Faculty of Engineering and Management

Engineering and Management

Bachelor's thesis

Comparison of Uncertainty Quantification Methods in Deep Learning with Respect to Computation Resource Limited Applications

Zübeyir Oflaz

Issued on: 20/04/2022

Submitted on: 20/09/2022

First Examiner: Prof. Dr. Alexander Schiendorfer

Second Examiner: Prof. Dr. Jürgen Bock

Table of Contents

Declaration

I hereby declare that this thesis is my own work, that I have not presented it elsewhere for examination purposes and that I have not used any sources or aids other than those stated. I have marked verbatim and indirect quotations as such.

Ingolstadt, Zübeyir Oflaz

List of Tables

List of Equations

List of Abbreviations:

AUROC: Area under receiver operating characteristics curve

CNN: Convolutional neural network

ECE: Expected calibration error

LA: Laplace Approximation

MIMO: Multi-Input Multi-Output

NLL: Negative log likelihood

RMSCE: Root mean square calibration error

Abstract

The ability of deep learning methods to create models that can recognize complex patterns with a relatively small amount of effort quickly led to their adoption in many real-world applications. However, the inability of these models to provide accurate uncertainty estimation regarding their results has hampered the many possible use cases for these models, and quantifying the uncertainty of their predictions has received many possible solutions from the academic world. This thesis investigated two such methods, namely Laplace Approximation and Multi Input Multi Output (MIMO) ensemble, in their ability to provide meaningful uncertainty estimation and create accurate predictions with near certain confidence under resource limitations. During this investigation, MIMO models have consistently outperformed the models with Laplace Approximation but have proven to be much harder to get a viable working model. Because of this, the ease of application for the Laplace Approximation method was still found to be an appealing solution when practicality is a serious concern.

Introduction

Deep learning has shown rapid growth in both the number of techniques developed and their possible applications and quickly became the state-of-the-art machine learning solution in many different fields (Pouyanfar et al. 2019). Their ability to map non-linear and complex patterns, which are sometimes not even known by people, with minimal supervision brought breakthroughs in many fields. The strengths of deep learning methods led to the quick adaptation of these methods to many real-world applications (LeCun et al. 2015).

However, the complex and non-linear structures of neural networks come with significant drawbacks such as non-interpretable decision making, inability to grasp what is actually learned by the neural network, and unexpected changes in network predictions due to extremely small changes in the input of a network. Moreover, most neural networks

provide point estimates which provide little value for gauging the uncertainty of a network, and the training process of a neural network often leads to miscalibrated networks with over or underconfident predictions (Guo et al. 2017b).

The aforementioned drawbacks of neural networks make them unfeasible for many applications that would provide great value for the user and/or society. What is even more worrying is the adaptation of these models in critical areas when they are not yet viable, which leads to unserviceable results (Roberts 2021) or even harmful consequences (Babic et al. 2021).

In order to expand the scope of application for and prevent the possibly harmful adaptation of deep learning, the development of practical methods that can provide better information regarding the uncertainty of a neural network is needed. These methods also need to be computationally feasible for their adaptation outside of the safety-critical applications where computational resources are limited. There has been a very large number of academic literature published with this goal in mind(Abdar et al. 2021), but the amount of further investigation that is done on these approaches is very limited, and our understanding of their limits and drawbacks is not yet established.

This thesis aims to investigate two of the promising methods that are being used for uncertainty estimation of neural networks, namely Laplace approximation (Azevedo-Filho and Shachter; Mackay 1995) and multi-input multi-output (MIMO) methods (Havasi et al. 2020), with models that can be run on systems with limited computational resources. The performances of these two methods has been be compared using three different datasets, their outputs have been analyzed and visualized to gain insights into their characteristics, and they have been evaluated by their ability to classify inputs with high level of confidence.

In order to evaluate the performance of these methods within a given resource limitation following analyses have been conducted: Runtime and memory requirements of the Laplace Approximation and MIMO models have been provided, and the changes in these requirements for different model configurations have been presented. The trade-off between model accuracy and flops/macs required by the models was also further investigated for one of the datasets.

Background Information

In order to grasp the task of creating reliable neural networks, a basic understanding of uncertainty is needed. There are two different types of sources of uncertainty that exists. Uncertainty can be the result of the inherent randomness of the issue that is being observed, our inability observe all factors related to the issue or the imperfections of the tools that we use in order to make the observations. In this case, the uncertainty is classified as aleatoric uncertainty. The uncertainty that is the result of inadequate knowledge about the observed process or inadequate data is categorized as epistemic uncertainty. The total uncertainty that is achieved from the combination of these two categories is called predictive uncertainty (Goodfellow et al. 2016).

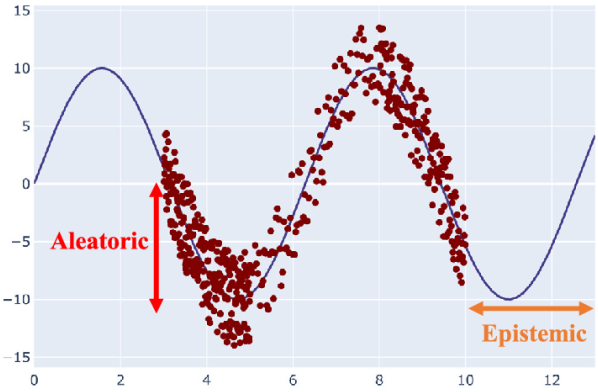


Figure 1 A schematic view of the main differences between aleatoric and epistemic uncertainties. Retrieved from Abdar et al. 2021 under Creative Commons license

In the case of epistemic uncertainty, the model performance can be improved by gathering more data or improving the model, but aleatoric uncertainty cannot be reduced by doing so since the uncertainty observed is the inherent property of the data.

In an ideal world, the decision-making process of a machine learning model would be understandable and the results of a neural network would be predictable and reflect the predictive uncertainty that exists. To achieve this four overlapping groups of challenges can be targeted: absence of theory, absence of casual models, sensitivity to imperfect data, and computational expenses (Abdar et al. 2021). In many of the areas where deep learning is being used we either lack a complete understanding of the observed process, or the creation of a machine learning method that can also be understood is not feasible. The non-linear and complex connections of any sizeable neural network make it nearly impossible for us to understand and describe the decision-making process of a model, though there are novel approaches being tried in order to create explainable AI (XAI) using neural networks (Tjoa and Guan 2019).

In some cases creating a conventional neural network that is well calibrated to provide reliable results can be achieved, but most of the time this is not realistic. The number of hyper-parameters that need to be explored by a machine learner is very high and this quickly becomes computationally intractable. One can assume that a random hyper-parameter search would be able to maximize the performance but this only happens in a small fraction of cases. Deep neural networks are particularly difficult to train and calibrate with many hyper-parameters to choose from. If one has enough data and time, it is possible to find a good model by means of cross-validation. However, in real world applications, most of the time this is not a viable option due to the computational cost of this approach.

However, neural networks can be made more reliable by employing methods like Laplace approximation or using special neural network structures like MIMO to reflect aleatoric and epistemic uncertainties with their results (Humt 2019; Havasi et al. 2020). In this thesis these methods will be applied to multiple datasets that show significant differences in their characteristics in order to gain insight into the strengths and weaknesses of these methods.

Laplace Approximation

A conventional neural network functions by approximating the posterior of a probability distribution by simply finding one single likeliest point under the posterior during the training process. This single point estimation is called maximum a posteriori (MAP). However, the MAP estimate does not capture the uncertainty around the maximum probability point called θ. Thus, it leads to model assigning wrong confidence levels in its prediction. Moreover, when such a network is given input wildly different from the instances in the training dataset it cannot recognize this fact and make overconfident estimates. Additionally, minute variations in the input data can lead to the model drastically changing its results and making wrong predictions.

The Bayesian Neural Networks offer significant improvements regarding these problems. First suggested in the ‘90s (Mackay 1992; Bickel et al. 1996), Bayesian neural networks (BNNs, Bayesian NNs) offer a probabilistic interpretation of deep learning models by inferring distributions over the models’ weights. The model offers

robustness to over-fitting, uncertainty estimates, and can easily learn from small datasets.

Bayesian NNs place a prior distribution over a neural network's weights, which induces a distribution over a parametric set of functions. Often a standard matrix Gaussian prior distributions over the weight matrices are placed for the layers in order to get a probabilistic distribution from the network rather than a single point result.

Despite their strength, BNNs haven't gained a lot of notoriety for which their difficulty in implementation, scaling, tuning along with the high cost of training are attributed (Daxberger et al. 2021).

Laplace Approximation (LA) promises to provide Bayesian probabilistic results with the use of conventional neural networks, thus granting the benefits of BNNs without the associated cost and challenges. With Laplace approximation, a Gaussian distribution is constructed around the mode of the posterior distribution through conventional model optimization, where the covariance matrix is the inverse curvature around the mode (Humt 2019). It can be used for hyperparameter selection while designing a model as well.

While application of LA to the whole model is still impractical for deep neural networks, approximation using novel hessian matrices (Martens and Grosse 2015) or layer-wise approximations in which LA is only applied to a selected layer still provides significant improvements for model calibration and uncertainty estimation (Daxberger et al. 2021).

Multi Input Multi Output Ensembles

The main idea of multi input multi output (MIMO) networks comes from the bigger literature regarding ensemble methods. In ensemble methods, the outputs gained from multiple neural networks are combined in order to create a range of predictions which can be seen as probabilistic distribution and can provide more reliable results over a single neural network (Hansen and Salamon 1990).

The use of multiple different neural networks for robust predictions has been an established method in the literature for quite some time and some derivatives of this approach have already proven significant benefits. However, the need for doing multiple

forward passes in an ensemble for each input increases the runtime and computational resources needed for the prediction.

MIMO method (Havasi et al. 2020) aims to overcome this disadvantage of ensemble methods by creating the ensemble members inside of a single neural network. If a MIMO model has n number of ensemble members, n different inputs are concatenated to create one input and fed into the neural network and n different predictions are extracted during the training phase of the neural network. During the evaluation phase, the same input is multiplied n times and concatenated, and the n results of the network are averaged to give the final result. By doing so, multiple results for an ensemble are obtained using a single pass.

Conventional neural networks can be pruned 70-80% without hurting their' performance (Frankle and Carbin) due to the overparameterization of neural networks. MIMO increases the information density contained in a neural network by fitting multiple ensemble members into the same network without, in theory, the need to increase the size of the neural network. This not only reduces the chance of overfitting, but also provides a further decrease in the run cost of a network.

Datasets

This thesis utilized three datasets during the evaluation in order to benchmark the two uncertainty methods. This section provides a short description of each datasets, their significant characteristics, their important characteristics and any processing that is applied to the datasets during their usage.

Arrhythmia Dataset

Arrhythmia dataset (Guvenir, H., Acar, Burak & Muderrisoglu, Haldun 1998) contains the information of 452 cardiac patients and their diagnoses. The aim is to distinguish between the presence and absence of cardiac arrhythmia and to classify it in one of the 16 groups. Class 0 refers to 'normal' ECG cases where no arrhythmia is detected, 01 to 14 refers to different classes of arrhythmia and class 15 refers to arrhythmia cases that don't belong to the 15 specified classes. The exact distribution of the number of observations per class is given in Figure 2. The specific names of these classifications are added to the Appendix section.

The dataset contains 279 different features for each observation; the first four are age, sex, height, and weight of the patient, and the rest include the data from different channels of an electrocardiograph machine and features derived from these data.

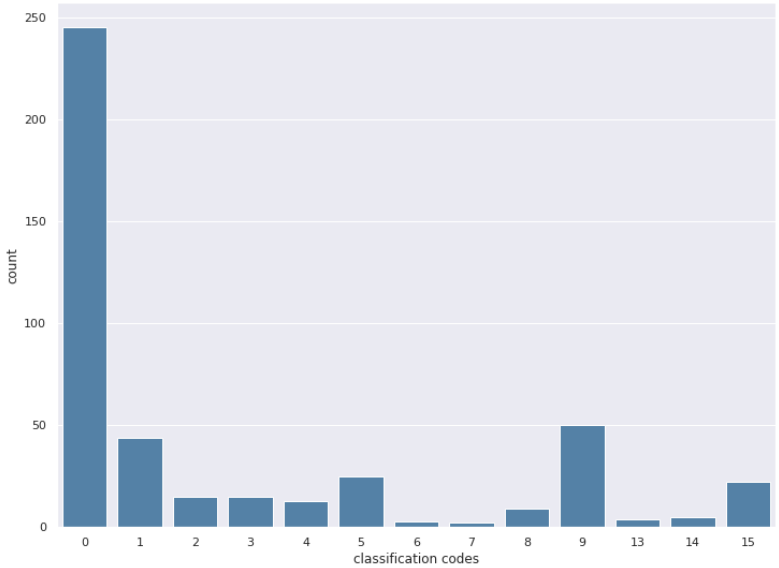


Figure 2 Number of observations in each class for the arrhythmia dataset

There are several notable characteristics of this dataset that makes it applicable for real-world uses and a real challenge for uncertainty estimation. The first of them is the unbalanced number of observations for each class. A significant number of observations belong to a small number of classes while the rest of the classes only contain

a handful of observations which is in line with the Pareto Principle. In fact, as it can be seen from here, the normal class contains half of all observations while three of the classes that are assigned (classes 10 to 12) don't have any observations associated with them. Such an unbalanced dataset tends to create neural networks that make poor predictions on the classes with low number of observations, and are overconfident on predictions related to classes with many observations.

The second important characteristic of this dataset that makes it valuable is the structure of data. The features contained in the dataset vary wildly from each other, and contain categorical and numerical data, some continuous features signify simple linear relationships while the others contain complex patterns (Guvenir et al. 1997). A lot of data points have missing values and some of the columns in the dataset don't differ in value between all of the observations as well. Because of these reasons, this dataset has been investigated in quite a significant number of academic papers (Dua and Graff 2017).

As can be seen from the figure below, the data range for different features differ markedly, which makes it more challenging for a neural network to train even more. The vast difference in the range also causes any simple scaling and normalization of the data to cause the loss of features. Since there are around 23 different types of features represented in the 279 different columns, creating a custom normalization or a creating custom-made neural network that contains different inputs for each feature set is highly impractical and would require the input of an expert from this field.

In order to combat the imperfections mentioned above, several efforts have been made while pre-processing the data. While the simple scaling of the dataset was leading to poor results, the Robust Scaler that is provided by the Scikit library (Pedregosa et al. 2011) displayed much better results. Robust Scaler removes the median and scales the data according to the quantile range (defaults to IQR: Interquartile Range). Centering and scaling happen independently on each feature by computing the relevant statistics on the samples in the training set. In addition to Robust Scaler, normalization of each feature separately from each other has also been done in an effort to improve model performance (these methods have been applied separately and not together).

