Certainty Groups: A practical approach to distinguish confidence levels in neural networks

Lukas Lodes[1], Alexander Schiendorfer[2]

[1,2] *Technische Hochschule Ingolstadt, Germany*
Lukas.Lodes@thi.de
Alexander.Schiendorfer@thi.de

ABSTRACT

Machine Learning (ML), in particular classification with deep neural nets, can be applied to a variety of industrial tasks. It can augment established methods for controlling manufacturing processes such as statistical process control (SPC) to detect non-obvious patterns in high-dimensional input data. However, due to the widespread issue of model miscalibration in neural networks, there is a need for estimating the predictive uncertainty of these models. Many established approaches for uncertainty estimation output scores that are difficult to put into actionable insight. We therefore introduce the concept of certainty groups which distinguish the predictions of a neural network into the normal group and the certainty group. The certainty group contains only predictions with a very high accuracy that can be set up to 100%. We present an approach to compute these certainty groups and demonstrate our approach on two datasets from a PHM setting.

1. INTRODUCTION

Modern production is a complex interaction of different parts and optimal usage of available production resources is hard to accomplish. Data analytics is used for decades to support decision making and gained additional attention in the last years through machine learning with deep neural networks (DNN). The potential of DNN-based classification is increasingly being explored in the context of prognostics and health management (PHM). There exists a wide variety of promising applications, including the ideal timing of maintenance intervals (predictive maintenance) or the prediction of health indices of the production line itself or of the products. Despite the availability of the technology, these tools are still not commonly used in the manufacturing industry. Besides the slow adoption of digitization in the manufacturing industry in general, there are further reservations about machine learning systems in critical applications like quality inspection. One problem are difficulties in understanding and judging the model's outputs, in particular, if they represent a probability distribution over a finite number of classes, as returned by a softmax or sigmoid activation. For neural networks, the widespread issue of model miscalibration (Guo, Pleiss, Sun, & Weinberger, 2017) leads to predictions in which the model assigns a high probability score to a wrong class. This makes it hard to trust the predictions of such a model. Even if a predictive ML model errs on some cases, it should provide a reliable estimate of its uncertainty in order to mitigate potential negative impact of false predictions. In a production system, this property could be used for *Hybrid Quality Inspection* (Ismail, Mostafa, & El-assal, 2021), in which automated quality inspection of the products is done only for a subset of all product instances – those considered difficult to judge would still undergo manual, i.e., human-operated tests. As stated, the ability of a neural network to express its predictive uncertainty is crucial for real world application. In the last few years, increasing attention has been paid to the field of uncertainty estimation in neural networks (an overview is given in Section 1.1). There exist several different approaches to uncertainty estimation in neural networks, and many of them share a common principle. The predictive uncertainty is often expressed by several metrics (usually mean and standard deviation) that are calculated on a sample of predictions done by slightly different estimators. These metrics however are hard to put into actionable insight as they are still relatively abstract. From the point of view of user-friendliness, a clear and concise classification of whether a prediction is based on a high or low uncertainty would be highly desirable. This is especially important in situations in which the estimated uncertainty of a prediction has to be converted into a binary decision, e.g., by a human worker in a production line. Additionally, many methods for estimating the predictive uncertainty in neural networks tend to involve a very large theoretical background (e.g., fully Bayesian methods involving a meticulous choice of priors) that further complicates the ap-

plication of these methods in applied machine learning applications such as predicting the health status of equipment or the quality of products. Thus, ease of use and manageability were further criteria in the development of our approach.

We therefore propose the concept of *certainty groups* which divides the predictions of a neural network into the normal group and the certainty group. The certainty group contains only predictions (and, consequently, instances) with a very high accuracy that can be set up to 100%. We assume that these instances are handled by an autonomous ML-system sufficiently well and don't require further action (e.g., established quality assurance methods). However, the other cases are potentially unsafe to be fully processed by an ML system and need to be more thoroughly examined. These instances represent the normal group, in which the predictions behave like usual, with them being correct with a certain accuracy. This leads to a division of the data set in different levels of model confidence. In the underlying paper, we present an approach based on ensembles to compute these certainty groups by measuring the disagreement between the ensemble members and sorting it based on a threshold value. We validate our approach using two datasets from a PHM setting, in which the method generates groups of samples that contain 26-51% of all instances with very few mispredictions compared to baseline models and approaches. We further show that our approach is highly flexible as it allows to configure the quality of the predictions contained in the certainty group.

1.1. Related Work

Due to the renewed interest in deep learning after breakthroughs in image classification or reinforcement learning, neural networks have received a lot of attention in the last 5 to 10 years. Therefore it's not surprising that there exist many recent studies that cover the application of neural networks in PHM related tasks like remaining useful life (RUL) prediction or predictive maintenance in general. Comprehensive surveys and reviews are provided in the works of (Zhao et al., 2016) and (Zhang et al., 2019). (Harshavardhanan & Nene, 2020) outlines why the consideration of uncertainty is beneficial when making predictions in a PHM related setting by explaining common sources of uncertainty and giving an overview on how to incorporate uncertainty into PHM systems. The area of uncertainty estimation in neural networks has been actively researched in the last couple of years. The first branch of works (see, e.g., (Guo et al., 2017; Tomani & Buettner, 2021; Naeini, Cooper, & Hauskrecht, 2015)) tries to directly solve the issue of miscalibration, either by post-processing a trained model or already during training (e.g. through additional loss functions). The second branch is based on the idea of ensembles, which are an established approach for estimating uncertainty in general and for obtaining more robust predictions (Lee, Purushwalkam, Cogswell, Crandall, & Batra, 2015; Wen, Tran, & Ba, 2020; Havasi et al., 2020). These works aim to further develop classical ensembling by increasing the accuracy of the ensemble prediction or improving the computational performance, which they achieve by different types and degrees of parameter sharing between the ensemble members. The works of (Daxberger et al., 2021) and (Liu et al., 2020) are examples for the integration of Bayesian methods in neural networks. (Daxberger et al., 2021) suggests an easy to use approach for converting conventional neural networks to bayesian neural networks by adding laplace approximation to the final layer, while (Liu et al., 2020) achieves distance aware outputs by adding a gaussian process to the network. Additionally, in the last years a variety of frameworks for uncertainty estimation were published. These frameworks cover a wide scope, ranging from implementations of neural network based approaches like Deep Ensembles (e.g. in (Weiss & Tonella, 2021)) to fully fledged probabilistic programming languages (Tehrani et al., 2020; Salvatier, Wiecki, & Fonnesbeck, 2016; Bingham et al., 2018; Phan, Pradhan, & Jankowiak, 2019) The mentioned works provide approaches for uncertainty estimation well-founded in probability theory – which also requires the selection of proper priors and sampling or variational inference for approximation. Our approach draws upon these methods and adds an additional layer of abstraction in order to improve the usability of uncertainty estimates.

