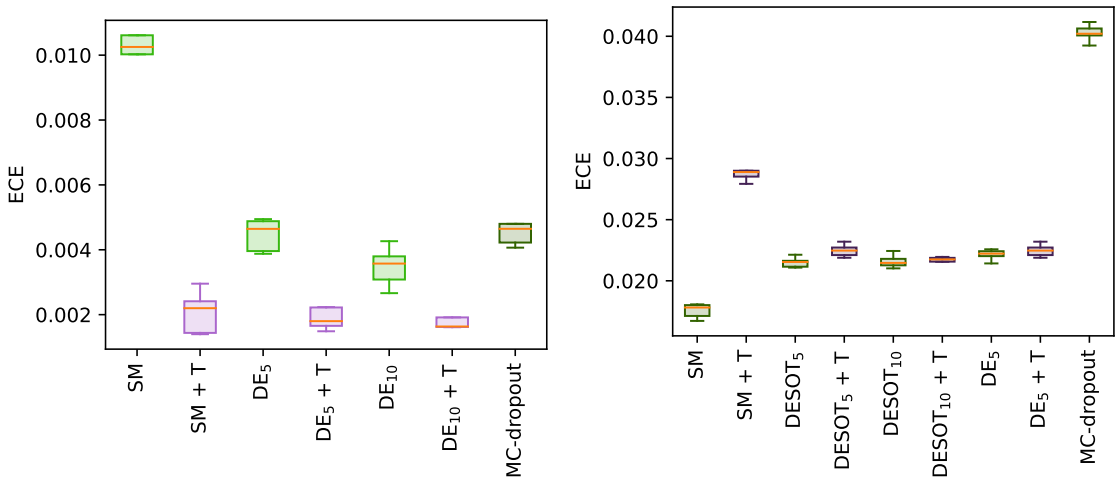
4.3 In-domain uncertainty quantification

When considering the results in this section, keep in mind the discussion regarding the difficulties in measuring calibration in subsection 3.4.4. Now, for in-domain uncertainty quantification, all models are not only compared against each other but also against their temperature scaled counterparts. The temperature scaling is optimized on the single frame validation set. In section A.3 in the appendix, additional results and observations about temperature scaling are presented. The results for in-domain calibration measured in Brier reliability are shown in Figure 4.4. The results when measured in ECE are shown in Figure 4.5.



(a) Single-frame test dataset. **(b)** Sequences dataset.

Figure 4.4: Uncertainty quantification performance for each model on in-distribution data measured in Brier reliability. Lower is better. Temperature scaling seems to significantly improve the calibration for ensembles both on the test dataset and sequences dataset. For single models, it instead seems to worsen calibration. Overall, MC-dropout is the worst calibrated out of all the models.



(a) Single-frame test dataset. **(b)** Sequences dataset.

Figure 4.5: Uncertainty quantification performance for each model on in-distribution data measured in ECE. Lower is better. Temperature scaling seems to significantly improve the calibration for both single models and ensembles on the test dataset. However, for the sequence dataset, temperature scaling seems to increase ECE. Overall, MC-dropout is the worst calibrated out of all the models.

4.3.1 Discussion on in-domain uncertainty quantification

The calibration of our models was measured using the Brier reliability, as detailed in subsection 3.4.4. This score is a good way of getting a measurement of calibration compared to ECE or MCE. Figure 4.4 shows how well the different models performed when estimating their own uncertainty. For the single-frame test dataset, the Brier reliability decreases for ensembles with more members, meaning that deep ensembles are better calibrated than single models by default. This confirms observations made by Ashukha *et al.* [12]. Temperature scaling also seems to improve the calibration of all models in terms of Brier reliability except for some single models. It is difficult to say why this is. In terms of ECE, all models are better calibrated with the addition of temperature scaling. This is consistent with the claims about the benefits of temperature scaling made by Guo *et al.* [1].

On the other hand, when testing on the sequences dataset, all models perform worse in absolute terms, and the Brier reliability score seems to increase with more members in the ensemble. For the single model, the temperature scaling is much worse than the standard model on both metrics. For the ensembles, temperature scaling results in a slight improvement. MC-dropout is not well calibrated on either dataset when considering Brier reliability and is the worst calibrated model. When looking at ECE, MC-dropout is better calibrated than the vanilla single model but is outperformed by its temperature scaled version.

We chose to temperature scale our models on the single frame validation set. This meant that each DESOT is temperature scaled jointly as a DE on single frames. The hope was that the calibration results would generalize from single frames to sequences. This would be helpful since labeled single-frame data is less expensive to produce than labeled sequence data. However, the results are somewhat unclear on this point, and overall, the results on in-domain uncertainty quantification are somewhat inconclusive. DEs are better calibrated than single models on the single-frame data, which supports previous work [6], [12], [13], but this does not seem to generalize to sequences with the DESOT models. The reason this might be the case is unclear but could be a result of the images in the sequences being different combined with the models of the ensembles being slightly different, the combination of which results in worse calibration. One thing that supports this reasoning is that the SM has a higher Brier reliability score on the sequences dataset than on the single-frame test set. The only difference between them is the averaging across a sequence instead of interpreting each frame by itself. Otherwise, the data is the same images as the last frame in the sequence. The other part of the result that hints that there is a combination of effects is that the Brier reliability score increases as the number of members in the DESOT increases, unlike the DE models.

4.4 Out-of-distribution uncertainty quantification

In the real world, an ML system will often encounter data that was not part of the training data, commonly referred to as out-of-distribution (OOD) data. Therefore, it is of great importance to test how the system behaves on such data. Especially, since it is common for machine learning systems to fail silently, giving high confidence for OOD examples that it has never seen before [2]. Because of this, we perform some experiments to characterize the behavior of the different systems. This is done in two ways: testing performance for gradual OODness by way of augmentations similar to Ovadia *et al.* [13], and by testing on fully unseen data, similarly to Lakshminarayanan *et al.* [6]. Testing for gradual OODness for a system in employment is useful since it allows for testing of common scenarios that might be encountered, such as rotated signs, the amount of daylight, or other variations in conditions. Importantly, for complete OOD data, sizable changes in output distribution shape compared to in-distribution data allow for OOD detection. This has the practical effect of allowing downstream subscribers of the model output to adjust their course of action.

4.4.1 Experiments on gradually augmented OOD data

Augmentations were used to generate data that is gradually more out of distribution as the severity of the augmentation increases. The concept of increasing the augmentation intensity was first introduced by Hendrycks and Dietterich [54], who introduced the ImageNet-C and ImageNet-P datasets that can be used to test for model robustness to various corruptions and perturbations. A similar approach was later employed by Ovadia *et al.* [13]. We used the same idea but with some changes in the effects used. The test consisted of eight different augmentations where the intensity was varied on an interval specific to the augmentation. The augmentation ranges and step sizes for each augmentation are displayed in Table 4.2.

Table 4.2: Augmentations and values used for OOD data creation. Brightness, saturation, and contrast all gradually decrease from their original values with increasing intensity.

Augmentation	**min**	**max**	**step size**
Rotation	0°	360°	15°
Brightness	0 %	100 %	5 pp
Hue	0 %	100 %	5 pp
Contrast	0 %	100 %	5 pp
Motion blur	0 px	60 px	3 px
Gaussian noise	0 %	100 %	5 pp
Shear	0°	180°	10°
Saturation	0 %	100 %	5 pp

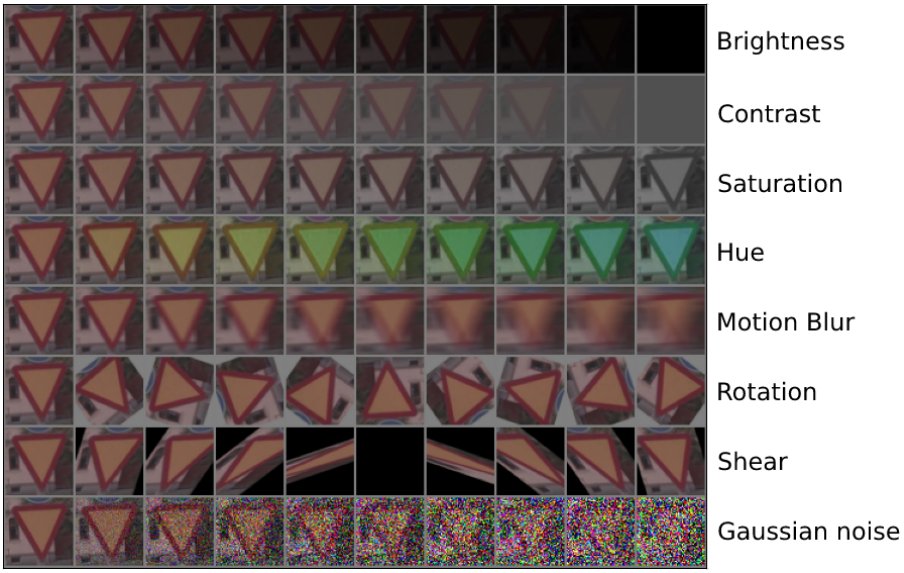


Figure 4.6: Illustration of the different augmentations used at various intensities, from no augmentation to maximal intensity.