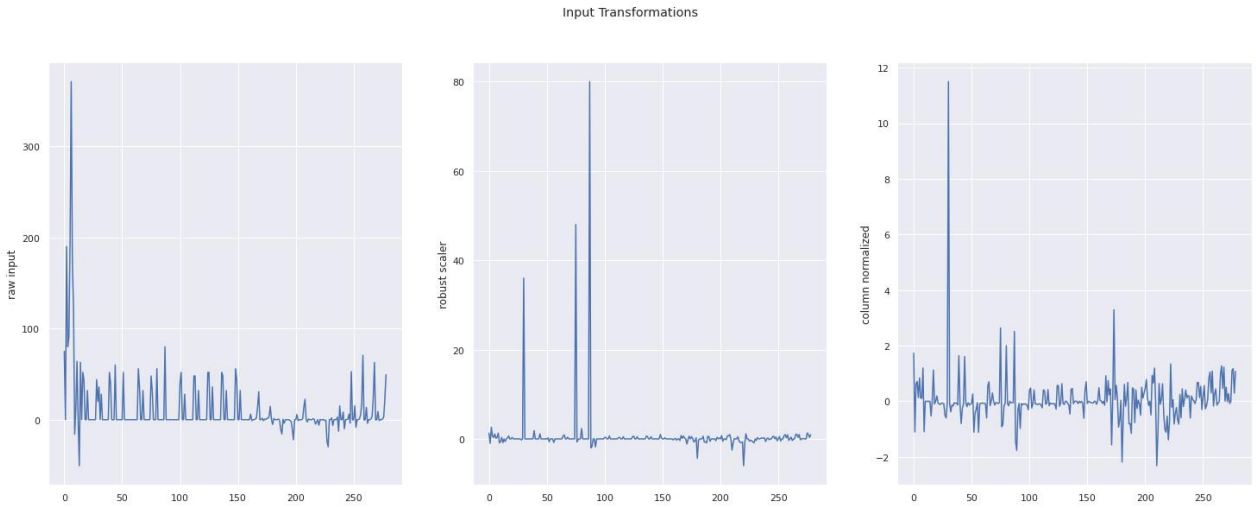


Figure 3 The line plot of an input. Left subplot shows the data without any change, and two remaining subplots represent the data after the application of Robust Scaling and normalization respectively

Figure 3 shows the line plot of the same input after each type of transformation. The missing value data points have been set to the median value of that column before any scaling or normalization and the columns that has the same value for all the observations simply set to zero. Removal of these columns and empty classes can assist in the creation of slightly smaller networks, but not removing shouldn't affect the model performance in any significant manner.

Casting Dataset

Casting dataset is a labeled visual dataset for the identification of defects during the production. It contains the top view photos of submersible pump impellers. These products are manufactured via the casting process some of which suffer from undesired irregularities in a metal casting process (Dabhi 2019). The dataset provides only binary classification even though there are many types of defects that can be observed in casting like blow holes, pinholes, burr, shrinkage defects, mold material defects, pouring metal defects, metallurgical defects, etc.

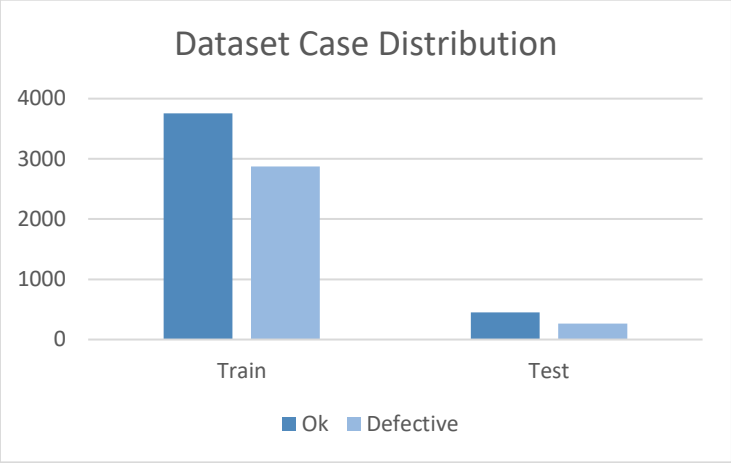


Figure 4 The distribution of images per group and class for the casting dataset

The dataset source contains 300x300 photos of impellers to which data augmentation via rotation and flipping of the images is already applied. The dataset contains somewhat more ok images than defect images as can be seen from figure 4 here. The ratio of ok to defective samples is preserved between training and test sets.

There are multiple reasons for the choice of this dataset for the thesis. Firstly, it applies very well to a real-world use case for neural networks during manufacturing, and it is used to create a working system for automated defect detection (Patel 2020). For removing defective products, companies need to have a quality inspection department and the inspection process is carried out manually. Inspection is a very time-consuming process and due to human errors, it is also not completely accurate.

Secondly, the detection of these defects is a relatively simple process for which a simple neural network with limited resources would be suitable. However, the very high accuracy of neural networks for this task was also a significant problem. Since even a small network was able to get more than 99% accuracy on the task measuring the real uncertainty of the network would not have been possible. Because of this, two different image transformations, namely motion blur and Gaussian filter that can be observed in manufacturing conditions are applied to the dataset. 75% of the images transformed with motion blur while the remaining images received Gaussian blur.

The images are further processed via Dataloader transformations during which they received random cropping and normalization. Finally, images are shrunk to 128x128 pixel size for the base and Laplace approximation models, and 64x64 pixel size for MIMO models. An example of these transformations can be seen in figure 5.

Figure 5 Casting dataset transformations. From left to right, it displays original image, motion blur, Gaussian blur, and final transformation result.

The reasoning behind the application of these transformations were to create input images that have artefacts which reflect issues that can be observed in a manufacturing setting where images can suffer from blurring due to the speed of a manufacturing line or problems with the camera, or not always provide perfectly centered input images due to the positional variations of the product or mistiming during the automated taking of the photo.

Ford A Dataset

Ford A dataset is a binary classification problem for time series data. It is used to detect whether a certain problem exists in an automotive subsystem or not. This dataset contains 500 measurements of engine noise from the sensors per observation and the corresponding label for the data (Bagnall 2008).

This dataset has also been chosen for its' practical use cases and doesn't require high computational power for significantly high classification accuracy. It also provides a different benchmark for comparison since time series data presents unique challenges to a neural network.

As it can be seen from figure 6 the distribution graph, instances per class are nearly at the same number in both train and test sets. The data from the sensors seem to be processed already and the application of scaling or normalization didn't lead to any significant improvements in the performance.

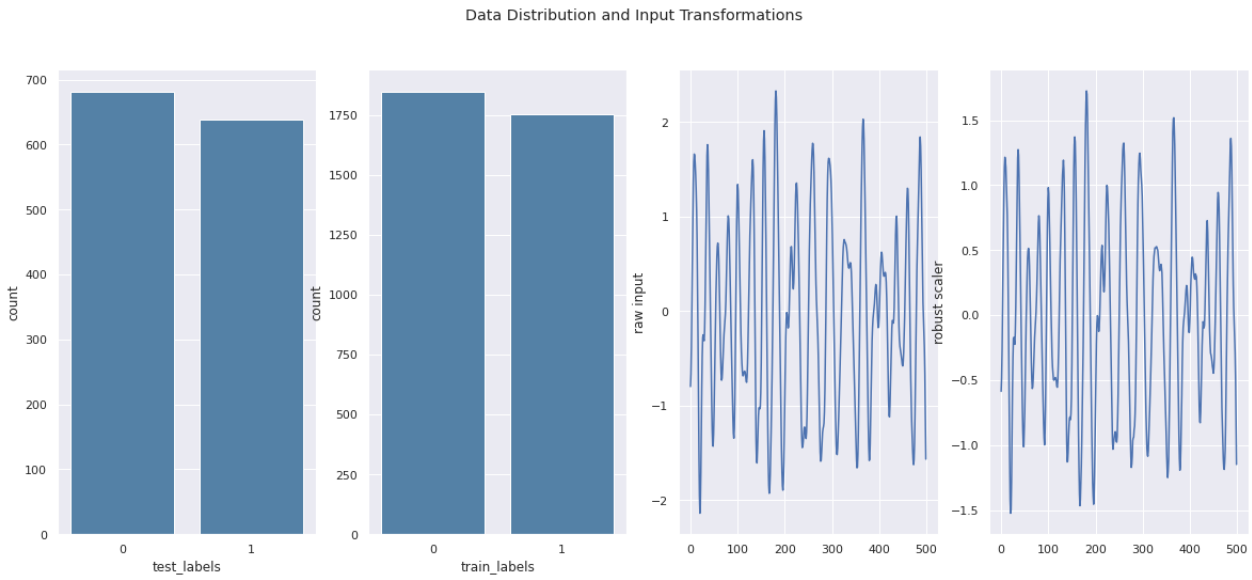


Figure 6 the class distribution of Ford A and Input Data Plot with and without scaling

Another interesting part of this dataset is the existence of FordB dataset where the same classification goal is tried to be achieved using the same sensor data with a noisy environment. This provides future study opportunity with regards to measuring robustness of the models and detection of out of distribution (OOD) inputs by the neural network.

Methodology

This section gives further information with regards to how the model selection was achieved, what methods were used to compare models with each other, and the libraries that were used during the implementation of the thesis and presentation of the results. By doing so, it provides transparency to how this paper come to conclusions as well as a level of reproducibility with regards to the results. The scripts that were used in order to create the models and measure the results can be obtained from the related GitHub repository (Oflaz 2022).

Model Design and Hyperparameter Selection

The model creation phase aimed to find the models with the highest accuracy in the test set within the given restrictions on the number of layers and number of neurons. Two different model selection for all the datasets was needed, one conventional neural network for the use of Laplace Approximation and one MIMO model. Finding two different types of models that perform very well for three dissimilar datasets was a significant challenge. In order to minimize the time needed to manually design models as well as the computation resources needed in order to achieve the best results, the Optuna library was used.

Optuna is an open-source hyperparameter optimization library that provides the implementation of multiple hyperparameter search algorithms with simple and dynamic construction of hyperparameter search spaces (Akiba et al. 2019). It provides easy usage of sampler algorithms for automated selection and testing of hyperparameters. With this library, a user can specify which hyperparameters and search spaces will be selected by the sampler during the creation and training of a neural network. Later, an optimization study is created during which the chosen sampler assigns values all the variables for each trial, and the success rate of the trial is recorded. Then this information can be used during later trials by the sampler in order to reduce the search space for finding the best performing neural network.

For the testing of Laplace Approximation, Optuna was used in order to find the conventional neural network with the highest accuracy rate, and Laplace Approximation was applied to this model. MIMO models were directly created by Optuna and received no further modification.

The Tree-structured Parzen Estimator (TPE) was used as the sampler in all of the Optuna studies that were created. TPE algorithm is a Bayesian optimization method that uses Parzen window estimators as the statistical model for density estimation. When an optimization study is created specified number of initial trials are conducted during which the variables are chosen randomly. By looking at the performances of these random trials, Parzen estimator can create density functions for each variable from individually

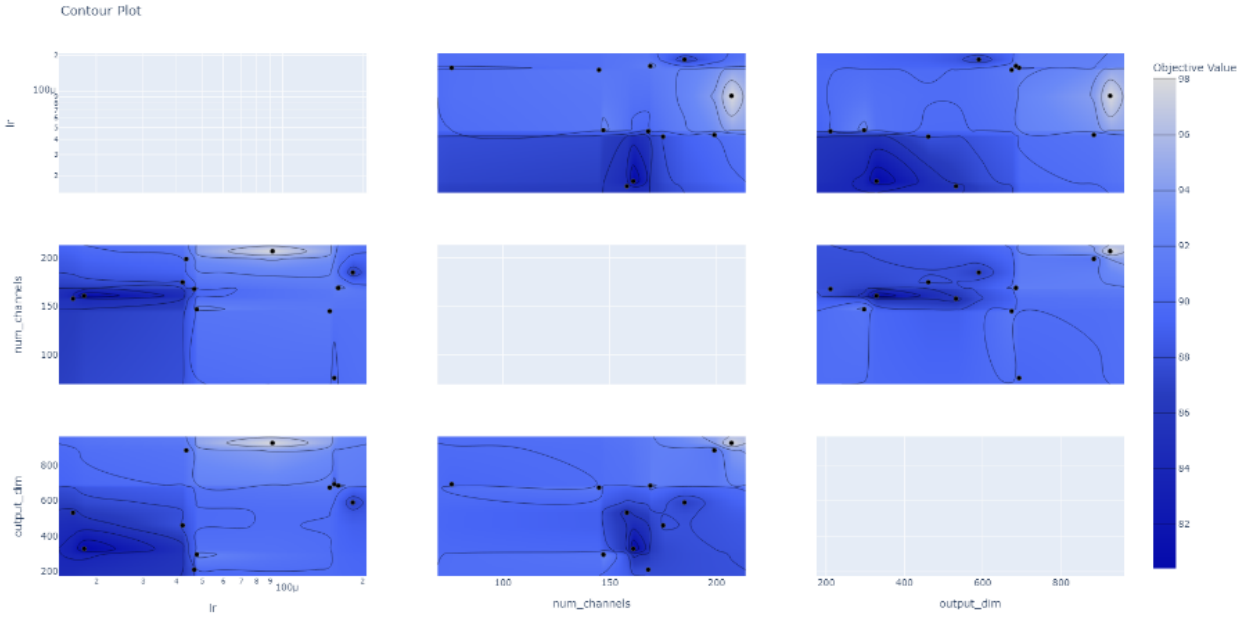


Figure 7Contour map of some of the variables used in a study. A more comprehensive contour map is added to the Appendix

continuous variables or discrete sets. These density functions give an idea to the sampler about the range of hyperparameter values that would be most likely to give the best results with respect to our Bayesian optimization problem. The range of the variables is truncated to give either Gaussian mixtures or weighted categorical sets and the remaining trials are conducted within these ranges in order to find the best set of hyperparameter (Bergstra et al. 2011). Figure 7 shows the contour map of different variables related to each other, with respect to the objective (in this case accuracy) which is a helpful visualization for seeing the relationship between different variables. A

more detailed version of this plot is added to the Appendix in order to provide better information.

Optuna also provides an effective pruning algorithm that stops trials that are not promising early. By doing so, the TPE sampler can quickly hone in on the optimal hyperparameter sets. A significant drawback of this sampler is its' inability to know or recognize the relationships among the variable set that is sampled. A later variation of TPE sampler introduced multivariate TPE sampling that can recognize the dependencies between variables (Falkner et al. 2018). A number of the studies that were done during this thesis employed multivariate TPE sampling.