2. Approach

As mentioned in the introduction, our approach is mainly motivated by creating an easy-to-use method with understandable outputs for practitioners in manufacturing. The targeted application area of our approach are production lines in which workers will have to work with the recommendations of a ML system. In the scope of this paper, we focus on binary classification problems – "OK" vs. "NOK" for a product, or "Machine defect" vs. "Machine intact" for health state estimation. We feel that for this use case the output of a ML system should be as easy as possible to understand to create transparent decision criteria and to make the workers more comfortable working with such a system. That's why we decided to provide again a binary decision whether a prediction is afflicted with uncertainty or whether the system is very certain about it. For a given instance x, we then have to first decide whether it falls into the certainty group or not and, second, if it belongs to class 0 or 1. The latter prediction is trusted more if x belongs to the certainty group.

2.1. Foundations

In the following, we consider a neural network model m having the form $f_m(x;\theta)$ where θ refers to the parameters that are optimized with respect to data set $D = (x_i, y_i)_{i=1}^N$ to obtain a point estimate (i.e., empirical risk minimization or maximum likelihood estimation). The output is assumed to be normalized, i.e., it is a real number that represents the

probability that x belongs to class 1:

$$p(y = 1 \mid x) = f_m(x; \theta) \tag{1}$$

In the case of neural networks, this is the direct output e.g. of a sigmoid or softmax layer. For example, $f_m(x; \theta) = 0.8$ represents estimating a probability of 80 % for class 1 and 20 % for class 0. In accordance with the literature (Guo et al., 2017), we refer to these normalized raw score outputs as *confidences*. The rounded confidences, i.e., $y = 0$ if $f_m(x; \theta) \leq 0.5$ and $y = 1$ otherwise, are called the *predictions*. A mismatch between a model's output confidence and the true correctness likelihood is called *miscalibration* (Guo et al., 2017). For example, miscalibration occurs if a model only reaches 60% accuracy on all test samples that it predicted to be 90% sure. The evaluation of a neural network m, i.e., calculating $f_m(x; \theta)$ for a particular input x and fixed parameters θ, is called *inference* (or, often, simply a forward pass) to distinguish the training from the productive stage.

In addition, we assume a strategy for uncertainty estimation $\zeta_m \geq 0$ where $\zeta_m(x \mid f_m, \theta) = 0$ means absolute certainty that $p(y = 1 \mid x)$ is indeed $f_m(x; \theta)$ and larger values indicate higher uncertainty. For example, given an ensemble of models f_{m_i} that are derived from m, ζ_m could refer to the sample standard deviation of the confidences $f_{m_i}(x; \theta)$ or other dispersion metrics. Other measures ζ_m could be retrieved from a Bayesian approach (i.e., having estimated a posterior over the parameter space $p(\theta \mid D)$) as the variance of the posterior predictive distribution

$$p(y = 1 \mid D, x) = \int_\Theta p(y = 1 \mid \theta, D, x)\, p(\theta \mid D, x)\, \mathrm{d}\theta \tag{2}$$

which might not be tractable but only approximated via sampling or variational methods. In principle, our approach targets both point estimates with sample statistics as well as Bayesian techniques – although the high computational costs of Bayesian inference leads us to turn to practically applicable sample-based methods.

2.2. Certainty Groups: A general definition

Our goal is to define certainty groups that isolate the instances for which the model's predictions are very reliable from the rest. In a manufacturing setting, these would be the parts that are definitely detected as scrap or good. Informally speaking, the *certainty group* of a model is created by an uncertainty estimator and a predicate that decides whether the estimated uncertainty of an instance's prediction is low enough. The instances whose estimated uncertainty is too high to be in the Certainty Group are contained in the *normal group*. They are usually significantly harder to classify and, thus, the model's predictions become less accurate. Returning to the above example, these would be the parts that have to undergo further testing and human intervention – as the model is not capable of determining their state. This leads to a binary distinction of the confidence levels in the underlying neural network. Figure 1 illustrates this concept.

In order to allow for a wide variety of uncertainty metrics ζ_m and to allow future extensions, we decided to define the Certainty Group of a model in a very open manner.

The certainty group CG of a model m with respect to data set D is then defined as follows:

$$CG(m, D) = \{(x, y) \in D \mid \varphi \text{ accepts } \zeta_m(x \mid f_m, \theta)\} \tag{3}$$

where φ is a boolean predicate (called the certainty criterion) that can consist of the operators $\wedge$ (AND), $\vee$ (OR), $\neg$ (NOT), $<$, $\leq$, $>$ and $\geq$. The specific predicate depends on the chosen approach to estimate the model's uncertainty and is further described later in this section.

2.3. Possible uncertainty measures for certainty groups

We tested two principles for computing certainty groups from model confidences and predictions. The first, main, approach uses ζ_m along with some thresholding for some model m to partition the instances into normal and certainty group. Along the way, some methods require to (temporarily) generate a finite number of models m_i derived from m (e.g., via different dropout masks) to calculate ζ_m based on the confidences and predictions retained from the individual m_i. We call these methods *sample-based*. The second, baseline, approach simply sets $\zeta_m = f_m$, i.e., it uses the raw confidences as an indicator for certainty – e.g., interpreting a model output $f_m(x; \theta) = 0.99$ to truly be 99% sure.

An uncertainty estimator can obtain the prediction y and standard deviation σ for an instance x as follows (note that this requires a model that implements a strategy for estimating uncertainty through multiple model outputs): First, it obtains n samples of the model's prediction for the instance, that can be formally described as $p_\theta(\hat{y} \mid x)$.

A sample-based uncertainty estimator first generates n samples, i.e., model confidences $(f_{m_i}(x; \theta))_{i \in \{1,\ldots,n\}}$ for a given instance x, e.g., $[0.85, 0.9, 0.92, \ldots]$.