The augmentations chosen can be seen in Figure 4.6 with varying intensity. These augmentations stem from some variations that could occur naturally in an AD system, where weather conditions or physics affect the image quality. Rotation can be seen if the road sign is placed at an angle to the car, brightness, contrast, and saturation could all be the result of bad weather such as rain or fog. The image can also be affected by the movement of the vehicle and the shape of the lens resulting in motion blur, shear, and color separation. This phenomenon can be seen in some frames of the ZOD dataset.

For this experiment, we expect model accuracy to decrease as the intensity of the augmentations increase. Ideally, we want the models to adjust their certainty to match this decrease in accuracy, such that calibration is maintained. If this is the case, one can trust the model uncertainty estimates, increasing the amount of actionable information available to downstream users. However, for high degrees of augmentations, calibration is not necessarily an informative way to measure uncertainty quantification performance. This is because calibration relies on the model being able to correctly classify the sequence, which is often not possible for high levels of augmentation. When the data becomes more out-of-distribution it instead becomes more relevant to be able to classify this as an OOD sample instead of having well calibrated class distribution, as the ground truth class loses its relevance. Thus, we have also included the mean entropy across the dataset for each model and augmentation severity.

Figure 4.7 shows how the different augmentations affect the accuracy, Brier reliability, and mean entropy of the models on the sequences dataset for various intensities. Results for the test frames dataset are omitted for the sake of brevity. For Brier reliability, a lower number means the model is better calibrated. Higher mean entropy suggests higher model uncertainty. The augmentations brightness, contrast,

and saturation have a stronger augmentation effect as the value decreases, where for example a brightness of 100 is the original image and 0 is a black image. The subplots for these augmentations have inverted x-axes to align all plots to have higher augmentation intensities towards the right.

Comparing the graphs for the accuracy and Brier reliability we can see that as the accuracy of a model decreases, the Brier reliability score increases, meaning that the models are worse calibrated the more OOD the data is. We also note that entropy seems to increase as the intensity of augmentation increases and the accuracy decreases. The odd shape of the graphs for the rotation augmentation is due to many of the signs having a rotational symmetry of less than 360°. Also note that a 90° shear results in a completely black image, which is the reason for the strange shape of the graphs in the resulting subplot.

It seems as though no model is clearly superior to any other, although MC-dropout is an outlier in some respects, achieving better calibration for strong augmentations. Single models are outliers in the other regard and have clearly worse performance than the other models tested. For example, it successfully maintains its calibration with decreasing brightness as other models fail, but it is also worse than many other models when it comes to maintaining calibration upon increased saturation. All models except MC-dropout seem to perform overconfident guesses on completely uninformative images, such as for extreme contrast and brightness augmentations. This is also reflected in MC-dropout producing predictive distributions with higher mean entropies than the other models for a given augmentation intensity. Overall, temperature scaling seems to help most for single models, which holds for all augmentations. Ensembles seem to be better calibrated than single models, but they still somewhat benefit from temperature scaling in terms of better calibration and higher entropy at high intensities of augmentation.

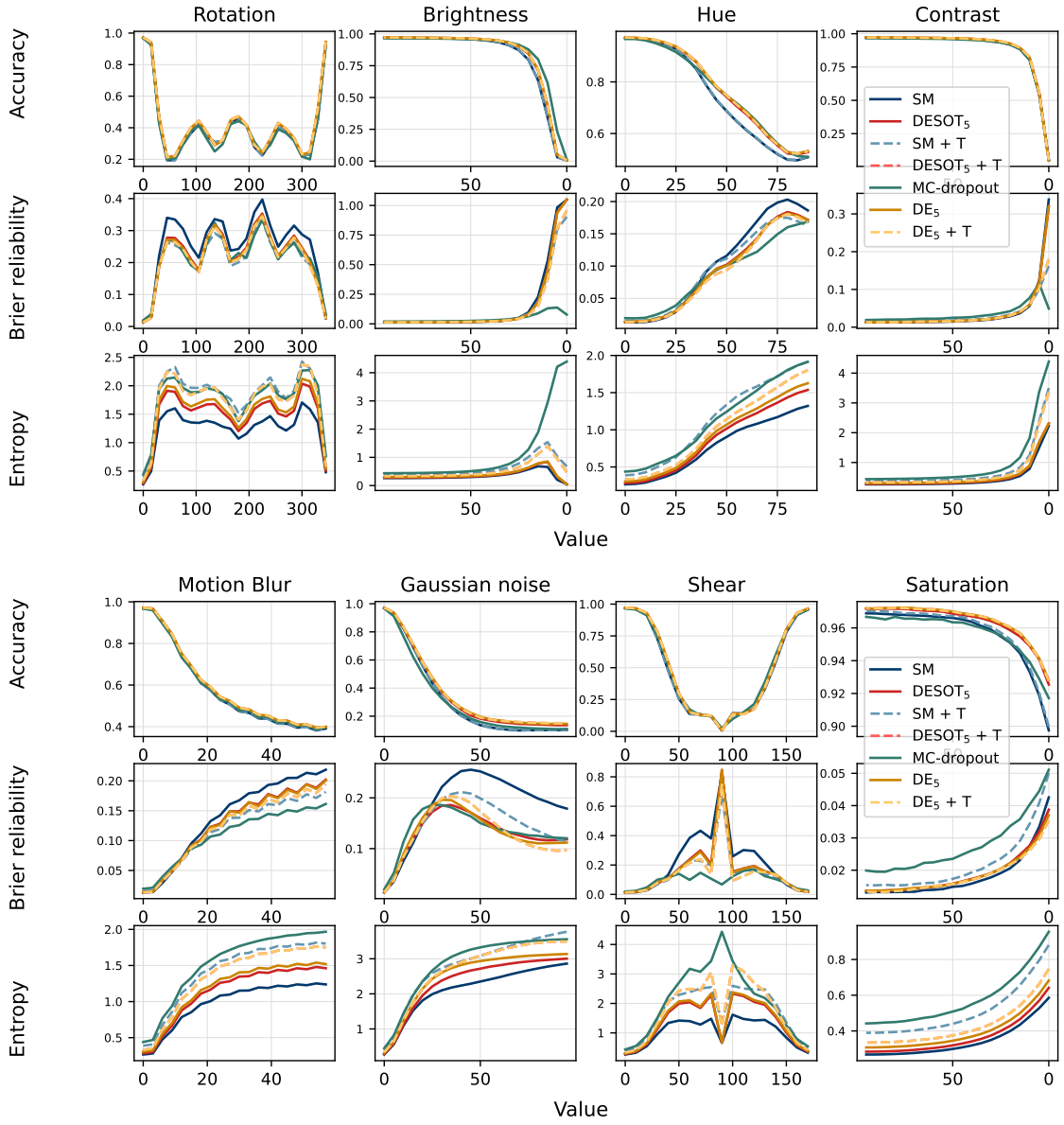


Figure 4.7: Uncertainty quantification performance for each model on augmented data of increasing intensity. The performance is measured in accuracy, Brier reliability, and mean entropy. Lower Brier reliability is better. Tested on the sequences dataset. No model is clearly better or worse, though single models and MC-dropout are outliers in some respects.

4.4.2 Discussion of performance on gradually augmented OOD data

From the results on the augmented OOD data, it seems that no model is more robust to the augmentations when it comes to predictive performance; they all deteriorate at about the same rate for increased intensities. Hendrycks and Dietterich [54] has previously noted that ResNets are very sensitive to perturbations and corruptions compared to other architectures. When it comes to the calibration of the models upon augmentation, there is more variation. Single models seem to be the worst at maintaining calibration at higher augmentation intensities, especially for rotations, hue, motion blur, and Gaussian noise. Ovadia *et al.* [13] saw similar results on rotational augmentations. Temperature scaling seems to help improve the calibration of single models for most augmentations, which is consistent with the results by Ovadia *et al.* [13]. However, we note that for rotational augmentations, temperature scaling seems to improve the calibration of single models to the level of DEs. This is not consistent with what is observed by Ovadia *et al.* [13]. Of note is that the datasets used vary greatly between their and our experiments, with Ovadia *et al.* [13] performing their experiments on the simple MNIST dataset, while our data is more complex. Additionally, we deal with sequences, and not single images.