Some of the trials that were conducted for this thesis contained around a dozen variables for optimization, and only 40 or so trials (20 of which were random trials) were needed until Optuna was able to achieve near maximum accuracy which is a very impressive result. However, there were significant drawbacks observed during the use of Optuna. The first important problem was the tendency of the TPE sampler to get stuck around local optima when it is given an unbalanced dataset and/or too big of a search space. The conventional solution of increasing the number of random trials and conducting multiple studies where often unhelpful, especially during the creation of MIMO models. In order to overcome this issue, a number of strategies were employed for two of the datasets which will be discussed in a later section.

The second problem observed during the usage of Optuna was the false pruning of promising models which again led to suboptimal final results. Again, this was a bigger problem for MIMO models where the accuracy curve was not linear, and more complex models were more prone to stay around a certain loss/accuracy level for a few epochs before showing improvements. This can be observed with the line plot in figure 8 which shows the accuracy levels of MIMO models during a study. Pruner frequently caused early stopping of more complex models in favor of simpler models that showed quick improvements in accuracy but only achieved suboptimal results in the end. In order to combat this issue, custom pruning algorithms were created that were better suited at identifying unpromising trials to the datasets they were applied to.

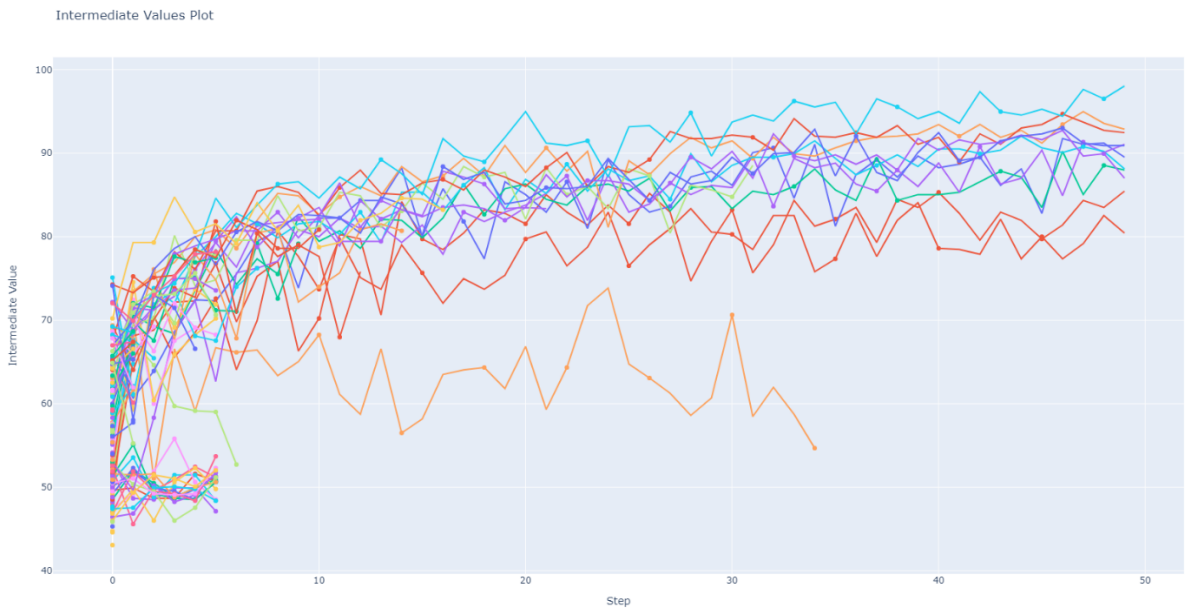


Figure 8 Accuracy graph of MIMO models in a study which highlights their nonlinear progress. X axis represents the epoch and Y represents the test set accuracy as a percentage

After the discovery of optimal models, Laplace Redux library was used in order to get the Laplace Approximation models from the conventional neural networks (Daxberger et al. 2021). With this library, the user can specify the neurons that will be the target of Laplace Approximation for which the library fits a Gaussian curve around. Then the method of optimizing the parameters of these Gaussian curves is given which is used to refine these distributions.

Quantifying Uncertainty and Related Work

This section will provide information regarding the metrics that are used in order to compare the performances of models between Laplace approximation and MIMO methods. It also describes the methods that are used in order to select certainty groups, which are the subset of predictions that can be considered having near certain accuracy, and related academic works that were used for certainty group determination and quantifying the divergence among samples or ensemble members. The more detailed description of certainty groups will be provided later on.

The process of evaluating model performances was twofold: Firstly, the model performances were benchmarked using well-established evaluation metrics, and secondly, their performance in classifying instances with very high certainty and the amount of divergence between different examples were analyzed.

The first evaluation metric that will be used will simply be model accuracy. The second metric that will be used is negative log-likelihood (NLL). NLL, which is also referred to as cross entropy loss in the context of deep learning, quantifies the loss of a neural network as

$$NLL = -\sum_{i=1}^{n} \log(\breve{\pi}(y_i|x_i))$$

Equation 1 Negative log likelihood

Where $\breve{\pi}(Y|X)$ denotes a probabilistic model and n denotes the number of samples (Guo et al. 2017a). For NLL lower value indicates better model performance. The third metric is Expected Calibration Error (ECE) which aims to quantify the difference between the confidence of a neural network's prediction and its accuracy. Expected Calibration Error ECE approximates this difference by partitioning predictions into M equally-spaced bins and taking the weighted average of the bins' accuracy/confidence difference (Guo et al. 2017a).The equation for Expected Calibration Error is as follows:

$$ECE = \sum_{m=1}^{M} \frac{|Bm|}{n} |Acc(Bm) - Conf(Bm)|$$

Equation 2Expected calibration error

Where n denotes the number of samples in a dataset, Acc(B_m) denotes accuracy of the particular bin and Conf(B_m) denotes the confidence of the bin. Root Mean Square Error which is another calibration error metric is also provided to evaluate performance (Detlefsen et al. 2022). RMSCE formula is as follows:

$$RMSCE = \sum_{m=1}^{M} \frac{|Bm|}{n} \sqrt{Acc(Bm) - Conf(Bm)}$$

Equation 3Root Mean Square Error for calibration

All the notations in RMSCE represents the same concepts in ECE. For both of the calibration errors lower value indicates more reliable model output.

Lastly, Area Under Receiver Operating Characteristics Curve (AUROC) is given as one of the evaluation metrics. Receiver Operating Characteristics Curve show the sensitivity/specificity trade-off of a classifier for all possible classification thresholds, and AUROC gives the total area under this curve. The value range for AUROC lies between 0.5 and 1, where 0.5 signifies a classifier with completely random outputs and 1 signifies a perfect classifier.

For the second portion of the evaluation two methods have been utilized. The first method was a variation of Kullback-Leibner Divergence proposed in the paper Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles (Lakshminarayanan et al. 2016) in which the divergence value Div of ensemble member n for a given input i can be explained as:

$$Div_{in} = \sum_{c=1}^{C} |m_c - p_{cn}|$$

Equation 4Divergence of an ensemble member

Where m_c is the mean value of probability among ensemble members for class c, p_{cn} is the probability p for the class c given by the ensemble member n, C is the total number of classes. The summation of all divergences among ensemble members as:

$$Div_i = \sum_{n=1}^{N} Div_{in}$$

Equation 5 Total divergence

Gives us the total divergence for instance i from all ensemble members. Same method is applied to the Laplace Approximation outputs where the divergence among different predictive samples were calculated and summed to give the total amount of variation seen among the samples.

This information regarding divergence provided three different functions during the evaluation. The first function was the measurement of disagreement observed among the different ensemble members in order to gain insights into the behavioral patterns of different ensemble members, and how much they contribute to the overall disagreement. Secondly, the total divergence enabled the observation of the correlation between divergence amount and model accuracy.

The second method used at this stage was the creation of certainty groups presented in the paper Certainty Groups: A practical approach to distinguish confidence levels in neural networks (Lodes and Schiendorfer 2022). This paper divides the network outputs into normal and certainty groups, where the members of certainty groups can be considered as having a very high probability of being classified correctly. The paper formalizes the definition of a certainty group as:

$$\mathbf{CG}\ (m, D) = \{(x, y) \in D \mid \boldsymbol{\phi}\ \text{accepts}\ \boldsymbol{\zeta m}\ (x \mid fm, \theta)\}$$

Equation 6 Certainty Group

CG refers to the certainty group for model m and dataset D, where the boolean predicate **ϕ** is applied to the uncertainty metric **ζm**. If the prediction from the neural network satisfies the determined uncertainty metric then the output is considered the part of the certainty group. The specific certainty group proposed in this paper is achieved by finding the mean and standard deviation of predictions done by a non-deterministic approach such as Laplace Approximation, or different ensemble members such as MIMO. The results of the neural network are investigated to find a threshold mean or standard deviation that gives a certainty group with an aimed level of accuracy.

The paper aims at creating a certainty group selection method that is practical requires little to no understanding of neural networks in order to apply it, and provides a reasonably large certainty group.

In this paper, the neural network outputs were processed and mean and standard deviation of each class. Sum and difference between these mean and standard deviation values were obtained in addition to the divergence amount as additional parameters as well.

The method of creating uncertainty groups were applied to the parameters that were showing promising patterns by finding the threshold value of certainty groups from the training dataset, and the accuracy and filtering rate of these certainty groups were reported for each dataset in the Dataset Analysis section.

Dataset Analysis:

This section will separate the results obtained during this thesis based on the dataset. In each part, the results for basic evaluation metrics will be provided, the neural network structures used to obtain these results will be shown, the certainty groups obtained using different metrics will be presented, and then the interesting outcomes will be further analyzed. Visual analysis of the neural network outputs is also provided in each section in order to gain further insights into the data distribution and analysis of outcomes.

Casting Dataset:

For the casting dataset, the overall results can be summarized as this. Laplace Approximation has shown modest improvement over the base model in many metrics and improved the overall accuracy. The MIMO model has provided the highest accuracy and AUROC in addition to providing remarkable results regarding the certainty groups even though it also has much higher NLL, ECE, and RMSCE compared to both base and Laplace models. The probable reasons for this outcome are discussed in-depth later on.

	Accuracy	NLL (↓)	ECE(↓)	RMSCE(↓)	AUROC
Base Model	0.9315	0.1577	0.0177	**0.0274**	0.9833
Laplace Model	0.9441	**0.1514**	**0.0162**	0.0342	0.9844
MIMO	**0.9622**	1.2001	0.3147	0.3154	**0.9910**

Table 1 The evaluation metrics results for the casting dataset

The neural network structures and the Laplace Approximation method used in order to gain these results have been added to the next page in figure 9 and below the figure. As it can be seen from the results in table 1, the Laplace Approximation provides an overall improvement with regard to the base model in nearly all metrics, and this was achieved with only a small amount of effort with the Laplace library. Because of this, it provides a very practical approach when it comes to calibrating an already trained network with minimal effort in order to get better results.

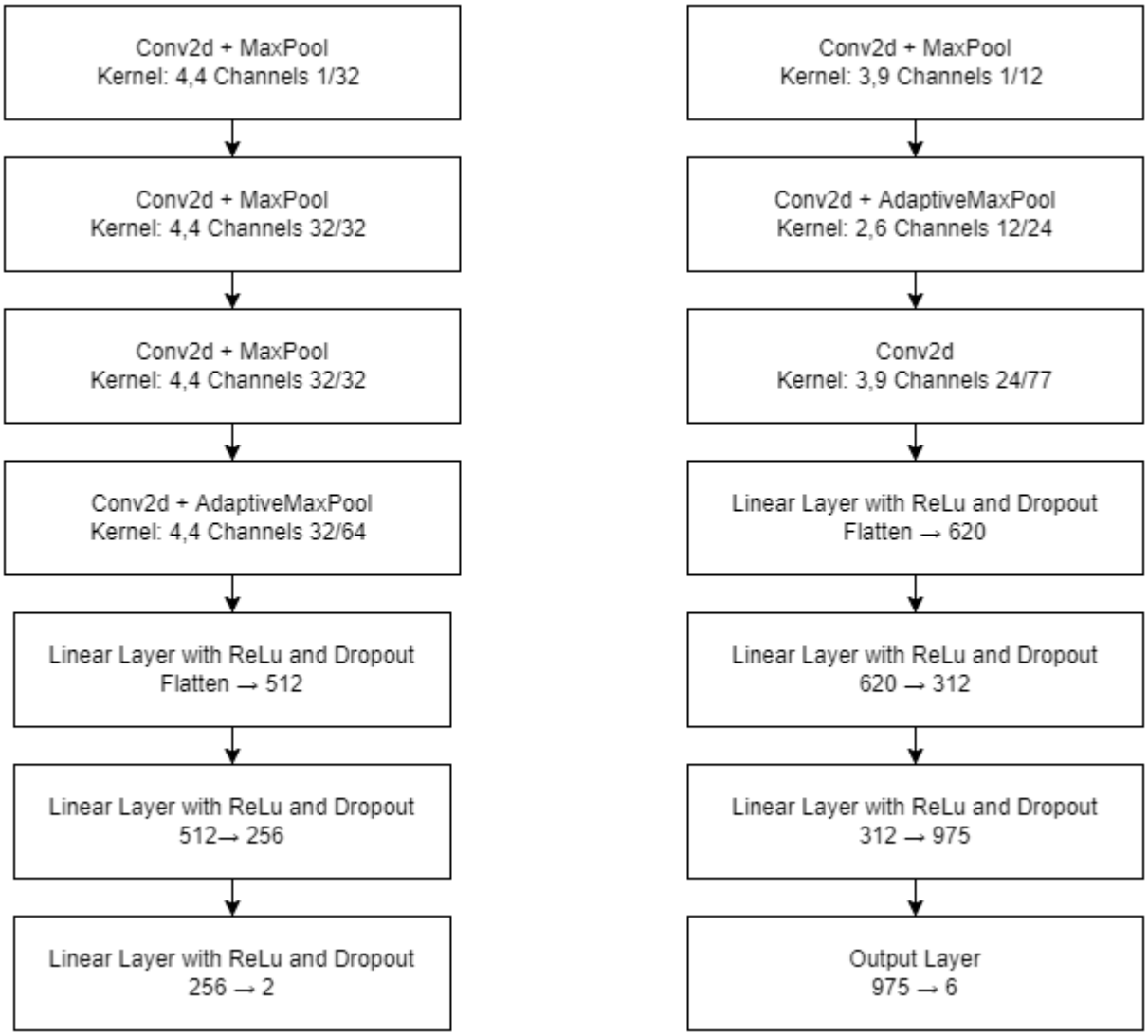


Figure 9 Neural Network Structures for the casting dataset. Base and LA model structure is presented on the left, and MIMO model is presented on the right

Model creation process summary: The base model receives the images in the size of 128x128 while the MIMO model gets 64x64 for each image due to the higher computational cost. MIMO model has three ensemble members and the three input images are concatenated sideways as the input.