Then it calculates the sample mean $\bar{f}$ and sample standard deviation s on these n sampled confidences and calculates the final class prediction y based on the sampled confidences by rounding $\bar{f}$. Finally, the decision whether this instance and prediction is contained in the certainty group is made by comparing s to an empirically determined threshold hyperparameter T_s. Referring to equation 3, this amounts to a straightforward predicate φ. In this case, the prediction is part of CG if

$$s \leq T_s \tag{4}$$

It is possible to calculate $\bar{f}$ and s based on the individual models' predictions (rounded confidence values) as well, but the

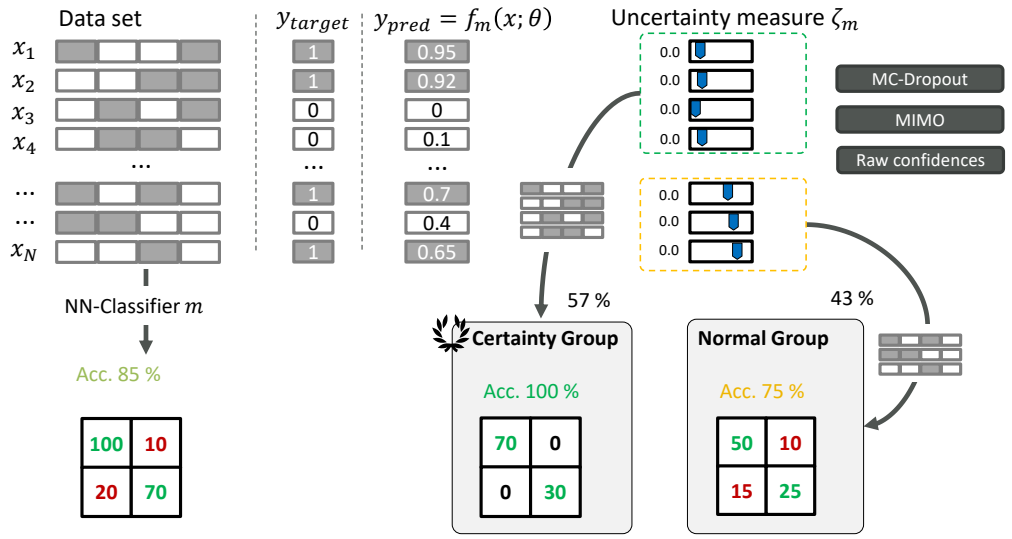


Figure 1. The predictions of a neural network m are evaluated in their uncertainty by an uncertainty measure ζ_m. If a prediction has a low enough uncertainty, it is assigned to the so called Certainty Group. The less-certain predictions of m are contained in the Normal Group. This usually leads to a significantly higher accuracy in the Certainty Group than in the Normal Group.

usage of raw confidence values allows for a much finer resolution w.r.t. φ, which we found to lead to larger certainty group sizes in our experiments, i.e., it was a more distinctive certainty criterion.

The introduction of the additional hyperparameter T_s allows for flexibility in terms of predictive accuracy in the certainty group. While we were initially aiming towards an accuracy of 100% in the certainty group, it's easily possible to lower the requirements for accuracy if the application allows it to get larger certainty group sizes.

The open definition of certainty groups in equation 3 allows us to define the baseline that only relies on the raw confidences. The most naive approach for computing certainty groups is to accept the confidence margins around 0 and 1 as safe instances and therefore include instances with model confidences in these margins in the certainty group. Formally speaking, an instance x is in CG for model m if

$$f_m(x;\theta) \leq T_{lower} \vee f_m(x;\theta) \geq T_{upper} \tag{5}$$

where T_{lower} and T_{upper} are two thresholds that define the border of the lower and upper decision margin.

2.4. Specific approaches to obtain ensembles from models

As stated in section 1.1, there is a wide variety of approaches that can be used either on top of existing neural networks or directly included in the architecture. From these approaches we chose a Dropout approach according to (Gal & Ghahramani, 2016) and multi-input multi-output configurations (MIMO) by (Havasi et al., 2020) as they fit our requirement for ease of use and manageability very well. In the following, we describe both approaches and discuss them to further justify our decision for these two approaches.

Dropout (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014) is a widely used regularizer in neural networks and is used to make a neural network more robust against overfitting the training data set. The idea of this approach is to randomly drop connections in a neural network with a certain probability, which leads to a slightly different network configuration in each inference step (i.e., forward pass or evaluation of the network). Gal et al. describe this mathematically by sampling binary variables (i.e., from a Bernoulli distribution) for every unit in the neural network with probability p_i for value 1 in each layer L_i except for the output layer. If the corresponding binary variable takes value 0, that particular unit is dropped by setting it to zero. Usually, dropout is only activated during training and deactivated in the test or deployment stage (once training finished). Gal et al. however suggested the activation of dropout during the test or deployment stages to enable a Bayesian approximation without the need for computationally expensive implementations of Bayesian neural networks. With activated dropout during model inference, the model's uncertainty can be estimated by doing n distinct inference passes – each corresponding to a different network configuration of deactivated connections. That way, we can obtain a sample-based uncertainty estimator ζ_m. Mathematically speaking, the goal is to calculate the predictive distribution $p(y \mid x, D)$ for a new input point x^* given the data set D. As this predictive distribution cannot be obtained analytically, it is approximated by the approximate predictive distribution $q(y \mid x, D)$. The authors show that the expected value of this distribution (serving as the model's confidence)

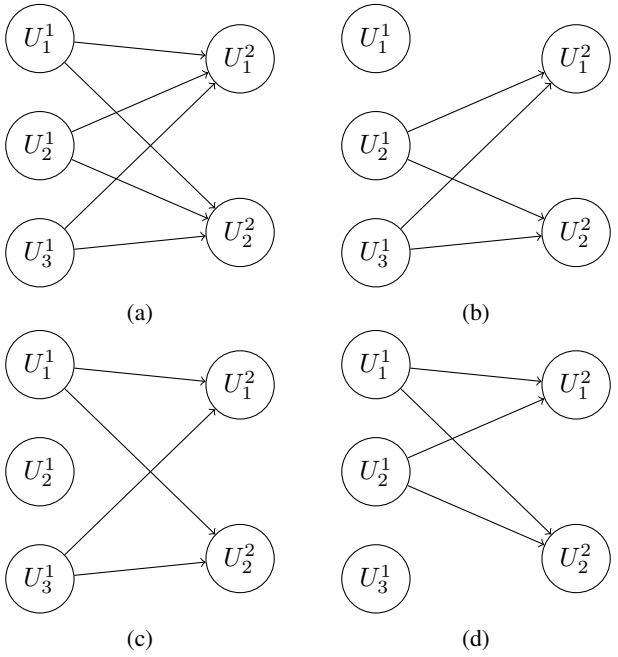
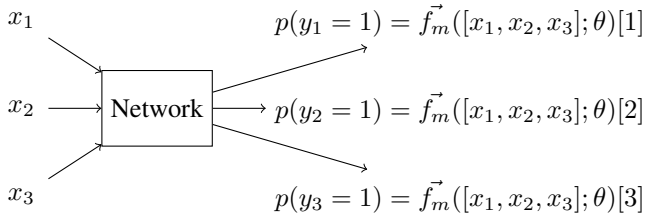
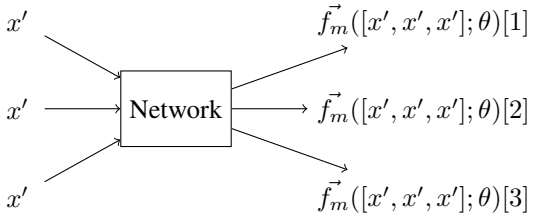


Figure 2. Visualization of MC Dropout with $T = 3$ inference passes in a network with 2 layers where U_i^n describes the i-th unit in layer n. (b), (c) and (d) show three different realizations of the base architecture shown in (a).