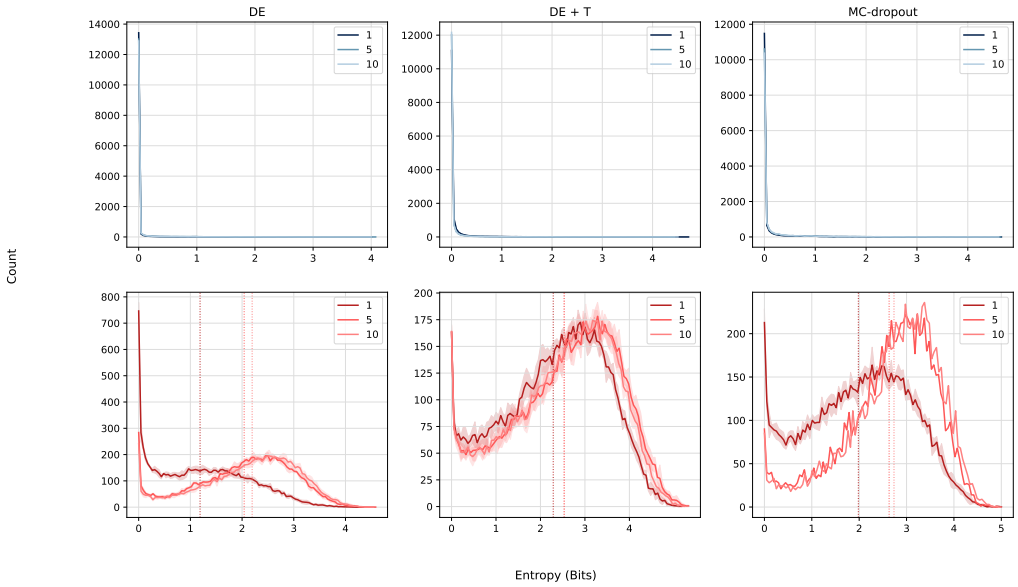
It generally seems as though DEs and DESOTs perform about the same in calibration measured in Brier reliability for any given augmentation intensity, and both perform better than single models. However, after temperature scaling, the difference between the ensembling methods and single models is not at all as great. The difference between ensemble and single models really manifests itself in the Gaussian noise corruption, with single models more clearly than other models failing to maintain calibration.

4.4.3 Experiments on complete OOD data

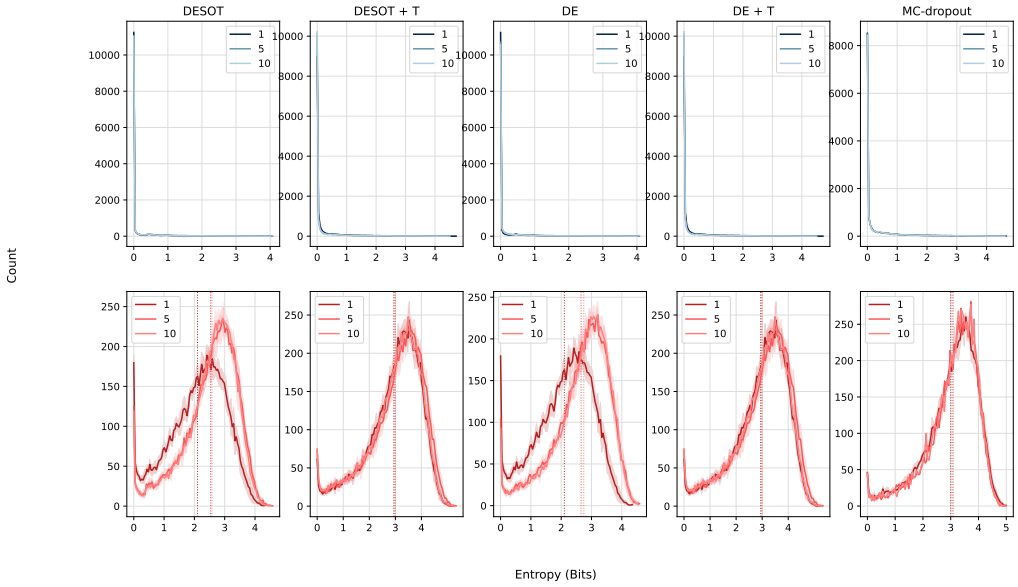
The ZOD dataset contains the class NotListed, a class of traffic signs that do not belong to any other class. These signs are for example destination signs, location signs, parking signs, general information, and other special or rare signs. For testing the behavior of our models on OOD data, using the NotListed class gives a set of mixed traffic signs that are not a part of the training set, and are thus unseen by the models.

As the model has not been trained on a class for the OOD data, metrics such as accuracy, F1-score, Brier score, or ECE naturally can not be used since they all rely on the model being able to predict the ground truth label. Instead, entropy is used (see Equation 2.11) and is assessed qualitatively. If the model's uncertainty increases, we expect the output distribution to be flatter, and the entropy to be higher. What we would like to observe on this sort of data is an increase in the entropy of the predictive output distributions of the models as compared to the in-distribution data, as a measure of if the models *know what they do not know*. To display the entropy in a graph, the entropy values for the test samples are divided into 100 equally sized bins, and the number of samples in each bin is counted.

In Figure 4.8, the results are presented in the form of a set of graphs, where each histogram contains the entropy scores for one model type of a varying number of members. The x-axis is the entropy in bits and the y-axis is the count of samples in the corresponding bin. The top graphs, in blue, are the results of the in-distribution data and show a big spike at 0 in entropy, and few samples with higher entropy. This is true for all models, regardless of the dataset. The bottom graphs, in red, show the results of the OOD NotListed data and shows a clear increase in entropy for all models compared to the in-distribution data. Table 4.3 has a summary of the average entropy for the model architectures on OOD data for various ensemble sizes.



(a) Single frame test dataset.



(b) Sequences dataset.

Figure 4.8: Entropy for OOD-data (red) compared to in-distribution data (blue) for the single-frame test and sequence datasets. The tests are run for various ensemble sizes $M \in \{1, 5, 10\}$, which are differentiated by color shade. Again, note that DE_1 and DESOT_1 are special cases, which are in effect both a single model (SM). The vertical dashed lines are the mean entropy for the model of the same color.

Table 4.3: Mean entropy on OOD data (NotListed class).

Sequence frames	**1 member**	**5 members**	**10 members**
DE:	1.187	2.040	2.195
DE + T:	2.287	2.534	2.532
MC-dropout:	1.982	2.632	2.735
Sequences:			
DESOT:	2.103	2.533	2.577
DESOT + T:	2.942	2.997	2.978
DE:	2.103	2.656	2.745
DE + T:	2.942	2.997	2.978
MC-dropout:	3.003	3.082	3.082

In Figure 4.9 the entropy of different ensemble techniques for in-distribution and OOD data is shown. All ensembles have 5 members, and temperature scaled versions are included. Note that this is the same information that is available in Figure 4.8, but presented in a way that allows for more direct comparisons between model architectures. For in-distribution data, all models show sharp predictions for both single-frames and sequences, with most predictive output distributions having close to zero entropy. For OOD data, all models successfully increase their average entropy significantly. On the sequence frames dataset, DE_5 with temperature scaling is the model with the highest average entropy, but MC-dropout has the smallest highly confident tail. The DE_5 with temperature scaling and the $DESOT_5$ with temperature scaling have an almost identical distribution of entropy, to the point where the curves are exactly overlapping. This is higher than for both their corresponding versions without temperature scaling. However, MC-dropout is the model with the highest average entropy.