The LA model is obtained by applying the Laplace Approximation to all the neurons for the last layer. Prior precision is optimized by applying neural network sampling predictive and cross validation. 100 predictive samples were obtained from the LA model when the confidence and standard deviation of observations were being calculated.

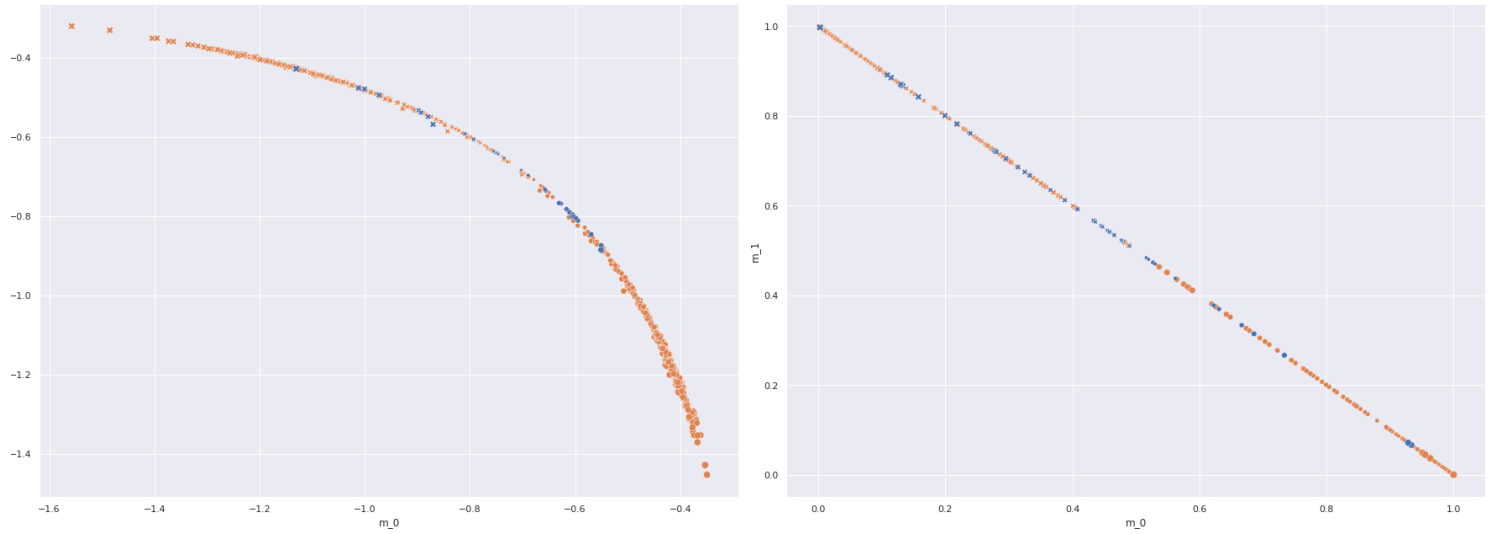


Figure 10 Class confidence distribution for the casting dataset. MIMO model on the left and LA model on the right

Figures 10 and 11 shows the mean confidence and standard deviation scatterplots for the casting dataset. m_0 and m_1 show the mean class confidences of models, and std_0 and std_1 give the standard deviation observed per class among ensemble members or samples. The MIMO outputs show the raw values from the log_softmax output layer of the model, while the LA model output is processed, and because of this sum of class means roughly always adds up to one, and the standard deviation per class is nearly identical to each other. In the figures of class confidence and standard deviation blue points signify false and orange points signify true predictions, and the size of the point signifies the total divergence for that observation with regards to its accuracy class. The different point styles present the predicted class for that observation when it is provided in the plot. These plots are obtained from the test dataset results.

The result plots for MIMO and LA models show a much better separation between true and false predictions in the case of the MIMO model, hence much more promising results when it comes to creating certainty groups.

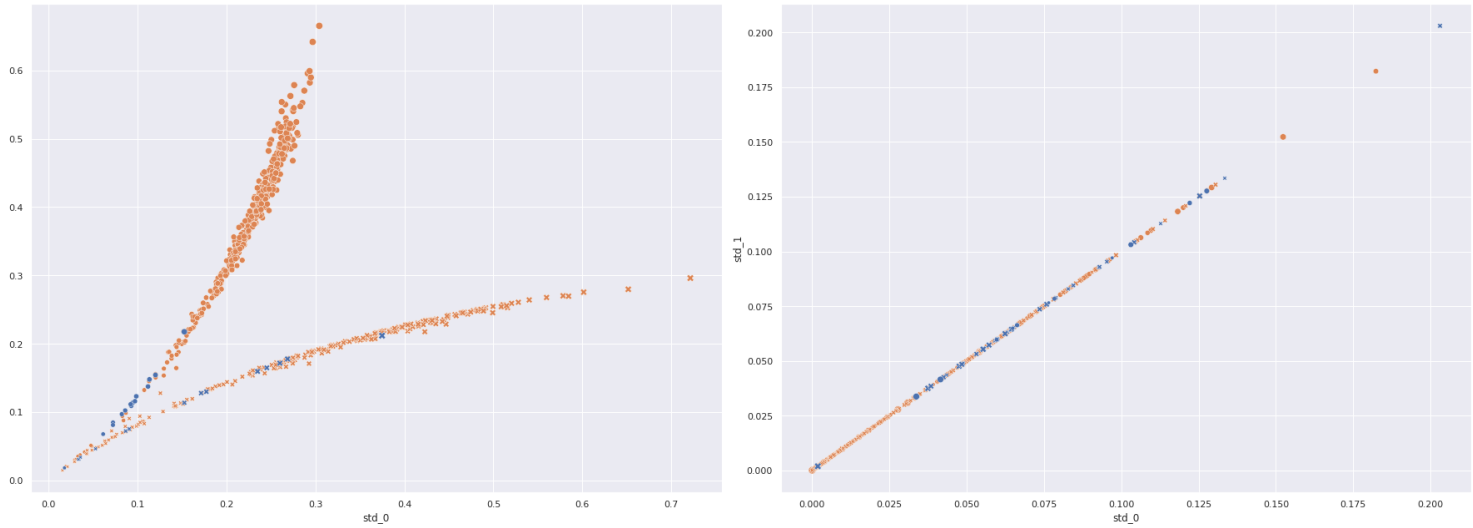


Figure 11 Standard deviation plot for the Casting Dataset. MIMO model on the left and LA model on the right

Figures 10, and 11 show a clear pattern where the prediction is much more likely to be correct as the difference between standard deviation and mean increases between classes in the case of the MIMO model. This is less likely so in the case of the LA model where some small number of false predictions lie at the end sections of the patterns.

The delta standard deviation doesn't provide a piece of significantly useful information in the case of the LA model, since the standard deviation value for each class is nearly identical between classes, and the highest difference is only around 1 x 10^{-8} value. While there is a clustering of false predictions around the mid-value for the standard deviation in the case of Laplace Approximation, the tail ends still show false predictions which shrinks the absolute certainty group when setting standard deviation as the threshold variable. Still, it is possible to get a sizeable absolute certainty group for the ok class with LA model.

	Sum Mean	Sum Standard Deviation	Total Divergence	Delta Mean	Delta Standard Deviation
Absolute Certainty	20.40%	24.33%	20.00%	55.80%	56.22%
>99% Certainty	89.93% (98.28%)	85.31% (99.01%)	85.59% (99.01%)	68.53% (99.79%)	64.19% (100%)

Table 2 Certainty groups for the casting dataset obtained by the MIMO model

	Total Divergence of Samples	Delta Mean	Standard Deviation
Absolute Certainty	-	48.67% (*)	-
Max Non-Absolute Certainty	98.0% (93.3%)	77.06%(99.09%)	73.7%(99.05%)

Table 3 Certainty groups for the casting dataset obtained by the LA model

Tables 2 and 3 provide the certainty groups obtained by setting thresholds using different parameters[1]. The values shown in parenthesis for the non-absolute certainty rows represent the accuracy rate of these uncertainty groups. Usage of some of the parameters utilized in the MIMO model was not possible to use in the case of the LA model. The reason for this was the different output layers between the models, which leads the sum of confidence means always being one and the standard deviation values being nearly identical to each other in the case of the LA model.

As it can be seen from these tables, MIMO model resulted with both highest absolute uncertainty and highest >99% certainty groups. Moreover, the divergence among the ensemble members was a meaningful predictor in the case of MIMO model, but the divergence obtained by the predictive samples of the LA model was a very weak predictor that didn't contribute any meaningful threshold for a certainty group.

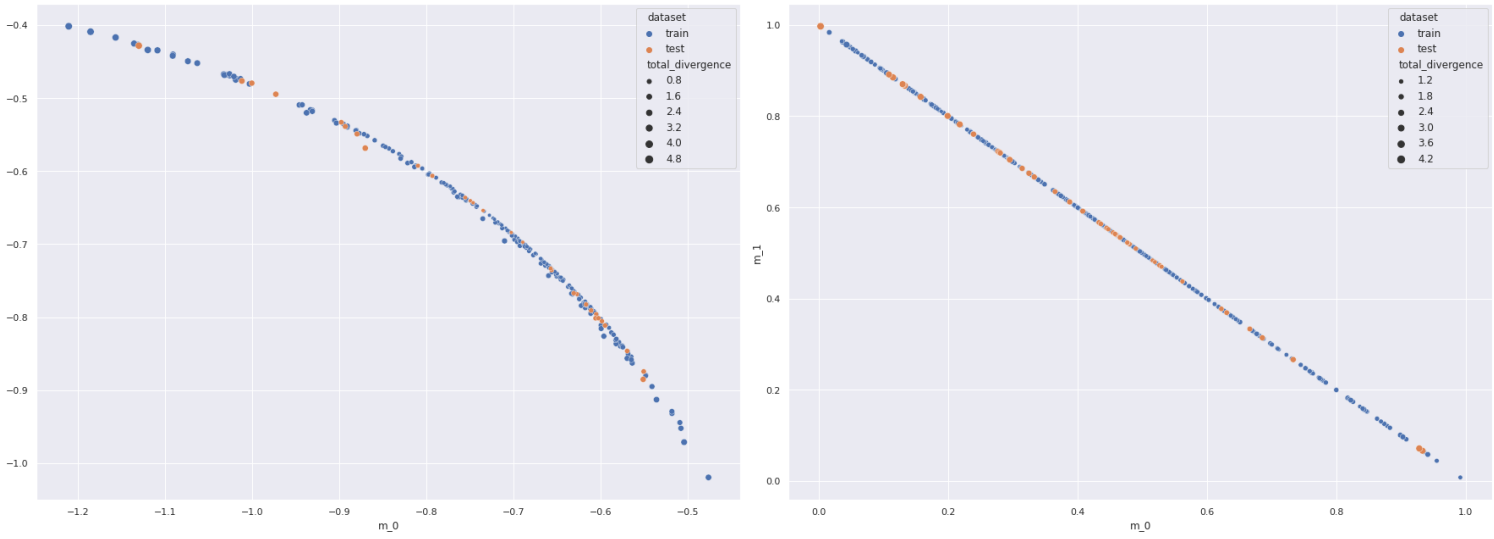


Figure 12 Mean confidences of false predictions per dataset. MIMO model on the left and LA model on the right

Figure 12 provides insight regarding the values of false predictions observed in the training dataset vs the test dataset with regards to mean class confidences. The false predictions occur with higher confidence values in the train dataset for the MIMO model, and this was also observed for the standard deviation values as well. Because of this, the certainty groups set by MIMO model was more conservative and has excluded

[1] * symbol next to the values in these tables indicate that this certainty group was obtained by increasing the certainty threshold by 5%

significant amount of instances that would've belonged to absolute certainty group if the certainty group was set with the test dataset threshold.

The problem was reversed for the LA model where test false positives sometimes shown the highest value. This was observed in the case of delta mean threshold for LA where a certainty group only became possible when the threshold was increased by 5% as it is indicated with the star next to the value in table 3.

As stated at the beginning of this section, the MIMO model resulted in the highest accuracy and AUROC as well as creating the biggest certainty groups, even though NLL and calibration errors from the MIMO model are remarkably larger. MIMO model being miscalibrated and showing high underconfidence in its predictions is suspected as the potential reason for this contradiction. The reason for this suspicion is the divergence patterns of model outputs. The total divergence amount of the model is much higher for correct predictions. Especially, one of the ensemble members consistently provides a much higher divergence than the other two for the correct predictions, while it shows a much smaller separation for the false predictions. This likely results in smaller confidence for the correct predictions which increases the NLL and calibration error levels. The ensemble member divergence is examined more closely in a later section specifically for MIMO models.

The miscalibration might also be the result of different sampling during the training of the MIMO model. As mentioned in the section on hyperparameter selection, Optuna very frequently favored models that always predict the class with more instances when the number of observations per class was not balanced. Because of this, the normal and defective classes were presented at an equal rate during the hyperparameter selection and training phase of the MIMO model. This might lead to higher levels of NLL can calibration error levels when the model is evaluated with the true dataset with the unequal class distribution.

It needs to be noted that all these possible reasons are speculations, the higher rates might also simply because these metrics need to be modified before applying to a MIMO model, and possible reasons need to be investigated for confirmation before any confident statement can be made. Still, the model with the highest accuracy and

certainty groups also providing the highest loss and calibration errors is a very interesting phenomenon and which might yield useful insights.