(a) In the train phase the network input consists of a concatenation of n instances. The network outputs n concatenated confidences corresponding to the different input instances.



(b) In the test phase the network input consists of n copies of an unseen input instance x. The network outputs n different confidences for the input instance.

Figure 3. Illustration of the MIMO configuration proposed by Havasi et al.

can be estimated by

$$\mathbb{E}_q(f_m) \approx \frac{1}{n} \sum_{i=1}^{n} f_{m_i}(x; W_1^i, ...W_L^i) \tag{6}$$

where m is a neural network with L layers and $\{W_1^i, ...W_L^i\}_{t=i}^n$ are the random variables representing the trainable parameters θ of a neural network with dropout. These random variables contain the actual dropout mask on the neural network's parameters. The authors refer to this Monte Carlo estimate as *MC Dropout*. As the activation of dropout during inference results in slightly different model architectures at each inference pass, it can be interpreted as an ensemble with n members $(f_{m_i})_{i=1}^n$ when calculating n inference passes. This process is illustrated in Figure 2. We chose this approach because dropout is already widely used and it is convenient to obtain an uncertainty estimate from this approach. Frameworks like PyTorch or Tensorflow allow an activation of dropout during the inference phase, which greatly reduces the required implementation effort. The most significant downside of this approach is the slow inference speed. As we need to compute n inference passes to create an ensemble with size n (and generally require large enough n to acquire reliable estimates), the approach is approximately n times slower than approaches that require only one inference pass. Our approach builds upon this method by implementing an actual decision rule for when to accept a prediction as "certain" and when as "uncertain". Gal et al. roughly propose the need for such a decision rule in the context of classification.

Similar to MC dropout, the usage of MIMO configurations from Havasi et al. aims towards the implicit creation of an ensemble. The foundation of this approach is given by the results of works covering sparsity in modern neural networks (Frankle & Carbin, 2018; Molchanov, Tyree, Karras, Aila, & Kautz, 2016; Zhu & Gupta, 2017). These works show that modern neural networks often are overparametrized for their tasks, i.e. they often could solve the task multiple times in terms of capacity. Havasi et al. use this assumption to concurrently train multiple independent subnetworks within one single larger network using the proposed multi-input, multi-output (MIMO) configuration. For this approach to work, only two small changes have to be made to an existing model architecture. In contrast to the normally used model inputs that consist of a single instance, n input instances are concatenated into one single instance. In this case, n represents the number of ensemble members contained inside the network. This can be formally described in the following way. Given a data set $D = (x_i, y_i)_{i=1}^N$ with size N where x_i is an instance with corresponding label y_i. Usually, the model output is given by $f_m(x_i; \theta)$. In the MIMO configuration however, the model input is given by a concatenation of n inputs $\{x_1, ..., x_n\}$. The model output does not consist of only one so called *head*, but of n *heads* that output n confidences as a

vector $\vec{f}_m$:

$$\vec{f}_m = \\ = (f_{m_1}([x_1, \ldots, x_n]; \theta), \ldots, f_{m_n}([x_1, \ldots, x_n]; \theta)) \quad (7)$$

In the training phase, the concatenated input instance consists of different instances, which leads to the creation of independent subnetworks that are contained inside the large network. During test time or inference in the deployment stage, the concatenated input instance is created by duplicating one unseen test input instance x' n times, so $x_1 = \ldots = x_n = x'$. This leads to n output confidences for the same instance. This way, one can obtain an ensemble with n members in only one inference pass, assuming that the overall model has enough capacity (i.e., parameters θ) to solve the given task n times. With respect to our sample-based uncertainty metrics, we consider the components of the output vector as individual derived confidences, i.e., $\vec{f}_m(x; \theta)[i] \widehat{=} f_{m_i}(x; \theta)$. The concept is illustrated in Figure 3. The authors propose to obtain robust predictions by using the mean confidence of the ensemble members as a combined output. We extend this by calculating a dispersion metric from the individual outputs and using it as a measurement of uncertainty and including it into decision making. The biggest advantage of this method is the fast inference speed as it is n times faster than an implicitly computed ensemble of the same size using dropout as the necessary modifications to the model architecture do not affect the overall number of computations in a meaningful way. However, in contrast to dropout, we cannot increase the ensemble to any desired size as the maximum possible ensemble size is bound by the model capacity and has to be determined empirically.

3. EXPERIMENTS

In this section, we evaluate the capabilities of our concept using the three methods presented in section 2.3. We performed experiments to answer the following questions:

1. Which method produces large and reliable certainty groups?
2. How does the target accuracy for Certainty Groups affect their size?
3. Do different methods detect the same instances in Certainty Groups?

3.1. Experimental Setup

Our approach is evaluated using two data sets originating from PHM settings: the FordA data set obtained from (Bagnall, 2022) and the AI4I predictive maintenance dataset (Matzka, 2020) obtained from (Dua & Graff, 2017) . The FordA dataset consists of 3601 train and 1320 test instances, each consisting of 500 sensor features. The goal is to classify whether the sensor measurement indicates a problem in an automotive subsystem or not. The AI4I Dataset is from synthetic origin an aims to reflect real predictive maintenance data. It consists of 10 000 data points with 6 Features (5 numeric, 1 categorical) and a label. Failures can result from five independent failure modes. As this dataset has a very imbalanced class distribution, we used SMOTE (Chawla, Bowyer, Hall, & Kegelmeyer, 2002) to overcome this issue during training. The experiments were performed on a system with a 8-Core CPU, a NVIDIA RTX 2080 Super GPU and 32GB RAM. For all experiments the same fixed random seed was chosen to ensure reproducibility. The code for these experiments is provided in (CertaintyGroups, 2022).

For each data set, we first developed two architectures, one for estimating uncertainty using MC dropout and one with a MIMO configuration. The dropout model was additionally used for the baseline approach in which the raw confidences are used as a certainty criterion. For these experiments, dropout was not active during inference. We then trained and evaluated both model architectures while performing hyperparameter optimization with respect to the model parameters (hidden dimensions, configurations of the convolutional layers) and training parameters (number of epochs, batch size, learning rate). Note that at this stage, the additional hyperparameter T_s is not yet optimized because it is not yet used. After having found a set of well-performing model hyperparameters, we started to optimize T_s by repeatedly computing certainty groups until we were satisfied with the predictive accuracy in the certainty groups. We then evaluated both the predictive performance and the certainty group behavior on a held-out test set.