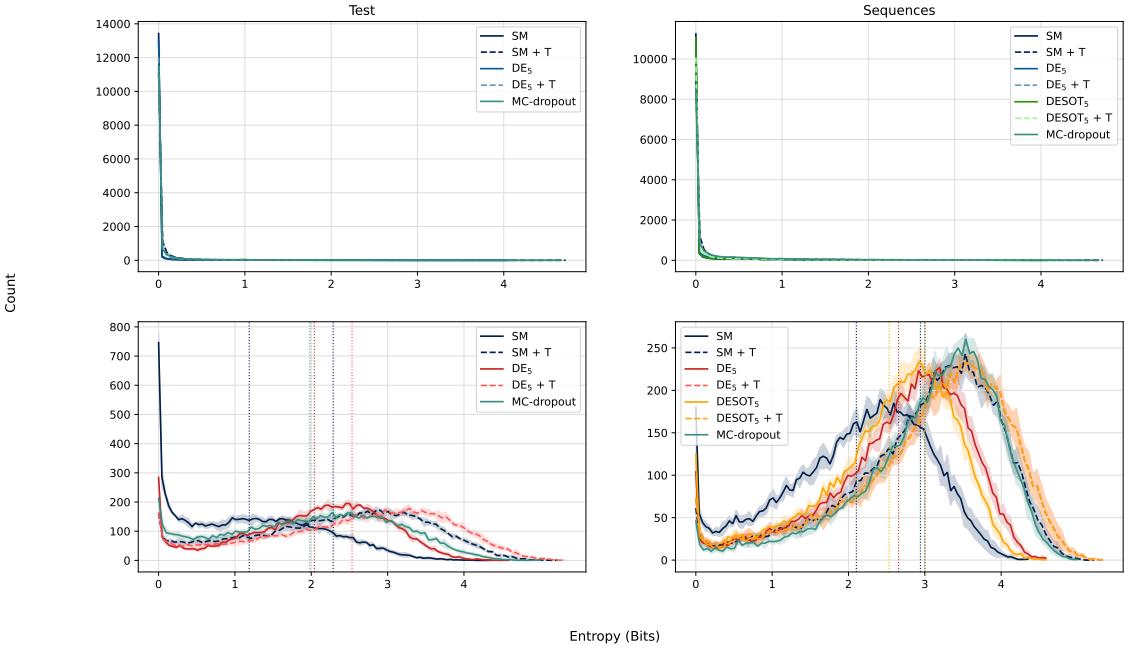


Figure 4.9: Entropy for in-distribution data (top row) compared to entropy for OOD data (bottom row) for the single-frame test dataset (left) and the sequences dataset (right).

OOD detection using a thresholding strategy

To test the potential for OOD detection for each model, a simple entropy thresholding strategy on output distribution entropy was used. This approach rewards methods with clearly separated in- and out-of-distribution entropy distributions. A threshold was fitted to each of the models using half of the available sequences for the binary classification problem of detecting in- or out-of-distribution samples. The OOD detection performance for each of the models is evaluated on the other half of the sequences dataset. These results can be seen in Table 4.4, where the entropy threshold value as well as accuracy, precision, recall, and F1-score are presented.

Table 4.4: Table of results from applying an entropy threshold for OOD detection on the sequences dataset.

	Threshold	Accuracy	Precision	Recall	F1
SM	0.709	0.895	0.811	0.905	0.855
SM + T	1.463	0.916	**0.856**	0.906	0.880
$DESOT_5$	1.048	0.919	0.852	0.922	0.886
$DESOT_5$ + T	1.143	0.919	0.850	**0.928**	0.887
DE_5	1.071	0.919	**0.856**	0.919	0.886
DE_5 + T	1.178	**0.920**	0.853	0.925	**0.888**
MC-dropout	1.428	0.910	0.845	0.900	0.872
MC-dropout + T	1.610	0.914	0.842	0.921	0.880

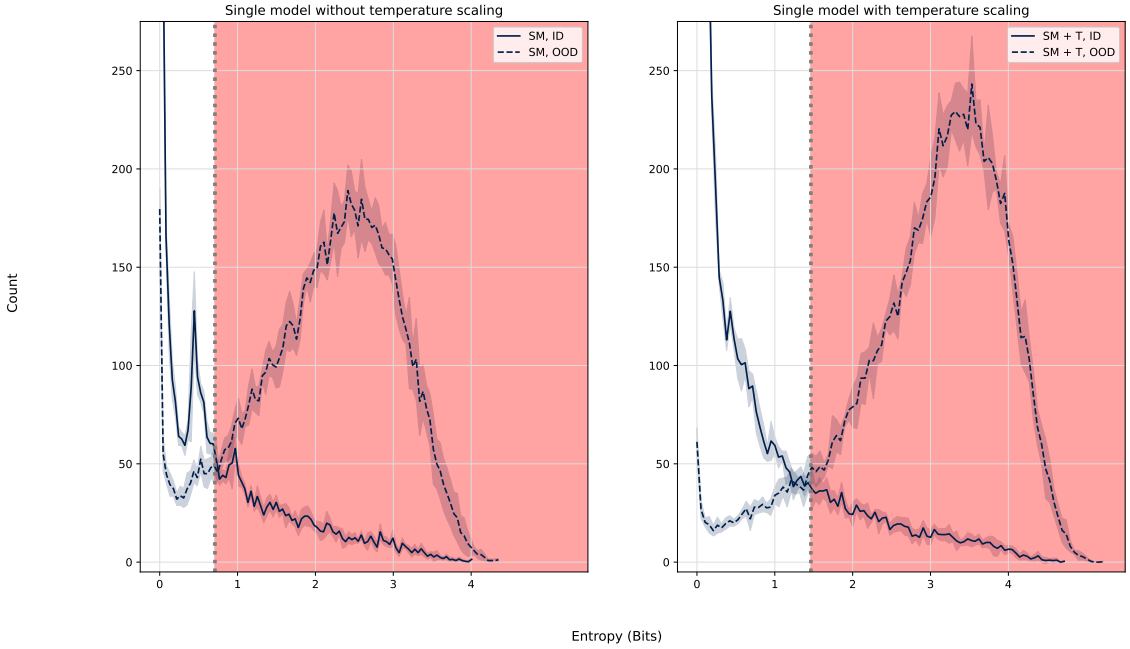


Figure 4.10: Illustration of the thresholding strategy applied to the entropy of a single model with and without temperature scaling. The solid line is the in-distribution entropy, and the dashed line is the OOD entropy. The vertical gray dotted line is the optimal threshold for that particular model, which is the same as in Table 4.4. All data points to the right of the threshold, inside the red area, are classified as OOD. Note that the increased separation between the in- and out-of-distribution lines for the temperature scaled version allows for higher OOD-detection performance.

Even though the value for the optimal threshold varies widely across models, the OOD detection performance is quite similar across models. Across all metrics, ensembles outperform single models. Temperature scaling gives a marginal increase in classification performance for most models, except for the single model where it helps increase OOD-detection performance a lot. This can be seen in Figure 4.10, where the thresholding strategy is visualized for a single model with and without temperature scaling. DESOT performs on par with a standard DE on all performance metrics.

4.4.4 Discussion of model performance on complete OOD data

Overall, the entropy of the output distribution of all models drastically increases on the OOD data compared to the in-distribution data. The single model has a quite flat histogram of entropies with a large spike at 0 on the single-frame test dataset. This shows that the vanilla single model is able to somewhat adjust its uncertainty on OOD data, but generally not that well. The DE_5 and DE_{10} models show a big increase in entropy over the single model on OOD data, which confirms the previous research [6], [12]. Something notable is that the difference between 5

Figure 4.11: Examples of images with low entropy from the NotListed class. These are very similar to the in-distribution data and may be incorrectly annotated.

and 10 members is not as big as the difference between 1 and 5 members, which seems to suggest that there is a saturation effect after some number of ensemble members.

On the sequence dataset, DE shows slightly higher entropy on OOD data than DESOT for a given ensemble size, but this difference disappears when the two models are temperature scaled. Overall, the ensembles get a small increase in entropy from around 2.5 to 3 bits on average with the addition of temperature scaling. The more surprising result is that the single models have an entropy distribution that is almost identical to the ensembles with an increase from 2 to 3 on average. MC-dropout also increases its entropy in line with DEs and DESOTs, even when a single dropout model is run on each frame. This is not in line with the conclusions drawn in Lakshminarayanan *et al.* [6], who instead observe that DEs have a far better separation between output entropy on in and out-of-distribution data than MC-dropout.

The results on OOD detection show that high predictive performance can be achieved by using a simple thresholding strategy. This serves as a baseline for other, more advanced OOD detection systems, but also shows that the models do not need to be trained on NotListed data in order to gain a high accuracy in detecting those signs. One advantage of this approach compared to simply training on the NotListed class is that other kinds of OOD data can be detected, such as those that might be produced by mistakes in an upstream object detection or tracking system.

Another thing to note is that there is still a spike at 0 in entropy, especially on the sequence frames dataset, which is generally not seen for ensembles in previous studies (*e.g.*, [6], [13]). This might be due to the dataset and that the OOD data

Figure 4.12: Examples of images in the NotListed class that are similar to in-distribution data. The top row contains samples from the training set (in-distribution) and the bottom row is similar signs that can be found in the NotListed class (out-of-distribution).

contains some samples that are very similar to in-distribution data, as can be seen in Figure 4.11. One type of such signs are the signs for weight, height, and length limitations of road vehicles. These are round with a white background, red border, and black numbers, and are very similar to the road traffic speed signs of many countries. Another example is the signs that use arrows to convey information, such as arrows to show where parking signs apply, lane direction signs that contain arrows, etc. These are sometimes similar to mandatory turn signs. There are also some signs in the NotListed class that are mislabeled and should be in one of the listed classes. Some examples of these kinds of signs are illustrated in Figure 4.12.