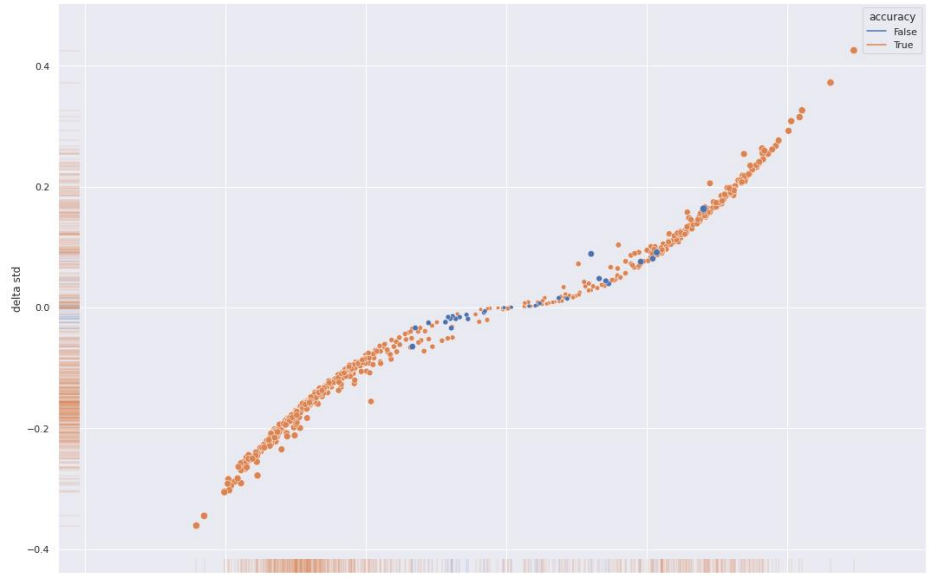


Figure 13 Confidence and Standard Deviation difference plot for MIMO Model for the casting dataset

Lastly, the plot in Figure 13 strongly indicates the creation of bigger certainty groups can be possible for the MIMO model by setting thresholds for multiple parameters. However, this removes the practicality of the certainty creation process proposed by Lodes et. al. and it makes the certainty group selection less predictable which is why this method was not tested within the scope of this thesis.

Arrhythmia Dataset

The results regarding this dataset can be summarized as follows. Laplace Approximation was able to increase the model performance partially and provided better results regarding NLL and RMSCE which can be observed in Table 4. AUROC metric was excluded from the results since the empty class instances caused an error during the calculation of this metric

	Accuracy	NLL (↓)	ECE(↓)	RMSCE(↓)
Base Model	0.9423	0.4200	**0.0381**	0.0958
Laplace Model	0.9423	**0.2758**	0.0684	**0.0131**

Table 4 Performances of Base and Laplace Models in the arrhythmia dataset

However, all attempts at creating a working MIMO model were unsuccessful for the arrhythmia dataset. Moreover, the Laplace model was able to classify the most common classes with near-perfect accuracy in both training and test datasets. This can be seen in the scatterplot in Figure 14. This plot provides the scatterplot for class 0, the normal

class, and class 9, right bundle branch blockage classes respectively. The x-axis shows the mean confidence for that class and the y-axis shows the standard deviation.

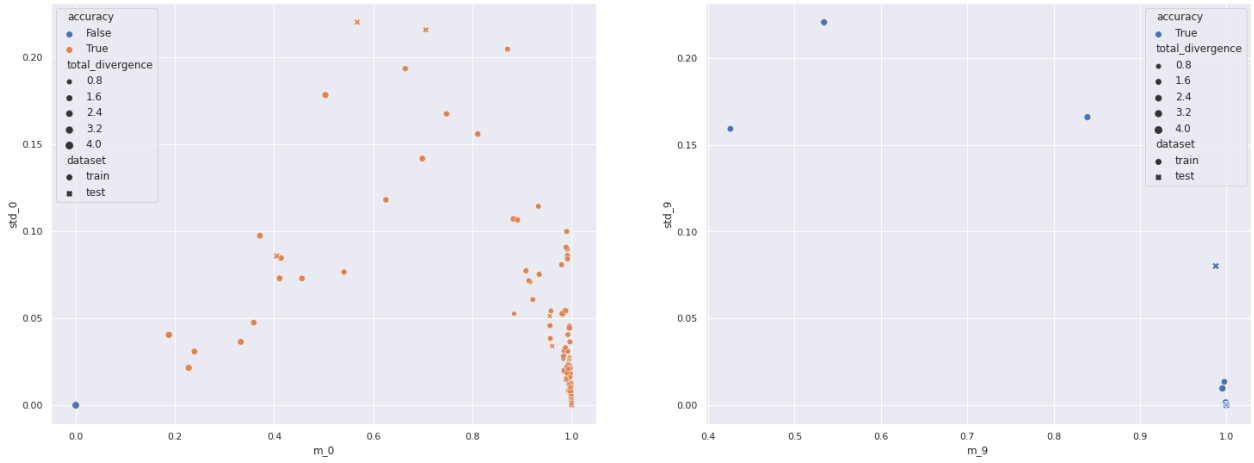


Figure 14 Mean and standard deviation plots for the class 0 and 9 in the arrhythmia dataset

The plot shows that only class 0 has 1 false prediction in the train dataset, and the remaining predictions for both classes in both datasets are all correct and display extremely high confidence. This issue resulted in the inability to find enough false predictions to create certainty groups. All the other groups were too small in size to have enough observations in both train and test datasets. Because of these reasons, the testing of certainty groups for this dataset was not possible to conduct.

The neural network that was used as a base for the Laplace Approximation can be seen in Figure 15. LA was applied to the last two layers of the neural network instead of only the last layer since the simplicity of this model allowed this with minimal additional resource requirement.

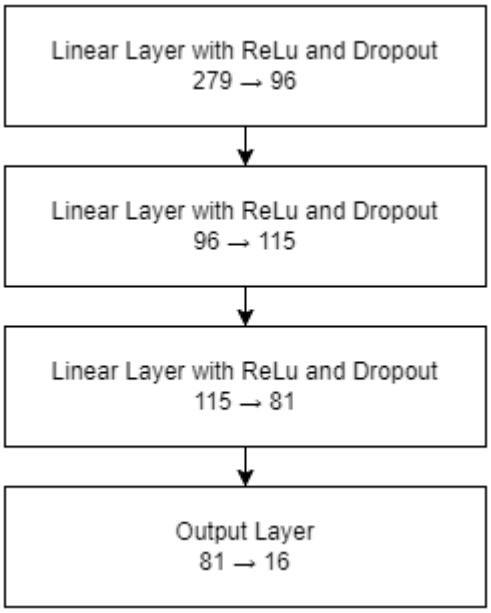


Figure 15 Arrhythmia base model network structure

Nearly all efforts in creating a viable MIMO model resulted in the creation of models that were either very good at classifying one or two of the classes, or models that only guess the most common class no matter the instance. The following strategies were tried in order to overcome this problem:

Manual model creation: Optuna had problems with MIMO models both in terms of getting stuck at some set of hyperparameters that give suboptimal results and false pruning. Because of this, manual model creation was tried with no success.

No resource limitation: Even though the arrhythmia base model is able to achieve its performance with a very little number of parameters (this can be seen in the later section with concrete numbers) some MIMO Optuna studies were given much bigger upper limits during their testing. This didn't end with success even though some of the models required nearly all 6GB memory of a GPU during training.

Convolutional early layers: A single observation of an arrhythmia dataset contains 279 different features and MIMO models multiply the number of input points which can require impractically large linear layers for performance. Because of this, the probability of needing convolutional layers as feature extractors to decrease the computational cost was seen as a possibility. However, Convolutional networks were also not able to achieve acceptable performance.

Uniform sampling: As mentioned earlier, MIMO models struggled with a slightly imbalanced dataset during experiments in the casting dataset, and it is very probable that this was causing issues for the experiments on this dataset as well. In order to combat this uniform sampling that was used in the casting experiments was also tried here in which the model is show roughly the same number of instances per class during training. This was also not successful, most probably because the observations that belonged to small classes were hundreds of times more likely to be shown than the instances in the normal class or other common classes. This led to the same instances being shown many times during a training epoch which was not helpful for model training.

Different data preprocessing: Even though the robust scaler was very successful in improving model performance for the base model, feature-wise data normalization was also tested in case the robust scaler was limiting the model performance which led to no performance improvement.

Multiple objective optimization: Since models were consistently getting stuck at the same accuracy, consideration of other metrics when evaluating model performance in

addition to accuracy was also considered as a potential solution and Optuna multi-objective optimization was used in order to test this hypothesis. Multiple objective optimization functionality lets users define multiple objectives and apply the TPE sampler for each objective (Akiba et al. 2019). The hyperparameter search space is narrowed by combining the optimal search ranges obtained from the TPE sampler for both objectives during the study. Additional metrics such as loss and ECE were used with accuracy during multi-objective studies, but this also did not result in a viable model.

The suspected reasons for this failure are the severe imbalance of class distributions and poor data quality, MIMO architecture needing a bigger dataset for such a task in order to effectively learn, or a possible error made during the application of the MIMO model for this dataset which is not found during investigations.

Ford A Dataset

The following observations were made regarding the performances of the MIMO and LA models for the Ford A Dataset. Laplace Approximation has provided significant benefits over the base model in all loss and calibration metrics. In contrast to the casting dataset, The AUROC score of MIMO model for the Ford A dataset was worse compared to base and LA models, while still having the issue of high NLL and calibration error levels. However, the base model used in LA has shown overfitting to the training dataset.

Even though LA was able to provide a slightly higher level of uncertainty this was not enough, and the issue of overfitting not only made the LA model extremely confident in its prediction but also made it impossible to get certainty group thresholds from the training dataset since only a handful of training instances were falsely classified.

More Optuna studies were conducted in order to find a better suited model, but all the best performing models across multiple studies were showing severe overfitting, even when both convolution and linear layers had high levels of dropout which led to the conclusion that Optuna was favoring models with high level of overparameterization.

To combat this issue multiple objective optimization feature was used of Optuna during which the study had the objective of maximizing accuracy while minimizing number of flops. Flops can be defined very simply as the number of operations that are done by the

	Accuracy	NLL (↓)	ECE(↓)	RMSCE(↓)	AUROC
Base Model	0.9311	0.1871	**0.0120**	0.0269	0.9812
Laplace Model	0.9311	**0.1834**	0.0134	**0.0264**	**0.9813**
MIMO	**0.9326**	0.9294	0.3542	0.3544	0.9783

Table 5 Performances of Base, Laplace and MIMO Models in the Ford A dataset

neural network for a single prediction. The result of this study was remarkable, leading to a neural network that was not only much smaller, but also behaved very similarly between test and train datasets for the fraction of accuracy loss. The results of this study was further investigated in the later section regarding model resources and limitations.

The table 5 shows the evaluation metrics obtained by this smaller base neural network, the Laplace Approximation applied to this network, and the MIMO model. The MIMO model again has a slightly higher accuracy but shows markedly larger NLL and calibration error, while Laplace Approximation provides a small improvement over the base model in most metrics. The neural network structures of these models can be seen in Figure 17 on the following page.

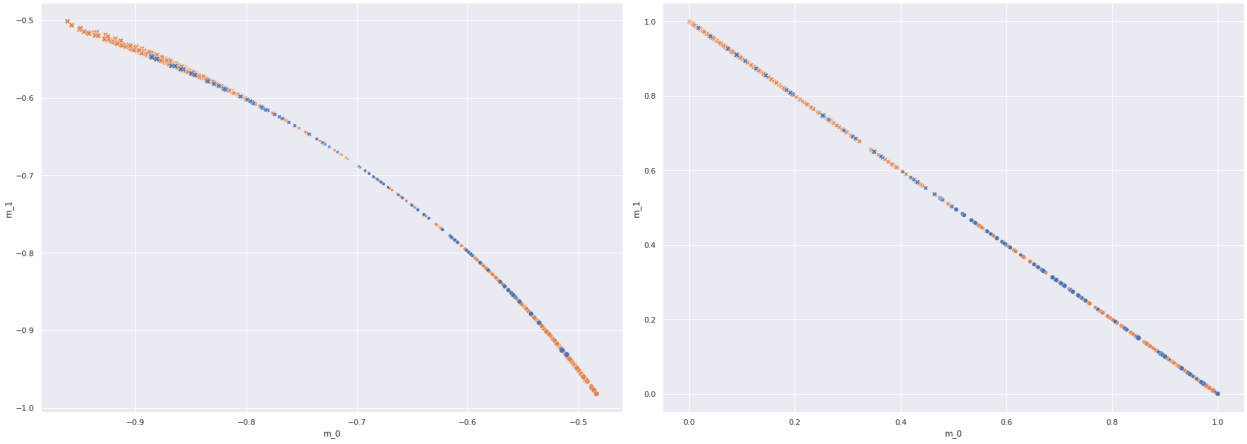
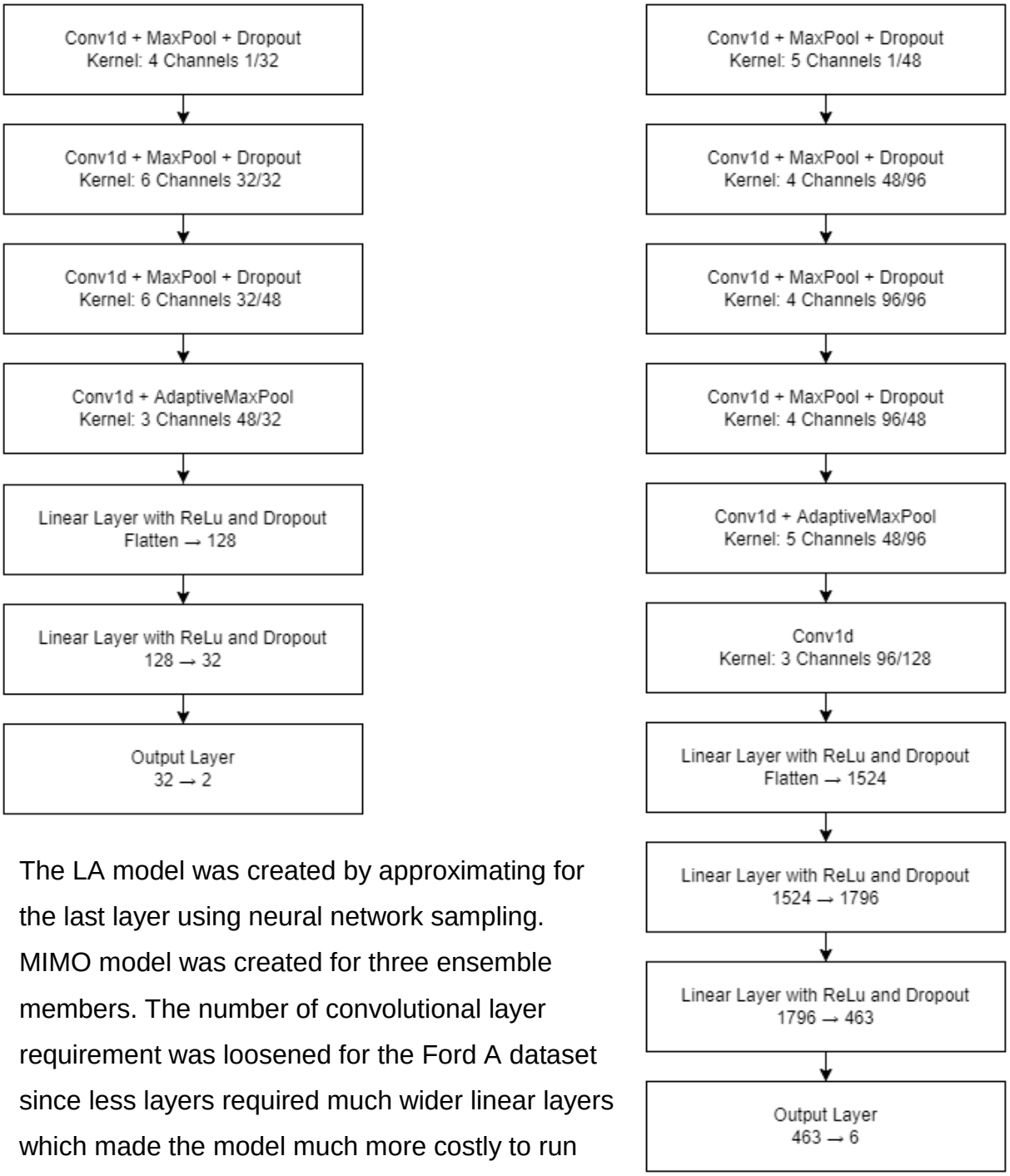