The model architectures for each data set are based on similar principles. For both approaches, we tested fully connected and convolutional architectures. The networks are 4 to 7 layers deep. The CNN approach is built like a LeNet-Style CNN (LeCun, Bottou, Bengio, & Haffner, 1998) with a feature extraction module built from convolutional layers at the beginning of the network and a final classification stage at the end of the network. In contrast to image classification, we use 1D-Convolutions as our problems only consist of 1D Data (e.g. sensor readings). Depending on the data set, minor changes to the internal dimensions of the layers were necessary. The main differences between the architectures for the dropout and the MIMO approach are dropout layers and reshape/concatenation operations before the first layer. These differences can be easily built into the base architectures. During the development of the model architectures, we followed the principles and goals stated in section 1. We therefore aimed to use architectures that are well manageable, which is the reason why we have refrained from using larger architectures. Figures 4 and 5 illustrate implementations for both approaches. Table 1 lists the architecture type per data set we chose for the evaluation. It can be seen that the fully connected architectures were used for the problem with fewer

Table 1. Used architecture types per dataset and approach for uncertainty estimation.

Dataset	**Dropout Model**	**MIMO Model**
AI4I	Fully Connected	Fully Connected
FordA	Convolutional	Convolutional

input features. Havasi et al. suggested 2 or 4 as an optimal number of subnetworks in a MIMO configuration. In our experiments with less complex datasets, we found that 16 (AI4I) and 32 (FordA) subnetworks lead to optimal results.

3.2. Results

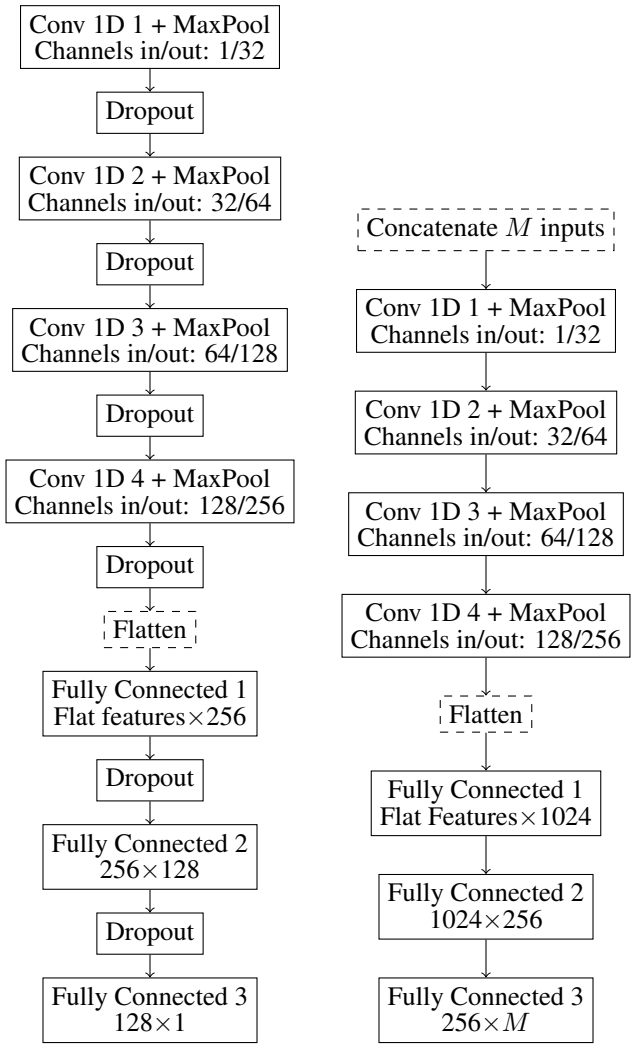
As a first step and to be able to judge the baseline model performance, we evaluated the predictive accuracy of the different models over the whole test set without partitioning them into normal and certainty group. As the AI4I dataset is not perfectly balanced, we additionally used *balanced accuracy (BA)* as a metric. Balanced accuracy is especially useful with imbalanced datasets in which a zero-rule classifier can lead to very high accuracy scores. In case of binary classification, it is defined as following:

$$BA = \frac{\textit{Sensitivity} + \textit{Specificity}}{2} \tag{8}$$

The accuracy and balanced accuracy scores for each dataset and architecture are shown in Table 2. For the FordA datasets we were able to achieve good accuracies well above 90%. On the AI4I dataset we achieved 84.0% and 80.74% accuracy. It was the hardest to train and showed fragile training behaviour, which may be caused by its low number of features. Note that we did not use the ensembling capabilities of the dropout and MIMO models during the evaluation of the predictive performance as this was not our main goal.

3.2.1. The 100% accuracy requirement for CGs

This section is dedicated to answer the first research question: *Which method produces large and reliable certainty groups?* For these experiments, we optimized the hyperparameter T_s for the dropout and the MIMO architectures w.r.t. maximum predictive accuracy in the certainty group. To accomplish this goal, we chose values for T_s so that the certainty group accuracy on a validation set was 100%. The chosen values for T_s were then evaluated on a dedicated test set. The results of these experiments can be found in tables 3 and 4. A certainty group size of $X\%$ means that $X\%$ of all instances-prediction pairs fulfill the certainty criterion and thus can be seen as "safe". We do not give the balanced accuracy scores in this case, as this metric looses its expresiveness when very few mispredictions (1-2 wrong predictions versus hundreds of correct) occur. Our concept was able to extract 26.30% to 50.61% of all predictions, with them having an accuracy



(a) Dropout (b) MIMO

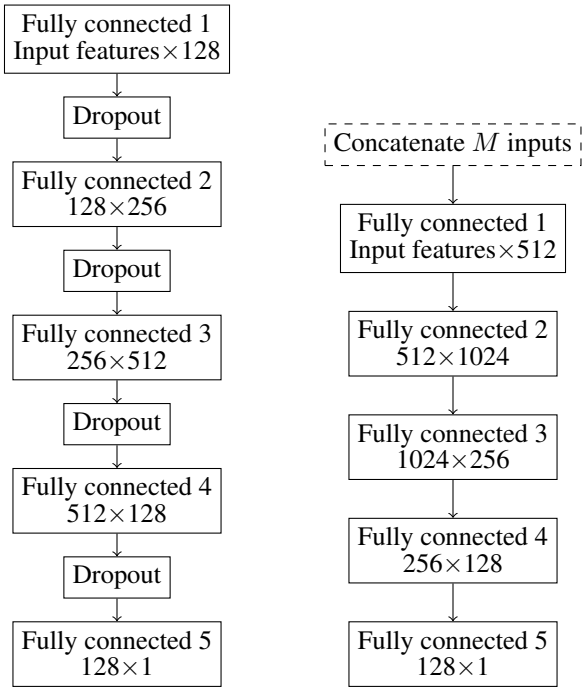
Figure 4. CNN based architectures for both approaches.