Examples of images with high entropy from both in- and out-of-distribution are shown in Figure 4.13. For the OOD data, we can see clear examples of signs that are not similar to images in the training set and are therefore easy to determine as OOD. For the in-distribution data, the images show clear signs where some of them are obstructed or skewed in some way which makes them harder for the models to classify. These are images that are misclassified as OOD by the simple thresholding strategy.

(a) Examples of in-distribution data with high entropy.

(b) Examples of OOD data with high entropy.

Figure 4.13: Examples of images with high entropy from the in-distribution and the OOD data. For the in-distribution data, we see the most problematic images for the model to classify. For the OOD data, these are the clearest examples of images of OOD data.

A simple thresholding strategy seems to be proficient for high-accuracy discrimination between in-distribution and OOD examples when applied to any of the tested models, despite some low-entropy OOD examples and some high-entropy in-distribution examples. This is of course useful in determining if the final prediction made by the model is likely to refer to a known sign or a sign not relevant to the car. This fact increases the amount of information available downstream and might help increase the system's robustness to upstream misdetections.

5

Conclusion

The results of this project have shown that our method, the DESOT, has potential when it comes to predictive performance where the model performs on par with a DE which has been aggregated across the frames of a sequence. This cuts down on the computational requirements for running a DE by a factor of M, M being the number of members in the ensemble. MC-dropout was found to be uncompetitive when it comes to predictive performance with the tested parameters, getting lower results than the single model. However, MC-dropout was not thoroughly tuned to achieve the best performance.

For uncertainty estimation, DESOT performs somewhat similarly to a single model and on par with a DE. We observe a trend of ensembles performing better than single models on single-frame data but this result does not translate to sequences. These results are hard to draw any conclusions from and need further investigation with more data and additional evaluation techniques to obtain clearer results.

On OOD data from the NotListed class, our model shows an improvement compared to the single model, although this difference is erased if temperature scaling is applied. In general, we find that temperature scaling is a good way to increase the difference in entropy between in- and out-of-distribution data, and works well on all models tested.

When shifting the distribution of the dataset using augmentations to generate OOD data, the results vary depending on what type of augmentation is used. For the most aggressive augmentations, the single model performed quite poorly and the MC-dropout worked well, with lower Brier reliability on most OOD data.

In conclusion, our model shows some positive results and few drawbacks. For predictive performance DESOT outperforms the lower single model baseline and performs on par with the upper DE baseline. For OOD uncertainty estimation it performs similarly to both baselines, with MC-dropout outperforming all other models. However, for in-distribution uncertainty estimation, no conclusion can be drawn. We believe that we have shown DESOT to be a promising way of limiting the computational load of DEs while maintaining performance when used for sequence classification, and a worthy candidate for further research.

5.1 Future research

While this project has provided some promising results, the research can be extended with more experiments. New datasets for other tasks might be utilized to see that our results generalize and are not specific to our dataset or task. One such task in the domain of self-driving cars could be classifying moving objects such as cars and pedestrians. It would also be interesting to see how our findings generalize to regression as well. To test for generalization, it would be interesting to test DESOTs with other architectures than ResNets. Perhaps especially, comparing on OOD robustness, since there is some work showing that ResNets are more sensitive to that sort of augmentation than other architectures.

The results on in-domain uncertainty quantification are somewhat vague. Investigating this aspect of model performance further is very important since it is essential to the safe application of machine learning. Importantly, there are other ways to measure calibration than using metrics such as ECE or Brier reliability as estimators. These include p-tests [39] and conformal prediction [55], which are both in some sense more statistically sound than the simpler metrics for point estimations of model calibration. It would be interesting to see how DESOT stacks up to the other models tested when using these ways of measuring UQ performance.

Limiting the computational load is an important aspect to consider when developing methods to be used on any mobile system, and was a primary motivation for this project. The comparison on this parameter was made using a proxy consisting of the number of forward passes made through a single model per timestep. However, this approach does not generalize if models of other architectures are to be compared. Additionally, there might be other factors that influence the runtime in a production system, such as being able to fit all ensemble members into memory. It would therefore be of interest to perform a controlled study of the inference time for the different models in a runtime system to see how the DESOT model compares in the real world.

Bibliography

[1] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On Calibration of Modern Neural Networks," *CoRR*, vol. abs/1706.04599, 2017. arXiv: 1706.04599.

[2] D. Amodei, C. Olah, J. Steinhardt, P. F. Christiano, J. Schulman, and D. Mané, *Concrete Problems in AI Safety*, 2016. DOI: 10.48550/arXiv.1606.06565.

[3] M. H. DeGroot and S. E. Fienberg, "The Comparison and Evaluation of Forecasters," *Journal of the Royal Statistical Society. Series D (The Statistician)*, vol. 32, no. 1/2, pp. 12–22, 1983, ISSN: 00390526, 14679884. DOI: 10.2307/2987588. JSTOR: 2987588. (visited on 01/20/2023).

[4] A. P. Dawid, "The Well-Calibrated Bayesian," *Journal of the American Statistical Association*, vol. 77, no. 379, pp. 605–610, 1982. DOI: 10.1080/01621459.1982.10477856.

[5] M. Abdar, F. Pourpanah, S. Hussain, *et al.*, "A review of uncertainty quantification in deep learning: Techniques, applications and challenges," *Information Fusion*, vol. 76, pp. 243–297, 2021, ISSN: 1566-2535. DOI: 10.1016/j.inffus.2021.05.008.

[6] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles," *International Conference on Neural Information Processing Systems (NIPS)*, vol. 31, pp. 6405–6416, 2017. DOI: 10.48550/arxiv.1612.01474.

[7] L. Hansen and P. Salamon, "Neural Network Ensembles," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 12, pp. 993–1001, Nov. 1990. DOI: 10.1109/34.58871.

[8] T. G. Dietterich, "Ensemble Methods in Machine Learning," in *Multiple Classifier Systems*, Springer Berlin Heidelberg, 2000, pp. 1–15, ISBN: 978-3-540-45014-6.

[9] M. Muhlbaier, A. Topalis, and R. Polikar, "Ensemble Confidence Estimates Posterior Probability," in *Multiple Classifier Systems*, N. C. Oza, R. Polikar, J. Kittler, and F. Roli, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 326–335, ISBN: 978-3-540-31578-0.

[10] A. Mehrtash, W. Wells, C. Tempany, P. Abolmaesumi, and T. Kapur, "Confidence Calibration and Predictive Uncertainty Estimation for Deep Medical Image Segmentation," *IEEE Transactions on Medical Imaging*, vol. PP, pp. 1–1, Jul. 2020. DOI: 10.1109/TMI.2020.3006437.

[11] J. Mukhoti, A. Kirsch, J. van Amersfoort, P. H. S. Torr, and Y. Gal, *Deep Deterministic Uncertainty: A Simple Baseline*, 2021. DOI: 10.48550/ARXIV.2102.11582. (visited on 01/26/2023).

[12] A. Ashukha, A. Lyzhov, D. Molchanov, and D. Vetrov, "Pitfalls of In-Domain Uncertainty Estimation and Ensembling in Deep Learning," in *International Conference on Learning Representations*, 2020.

[13] Y. Ovadia, E. Fertig, J. Ren, *et al.*, *Can You Trust Your Model's Uncertainty? Evaluating Predictive Uncertainty Under Dataset Shift*, 2019. DOI: 10.48550/ARXIV.1906.02530.

[14] J. Van Brummelen, M. O'Brien, D. Gruyer, and H. Najjaran, "Autonomous vehicle perception: The technology of today and tomorrow," *Transportation Research Part C: Emerging Technologies*, vol. 89, pp. 384–406, Apr. 2018, ISSN: 0968-090X. DOI: 10.1016/j.trc.2018.02.012.

[15] D. Gruyer, V. Magnier, K. Hamdi, L. Claussmann, O. Orfila, and A. Rakotonirainy, "Perception, information processing and modeling: Critical stages for autonomous driving applications," *Annual Reviews in Control*, vol. 44, pp. 323–341, Jan. 2017, ISSN: 1367-5788. DOI: 10.1016/j.arcontrol.2017.09.012.

[16] M. Mathias, R. Timofte, R. Benenson, and L. Van Gool, "Traffic sign recognition — How far are we from the solution?" In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, 2013, pp. 1–8, ISBN: 978-1-4673-6129-3. DOI: 10.1109/IJCNN.2013.6707049.

[17] T. Huo, J. Fan, X. Li, H. Chen, B. Gao, and X. Li, "Traffic Sign Recognition Based on ResNet-20 and Deep Mutual Learning," in *2020 Chinese Automation Congress (CAC)*, 2020, pp. 4770–4774. DOI: 10.1109/CAC51589.2020.9327282.