Figure 16 Class confidence distribution for the Ford A dataset. MIMO model on the left and LA model on the right



The LA model was created by approximating for the last layer using neural network sampling. MIMO model was created for three ensemble members. The number of convolutional layer requirement was loosened for the Ford A dataset since less layers required much wider linear layers which made the model much more costly to run

Figure 17Neural network structures of the Ford A dataset, base/Laplace model on the left MIMO model on the right

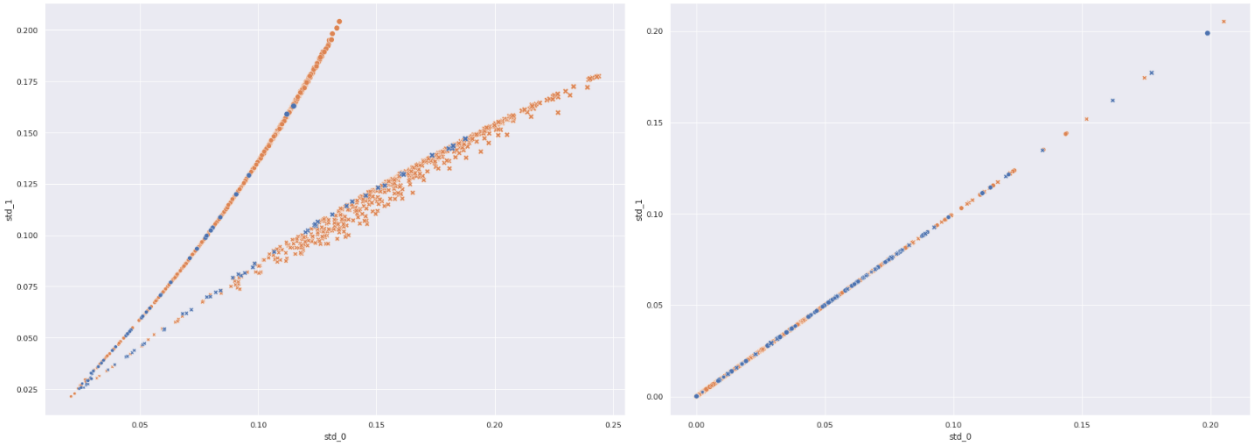


Figure 18 Standard deviation plot for the Ford A Dataset. MIMO model on the left and LA model on the right

Figures 16 and 18 show the mean confidence and standard deviation scatterplots for the Ford A dataset. m_0 and m_1 show the mean class confidences of models, and std_0 and std_1 give the standard deviation observed per class among ensemble members or samples. As can be seen from these figures, the separation between correct and incorrect predictions is less pronounced in the case of the Ford A dataset compared to the casting dataset. Still, the overlap of false predictions between train and test datasets for Laplace Approximation is much better than the previous base model and it enabled the creation of some certainty groups. The false predictions per dataset can be observed in Figure 19.

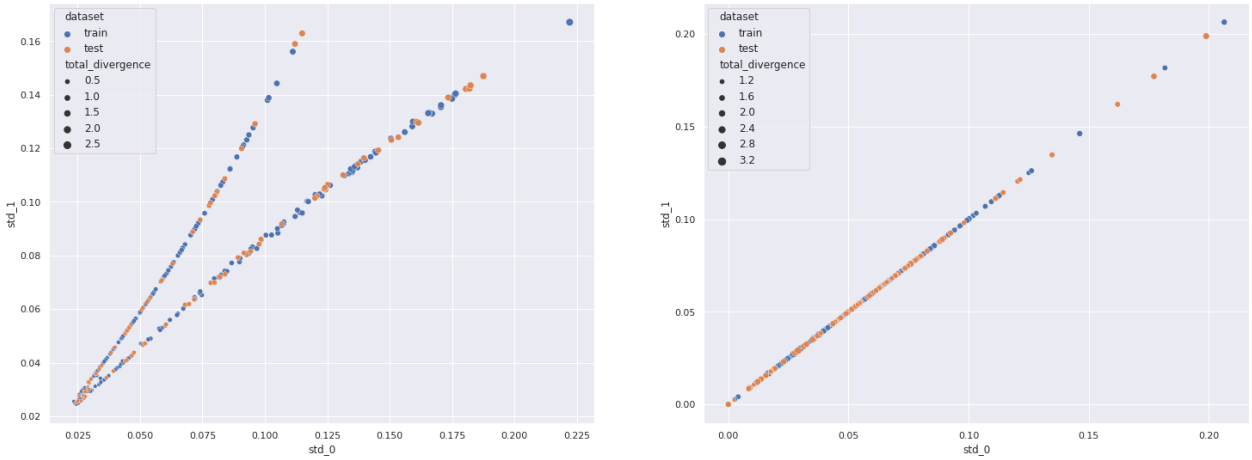


Figure 19 False prediction plots for MIMO and LA models respectively

Table 6 and 7 shows the certainty groups that were obtained using MIMO and LA models respectively. The results that show * next to their names are obtained by adding

a 5% safety margin to what was obtained from the validation group, meaning the threshold value found was not enough to get an absolute certainty group and threshold value was increased by 5%.

	Sum Mean	Sum Standard Deviation	Total Divergence	Delta Mean	Delta Standard Deviation
Absolute Certainty	5.68%	1.14%	0.76%	11.14%(*)	-
>99% Certainty	45.98% (99.01%)	10.30% (99.26%)	10.53% (99.28%)	53.9% (99.01%)	41.4% (99.1%)

Table 6 MIMO certainty group results for the Ford A dataset

	Total Divergence of Samples	Delta Mean	Standard Deviation
Absolute Certainty	-	18.56%(*)	-
Max Non-Absolute Certainty	7.27% (99.01%)	69.32%(99.2%)	10.75% (99.3%)

Table 7 Laplace Approximation Results for the Ford A dataset

As can be seen from the tables, even a small increment of the threshold by 5% not only enabled the creation of safety thresholds when it was previously not possible but also led to the biggest absolute certainty groups. The metrics that required safety margin also produced the biggest >99% certainty group. Similar to the casting dataset, the MIMO model outperformed Laplace Approximation, but LA outperformed the MIMO model in the delta mean threshold in both absolute certainty and >99% certainty groups.

MIMO Models, General Assessment

In general, MIMO models provided a higher level of performance when it came to creating certainty groups even though the NLL and calibration error losses observed were quite higher than observed in base and LA models. Ford A model behaving similarly in this issue to casting MIMO model decreases the probability of casting model showing high loss and error because of training method. It also increases the probability of the reason for higher loss/error seen for MIMO models being the high amount of divergence shown by the MIMO model for correct predictions.

There are very similar response patterns seen between the Ford A and casting MIMO models even though they work on different datasets and have significantly different structures. The first similarity is the frequently divergent results of one of the ensemble members for true predictions while the other two ensemble members display a similar amount of divergence in the vast majority of the instances. Figure 20 shows this behavior as a line plot. Different lines indicate the divergence amount of different

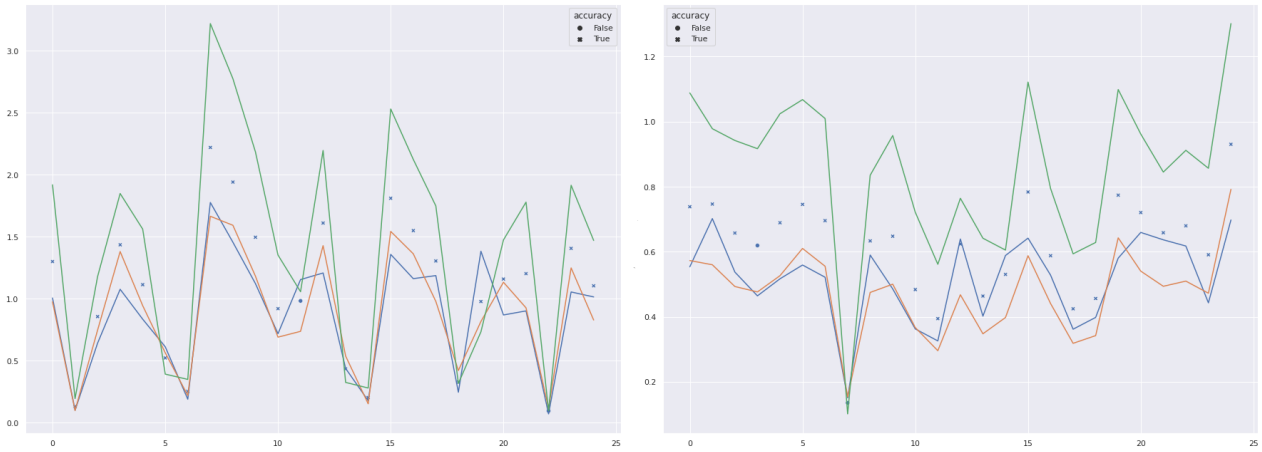


Figure 20 MIMO divergence plots of Casting and Ford A models

ensemble members, the x-axis displays different observations, and the y-axis displays the divergence amount. The points added to the plot show the mean divergence value for that observation, and the style of the point shows whether the instance was predicted correctly by the model. While some correct predictions also display small total divergence, this is much more common in the case of incorrect predictions for both of the models.

Another interesting behavior observed was in the clustering of the ensemble responses. Figure 21 shows the cluster map of ensemble members for Ford A dataset, where y-axis indicates different observations and x-axis indicates different ensembles as columns. The row colors next to the cluster map display prediction accuracy where red indicates

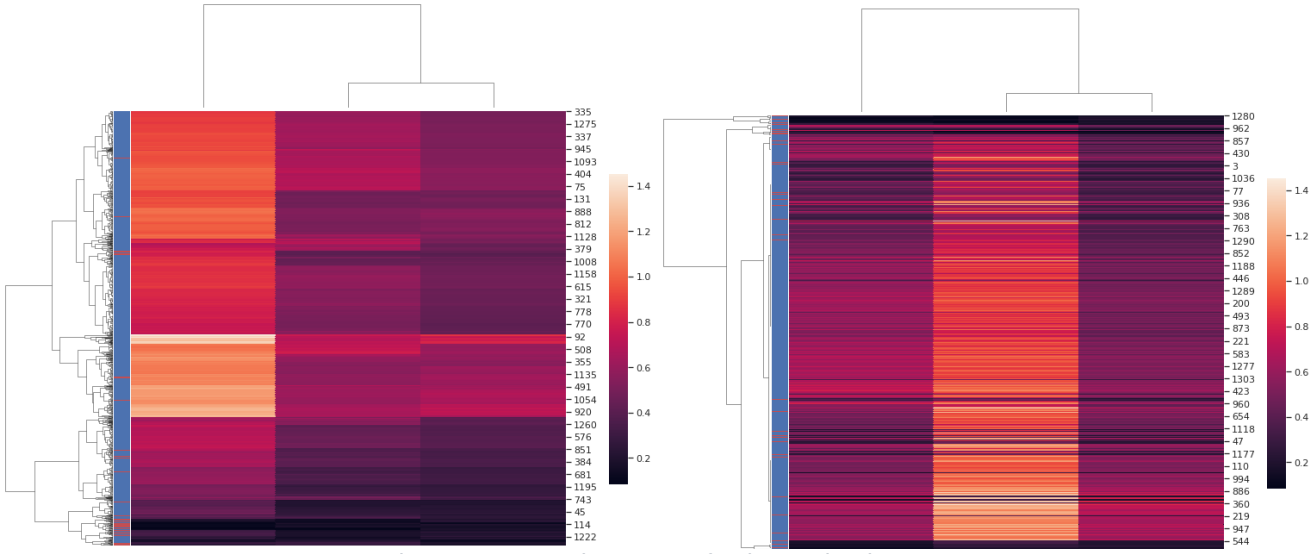


Figure 21 MIMO Clustermaps for the Ford A dataset

false and blue indicate true prediction. The left cluster map is created by using Euclidian distance as a clustering metric, and the right cluster map is created by using correlation as the metric.

Unsurprisingly, the Euclidian clustering process clusters the ensemble members with smaller divergence closer, but surprisingly correlation clustering process clusters the divergent ensemble member close to another ensemble member, which indicates the patterns of response these ensemble members are more similar even though the scale of the response significantly differs. This interesting phenomenon was observed in both Ford A and MIMO models.

Regarding the challenges of working with MIMO models, the following points can be made. Firstly, obtaining a working MIMO model was much harder to achieve, and small differences in the hyperparameters regularly resulted in a drastic decline in the model performance, and class distribution imbalances that can be easily handled by normal neural networks provided challenges for MIMO models. Secondly, the loss and accuracy results during the training of MIMO models were not predictable. This resulted in the need to train longer and be much more conservative when pruning a trial, which resulted in very long training periods even for these small datasets. Thirdly, the efforts of creating a MIMO model for the arrhythmia dataset indicate that MIMO models require a bigger dataset in order to provide comparable results to a basic neural network which makes them a poor fit for datasets with limited sample size and/or poor data quality.

Regardless of these challenges, MIMO models consistently provided a meaningful level of divergence between ensemble members, and provide a useful distinction between true and false classes while, as it will be shown later, not needing a significantly higher computational resource requirement than a base neural network. This makes them a good candidate for giving information on prediction uncertainty and the creation of certainty groups.