Table 2. Accuracies of the models used for the Certainty Group Computation

Dataset	Dropout Model	MIMO Model
AI4I	Acc.: 84.0%; BA 87.06%	Acc.: 80.74%; BA: 79.13%
FordA	Acc.: 91.66%; BA: 91.73%	Acc.: 93.06%; BA: 93.02%

Table 3. Certainty group sizes on the test set when tuned to 100% accuracy on the validation set. If no accuracy score is presented there were no mispredictions

Dataset	Dropout Model	MIMO Model	Confidence
AI4I	26.3%	47.1% with Acc. 99.78%	42.2 %
FordA	45.30% with Acc. 99.66%	50.61% with Acc. 99.25%	54.69% with Acc. 98.61%



(a) Dropout (b) MIMO

Figure 5. Fully connected architectures for both approaches.

score of 99.25% to 100.0%. It can be seen that the targeted accuracy of 100% in the certainty group is not always achieved. Especially the MIMO models were not able to translate this accuracy from the validation set to the test set. If perfect predictions in the certainty group are a strict requirement, a more conservatively chosen value for T_s may be necessary. On all data sets it can be seen that the corresponding normal groups have a clearly worse predictive accuracy. For comparison with the dropout and MIMO approaches, we used the raw confidences of the dropout architecture to compute certainty groups by only accepting predictions with a confidence ≥ 0.99 or ≤ 0.01 into the certainty group. We chose these values for T_{lower} and T_{upper} because we saw them as a natural choice for viewing predictions as "safe" if no uncertainty estimation strategy is implemented. Using this method, our concept extracted 42.2% and 54.69% of all predictions, with them having an accuracy score of 98.61% and 100.0%. It outperformed the dropout approach on both datasets and was able to outperform the MIMO approach on the FordA dataset, however with a slightly lower accuracy. In section 4.1 we take a closer look at this result.

3.2.2. The 98% accuracy requirement for CGs

To demonstrate the flexibility of our approach, we performed additional experiments in which we tuned T_s on an already trained model for ca. 98.0%-98.5% accuracy in the certainty group. For the confidence approach, we tuned the margin thresholds T_{lower} and T_{upper} until the desired accuracy was reached. This was done to answer the question of *how does the target accuracy for Certainty Groups affect their size*. It is especially interesting for use cases that don't require a maximum amount of certainty. The results of these experiments can be found in Table 5. It can be seen that the certainty groups contain 50.6% to 83.40% of all instances. These predictions have accuracy scores ranging from 96.7% up to 98.33%. With more instances and false predictions in the certainty group, we added balanced accuracy scores for these experiments to put the accuracy scores into perspective. This is important to interpret the MIMO model's performance

Table 4. Corresponding Sizes and test set accuracies of the normal groups to the certainty groups from Table 3.

Dataset	Dropout Model	MIMO Model	Confidence
AI4I	73.7%, Acc.: 76.11%	52.8%, Acc. 70.88%	57.8%, Acc.: 72.31%
FordA	54.70%, Acc. 87.81%	49.39%, Acc. 88.19%	45.3%, Acc.: 83.27%

Table 5. Certainty Group Sizes on the test set when tuned to ca. 98% Accuracy on the validation set.

Dataset	Dropout Model	MIMO Model	Confidence
AI4I	50.60%, Acc. 98.22%, BA 99.09%	56.49%, Acc. 98.93%, BA 74.73%	78.0%, Acc. 98.33%, BA 95.37%
FordA	75.60%, Acc. 97.99%, BA 97.99%	83.40%, Acc. 96.73%, BA 96.73%	68.78%, Acc. 97.57%, BA 97.44%

AI4I data set In this case, the approach was able to extract large portions of the predictions into the certainty group, but showed flawed predictive performance w.r.t one data set class. In these experiments the naive confidence approach behaved slightly different compared to the experiments in which we aimed towards 100.0% accuracy in the certainty group. While it previously performed best on the FordA data set, it now performs worst. On the AI4I data set however the confidence approach was able to outperform the other approaches by a significant margin. The dropout approach was always able to keep both the balanced and the standard accuracy score on a similar level and performed as desired or even better.

3.2.3. Evaluation of intersection

We performed the following experiments because we were interested in the agreement of the different approaches, i.e. *do different methods detect the same instances in Certainty Groups?* That would be an indication that some instances are inherently harder or easier to predict, perhaps due to the available measurements, quality of data, and non-ambiguity. A real-world example would be the difference in "hardness" between a very sharp, high-resolution image and a blurry, very low-resolution image. We assumed that if an instance is placed into the certainty groups by *each* approach, it really must be a certain prediction and an easy-to-classify instance. Another perspective on this assumption is the difference between *aleatoric* and *epistemic* uncertainty. While epistemic uncertainty is a result of insufficient model capability, aleatoric uncertainty arises directly from randomness and/or variablities of the underlying data source (Murphy, 2022). If instances are accepted to the certainty group by multiple approaches, they have a low aleatoric uncertainty. For quantifying the agreement between the different approaches, we developed a metric that measures how many of the instances in the smallest certainty group are contained in the common certainty group consisting of instances that have been accepted by every approach. The metric measures the ratio of the overlap of all certainty groups and the size of the smallest certainty group. We assumed that it's the best case if all accepted instances of a weaker approach (w.r.t. the maximum possible certainty group size) are part of the overlap of all certainty groups. We call this metric *intersection* and it is defined as following:

$$\frac{Overlap}{|\min(CG_{Dropout}, CG_{MIMO}, CG_{Confidence})|} \tag{9}$$

The described best case scenario leads to an intersection score of 100%. The intersection results for all two datasets are shown in Table 6. The results were calculated using the results of the 100% target experiments. It can be seen that in all two datasets the smallest certainty group is contained to at least 75% in the overlap group. This suggests that the combination of established approaches for uncertainty estimation and the concept of certainty groups is able to extract instances from an underlying dataset that are inherently easier and clearer to predict than other instances.

4. Discussion

4.1. Analysis

A conclusive assessment of all the approaches evaluated is difficult to make as the approaches show different strengths and weaknesses. If we only consider the sizes of the certainty groups, the MIMO approach was the best performing although it is outperformed by the confidence approach on one data set both in the 100% and 98% target experiments. Despite showing the greatest potential in our initial experiments, the MIMO approach had some problems when taking all results into account. It was the only approach that was not able to produce certainty groups with no mispredictions on a held-out test set when tuned on a validation set. Additionally, in the 98% target certainty group the balanced accuracy was not good in one of the two datasets. Its greatest asset in our experiment was the extremely fast execution speed as it only requires one inference pass and its good performance in the 100% target certainty group.