[18] L. Canese, G. C. Cardarilli, L. Di Nunzio, *et al.*, "Sensing and Detection of Traffic Signs Using CNNs: An Assessment on Their Performance," *Sensors*, vol. 22, no. 22, 2022, ISSN: 1424-8220. DOI: 10.3390/s22228830.

[19] S. Ji, W. Xu, M. Yang, and K. Yu, "3D Convolutional Neural Networks for Human Action Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 221–231, Jan. 2013, ISSN: 1939-3539. DOI: 10.1109/TPAMI.2012.59.

[20] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, *A Closer Look at Spatiotemporal Convolutions for Action Recognition*, Apr. 2018. DOI: 10.48550/arXiv.1711.11248. arXiv: 1711.11248. (visited on 05/09/2023).

[21] World Health Organization, *Global status report on road safety 2018*, Online Multimedia, 2018.

[22] A. Taeihagh and H. S. M. Lim, "Governing autonomous vehicles: Emerging responses for safety, liability, privacy, cybersecurity, and industry risks," *Transport Reviews*, vol. 39, no. 1, pp. 103–128, 2019, ISSN: 0144-1647. DOI: 10.1080/01441647.2018.1494640.

[23] T. Litman and V. T. P. Institute, "Autonomous Vehicle Implementation Predictions, Implications for Transport Planning," Victoria Transport Policy Institute, Report, Nov. 2022.

[24] T. Keeney, "Mobility-As-A-Service: Why Self-Driving Cars Could Change Everything," ARK Invest, Report, 2017.

[25] Z. Yang, L. Li, X. Xu, B. Kailkhura, T. Xie, and B. Li, "On the certified robustness for ensemble models and beyond," *ICLR*, 2022.

[26] K. He, X. Zhang, S. Ren, and J. Sun, *Deep Residual Learning for Image Recognition*, Dec. 2015. arXiv: 1512.03385 [cs]. (visited on 04/24/2023).

[27] M. Sokolova, N. Japkowicz, and S. Szpakowicz, "Beyond Accuracy, F-Score and ROC: A Family of Discriminant Measures for Performance Evaluation," in *AI 2006: Advances in Artificial Intelligence*, A. Sattar and B.-h. Kang, Eds., ser. Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2006, pp. 1015–1021, ISBN: 978-3-540-49788-2. DOI: 10.1007/11941439_114.

[28] L. Boltzmann, "Studien über das Gleichgewicht der lebendigen Kraft zwischen bewegten materiellen Punkten," *Wiener Berichte*, vol. 58, pp. 517–560, 1868.

[29] J. S. Bridle, "Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters," in *Advances in Neural Information Processing Systems*, D. Touretzky, Ed., vol. 2, Morgan-Kaufmann, 1989, pp. 211–217.

[30] G. W. Brier, "Verification of Forecasts Expressed in Terms of Probability," *Monthly Weather Review*, vol. 78, no. 1, pp. 1–3, 1950. DOI: 10.1175/1520-0493(1950)078<0001:VOFEIT>2.0.CO;2.

[31] A. H. Murphy, "A new vector partition of the probability score," *Journal of Applied Meteorology and Climatology*, vol. 12, no. 4, pp. 595–600, 1973. DOI: 10.1175/1520-0450(1973)012<0595:ANVPOT>2.0.CO;2.

[32] T. Gneiting and A. E. Raftery, "Strictly Proper Scoring Rules, Prediction, and Estimation," *Journal of the American Statistical Association*, vol. 102, no. 477, pp. 359–378, 2007, ISSN: 0162-1459. DOI: 10.1198/016214506000001437.

[33] J. Bröcker, "Reliability, sufficiency, and the decomposition of proper scores," *Quarterly Journal of the Royal Meteorological Society*, vol. 135, no. 643, pp. 1512–1519, Jul. 2009, ISSN: 0035-9009. DOI: 10.1002/qj.456. (visited on 04/18/2023).

[34] S. Siegert, "Simplifying and generalising Murphy's Brier score decomposition," *Quarterly Journal of the Royal Meteorological Society*, vol. 143, no. 703, pp. 1178–1183, 2017, ISSN: 1477-870X. DOI: 10.1002/qj.2985. (visited on 04/18/2023).

[35] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, Jul. 1948, ISSN: 0005-8580. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

[36] A. Niculescu-Mizil and R. Caruana, "Predicting good probabilities with supervised learning," in *ICML 2005 - Proceedings of the 22nd International Conference on Machine Learning*, Jan. 2005, pp. 625–632. DOI: 10.1145/1102351.1102430.

[37] M. P. Naeini, G. F. Cooper, and M. Hauskrecht, "Obtaining Well Calibrated Probabilities Using Bayesian Binning," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, ser. AAAI'15, AAAI Press, 2015, pp. 2901–2907, ISBN: 0-262-51129-0.

[38] J. Nixon, M. Dusenberry, L. Zhang, G. Jerfel, and D. Tran, *Measuring Calibration in Deep Learning*, 2019. arXiv: 1904.01685.

[39] J. Vaicenavicius, D. Widmann, C. Andersson, F. Lindsten, J. Roll, and T. B. Schön, *Evaluating model calibration in classification*, Feb. 2019. arXiv: 1902.06977 [cs, stat]. (visited on 05/20/2023).

[40] T. Dimitriadis, T. Gneiting, and A. I. Jordan, "Stable reliability diagrams for probabilistic classifiers," *Proceedings of the National Academy of Sciences*, vol. 118, no. 8, e2016191118, Feb. 2021. DOI: 10.1073/pnas.2016191118. (visited on 04/19/2023).

[41] M. Minderer, J. Djolonga, R. Romijnders, *et al.*, "Revisiting the calibration of modern neural networks," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34, Curran Associates, Inc., 2021, pp. 15 682–15 694.

[42] A. D. Kiureghian and O. Ditlevsen, "Aleatory or epistemic? Does it matter?" *Structural Safety*, vol. 31, no. 2, pp. 105–112, Mar. 2009, ISSN: 01674730. DOI: 10.1016/j.strusafe.2008.06.020. (visited on 01/26/2023).

[43] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014, ISSN: 1532-4435.

[44] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *Proceedings of the 33rd International Conference on Machine Learning - Volume 48*, ser. ICML'16, JMLR.org, 2016, pp. 1050–1059. DOI: 10.48550/arXiv.1506.02142.

[45] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Selected Papers from IJCNN 2011*, vol. 32, pp. 323–332, Aug. 2012, ISSN: 0893-6080. DOI: 10.1016/j.neunet.2012.02.016.

[46] A. Mogelmose, M. M. Trivedi, and T. B. Moeslund, "Vision-Based Traffic Sign Detection and Analysis for Intelligent Driver Assistance Systems: Perspectives and Survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, pp. 1484–1497, 2012. DOI: 10.1109/TITS.2012.2209421.

[47] A. F. Magnussen, N. Le, L. Hu, and E. W. Wong, "A Survey of the Inadequacies in Traffic Sign Recognition Systems for Autonomous Vehicles," *International Journal of Performability Engineering*, vol. 16, no. 10, pp. 1588–1597, 2020. DOI: 10.23940/ijpe.20.10.p10.15881597.

[48] A. Paszke, S. Gross, F. Massa, *et al.*, *PyTorch: An Imperative Style, High-Performance Deep Learning Library*, Dec. 2019. arXiv: 1912.01703 [cs, stat]. (visited on 04/21/2023).

[49] M. Alibeigi, W. Ljungbergh, A. Tonderski, *et al.*, "Zenseact Open Dataset: A large-scale and diverse multimodal dataset for autonomous driving," 2023. DOI: 10.48550/ARXIV.2305.02008. (visited on 05/19/2023).

[50] Y. Zhang, P. Sun, Y. Jiang, *et al.*, *ByteTrack: Multi-Object Tracking by Associating Every Detection Box*, Apr. 2022. arXiv: 2110.06864 [cs]. (visited on 04/25/2023).

[51] X. R. Lim, C. P. Lee, K. M. Lim, and T. S. Ong, "Enhanced Traffic Sign Recognition with Ensemble Learning," *Journal of Sensor and Actuator Networks*, vol. 12, no. 2, 2023, ISSN: 2224-2708. DOI: 10.3390/jsan12020033.

[52] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," 2017. DOI: 10.48550/ARXIV.1711.05101. (visited on 05/23/2023).

[53] I. Loshchilov and F. Hutter, "SGDR: Stochastic Gradient Descent with Warm Restarts," 2016. DOI: 10.48550/ARXIV.1608.03983. (visited on 05/23/2023).

[54] D. Hendrycks and T. Dietterich, *Benchmarking Neural Network Robustness to Common Corruptions and Perturbations*, Mar. 2019. arXiv: 1903.12261 [cs, stat]. (visited on 03/23/2023).