Laplace Approximation, General Assessment:

During the experiments, Laplace Approximation has left the following impressions. In general Laplace, Approximation was a beneficial tool for improving model calibration and adding model uncertainty, and the simplicity of adding LA to the models using the

Laplace Redux library presented this method as an easy and viable tool for introducing model uncertainty.

However, the improvements that were seen from this method were limited and these improvements haven't been effective enough in the cases where there was a significant miscalibration in the model. Moreover, adding more layers for approximation besides the last layer presented negligible improvements in model performance while vastly increasing the resources needed. It also consistently performed worse than the MIMO model when it was used for certainty group creation in two of the datasets which decreased the attractiveness of this method compared to MIMO.

Nevertheless, one needs to consider the following while evaluating these results. While Laplace Approximation can be applied to all conventional neural networks, the performance that can be expected from it highly depends on the neural network it is applied to. Laplace Approximation is accurate for convex functions that have most of their mass concentrated to the maximum (Peng 2022) and suffers significant performance issues on neural networks that have non-convex and/or chaotic loss landscapes. Because of this, a more complex model selection process that also evaluates the model loss landscape would be needed in order to see the potential of this method more clearly. One way to achieve this would be to record the eigenvalue curvature matrices or visualize loss landscapes presented in the paper Laplace Approximation for Uncertainty Estimation of Deep Neural Networks (Humt 2019). This can be applied to promising neural networks and indicate the level of success that can be expected from that network when used for Laplace Approximation.

Uncertainty Estimation under Resource Constraints

This section will discuss the objective of uncertainty estimation under limited computational resources in two different sections. First, it will give a short overview of the relationship between deep learning model performance and model size, with a concrete example that was observed during this study, then it will talk about the limitations that were put during the creation of the models in order to comply with resource limitation and present the sizes of the networks that were used in this research.

Model Accuracy and Computational Resource Trade-off

The application of certainty groups with strict computation limitations provides an interesting challenge with seemingly contradictory goals. On the one hand, the process of certainty group setting requires an exceptionally well-performing model that can recognize even the minute patterns in a given data, since the certainty group setting requires both very high confidence from a prediction and exceptional calibration of the network. Any reduction in the optimal performance caused by the computational limit can completely prevent the use of certainty groups and negatively affect the uncertainty estimation.

On the other hand, this level of recognizing complex patterns requires a significant amount of computation for any classification task that is complex enough to require neural networks and cannot feasibly be solved by a machine learning method that can be understood and predictable. Because of these two opposing forces, the certainty group setting may be seen as not possible with any significant restrictions on resources.

However, as this paper hinted earlier in the section related to Ford A dataset results, resource limitation can be a tremendous help when it comes to creating better-performing networks. Overparametarization of deep learning models is a significant issue that results in overfitting and miscalibration even by the best-performing public models (Guo et al. 2017a). These model performance issues even come with rapidly increasing power requirements that are becoming a significant contributor to our carbon

footprint where the model training of modern complex models is estimated to produce the lifetime carbon emissions of five cars (Strubell et al. 2019), and even running these models requires server clusters with high capabilities.

Because of these reasons, limiting the model size and complexity to a level that can achieve what is needed from the model should be one of the important considerations even when such a limit is not required. This is especially important for models that are going to be used for Laplace Approximation since the chaotic loss landscapes that are caused by complex models hamper the Gaussian approximation and reduce the performance improvement that can be achieved by Laplace Approximation (Humt 2019).

The information presented in Table 8 shows the evaluation metrics for two different Ford A models. The second model is the original model that was obtained by maximizing model accuracy with loose resource constraints, and the first model shows the smaller model that was created by optimizing both accuracy and number of flops together.

	Accuracy	NLL (↓)	ECE(↓)	RMSCE(↓	AUROC
New Model Test Dataset	0.9311	0.1871	**0.0120**	0.0269	0.9812
New Model Train Dataset	0.9486	0.1362	0.0249	0.0323	0.9912
Old Model Test Dataset	0.9485	0.2197	0.0323	0.0444	0.9858
Old Model Train Dataset	0.9986	0.0098	0.0048	0.0190	0.9999

Table 8 Ford A base model performance comparison

As it can be clearly observed, the smaller model has a much lower NLL and calibration error, and it performs at a very similar level between the training and test dataset, meaning it has significantly lower overfitting and it is much more useful for setting certainty thresholds. This comes at the expense of slightly lower accuracy, which is acceptable when the primary goal is not just to maximize the number of correct predictions but also to expect reliability.

	flops	parameters	runtime(in microseconds)
Ford A Old	41,842,944	4,602,530	0.439
Ford A New	3,726,528	540,658	0.347

Table 9 Ford A base models resource comparison

Table 9 shows the number of flops and parameters each model has as well as their runtime. The bigger model has 13 times more flops and 9 times more parameters than

the smaller model. The small difference in runtime is misleading and caused by both models being run on a system that has much higher resources than these models require. In the case where the models are run on systems with little capabilities such as microcontrollers, the runtime of the bigger model would require considerably more time relative to the smaller model.

However, there is certainly a compromise between these two forces. In order to investigate this trade-off, the Optuna study with multiple objective optimizations was conducted. As explained earlier, this study aimed to minimize the number of flops needed by the model and maximize the model accuracy on the test dataset. Figure 22 shows the scatter plot of this study with 350 different trials where each point signifies the results of a trial. The blue hue indicates the trial's order, and the red hue indicates the model performance perceived by Optuna. The green dot on this figure indicates the final model that was chosen as Ford A base model. Y axis shows the number of flops (multiplied by 16 due to the batch size) and x shows the final accuracy of the models.

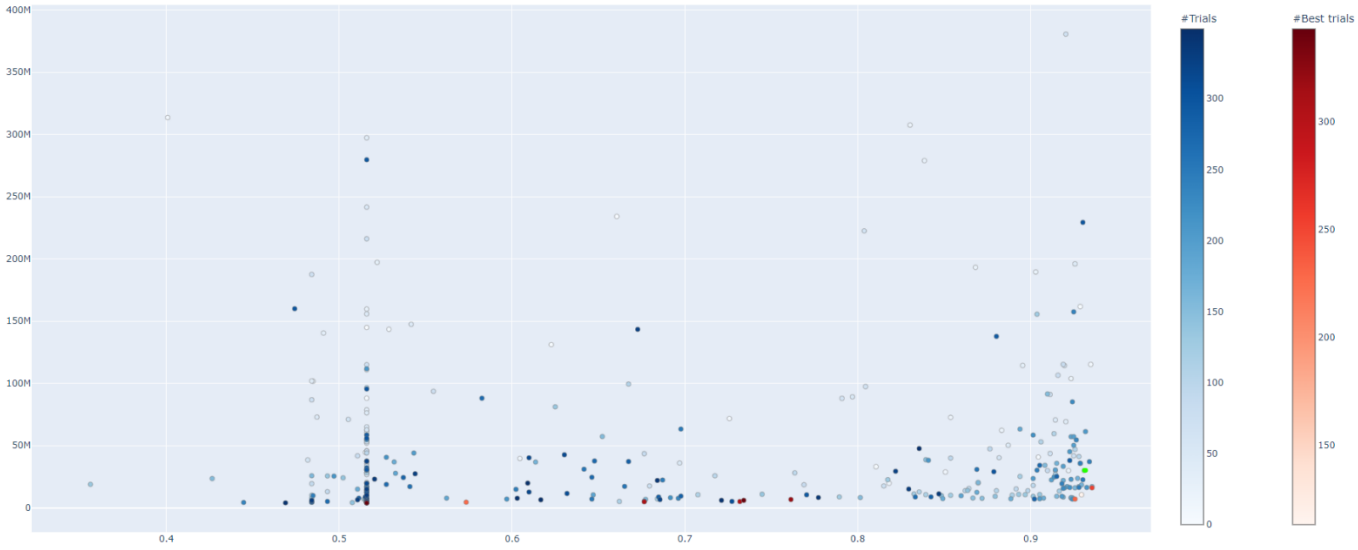


Figure 22 Ford A multi-objective optimization plot

There are multiple interesting points that were derived from this study regarding the viability of Optuna for finding computationally efficient models. Firstly, Optuna was able to understand the relationship between the number of flops and hyperparameters much quicker, resulting in a smaller time spent needed to optimize the number of flops, which is highlighted by the clustering of points much closer to the lower end of the y-axis.

However, since Optuna gives equal importance to both reducing flops and increasing accuracy the models that are seen as best by Optuna perform very poorly on accuracy. Moreover, Optuna currently doesn't support pruning for multiple objective optimizations and because the vast majority of the trials resulted in models with unusable performance, a huge chunk of the runtime for the trial was spent on trials that were clearly not helpful. Because of these reasons, multiple objective optimization is only a viable strategy for small and relatively simple model creation where the cost of wasted trials is negligible. In order to make it viable for more complex applications a more sophisticated objective function that discourages the minimization of the number of flops in exchange of a significant reduction in accuracy, and/or pruning functionality for unviable trials is needed.

Model Resource Comparison

During the creation of the neural networks, the following constraints were applied:

- Maximum three hidden layers
- Maximum four convolutional layers
- Maximum 128 channels for convolutional layers except for the last convolutional layer which had 256 channel upper limit
- Maximum 2056 neurons for linear layers

The adaptive Max Pool layer was used as the last pooling layer in order to make sure that the flattened last convolution layer wouldn't lead to an exceptionally large number of neurons. As mentioned previously, the number of convolution layers was increased for the Ford A MIMO model in order to compress the model size that would've been needed with only 4 layers, and the image size was shrunk to 64 pixels rather than 128 pixels for the casting dataset for the reduction of computational resource needed

Table 10 shows the number of flops and number of parameters each model has in addition to the runtime of each model. Two results are particularly interesting among these results. Firstly, the casting MIMO model was able to get higher accuracy than the base model even with significantly smaller calculations needed to run it. Secondly, the Ford A MIMO model is much larger than what was expected which is caused by the model having a very high total number of channels when all of its convolutional layers

combined. The probability of being able to create a much smaller MIMO network for the Ford A dataset is quite high.

	flops	parameters	runtime(in microseconds)
Casting Base Model	80,175,616	4,917,154	4.861
Casting MIMO Model	63,012,075	14,987,071	4.162
Ford A Base Model	3,726,528	540,658	0.347
Ford A MIMO Model	41,781,170	22,087,858	1.510
Arrhythmia Base Model	48,435	48,743	0.058

Table 10 Benchmarking results for the used neural networks

In general, models are relatively small and require only a small number of computations to run. While the constraints employed prevented the creation of very large models, there would certainly be a significant amount of optimization that can be done.

In the case of Laplace models, it wasn't possible to calculate their number of flops and parameters due to Laplace models not working with any of the libraries that calculate these values. However, Figure 23 gives an idea about the runtime of these models. The x-axis shows the number of samples requested per instance from the model and the y-axis shows the runtime. The blue line shows the runtime of LA model created with neural network sampling, while the orange line shows the generalized linear model sampling method. Runtime wise both methods show nearly equivalent results. However, the memory requirements of the methods vary considerably. The neural network sampling creates models that require ten times more memory (as an example ~30MBs for glm Ford A model vs ~300MB for nn Ford A model) and if this problem isn't simply caused by inefficiencies in code, it would make neural network sampling not viable for systems with constraints on available memory.

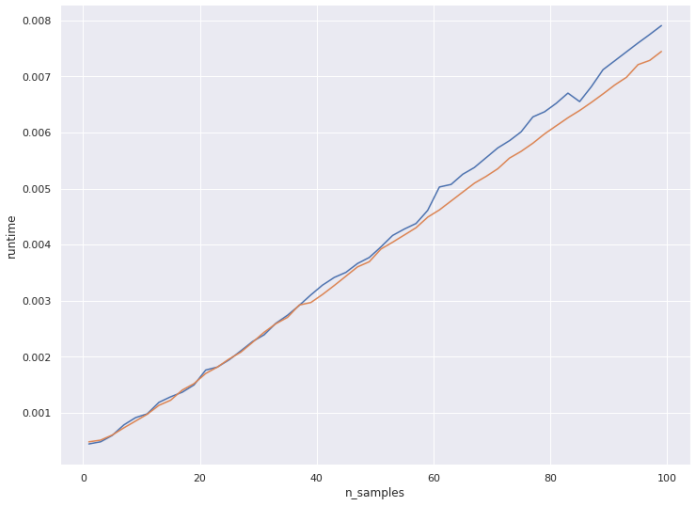


Figure 23 Runtime of Ford A Laplace model with relation to the number of samples requested

Conclusion

The overall conclusions in this paper can be summarized by these few points. Firstly, high levels of certainty groups are possible for simple tasks and well-calibrated neural networks as in the case of the casting dataset with MIMO. The computational resource limitation can even be an assistance in creating certainty groups rather than an obstacle.

MIMO models consistently performed better than Laplace Approximation in creating certainty groups and were only outperformed by LA in the case of >99% certainty group for the Ford A dataset. It also provided more metrics to evaluate since the divergence of ensemble members was a good predictor for certainty in the case of MIMO as opposed to the divergence of sample results from the Laplace Approximation.

Laplace Approximation was much easier to implement, and with the Laplace Redux library, it only required a few lines of code. Obtaining a viable MIMO model came with many challenges for all of the datasets, required many times more computation time for finding the best performing model, and it couldn't even be achieved for the arrhythmia dataset. This makes Laplace Approximation still a desirable option in order to introduce uncertainty to the model, especially when the model's loss function is fit to be approximated by this approach.

Obtaining a reliable model and certainty group selection process requires a much bigger dataset than obtaining a model with a very high level of accuracy as has been observed from the arrhythmia dataset.

None of the metrics that were tested as a certainty threshold consistently outperformed the others, indicating that the certainty group setting depends highly on the dataset and a metric that would consistently perform the best doesn't seem to exist.

Further Research

This thesis signaled multiple future research topics which can help find further answers in the area of uncertainty quantification. As mentioned earlier, using metrics to make sure a model is fit to be used in Laplace Approximation would increase the benefits that can be obtained from Laplace Approximation and provide a more meaningful result regarding the potential of Laplace Approximation.

Additionally, accuracy alone is a poor predictor of model calibration and robustness. Because of this, finding additional metrics to use during the hyperparameter selection would certainly be useful, and can be a good additional objective optimization parameter while sampling model hyperparameters.

The ability to maximize accuracy while minimizing computational cost was certainly a great tool but was unfeasible to apply in most cases. Finding a more sophisticated objective function that can enable the TPE sampler to minimize flops without sacrificing significant levels of accuracy during multi-objective optimization would also be a very significant contribution.