The fact that the confidence approach worked so well was surprising for us. We have therefore carried out further analysis with the framework introduced in (Küppers, Kronenberger, Shantia, & Haselhoff, 2020) to get a clearer picture of the

Table 6. Certainty Group sizes for each approach and dataset and the resulting intersection scores. The Certainty Group sizes are the same as in Table 3 since the same models and certainty criteria were used.

Dataset	Dropout CG	MIMO CG	Confidence CG	Intersection
AI4I	26.30%	47.10%	42.20%	76.42%
FordA	45.30%	50.61%	54.69%	75.91%

actual model calibrations. All models we used for the confidence approach were significantly miscalibrated. This finding collided with our hypothesis that a model has to be well calibrated in order to work well with this approach. It even appears that miscalibration in the margins used for the certainty criteria is beneficial for this approach as it helps to avoid mispredictions when the used margins are large (e.g. confidence $>$0.7 or $<$0.3). If a model is perfectly calibrated, predictions at confidence 0.75 should be 75% accurate. If the model is massively miscalibrated in this region (e.g. confidence 0.75 yields 100% accuracy), this helps in the setting of certainty groups. Overall, this approach seems to be very dependent on the model calibration, which poses risks for real-world applications.

The MC-dropout approach was the most consistent performing. Although it was not able to consistently outperform the other two approaches w.r.t. certainty group sizes, it always worked as expected across all datasets and experiments. This is underlined in the 98% target scenario, in which the dropout approach outperformed both other methods w.r.t. accuracy and/or balanced accuracy. On the AI4I dataset, its balanced accuracy score was clearly better, while on the FordA dataset it outperformed the accuracy score of the other two methods. However, a major disadvantage of this approach is the slow execution speed that was clearly noticeable in the experiments.

In summary, the confidence approach entails risks for real world operation as the model calibration plays a massive role in the creation of the certainty groups. The MIMO approach is able to generate relatively large certainty groups but works best when $>$99.5% accuracy is targeted in the certainty group. The dropout approach is overall very consistent but suffers from slow execution speed.

4.2. Possible applications

The results have made us optimistic that our approach is suitable for several applications in PHM. It can be applied to use cases in which a clear and concise decision is needed for an ML-system to add value. The results show that our approach is able to extract a subset of instances with highly accurate predictions. This can enable a transition towards increased automation of data based decisions in industrial applications since not all decisions are put into the hands of an ML-based system. It thus represents a compromise between conservative and progressive data analytics approaches.

The experiments show that our approach is able to extract subsets of the dataset with a higher predictive accuracy compared to the whole dataset. Depending on the setting, our approach is also capable of making models with mediocre predictive quality at least partially useful (e.g. MIMO on AI4I with the 100% target). In situations where, for example, the data source or the available hardware does not allow a more powerful model, this could prove useful. This could enable multi-level architectures in which data points "flow" through multiple levels of machine learning models. A conceivable use case is the distributed IT architecture in manufacturing plants in which very small computational units (the so called "edge") represent the lowest level of devices and high performance computer systems either in the cloud or in local data centers represent the highest level of computational capability. Our approach could allow very small but not well performing models to be deployed as close to the production line as possible. In this stage, predictions in the certainty group are considered "clear" and safe to proceed without further supervision. The instances in the normal group can then flow towards higher levels of computational capability in which the same process is repeated with more capable ML methods. At the end stands either a completely automatic decision or human supervision for very hard cases. This concept can be interpreted as an iterative reduction of *epistemic uncertainty* as increasingly capable models are applied to the underlying problem.

4.3. Limitations and future work

Despite showing potential for several use cases (as described in section 4.2), our approach in its current state suffers from a couple of limitations, which yield potential for further research of the presented concept.

Currently, our approach does only work with binary classification. In initial experiments on multiclass problems conducted at an early stage of research, we found that the expressiveness of the predictive standard deviation is lower than in a binary setting. We plan to investigate other dispersion metrics, perhaps information-theoretic ones based on entropy or KL-divergence. Using our collected knowledge, an adoption of certainty groups for multiclass classification would be the next logical step. A variation of the problem in the form of one versus all would be conceivable as in this case a binary aspect would still be present.

Despite the possibility of tuning T_s such that the accuracy on

a validation set is 100%, our approach cannot guarantee perfect predictions in a real world application. As shown in the results, it is still possible that all ensemble members perform wrong predictions with a very similar confidence that leads to a containment in the certainty group. Additionally, our approach does not take distribution shifts into account. In a real world application in which distribution shift naturally occurs (e.g. through wear and tear of multiple machine components), some type of out-of-distribution (OOD) detection would be necessary to prevent false predictions in the certainty group.

So far we have tuned T_s manually to result in a desired validation accuracy for the certainty group. An automatic solution for this task would be another next logical step, perhaps using Bayesian optimization (Mockus, 2012) or other optimization techniques. This could also involve requirements for the certainty group that are more complex than only accuracy requirements. An optimization with regards to very specific predictive behavior could be implemented by more complex certainty and optimization criteria. A conceivable example is the avoidance of false negative predictions in a predictive quality setting. In this case, a false negative prediction (defect not detected) is much more problematic than a false positive prediction. An optimizing solution could look for a certainty criterion on the uncertainty measure ζ_m that matches the target metrics for the certainty group.

While we only considered neural networks in this work, our approach is not limited to them. An in-depth comparison of the certainty group behavior of the used neural networks to Bayesian approaches (e.g. Gaussian processes or Laplace approximations) as hinted in Sect. 2.1 would be very interesting. So far we have evaluated our approach only with relatively small architectures. We have also not investigated the impact of the actual model architecture (e.g. convolutional, fully connected, recurrent) onto the quality and size of the certainty groups. Therefore, a direct comparison between the smaller architectures and architecture types we used to significantly larger architectures and different architecture types would add to the understanding of the presented concept. As the confidence approach showed serious potential, the application of calibration techniques like in (Guo et al., 2017) or (Tomani & Buettner, 2021) could result in interesting findings and are worth investigating.

5. Conclusion

In this paper, we introduced the concept of certainty groups that divides the predictions of neural networks into a normal group and the certainty group. Through an additional hyperparameter, the quality of the predictions in the certainty group can be controlled, which can enable interesting applications in manufacturing applications relevant to prognostics and health management. We evaluated three approaches for computing certainty groups, with an approach based on MC dropout performing most consistent despite not generating the largest certainty groups. We further showed that the three approaches accept very similar instances into the certainty group. Our approach is characterized by its simplicity and practical motivation as it combines advanced uncertainty estimation techniques with a clear certainty criterion.