[55] G. Shafer and V. Vovk, *A tutorial on conformal prediction*, Jun. 2007. arXiv: 0706.3188 [cs, stat]. (visited on 05/31/2023).

[56] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Mathematical Programming*, vol. 45, pp. 503–528, 1989.

[57] R. Rahaman and A. H. Thiery, *Uncertainty Quantification and Deep Ensembles*, Nov. 2021. arXiv: 2007.08792 [cs, stat]. (visited on 02/17/2023).

A

Additional experiments

In this appendix, some additional experiments are presented to the interested reader.

A.1 Sequence quality

In the sequence dataset, only the last frame is annotated by humans and the rest of the frames in any given sequence lack annotations. In order to obtain approximate annotations for these frames, the traffic signs are tracked across frames using a ByteTracker. One might expect this to result in diminishing crop quality as the distance to the original annotated frame increases due to tracking imperfections or data variations. To test this hypothesis, the data loader for the sequence data set is augmented to only present frames from a certain point (timestep) in every sequence. Then, 5-member DE and a single model were evaluated on each of the timestep-wise data loaders (see Figure A.1).

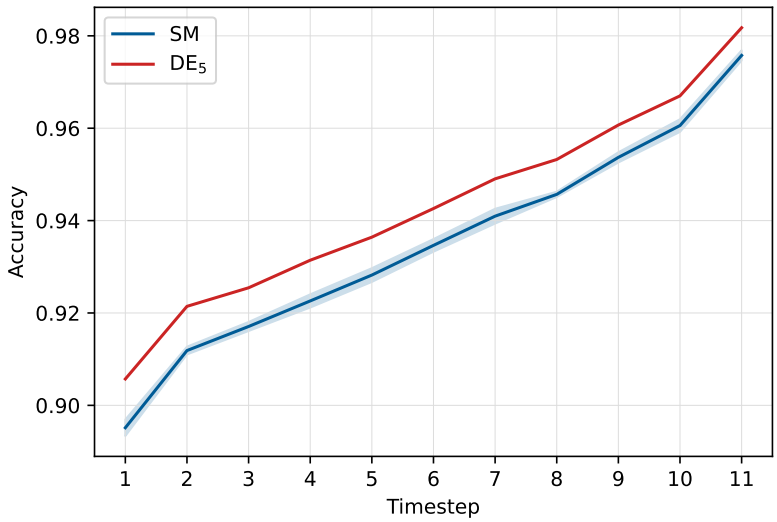


Figure A.1: Accuracy per frame at each timestep of the sequence dataset for a vanilla ResNet18 model and a 5-member ResNet18 ensemble.

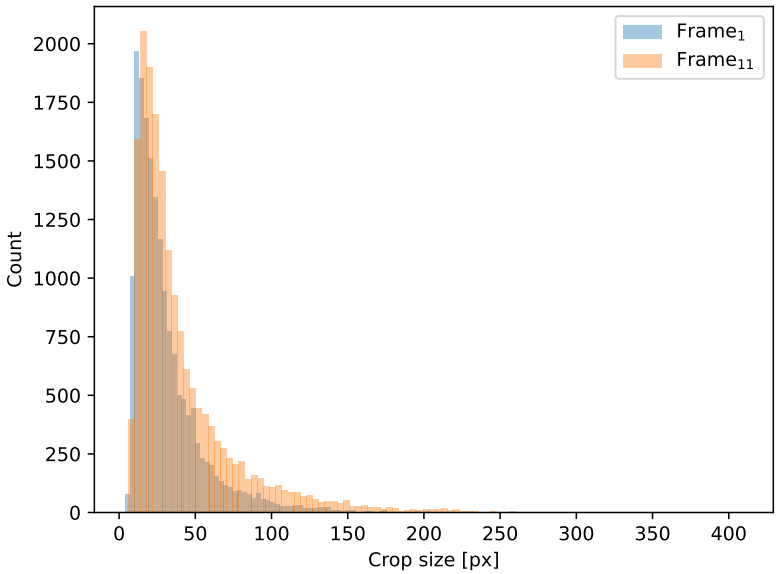


Figure A.2: The crop size distribution for frames 1 and 11 across all sequences, plotted with 100 bins. The crop size is defined as the size (in pixels) of the smallest crop dimension.

The results seem to indicate that the average single-frame quality deteriorates, as previously asserted. In order to determine what kind of deterioration that is occurring, the crop size distribution for the first and the last frames of the sequences is plotted (see Figure A.2). This shows that the crop size trends downward as we move further away in time from the annotated frame, which makes sense since the traffic signs will on average be further away from the front-facing camera of the car in the early frames.

It is difficult to quantitatively disentangle the effects of smaller crop sizes and any deterioration in the tracking across time, so a rough qualitative review was also conducted by finding sequences where the first frame was misclassified and the final frame was correctly classified. Most of those sequences constitute occlusions or sequences where the tracker progressively loses the traffic sign. The occlusions include instances where other cars pass in front of the camera, or when a tree branch or other object progressively obstructs more and more of the tracked traffic sign.

An interesting point is if averaging model predictions across sequences helps alleviate the problems with sequence quality degradation. To test this, two different single-frame datasets are created. The first method is to take all the images from the sequences and sample each image randomly without replacement. We call this the sequence frames dataset. The second method is to take only the last frame in the sequence that has the original and therefore best annotation and use that as the comparison. This is the single-frame test dataset used in the rest of the report. The problem with the first option is the average quality degradation in the sequence, causing a lower predictive performance than expected. The second option, on the

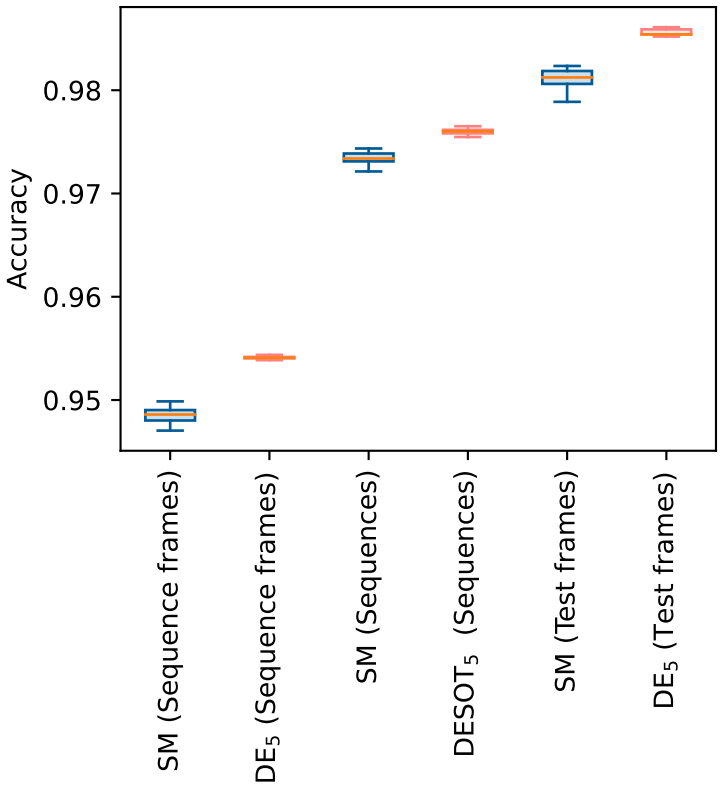


Figure A.3: Comparing the effects from the different methods of creating a single frame dataset. The sequence frames are all the frames from the sequence randomly sampled without replacement. The Test frames are the last image in the sequence, which contains the most high-quality information. For the sequences dataset, a DESOT is applied. It seems that average input quality dominates quantity.

other hand, completely disregards the other frames that have a degraded average quality. In Figure A.3, the performance of DE and DESOT on the different datasets are compared. As can be seen, the performance on the sequences dataset is closer to the performance on the test frames dataset than to the sequence frames. This seems to show that the aggregation across the sequence helps to mitigate the effects of the lacking sequence quality, but can not outperform the last frame which has the highest-quality information. Thus, it seems that the aggregation across sequences adds robustness to low-quality crops, and can make up for some of the negative effects of poor tracking.