Applying simple machine learning and statistical methods while creating certainty groups can certainly lead to bigger and/or more reliable certainty groups, and can still be understandable and predictable in nature. This was talked about during the discussions regarding the casting dataset, and it was also seen as a promising avenue of research in the analysis of Ford A dataset results. Figure 24, the sum of standard deviation and delta mean scatterplot of Ford A dataset shows this possibility. As it can be observed the false predictions all fall into two linear lines with respect to the delta mean, while a significant portion of the true predictions lies outside of this line, which is indicated with the red ellipsoid. Because of this, there are

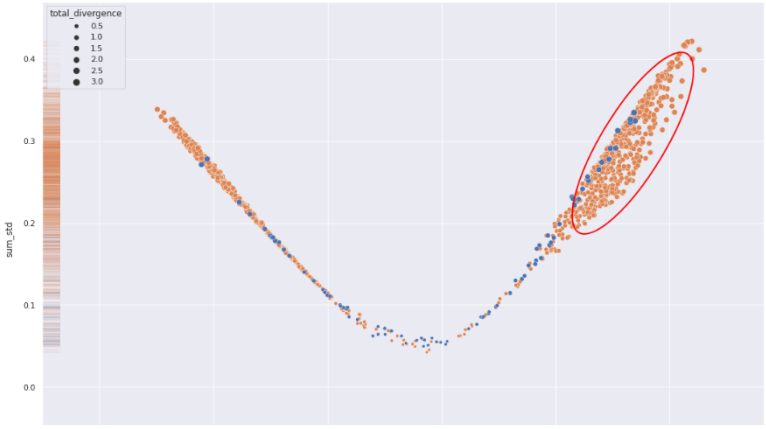


Figure 24 Ford A dataset standard deviation and mean plot for the MIMO model

observations that can be easily classified correctly with absolute certainty if a method that evaluates the relationship between parameters was used while creating the certainty groups.

Clustering of ensemble divergence has also shown that unlabeled statistical methods may also prove useful in uncertainty estimation and would enable skipping the costly labeling process. Moreover, the interesting response patterns of ensemble members that were discussed in the related section may also be a worthwhile area of research.

Lastly, using both MIMO and Laplace approximation in tandem can lead to an improvement over even the results of the MIMO model. Unfortunately, testing this was not possible due to MIMO models needing customized training method that is incompatible with the Laplace library, however, this was only a technical limitation. As mentioned in the literature, while ensemble methods provide uncertainty quantification they are not Bayesian by nature, and using an ensemble method that achieves Bayesian approximation can lead to superior results compared to both of these methods alone(Pearce et al.).

Publication bibliography

Abdar, Moloud; Pourpanah, Farhad; Hussain, Sadiq; Rezazadegan, Dana; Liu, Li; Ghavamzadeh, Mohammad et al. (2021): A Review of Uncertainty Quantification in Deep Learning: Techniques, Applications and Challenges. In *Information Fusion* 76 (1), pp. 243–297. DOI: 10.1016/j.inffus.2021.05.008.

Akiba, Takuya; Sano, Shotaro; Yanase, Toshihiko; Ohta, Takeru; Koyama, Masanori (2019): Optuna: A Next-generation Hyperparameter Optimization Framework. Available online at https://arxiv.org/pdf/1907.10902.

Azevedo-Filho, Adriano; Shachter, Ross D.: Laplace's Method Approximations for Probabilistic Inference in Belief Networks with Continuous Variables. In *UAI-P-*. Available online at https://arxiv.org/pdf/1302.6782.

Babic, Boris; I. Glenn, Cohen; Theodoros, Evgeniou; Sara, Gerke (2021): When Machine Learning Goes Off the Rails. A guide to managing the risks. In *Harvard Business Review*. Available online at https://hbr.org/2021/01/when-machine-learning-goes-off-the-rails.

Bagnall, Anthony (2008): Ford A Dataset. Sensor data file.

Bergstra, James; Bardenet, Rémi; Bengio, Yoshua; Kégl, Balázs (2011): Algorithms for Hyper-Parameter Optimization. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, K.Q. Weinberger (Eds.): Advances in Neural Information Processing Systems, vol. 24: Curran Associates, Inc. Available online at https://proceedings.neurips.cc/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf.

Bickel, P.; Diggle, P.; Fienberg, S. (1996): Bayesian Learning for Neural Networks. With assistance of P. Diggle, S. Fienberg. N. New York: Springer New York (Lecture Notes in Statistics Ser, v.118). Available online at https://ebookcentral.proquest.com/lib/kxp/detail.action?docID=3074890.

Dabhi, Ravirajsinh et al. (2019): Casting Dataset. Collection of Image Files.

Daxberger, Erik; Kristiadi, Agustinus; Immer, Alexander; Eschenhagen, Runa; Bauer, Matthias; Hennig, Philipp (2021): Laplace Redux -- Effortless Bayesian Deep Learning. Available online at https://arxiv.org/pdf/2106.14806.

Detlefsen, Nicki; Borovec, Jiri; Schock, Justus; Jha, Ananya; Koker, Teddy; Di Liello, Luca et al. (2022): TorchMetrics - Measuring Reproducibility in PyTorch. In *JOSS* 7 (70), p. 4101. DOI: 10.21105/joss.04101.

Dua, Dheeru; Graff, Casey (2017): UCI Machine Learning Repository. Available online at http://archive.ics.uci.edu/ml.

Falkner, Stefan; Klein, Aaron; Hutter, Frank (2018): BOHB: Robust and Efficient Hyperparameter Optimization at Scale. In Jennifer Dy, Andreas Krause (Eds.): Proceedings of the 35th International Conference on Machine Learning, vol. 80: PMLR (Proceedings of Machine Learning Research), pp. 1437–1446. Available online at https://proceedings.mlr.press/v80/falkner18a.html.

Frankle, Jonathan; Carbin, Michael: The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *ICLR*. Available online at https://arxiv.org/pdf/1803.03635.

Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron (2016): Deep learning. Cambridge, Massachusetts: The MIT Press (Adaptive computation and machine learning).

Guo, Chuan; Pleiss, Geoff; Sun, Yu; Weinberger, Kilian Q. (2017a): On Calibration of Modern Neural Networks. Available online at https://arxiv.org/pdf/1706.04599.

Guo, Chuan; Pleiss, Geoff; Sun, Yu; Weinberger, Kilian Q. (2017b): On Calibration of Modern Neural Networks. Available online at https://arxiv.org/pdf/1706.04599.

Guvenir, H. A.; Acar, B.; Demiroz, G.; Cekin, A. (1997): A supervised machine learning algorithm for arrhythmia analysis. In Alan Murray, Steven. Ed Swiryn (Eds.): Computers in cardiology. Computers in Cardiology 1997. Lund, Sweden, 7-10 Sept. 1997. Institute of Electrical and Electronics Engineers: IEEE, pp. 433–436.

Guvenir, H., Acar, Burak & Muderrisoglu, Haldun (1998): Arrhythmia.

Hansen, L. K.; Salamon, P. (1990): Neural network ensembles. In *IEEE Trans. Pattern Anal. Machine Intell.* 12 (10), pp. 993–1001. DOI: 10.1109/34.58871.

Havasi, Marton; Jenatton, Rodolphe; Fort, Stanislav; Liu, Jeremiah Zhe; Snoek, Jasper; Lakshminarayanan, Balaji et al. (2020): Training independent subnetworks for robust prediction. Available online at https://arxiv.org/pdf/2010.06610.

Humt, Matthias (2019): Laplace Approximation for Uncertainty Estimation of Deep Neural Networks.

Lakshminarayanan, Balaji; Pritzel, Alexander; Blundell, Charles (2016): Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. Available online at https://arxiv.org/pdf/1612.01474.

LeCun, Yann; Bengio, Yoshua; Hinton, Geoffrey (2015): Deep learning. In *Nature* 521 (7553), pp. 436–444. DOI: 10.1038/nature14539.

Lodes, Lukas; Schiendorfer, Alexander (2022): Certainty Groups: A Practical Approach to Distinguish Confidence Levels in Neural Networks. In *PHME_CONF* 7 (1), pp. 294–305. DOI: 10.36001/phme.2022.v7i1.3331.

Mackay, David J. C. (1992): The Evidence Framework Applied to Classification Networks. In *Neural Computation* 4 (5), pp. 720–736. DOI: 10.1162/neco.1992.4.5.720.

Mackay, David J. C. (1995): Probable networks and plausible predictions — a review of practical Bayesian methods for supervised neural networks. In *Network: Computation in Neural Systems* 6 (3), pp. 469–505. DOI: 10.1088/0954-898X_6_3_011.

Martens, James; Grosse, Roger (2015): Optimizing Neural Networks with Kronecker-factored Approximate Curvature. Available online at https://arxiv.org/pdf/1503.05671.

Oflaz, Zübeyir (2022): Deep-Learning-Uncertainty-Quantification-Methods: GitHub. In *GitHub repository*.

Patel, Ronak (2020): AI in Manufacturing | Quality Inspection | AI project. Industry 4.0 Analysis. Available online at https://www.youtube.com/watch?v=4sDfwS48p0A, checked on 8/31/2022.

Pearce, Tim; Leibfried, Felix; Brintrup, Alexandra; Zaki, Mohamed; Neely, Andy: Uncertainty in Neural Networks: Approximately Bayesian Ensembling. In *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS*. Available online at https://arxiv.org/pdf/1810.05546.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O. et al. (2011): Scikit-learn: Machine Learning in Python. In *Journal of Machine Learning Research* 12, pp. 2825–2830.

Peng, Roger D. (2022): Advanced Statistical Computing. Leanpub. Available online at https://bookdown.org/rdpeng/advstatcomp/.

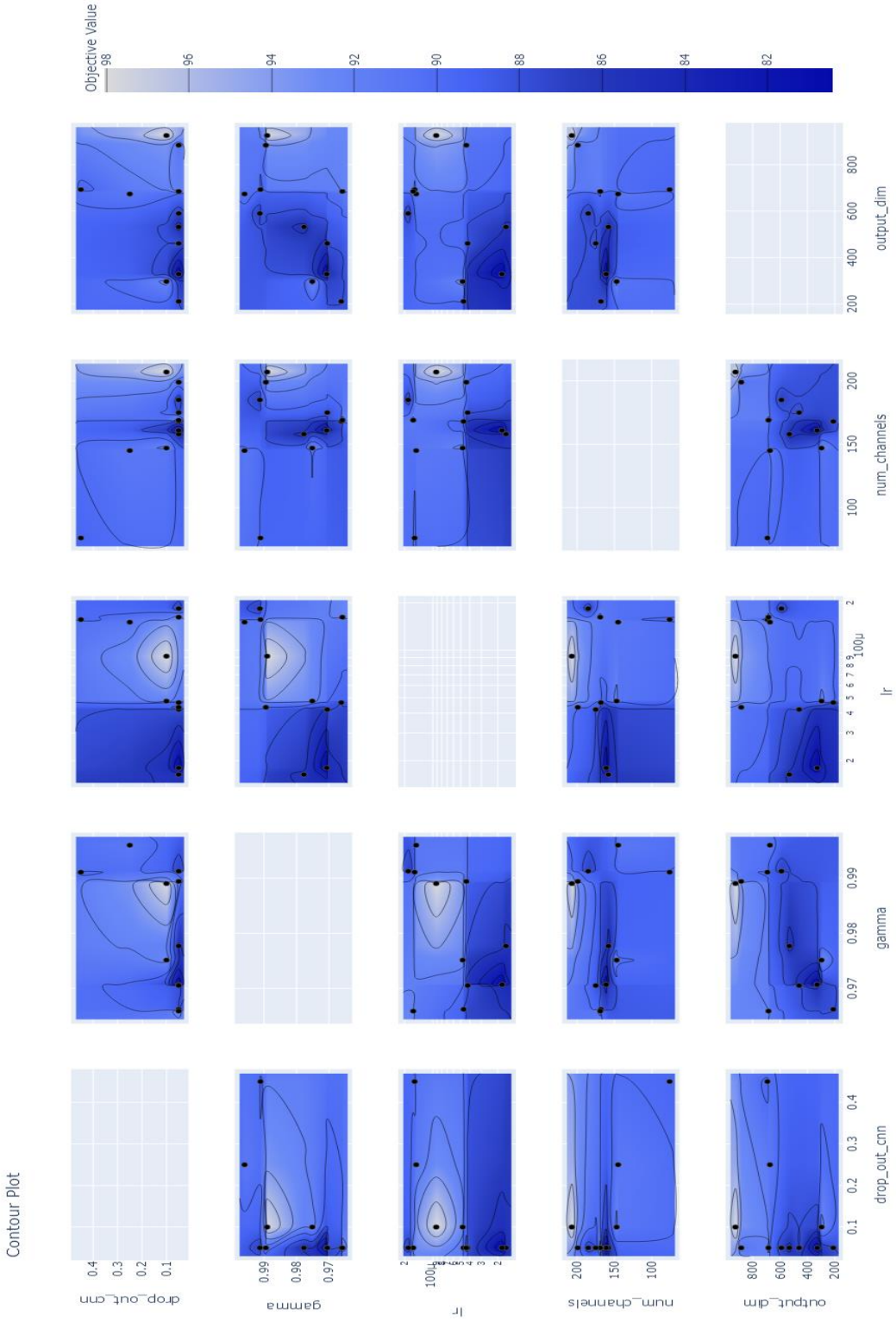
Pouyanfar, Samira; Sadiq, Saad; Yan, Yilin; Tian, Haiman; Tao, Yudong; Reyes, Maria Presa et al. (2019): A Survey on Deep Learning. In *ACM Comput. Surv.* 51 (5), pp. 1–36. DOI: 10.1145/3234150.

Strubell, Emma; Ganesh, Ananya; McCallum, Andrew (2019): Energy and Policy Considerations for Deep Learning in NLP. Available online at https://arxiv.org/pdf/1906.02243.

Tjoa, Erico; Guan, Cuntai (2019): A Survey on Explainable Artificial Intelligence (XAI): Toward Medical XAI (11). Available online at https://arxiv.org/pdf/1907.07374.

Appendix

Optuna Detailed Contour Map



Arrhythmia Classification Names

Class Number	Classification Name
0	Normal
1	Ischemic changes (Coronary Artery Disease)
2	Old Anterior Myocardial Infarction
3	Old Inferior Myocardial Infarction
4	Sinus tachycardy
5	Sinus bradycardy
6	Ventricular Premature Contraction (PVC)
7	Supraventricular Premature Contraction
8	Left bundle branch block
9	Right bundle branch block
10	1. degree AtrioVentricular block
11	2. degree AV block
12	3. degree AV block
13	Left ventricule hypertrophy
14	Atrial Fibrillation or Flutter
15	Others