References

Bagnall, A. (2022). *Time Series Classification.* Retrieved from http://www.timeseriesclassification.com/

Bingham, E., Chen, J. P., Jankowiak, M., Obermeyer, F., Pradhan, N., Karaletsos, T., ... Goodman, N. D. (2018). Pyro: Deep Universal Probabilistic Programming. *Journal of Machine Learning Research*.

CertaintyGroups. (2022). *Certainty Groups Code on Anonymous GitHub.* Retrieved from https://anonymous.4open.science/r/CertaintyGroups

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, *16*, 321–357.

Daxberger, E., Kristiadi, A., Immer, A., Eschenhagen, R., Bauer, M., & Hennig, P. (2021). Laplace redux - effortless bayesian deep learning. *CoRR*, *abs/2106.14806*. Retrieved from https://arxiv.org/abs/2106.14806

Dua, D., & Graff, C. (2017). *UCI machine learning repository.* Retrieved from http://archive.ics.uci.edu/ml

Frankle, J., & Carbin, M. (2018). The lottery ticket hypothesis: Training pruned neural networks. *CoRR*, *abs/1803.03635*. Retrieved from http://arxiv.org/abs/1803.03635

Gal, Y., & Ghahramani, Z. (2016). *Dropout as a bayesian approximation: Representing model uncertainty in deep learning.*

Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural networks. In *International conference on machine learning* (pp. 1321–1330).

Harshavardhanan, S., & Nene, M. J. (2020). Capturing and modeling uncertainty in prognostics and health management using machine learning. In *2020 5th international conference on communication and electronics systems (icces)* (pp. 1235–1241).

Havasi, M., Jenatton, R., Fort, S., Liu, J. Z., Snoek, J., Lakshminarayanan, B., ... Tran, D. (2020). Training independent subnetworks for robust prediction. *CoRR*, *abs/2010.06610*. Retrieved from https://arxiv.org/abs/2010.06610

Ismail, M., Mostafa, N. A., & El-assal, A. (2021). Quality monitoring in multistage manufacturing systems by us-

ing machine learning techniques. *Journal of Intelligent Manufacturing*, 1–16.

Küppers, F., Kronenberger, J., Shantia, A., & Haselhoff, A. (2020, June). Multivariate confidence calibration for object detection. In *The ieee/cvf conference on computer vision and pattern recognition (cvpr) workshops.*

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Lee, S., Purushwalkam, S., Cogswell, M., Crandall, D. J., & Batra, D. (2015). Why M heads are better than one: Training a diverse ensemble of deep networks. *CoRR*, *abs/1511.06314*. Retrieved from http://arxiv.org/abs/1511.06314

Liu, J. Z., Lin, Z., Padhy, S., Tran, D., Bedrax-Weiss, T., & Lakshminarayanan, B. (2020). Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. *CoRR*, *abs/2006.10108*. Retrieved from https://arxiv.org/abs/2006.10108

Matzka, S. (2020). Explainable artificial intelligence for predictive maintenance applications. In *2020 third international conference on artificial intelligence for industries (ai4i)* (pp. 69–74).

Mockus, J. (2012). *Bayesian approach to global optimization: theory and applications* (Vol. 37). Springer Science & Business Media.

Molchanov, P., Tyree, S., Karras, T., Aila, T., & Kautz, J. (2016). Pruning convolutional neural networks for resource efficient transfer learning. *CoRR*, *abs/1611.06440*. Retrieved from http://arxiv.org/abs/1611.06440

Murphy, K. P. (2022). *Probabilistic machine learning: An introduction*. MIT Press. Retrieved from probml.ai

Naeini, M. P., Cooper, G., & Hauskrecht, M. (2015). Obtaining well calibrated probabilities using bayesian binning. In *Twenty-ninth aaai conference on artificial intelligence.*

Phan, D., Pradhan, N., & Jankowiak, M. (2019). Composable effects for flexible and accelerated probabilistic programming in numpyro. *arXiv preprint arXiv:1912.11554*.

Salvatier, J., Wiecki, T. V., & Fonnesbeck, C. (2016). Probabilistic programming in python using pymc3. *PeerJ Computer Science*, *2*, e55.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(56), 1929–1958. Retrieved from http://jmlr.org/papers/v15/srivastava14a.html

Tehrani, N., Arora, N. S., Li, Y. L., Shah, K. D., Noursi, D., Tingley, M., ... others (2020). Bean machine: A declarative probabilistic programming language for efficient programmable inference. In *International conference on probabilistic graphical models.*

Tomani, C., & Buettner, F. (2021). Towards trustworthy predictions from deep neural networks with fast adversarial calibration. In *Thirty-fifth aaai conference on artificial intelligence* (Vol. 3).

Weiss, M., & Tonella, P. (2021). Uncertainty-wizard: Fast and user-friendly neural network uncertainty quantification. *CoRR*, *abs/2101.00982*. Retrieved from https://arxiv.org/abs/2101.00982

Wen, Y., Tran, D., & Ba, J. (2020). Batchensemble: An alternative approach to efficient ensemble and lifelong learning. *CoRR*, *abs/2002.06715*. Retrieved from https://arxiv.org/abs/2002.06715

Zhang, L., Lin, J., Liu, B., Zhang, Z., Yan, X., & Wei, M. (2019). A review on deep learning applications in prognostics and health management. *Ieee Access*, *7*, 162415–162438.

Zhao, R., Yan, R., Chen, Z., Mao, K., Wang, P., & Gao, R. X. (2016). Deep learning and its applications to machine health monitoring: A survey. *CoRR*, *abs/1612.07640*. Retrieved from http://arxiv.org/abs/1612.07640

Zhu, M., & Gupta, S. (2017). To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*.

BIOGRAPHIES

Lukas Lodes grew up in Bavaria, Germany. He studied computer science at the University of Augsburg (Germany), where he obtained his Bachelor's and Master's degree. His master thesis focused on the application of graphical neural networks for the prediction of CFRP manufacturing results. He is currently working as a research assistant and PhD student under the supervision of Alexander Schiendorfer at THI, the Technical University of Applied Sciences Ingolstadt.

Alexander Schiendorfer grew up in Upper Austria where he also obtained his Bachelor's degree in Software Engineering at the UAUaS in Hagenberg. He then went on to get a joint Master's degree in Software Engineering from the University of Augsburg, TUM (Technical University of Munich), and LMU (the University of Munich). After receiving his doctoral degree in Computer Science at the University of Augsburg in 2019 and two years as a postdoctoral scholar, he accepted a professorship at Technische Hochschule Ingolstadt (THI). His research interests lie in the application of combinatorial optimization and machine learning to the manufacturing domain – in particular constraint programming, probabilistic machine learning, and reinforcement learning. He is a member of the Association for Constraint Programming (ACP) and won the University of Augsburg doctoral dissertation award in 2019.