A.2 Comparing model architecture size

ResNet models drastically vary in size, from tens of layers deep to as deep as hundreds or thousands of layers [26]. Since traffic sign recognition is not very parameter intensive, with smaller CNNs performing well [17], a small study was conducted to

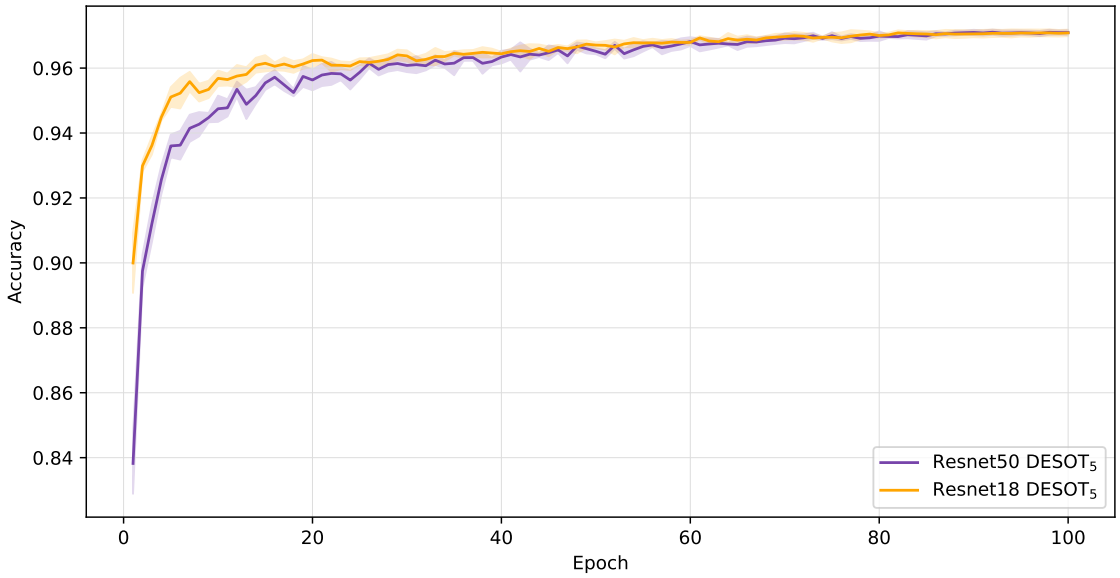


Figure A.4: ResNet18 and ResNet50 compared on accuracy on the validation dataset across training epochs.

compare the performance of ResNet50 with the smaller ResNet18. The study was conducted by training five instances of each of the two networks until convergence. Then, these were combined into a 5-members DE, and their predictive performance on the single-frame validation set was then compared (see Figure A.4). As can be seen in the graph, the two different architectures are similar in F1-score after convergence and thus the bigger model does not give any advantage for this application. The smaller ResNet18 appears to train the initial epochs faster which helps achieve better results with limited time and resources.

This study is also used as a basis for the choice of how many epochs to train every single model for. The choice to train the final models only for 30 epochs is that it cuts down the training time and allows us to train more models, which in the end leads to better performance estimates for the final models.

A.3 Temperature scaling

In this part of the appendix, some extra results concerning temperature scaling are presented (see subsection 2.2.3 for a theoretical overview of temperature scaling). The temperature scaling is optimized on the single frame validation set using negative log-likelihood as a loss function on the L-BFGS optimizer [56] with a learning rate of 0.05. A maximum of 50 iterations are used, with an early bailout tolerance of 10^{-9} and early bailout gradient tolerance of 10^{-7}. The optimization starts off with unit temperature ($T = 1.0$). We base our implementation of temperature scaling on an implementation by Geoff Pleiss, co-author of the original temperature scaling paper [1]. In Figure A.5, the optimal temperature for different numbers of trained epochs is shown. The optimal temperature rises consistently with the number of

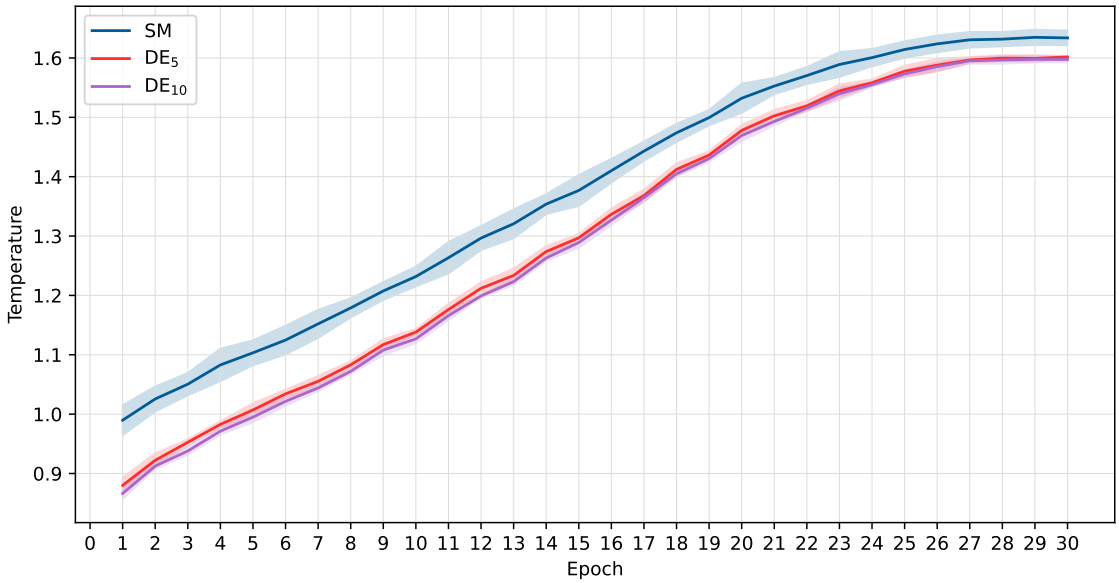
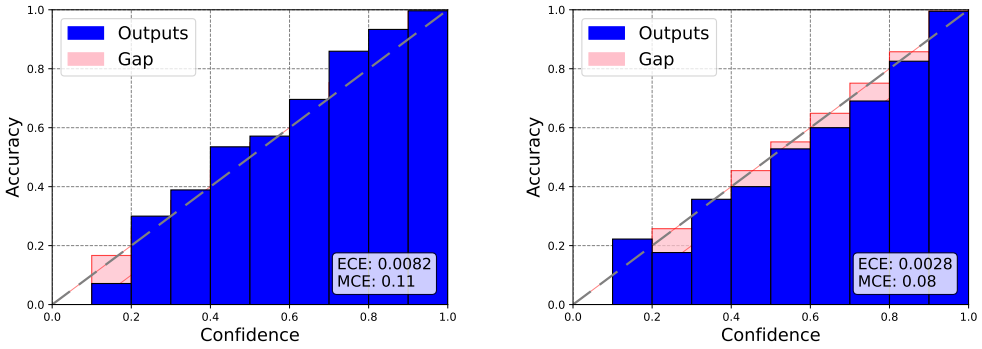


Figure A.5: The optimal temperature for models trained for different numbers of epochs.

epochs trained, which indicates that the models become more and more overconfident as they train more. This seems to make sense, given that Guo *et al.* [1] claim that modern neural networks are overconfident, and hypothesize that these modern networks trade low classification error against calibration performance. This might be caused by the fact that using cross entropy as the loss function incentivizes sharpness in model prediction (assigning high probabilities to the predicted class), which implicitly encourages highly confident predictions. This also means that misclassifications will often be highly confident.

It seems that the ensembles generally are less overconfident out-of-the-box, compared to single models and are even underconfident at early epochs. They are, however, not perfectly calibrated and also follow the trend of becoming more overconfident as training progresses.

One thing to note is that it is not apparent how to temperature scale an ensemble of models. We are not the first to temperature scale ensembles. Ashukha *et al.* [12] even go as far as to say that deep ensembles should always be temperature scaled, but fail to include how they do it. Two different ways come to mind: creating ensembles from single models that have been individually temperature scaled to be well-calibrated, or directly temperature scaling the ensemble itself. The first approach uses individually scaled models where each model is temperature scaled just as if it was a single model, then combined into an ensemble as usual. We refer to this way of temperature scaling an ensemble as *individual* temperature scaling. The second approach is to combine the single models with no temperature scaling into an ensemble, and then temperature scaling the ensemble as if it were a single model. We refer to this approach as *joint* temperature scaling.



(a) Individually temperature scaled models.

(b) Jointly temperature scaled models.

Figure A.6: The reliability plots comparing temperature scaling for a DE_5 using two different techniques – individual temperature scaling and joint temperature scaling. The results are shown for the single-frame validation dataset that the models are temperature scaled using.

In our testing, joint temperature scaling seems to be less under-confident when considering the reliability plots shown in Figure A.6. The joint temperature scaled models also achieve lower ECE and MCE, showing that when temperature scaling an ensemble, jointly scaling is the preferred method. This same conclusion is reached by Rahaman and Thiery [57], who more thoroughly investigate different ways of temperature scaling deep ensembles. For these reasons, all temperature scaling results in the main text are produced using joint temperature scaling.

DEPARTMENT OF ELECTRICAL ENGINEERING
DIVISION OF SIGNAL PROCESSING AND BIOMEDICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY